Survey

Hao WU, Li CHEN

May 15, 2016

1 Application

Integration of geographical information into network service has rendered new services possible [6], which include

1. *Adding location attributes to user profiles.* The geographical information is called geo-Tag, which could be used for later analysis, such as population survey.
2. *Finding people.* People usually searched for can be divided into two classes: persistently connected to or temporarily connected to the query launcher. Persistent connections includes friendship, family relation, etc. Temporary connections are connections exist for a short period of time. When a man needs special services, for example, he loses his key, he searches the nearby for special assistance. The service provider and the man connects temporarily.
3. *Sharing content and collaborating.* People don't know each other are able to collaborate with each other via geo-social network. Google Map has enabled users to add description to a particular site, which could be a restaurant, a tourist spot, a hotel etc. Users can refer to previous descriptions when searching on the Map.
4. *Intelligent recommendation.* Make recommendation of potential users and places of interests.

2 Queries

2.1 Geo-Circle of Friend Query

W. Liu et al. [8] proposed the Geo-Circle of Friend Query (gCoFQ) which, given a query point q (an user in the social network), returns a group of q's friends who are both socially and geographically close to each other.

To make the problem more clear, consider the undirected weighted graph $G =< V, E, W >$ with vertex set V, edge set E and function W:$E \rightarrow R$. For $u, v \in V$, $(u, v) \in E$ if u and v are friends and $W(u, v)$ is the measure of friendship between u and v. The form of function W varies with respect to different scenarios. In the case of messenger app, W could be defined as the frequency of communication between u and v. In the case of co-authorship, $W(u, v)$ can be $1/\sum_i \frac{1}{x_i}$, where x_i is the number of authors in the ith paper u and v co-write. Even in the same application scenario, W may be constructed in different ways. We will come back to this issue latter.

Based on the definition of W, we define closeness between u and v as

$$closeness(u, v) = \text{the accumulative weights of the shortest path between u and v in G}$$

For geographical distance between u and v, we use Euclidean distance, which is denoted by $||u, v||$.

Given the social closeness and geographical closeness of u and v, we are able to define the geo-social distance between u and v,

$$dist(u, v) = \lambda \frac{||u, v||}{||V||} + (1 - \lambda) \frac{closeness(u, v)}{closeness(V)}$$

where λ is a parameter, $||V||$ is the maximum Euclidean distance between any pair of user in V and closeness(V) the maximum social distance between any pair of users.

The ranking function of a group of users S is

$$dist(S) \doteq \max_{u,v \in S} dist(u, v)$$

Geo-Social Circle of Friend Query Given a query point q and parameter k, Geo-Social Circle of Friend Query returns a subset $V_q \in V$, s.t.

$$q \in V_q \quad (1)$$

$$|V_q| = k + 1 \quad (2)$$

$$dist(V_q) \text{ is the smallest} \quad (3)$$

2.2 Socio-Spatial Group Query

Similar to Geo-Social Circle of Friend Query, given a parameter k, Social-Spatial Group Query(SSGQ) [11] returns a group of k users. The difference is that Geo-Social Circle of Friend Query will returns k + 1 users which includes the query point q. On the other hand, the query point q in Socio-Spatial Group Query is not a user but a location instead.

The change of query point from a user to a location makes sense for real world applications. For example, location-based advertisements can leverage SSGQ to a group of friends for a preferred restaurant to order to push mobile coupon.

Borrowing the notion defined in Geo-Social Circle of Friend Query, we could immediately give a similar problem statement.

Socio-Spatial Group Query Given a query point q(a location) and parameter k, Socio-Spatial Group Query returns a subset $V_q \in V$, s.t.,

$$|V_q| = k \quad (4)$$

$$\max\{dist(V_q), \max_{v \in V_q} \lambda \frac{||q, v||}{||V||}\} \text{ is the smallest.} \quad (5)$$

The intuition behind equation (5) is that the selected k friends should be socially close to each other and geographical close to each other as well as the query point.

