Towards Location-based Social Networking Services*

Chi-Yin Chow[1] Jie Bao[2] Mohamed F. Mokbel[2]
[1]Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong
[2]Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, USA
chiychow@cityu.edu.hk {baojie,mokbel}@cs.umn.edu

ABSTRACT

Social networking applications have become very important web services that provide Internet-based platforms for their users to interact with their friends. With the advances in the location-aware hardware and software technologies, location-based social networking applications have been proposed to provide services for their users, taking into account both the spatial and social aspects. Unfortunately, none of existing location-based social networking applications is a holistic system nor equips database management systems to support scalable location-based social networking services. In this paper, we present GeoSocialDB; a holistic system providing three location-based social networking services, namely, location-based news feed, location-based news ranking, and location-based recommendation. In GeoSocialDB, we aim to implement these services as query operators inside a database engine to optimize the query processing performance. Within the GeoSocialDB framework, we discuss research challenges and directions towards the realization of scalable and practical query processing for location-based social networking services. In general, we discuss the challenges in designing location- and/or rank-aware query operators, materializing query answers, supporting continuous query processing, and providing privacy-aware query processing for our three location-based social networking services.

Categories and Subject Descriptors

H.2 [**Database Management**]: Miscellaneous

General Terms

Design, Architecture

*The work described in this paper was supported in part by a grant from City University of Hong Kong (Project No. 7200216), by the National Science Foundation under Grants IIS-0811998, IIS-0811935, CNS-0708604, IIS-0952977 (NSF CAREER), and by a Microsoft Research Gift.

ACM LBSN '10, November 2, 2010. San Jose, CA, USA

1. INTRODUCTION

Social networking applications have become one of the most important web services, e.g., Facebook [12] and Twitter [39], which provide Internet-based platforms for users to interact with other people that are socially relevant to them, e.g., their friends. With the advances in location-aware mobile devices (e.g., GPS-enabled portable devices), wireless communication technologies (e.g., 3G and Wi-Fi), map services (e.g., Google Maps [15], Microsoft Bing Maps [5], and Yahoo! Maps [42]), and (spatial) database management systems (DBMSs), location-based social networking applications have been taking shape at fast pace. Examples of such applications include Google Buzz Mobile [6], Loopt [29], and Microsoft Geo-Life [47].

Location-based social networking systems not only provide services with social relevance for users, but they also provide services with spatial relevance for the users. Potential applications of these systems include that a user wants to receive nearby geo-tagged messages submitted by his or her friends and a user wants to find new restaurants within a certain area based on his or her friends' opinions. Although location-based services have been widely studied, they heavily rely on nearest-neighbor queries, range queries, and skyline queries to get relevant information to their users based on only their users' locations, and thus, completely ignoring the social aspect in social networking services. On the other hand, existing location-based social networking applications focus on very specific services, which include sharing geo-tagged messages [7], providing recommendations for users based on their historical trajectory data [46, 45], and supporting privacy-preserving buddy search [24, 36]. To our best knowledge, none of existing location-based social networking applications is a holistic system nor equips the query processing engine inside the DBMS to support scalable location-based social networking services.

In this paper, we present GeoSocialDB; a holistic location-based social networking system, which is currently under joint development by the City University of Hong Kong and the University of Minnesota. GeoSocialDB provides the following three new location-based social networking services:

- **Location-based news feed.** In GeoSocialDB, each message submitted by a user is tagged with the user's location by the user device. When a user logs in GeoSocialDB through its web-based user interface or refreshes the user interface, the system generates a log-on query like "*Q1: Send me the messages submitted by my friends with tagged locations within d miles of my location*". Then, GeoSocialDB processes query *Q1* and

returns a set of messages that are socially and spatially relevant to the user as a query answer.

- **Location-based news ranking.** Since query *Q1* may return a large number of messages to a user, GeoSocialDB allows the user to limit the number of received messages, i.e., only the k most relevant messages are sent to the user, by sending a ranking log-on query like "*Q2: Send me the k most relevant messages submitted by my friends with tagged locations within d miles of my location*". The location-based news ranking service can rank the messages previously retrieved by the location-based news feed service based on the user's personalized ranking preferences on different domains, e.g., spatial domain, temporal domain, and user interest domain.