Indeed the query proposed by Yang, D.N. et al. [11] is slightly different from the one stated above. The difference is twofold.

1. **Quantification of the idea that returned users should be close to query point.** What we propose is that $\max_{v \in V_q} ||q, v||/||V||$ should be as small as possible. What Yang D.N. [11] suggested is to optimize $\sum_{v \in V_q} ||q, v||/||V||$, i.e., the sum of distance of vertex in V_q to query point q.

2. **Social Closeness** While Geo-Social Circle of Friend Query use the accumulative weight of shortest path between vertex pair to measure the friendship, in Socio-Spatial Group Query, the average of people a user $v \in V_q$ doesn't know in group V_q is used to measure the social connectivity between users.

Socio-Spatial Group Query Given a query point q, a parameter k, and a control parameter n, the Socio-Spatial Group Query returns a group of users $V_q \in V$, s.t.,

$$|V_q| = k \tag{6}$$

$$\sum_{v \in V_q} (|V_q| - 1 - N_v)/|V_q| \leq n \tag{7}$$

$$\sum_{v \in V_q} ||v, q|| \text{ is the smallest} \tag{8}$$

2.3 Geo-Social Ranking Top-k Query

The aforementioned queries all focus on returning a group of users instead of individual user. Also, the social connectivity is evaluated within a specific group of users. Geo-Social Ranking Top-k(GSR Top-k query) Query proposed by Armenatzoglou, N. [4] is able to extract the top-k users considering the their distance to the query point q, the number of friends in the vicinity of q, and possible social connectivity of those friends.

In Geo-Circle of Friends query, we use $dist(V_q)$ to measure a group of user V_q. In Socio-Spatial Group Query, $\max\{dist(V_q), \max_{v \in V_q} \lambda \frac{||q,v||}{||V||}\}$ is used to filter the best group. Similarly, in Geo-Social Ranking Top-k Query, we need an evaluation function for each user. In the following discussion, we denote this function f.

GSR Top-k query. Given a query point q, a positive integer k, and a GSR function f, the query returns a list of k tuples $R = (\{v_1, f(q, v_1), V_1\}, ...,$ $\{v_k, f(q, v_k), V_k\})$ such that for each $1 \leq i \leq k$:

$$f(q, v_i) \geq f(q, v_{i+1}) \text{ and} \tag{9}$$

$$\nexists \{u, f(q, u), V_u\} \notin R : f(q, v_k) < f(q, u) \tag{10}$$

The remaining problem becomes how to define f. Suppose that we want a selected vertex v to process the following tow properties:

1. **v should be close to q**. This can be achieved by minimizing $||v, q||$.

2. **v have a set of friends that are close to q**. This notion could be more trickier. Let the set of vertex that are adjacent to v be V_v. Not all vertex in V_v are close to q. So we could not quantify this notion by $|V_v|$. We denote the set of friends close to q by $N_v \subset V_v$. Now the question becomes: what vertex in V_v should be included in N_v?

There could be many way to find N_v. The simplest way could be use a fix threshold r: $N_v = \{u|u \in V_v \&\& ||u, v|| \leq r\}$. But how can we determine the r?

One solution is to choose an r such that it gives the optimized value of f. We can find the r only after we know the form of f.

One simplest form of f could be

$$f = \lambda|N_v| + (1 - \lambda)(-||v, q||) \tag{11}$$

As before, λ is a parameter controlling the relative importance of the two terms. By optimizing this function we can not find a set of N_v that are in the vicinity of q, since it imposes no restriction on the location of N_v. Inspired by this, we change it to

$$f = \lambda|N_v| + (1 - \lambda)(-\sum_{u \in N_v} ||u, q||) \tag{12}$$

Adding the normalization terms F ($F \doteq \max_{v \in V} |V_v|$) the maximum degree of any vertex in G and C ($C \doteq \max_{v \in V} ||v, q||$) the maximum distance from any vertex to q,

$$f = \lambda \frac{|N_v|}{F} + (1 - \lambda)(1 - \frac{\sum_{u \in N_v} ||u, q||}{F * C}) \tag{13}$$

To see how f changes if we include one more v's friend u in N_v, as $|N_v|$ increases by one, f would increase by $\frac{\lambda}{F}$. Also, f would decrease by $(1-\lambda)\frac{||q,u||}{F*C}$.