- **Location-based recommendation.** GeoSocialDB provides recommendations for users with respect to not only their interests and preferences (i.e., their ratings of various items or places) but also their personalized spatial and social preferences. The users can issue location-based recommendation queries like "*Q3: Recommend me the best k restaurants within d miles of my location based on my friends' opinions*".

This paper aims to raise the research challenges and provide research directions to realize GeoSocialDB. In general, within the framework of GeoSocialDB, we identify four major research challenges that need to be addressed to accomplish the realization of a scalable and practical database system for location-based social networking services. These research challenges are: (1) designing location- and/or rank-aware query operators that take into account both the spatial and social aspects, (2) investigating how to utilize materialization techniques to reduce system overhead and improve query response time, (3) supporting continuous queries that are ubiquitous in mobile environments, and (4) providing privacy-aware query processing to preserve user location privacy.

The rest of this paper is organized as follows. Section 2 presents the system architecture of GeoSocialDB. Sections 3 to 5 discuss the research challenges addressed by GeoSocialDB with pointers to research directions and solutions for location-based news feed, location-based news ranking, and location-based recommendation services, respectively. Related work is highlighted in Section 6. Finally, Section 7 concludes the paper.

2. SYSTEM ARCHITECTURE

Figure 1 gives the system architecture of GeoSocialDB, where there are three main modules for our proposed location-based social networking services, namely, *location-based news feed*, *location-based news ranking*, and *location-based recommendation*. GeoSocialDB also maintains three stored data, namely, *messages*, *user profiles*, and *suggestions*, and takes three main types of user inputs, namely, *user updates*, *log-on query*, and *recommendation query*. We present the details of the system architecture of GeoSocialDB, according to the three main types of user inputs.

User updates. The thin arrows indicate each user input's corresponding stored data. (1) User-generated and *geo-tagged messages* with plain text or multimedia data are stored in the messages stored data. When a user submits

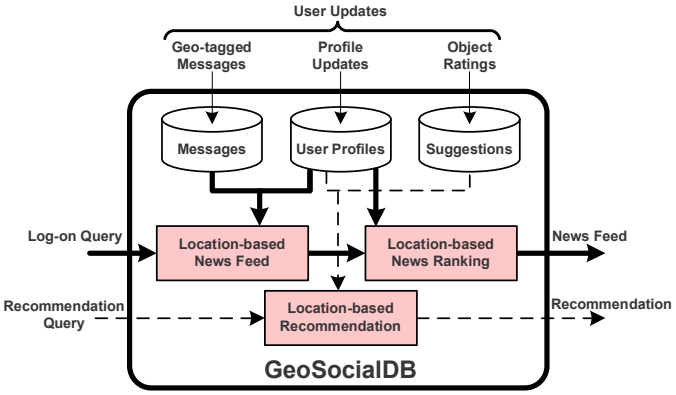
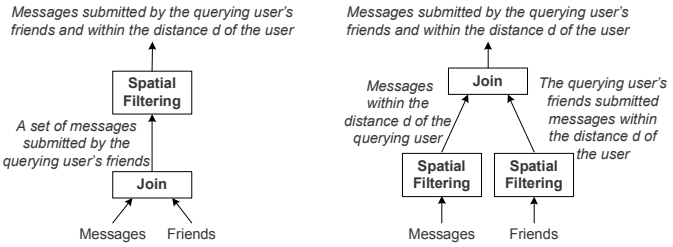


Figure 1: The system architecture of GeoSocialDB.

a message to GeoSocialDB, the message is automatically tagged with the user's location by the user device. After the message is stored in the messages stored data, it could be delivered to the user's friends when they issue log-on queries with a user specified range distance that covers the message's tagged location. (2) User profiles include user's personal information, e.g., identity and contact information, a list of the user's friends, and the user's preferences for the location-based news ranking service. The user can update his or her profile at any time, and the update is stored in the user profiles stored data. (3) Object ratings are the user opinions for objects or places, e.g., restaurants and hotels. The user can give a score in a scale from one (bad) to five (good) through the web-based user interface for any object or place stored in the database of GeoSocialDB. The object ratings are stored in the suggestions stored data.

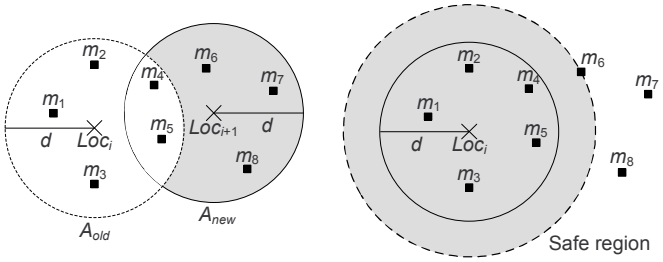
Log-on query. The thick arrows indicate the processing flow of log-on queries for the location-based news feed and location-based news ranking services. When a user logs in GeoSocialDB through its web-based user interface or refreshes the user interface, the system generates a log-on query with the user's location and a user specified range distance d to the location-based news feed module. Then, the module retrieves a set of messages that are socially and spatially relevant to the user, where each retrieved message (a) is sent by the user's friend, i.e., the social relevance, and (b) is tagged with a location within the range distance d of the user, i.e., the spatial relevance. If the user wants to limit the number of received messages to k, the set of retrieved messages is passed to the location-based news ranking module as input. The location-based news ranking module gets the user's preferences for the location-based news ranking service from the user profiles stored data and ranks the retrieved messages based on the user's ranking preferences. Finally, the k most relevant messages are returned to the user and displayed on the user interface with their tagged locations indicated by markers on the underlying map.

Recommendation query. The dotted arrows indicate the processing flow of users' recommendation queries for the location-based recommendation service. When a user requests recommendations for a specific object type, e.g., restaurants or hotels, within a range distance from the user's location through the GeoSocialDB's web-based user interface, GeoSocialDB generates a recommendation query to the location-based recommendation module. This module gets



(a) A straightforward way (b) GeoSocialDB

Figure 2: The query plans of executing query *Q1*.



(a) Incremental query processing (b) Safe region

Figure 3: Optimizations for continuous query processing.

the user's list of friends from the user profiles stored data, selects the objects located within the specified range distance of the user, and the ratings of the selected objects, which are given by the user's friends, from the suggestions stored data. Then, the location-based recommendation module predicts a score for each of the selected objects based on the retrieved object ratings to determine how it is relevant to the user, and returns the k objects with the highest predicted scores to the user. Finally, the recommended objects are displayed on the user interface with their locations indicated by markers on the underlying map.

3. LOCATION-BASED NEWS FEED

When a user logs in GeoSocialDB through its web-based user interface or refreshes the user interface, GeoSocialDB generates a log-on query *Q1* given in Section 1 to the location-based news feed module. Query *Q1* can be written in SQL as follows:

```
Q1: SELECT    M.Sender, M.Location, M.Content,
              DistFunc(M.Location, MyLocation)
              AS Distance
    FROM      Messages M, Friends F
    WHERE     F.ID=MyID AND F.Friend_ID=M.Sender
              AND Distance ≤ d
```

where *MyID* is a querying user's identity, *MyLocation* is the querying user's reported location, and the two tables *Messages* and *Friends* that are part of the messages and user profiles stored data, respectively. In the table *Friends*, each tuple stores two user identities (*ID*, *Friend_ID*) to indicate a friend relationship.

3.1 Challenge I: Location-Aware Query Operators

A straightforward way to execute query *Q1* is to join the two tables *Messages* and *Friends* to select a set of messages S submitted by a querying user's friends, followed by a spatial filtering operation that finds the messages with tagged locations within a range distance d of the querying user from S as a query answer, as illustrated in Figure 2a. However, this straightforward execution of query *Q1* is extremely inefficient. This is because the most challenging part in query *Q1* is the join operation in the `FROM` clause, as a social networking system typically stores a huge number of messages[1]. Figure 2b gives a much better solution to this challenge, where GeoSocialDB uses a spatial index (e.g., R-tree or grid structure) to index the geo-tagged messages with their sender identities in the system, and implements location-aware selection query operators inside the query processing engine of GeoSocialDB. The selection query operators are pushed below the expensive join operator to utilize the underlying spatial index to select (a) a set of messages from the table *Messages* that satisfy the spatial criterion in the `WHERE` clause and (b) a set of the querying user's friends from the table *Friends* who have sent messages satisfying the same spatial criterion. Then, the join operator only joins the two much smaller data sets (a) and (b). This solution is analogous to pushing the non-spatial selection operations below the join operation as a filtering condition in traditional database systems [8].