In order for u to be included in f, the positive contribution should exceed the negative:

$$\frac{\lambda}{F} \geq (1 - \lambda)\frac{||q, u||}{F * C} \tag{14}$$

$$\longrightarrow ||q, u|| \leq \frac{\lambda C}{1 - \lambda} \tag{15}$$

This suggests that $r = \frac{\lambda C}{1-\lambda}$, and $N_v = \{u|u \in V_v \&\& ||u, v|| \leq \frac{\lambda C}{1-\lambda}\}$

Armenatzoglou, N.[4] also mentions several other way of constructing f. The difference between these functions is threefold:

1. The way to characterize the closeness between a user v and the query point q.

2. The way to characterize the closeness of a user v's relevant friend to query point q.

3. The way to combine the above two features. What we have explored so far are all linear combination. One advantage of linear combination is that it is easy to include more than two features. The queries we investigate so far consider only social and spatial features.

2.4 Geo-Social Keyword Search

We have mentioned that linear combination allows easy extension of features to higher dimensions. Geo-Social Keyword Search follows just this line and textual information is utilized.

To investigate the motivation of doing so, we take Google Map as an example of geographic and textual integration. Typical example of such search may be "Chinese restaurants nearby".

3 GeoSN Query Process System

We overview the literature on GeoSN query processing systems in industry and academia. In what follows, we examine the API exposed by commercial online services, as well as the algorithm and data structure reported in academic projects.

3.1 Commercial Products

3.1.1 Foursquare

Foursquare's Radar return the friends who are currently in the vicinity of a user. Foursqaure also expose their API for developers of LBS. The Foursquare API provides methods for accessing a resource such as a venue, tip, or user. In spirit with the RESTful model[9], each resource is associated with a URL. For example, information about Clinton Street Baking Co can be found as follows (assuming credentials for such information is verified cryptographically beforehand):

```
https://api.foursquare.com/v2/venues/40a55d80f964a52020f31ee3
```

Given a resource, you can then drill into a particular aspect, for example

```
https://api.foursquare.com/v2/venues/40a55d80f964a52020f31ee3/tips
```

Each returned tip will have its own ID, which corresponds to its own resource URL, for example

```
https://api.foursquare.com/v2/tips/49f083e770c603bbe81f8eb4
```

Foursquare does not expose user's GPS location via its API, and only allows access to venue information (in the `venuehistory` and `checkins` aspects of user API). Foursquare also reveals the social network between users via the `friends` property of any users, which developers can use to get list of friends as follows:

```
https://api.foursquare.com/v2/users/USER_ID/friends
```

Foursquare does not report how they manage their social network database. However, they use MongoDB for the storage of venues and check-ins[2]. The original Foursquare application relied on a single relational database. With this relational architecture, foursquare could not simply and easily scale to many nodes required for a high traffic application. As the company experienced rapid growth, it split the data to two nodes: one for checkins (the biggest data set) and one for everything else. Yet it was clear that check-ins would grow beyond what a single machine could handle, and that a long-term, scalable solution to Foursquare's growth was needed. Foursquare benefits from MongoDB's support for geospatial indexing, allowing it to easily query for location-based data.

MongoDB's document model, with independent JSON-like objects, maps well to object-oriented programming, in contrast with the schema-enforced table structures of relational databases. MongoDB allows foursquare to dramatically simplify its data model. For instance, rather than storing tags ('has wifi", "great for dates", "hotspot", etc.) in a separate table and relying on mapping tables and costly JOINs, in MongoDB tags are embedded directly into the document representing a venue. This is both more efficient at run-time, and easier for engineers to understand and manipulate.