[1]For example, Facebook has more than 500 million active users and 30 billion pieces of content (web links, news stores, blog posts, notes, etc.), and an average user creates 90 pieces of content each month [1].

3.2 Challenge II: Answer Materialization

Since GeoSocialDB aims to serve a large number of concurrent users, it is important to minimize query response time to guarantee the quality of services. Materialization techniques are known to be used to minimize query response time, as a materialized answer is immediately ready to be returned to a user (e.g., [3, 17, 37]). However, maintaining the materialized answer of each user would increase the overall system overhead substantially, and thus, the major challenge is to select an appropriate set of users to maintain their materialized answers, in order to minimize the computational overhead of GeoSocialDB. The basic idea is that GeoSocialDB should maintain a materialized answer for a user if the cost of maintaining the materialized answer is smaller than the cost of executing query *Q1* from scratch. The solution to this challenge is to determine the query rate *QueryRate* and the query cost *QueryCost* for each user, and predict the update rate *UpdateRate* and the update cost *UpdateCost* of each user's materialized answer. GeoSocialDB should maintain a materialized answer for a user if the following inequality takes place:

$$QueryRate \times QueryCost > UpdateRate \times UpdateCost. \quad (1)$$

To use this equation, GeoSocialDB keeps statistics to estimate *QueryRate* and *UpdateRate*, while *QueryCost* and *UpdateCost* can be determined based on the access methods and query plans used by the query processor.

3.3 Challenge III: Continuous Query Processing

Since many users accessing social networking services through mobile devices[2], GeoSocialDB should have scalable continuous query processing methods to maintain the users' answers for the frequent changes of their locations. Evaluating query *Q1* from scratch for every user location update is extremely inefficient, in terms of computational and communication overhead. GeoSocial uses two common optimization techniques, *incremental query processing* and *safety region*, to improve system performance (e.g., [19, 31, 32]).

Figure 3a illustrates the basic idea of the incremental query processing optimization, where m_1 to m_8 are the messages submitted by a querying user's friends. After GeoSocialDB finds the answer of query *Q1* for a user, it stores the answer with the searched area A_{old} (represented by a dotted circle), i.e., a circular area centered at the user's reported location, Loc_i, with a radius of the user specified range distance d. When the user reports a new location, Loc_{i+1}, to GeoSocialDB, if the new search area A_{new} (represented by a circle), i.e., a circular area centered at the user's newly reported location with a radius of d, overlaps with the old searched area A_{old}, the incremental query processing optimization only requires computation over the area of A_{new} that does not overlap with A_{old}, i.e., the required search area is $A_{new} \setminus (A_{new} \cap A_{old})$ (represented by a shaded area). GeoSocialDB only computes an answer for the required search area, i.e., $\{m_6, m_7, m_8\}$, and returns it to the user. On the client side, the user simply removes the messages that are outside A_{new} from the previous answer, i.e., $\{m_1, m_2, m_3\}$, and adds the newly returned answer to the previous one to constitute the answer, i.e., $\{m_4, m_5, m_6, m_7, m_8\}$.

The idea of the safe region optimization is that after GeoSocialDB computes a query answer, it also computes a safe region such that the answer remains the same as long as subsequent required search areas are within the safe region. Figure 3b gives a straightforward way to find a safe region for a user, where after finding an answer for a searched area (represented by a circle), we select a message, i.e., m_6, which is submitted by the user's friend and outside the searched area. Then, the safe region is a circular area centered the user's location with a radius of the distance between the user and the selected message (represented by a shaded area). More sophisticated incremental query processing and safe region optimizations can be developed to further improve the continuous query processing performance of GeoSocialDB.

3.4 Challenge IV: Privacy-Aware Query Processing

One of the major challenges in location-based services is the user location privacy for both the geo-tagged messages and the location information of log-on queries. A straightforward approach to preserve the user location privacy is to reduce the location resolution, e.g., Google Latitude provides city-level privacy for users [26]. Since much more messages will satisfy the spatial criterion in the `WHERE` clause in query *Q1*, the query processor has to process much more messages and return a much larger number of messages to the user. Although this approach is simple, the computational and communication overhead of processing such low resolution location data is very high. Existing more efficient techniques for privacy-preserving location-based services can be categorized into two main classes. (1) *Spatial cloaking*. The idea of this technique is to blur a user location into a cloaked area that satisfies the user specified privacy requirements, e.g., k-anonymity and/or minimum area privacy requirements (e.g., [4, 9, 10, 13, 16, 21, 30, 41]). Since most of spatial cloaking algorithms require a fully trusted third party placed between the user and the database server to perform spatial cloaking for every location update and query, the third party may become a single point attack or a system bottleneck. (2) *Data transformation*. This class of techniques transforms the data and query locations into a transformed (or encrypted) space, and processes location-based queries in the transformed space to support nearest-neighbor queries (e.g., [14, 23, 40, 44]) and range queries (e.g., [44]). Among these techniques, only the work [44] can deal with location-based range queries. However, the social aspect, i.e., the data (messages) is submitted by users and only their friends can read the messages, has not been discussed, and thus, we need to study how to extend this work to our GeoSocialDB to provide privacy-preserving location-based news feed service.

[2]For example, there are more than 150 million out of 500 million active users accessing Facebook through their mobile devices [1].

4. LOCATION-BASED NEWS RANKING

Similar to the location-based news feed service, when a user logs in GeoSocialDB through its web-based user interface or refreshes the user interface, our system generates a log-on query to the location-based news feed module. Since the location-based news feed module may return a large number of messages to the user, the user can trigger the location-based news ranking service by specifying his or her news ranking preferences to get the k most relevant messages like query *Q2* given in Section 1. Query *Q2* can be written in SQL as follows:

```
Q2: SELECT    M.Sender, M.Location, M.Content,
              DistFunc(M.Location, MyLocation)
              AS Distance
    FROM      Messages M, Friends F
    WHERE     F.ID=MyID AND F.Friend_ID=M.Sender
              AND Distance ≤ d
    ORDER BY  RankScore(M.ID, MyProfile) DESC
    LIMIT     k
```

where the user profile *MyProfile* stores the user's preferences for news ranking, the function *RankScore* in the `ORDER BY` clause computes a ranking score for each message retrieved by the location-based news feed module based on the user profile, and the `LIMIT` clause indicates at most k messages with the highest ranking scores are returned to the user.

There are many possible ways to define the function *RankScore* to compute a ranking score for a message. We here consider a general weighting function for GeoSocialDB based on three domains, namely, spatial domain, temporal domain, and user interest domain (i.e., the user's ratings of messages). Basically, the user can specify his or her preference for each domain through the web-based user interface as a weight in a scale from 0 (not important) to 1 (very important), and the sum of the weights of all the domains is one. To combine these domains in a fair and comparable way, we normalize each value with respect to the best one in its domain to a scale from 0 (bad) to 1 (good). For example, we assume that a user specifies weights 0.6, 0.2, and 0.2

for the spatial, temporal, and user interest domains, respectively. Then, given a message with normalized values x, y, and z for the spatial, temporal, and user interest domains, respectively, the ranking score of this message computed by the function *RankScore* is $0.6x + 0.2y + 0.2z$. Although we only consider the spatial, temporal, and user interest domains, any additional domain can be added to the function *RankScore* if it can be normalized to the required scale.

4.1 Challenge I: Location- and Rank-Aware Query Operators

Similar to query *Q1* in Section 3, GeoSocialDB implements the spatial selection query operators and pushes them below the expensive join operation. The spatial selection query operators select a set of messages from the table *Messages* that satisfy the spatial criterion in the `WHERE` clause and a set of a querying user's friends from the table *Friends* that have submitted messages satisfying the spatial criterion. Then, a join operator joins these two selected data sets. However, unlike query *Q1*, query *Q2* can have another optimization by pushing the ranking operation inside the expensive join operation (e.g., [20, 27]). The basic idea of this optimization is to sort the selected messages in a particular order based on the most important user preference (i.e., the preference has the largest weight), and then join them to the selected friends one by one in the same order. After the rank-aware join operator finds a message that is submitted by the querying user's friend, the function *RankScore* computes a ranking score for the message. Sorting the messages in the particular order enables the rank-aware join operator to determine the largest possible ranking score for the next unprocessed message. After the rank-aware query operator processed at least k messages, it can find a potential answer. If the largest possible ranking score of the next unprocessed message is less than or equal to the smallest ranking score of the messages in the potential answer, the rank-aware join operator terminates the query processing and returns the potential answer to the user. The reason is that the actual ranking scores of all the remaining unprocessed messages cannot be better than that of the messages in the potential answer. This early termination ability of the rank-aware join operator can reduce computational overhead, as it needs not process all the selected messages to process query *Q2*.