3.1.2 Facebook

Facebook's Graph API is the primary way for developers to get data in and out of Facebook's platform. Aside from social graph, developers can obtain user's location via `place` API as follows.

```
/* make the API call */
FB.api(
    "/{place-id}",
    function (response) {
      if (response && !response.error) {
        /* handle the result */
      }
    }
);
```

Each place has a field called `overall_rating`, which is the overall rating of Place, on a 5-star scale.

3.1.3 ArcGIS

ArcGIS[1] exposes a variety of functionalities to developers, including location services (directions/guidance, geo-trigger[1]), mapping, and imagery. They claim that their data can be accessed via a RESTful API, but do not disclose details of how their data is managed. It is also unclear whether they have social networking functions in their system based on public information.

3.2 Academic Efforts

There has been few literature on GeoSN query processing[3, 12, 7, 10]. These works did not focus on important data management issues, which potentially undermine their practicality. More specifically, they tie the algorithms with specific data representations and indices, which may incurr significant overhead in large GeoSNs, because data representation scheme greatly affects the performance of any algorithm. Also, all of them assume that all the data are owned by a single entity, and can be handles by a single machine. Their specific problems are as follows:

[1] Geo-trigger notifies a user when friends enter a certain range.

- [7] uses an adjacency matrix for keeping info about the social graph, which may incur prohibitive storage overhead.
- [3, 10] make use of hybrid indices of both social and spatial data, which may suffer from enormous maintenance costs due to high check-in rates.
- [12, 10] do not specify how the social graph is stored.

3.3 A General Framework for Geo-Social Query Processing[5]

A general framework for processing GeoSN queries is proposed in [5]. The proposed architecture consists of three modules, depicted in Figure1: a social module (SM), a geographical module (GM), and a query processing module (QM). The SM stores only the social graph (e.g., friendship relations), and the GM keeps only geographical information (e.g., check-ins, venues, ratings). The QM is responsible for receiving GeoSN queries from users, executing them, and returning the results. The users do not communicate directly with the SM and GM. The SM, GM and QM can either be three separate servers, three separate clouds, or a single system (server or cloud). However, the tasks of the three modules are segregated.

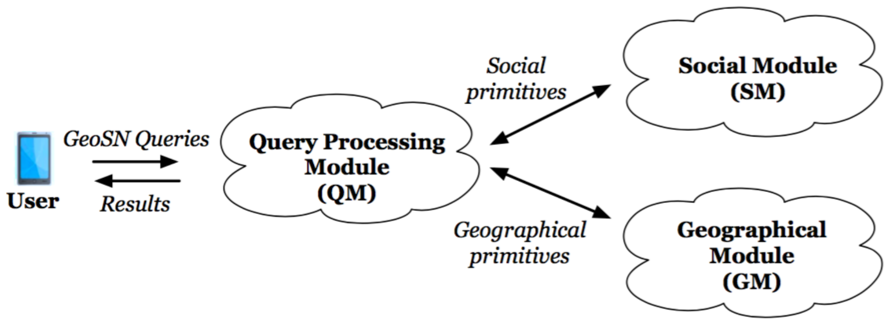

Figure 1: The proposed architecture[5]

The SM and GM interact only with the QM using well-defined social and geographical primitive queries. The SM and GM only execute their corresponding primitives on their stored data, and the primitives are given by QM based on the algorithms used to process the queries. The QM eventually assembles the final results by combining the outputs of the primitives, optionally exploiting auxiliary indices maintained locally.

The segregation of SM and GM allows their administration by different entities, e.g., the SM (GM) can be maintained by a company with expertise in social networking (resp. location-based services).

For instance, in UK and Japan, Facebook Places [3] cooperates with Factual [4], which provides infrastructure for location- based services. Glancee [12], a location-based service app, uses Facebook's social graph to connect nearby users. Another example is the cooperation of pure commercial social networks, e.g., Twitter or Facebook, and GeoSNs like Foursquare. A user who has both a Twitter or Facebook and a Foursquare account can post his Foursquare check-in at Twitter or Facebook [15]. Thus, if Facebook or Twitter needs the geographical information of users' check-ins to execute a GeoSN query, it obtains it from Foursquare. The separation of QM enables third-party companies that do not own any social or geographical data to implement GeoSN queries by solely interacting with the APIs of SM and GM (e.g., Agora [1]).

In addition, separating the functionality of SM and GM renders the management of social and geographical data more flexible, because the frequent check-in updates do not burden the relatively static social structures.

For example, due to an unexpected high rate of check-ins recently, Foursquare's system had a very long down- time. The problem was caused because their data are spread across multiple balanced database shards. When a shard is overused, a new one is added, followed by rebalancing. The rebalancing of the entire database caused the crash. In a segregated system, such a crash in GM would not affect SM.

First, our architecture can readily integrate modifications (e.g., a new, more efficient structure) in the implementation of SM without modifying GM, and vice versa. Second, novel GeoSN query types and algorithms can be devised, either by using a different combination of existing primitives or by implementing new ones, without the need of altering the SM and GM infrastructures. Last, social (geographical) data can be used independently for pure social (resp. geographical) queries, potentially through the same primitive operations utilized by GeoSN queries. As a result, a "traditional" social network can adopt our architecture without extra effort.

References

[1] Arcgis api guide. `https://developers.arcgis.com/javascript/latest/guide/index.html`. Accessed: 2016-05-10.

[2] Mongodb: Foursquare. `https://www.mongodb.com/customers/foursquare`. Accessed: 2016-05-10.

[3] Amir, A., Efrat, A., Myllymaki, J., Palaniappan, L., and Wampler, K. Buddy tracking—efficient proximity detection among mobile friends. *Pervasive and Mobile Computing 3*, 5 (2007), 489–511.

[4] Armenatzoglou, N., Ahuja, R., and Papadias, D. Geo-social ranking: functions and query processing. *The VLDB Journal—The International Journal on Very Large Data Bases 24*, 6 (2015), 783–799.

[5] Armenatzoglou, N., Papadopoulos, S., and Papadias, D. A general framework for geo-social query processing. *Proceedings of the VLDB Endowment 6*, 10 (2013), 913–924.

[6] Huang, Q., and Liu, Y. On geo-social network services. In *Geoinformatics, 2009 17th International Conference on* (2009), Ieee, pp. 1–6.

[7] Khoshgozaran, A., and Shahabi, C. Private buddy search: Enabling private spatial queries in social networks. In *Computational Science and Engineering, 2009. CSE'09. International Conference on* (2009), vol. 4, IEEE, pp. 166–173.

[8] Liu, W., Sun, W., Chen, C., Huang, Y., Jing, Y., and Chen, K. Circle of friend query in geo-social networks. In *Database Systems for Advanced Applications* (2012), Springer, pp. 126–137.

[9] Richardson, L., and Ruby, S. *RESTful web services.* " O'Reilly Media, Inc.", 2008.

[10] Scellato, S., Mascolo, C., Musolesi, M., and Latora, V. Distance matters: Geo-social metrics for online social networks. In *WOSN* (2010).

[11] Yang, D.-N., Shen, C.-Y., Lee, W.-C., and Chen, M.-S. On socio-spatial group query for location-based social networks. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining* (2012), ACM, pp. 949–957.

[12] Yiu, M. L., Šaltenis, S., Tzoumas, K., et al. Efficient proximity detection among mobile users via self-tuning policies. *Proceedings of the VLDB Endowment 3*, 1-2 (2010), 985–996.