4.2 Challenge II: Answer Materialization

Since the location-based news ranking service is an add-on function to the location-based news feed service, the basic idea of its solution to the answer materialization challenge is the same as in the location-based news feed service (Section 3.2). However, GeoSocialDB needs to have a more sophisticated way to estimate the query execution cost *QueryCost* and the update cost of a materialized answer *UpdateCost* of query *Q2* based on the underlying access methods, execution query plans, and the internal operations of the developed rank-aware join operator to select an appropriate set of users to materialize their query answers.

4.3 Challenge III: Continuous Query Processing

Unlike the location-based news feed query, the location-based news ranking query is *order-sensitive*, i.e., only top-k messages satisfying the spatial criterion in the `WHERE` clause in query *Q2* are returned to a querying user. The optimizations of continuous query processing for the location-based news feed service, *incremental query processing* and *safe region*, have to be modified to deal with the order-sensitivity requirement.

The modification of the incremental query processing optimization is that GeoSocialDB stores the previous answer with the searched area A_{old} of query *Q2* for a user. Then, when the user reports a new location to GeoSocialDB, if the new search area of A_{new}, i.e., a circular area centered at the user's newly reported location with a radius of distance d, overlaps with the previously searched area A_{old}, GeoSocialDB only needs to compute a new answer for the area A_{new} that does not overlap with A_{old}, i.e., the area in $A_{new} \setminus (A_{new} \cap A_{old})$. To find a result answer for the newly reported location, GeoSocialDB ranks the messages in the previous and new answers based on their ranking scores, and selects the k messages with the highest ranking scores. Finally, GeoSocialDB sends the new messages, which are not included in the previous answer, and an eviction list, which includes the messages in the previous answer but not in the new answer, to the user. Finally, the user updates his or her cached answer accordingly.

To deal with the order-sensitivity requirement of the location-based news ranking queries, the modified safe region optimization has to select a message that is outside the searched area of a query answer, is the nearest one to the user, and has a higher ranking score than the messages included in the query answer. Thus, the safe region is a circular area centered at the user's location with a radius of the distance between the user and the selected message. The user needs not send location updates to GeoSocialDB as long as the subsequent required search areas are within the safe region. However, in case that there is a new message submitted by a querying user's friend within the safe region, if the ranking score of the new message is larger than the smallest ranking score of the messages in the querying user's previous query answer, GeoSocialDB notifies the querying user to report his or her current location and re-evaluates the location-based news ranking query if necessary. In addition, it is worthy to note that the design of the safe region optimization will be more complex if the ranking function *RankScore* considers the distance between a message and a querying user. The reason is that the movement of the querying user affects the ranking scores of the messages within the previously searched area and/or the safe region.

4.4 Challenge IV: Privacy-Aware Query Processing

Since the location-based news ranking service is the add-on function to the location-based news feed service, the discussion of the location privacy challenge of the location-based news ranking service is the same as in the location-based news feed service (Section 3.4). The only difference here is that the privacy-preserving data transformation scheme [44] cannot give the actual distance between the encrypted location information of a message and a querying user for the function *RankScore*, and thus, we need to study how to make the spatial domain applicable to the case of privacy-preserving location-based news ranking service.

5. LOCATION-BASED RECOMMENDATION

In GeoSocialDB, a user is able to issue a location-based recommendation query *Q3* presented in Section 1 to request

recommendations for a specific type of objects of interest, e.g., restaurants. Query *Q3* can be written in SQL as follows:

```
Q3: SELECT     R.ID, R.Name, R.Address,
               DistFunc(R.location, MyLocation)
               AS Distance
    FROM       Restaurants R
    WHERE      Distance ≤ d
    ORDER BY   RecScore(R.ID, MyProfile) DESC
    LIMIT      k
```

where the user profile *MyProfile* stores the user's ratings of objects, the function *RecScore* in the `ORDER BY` clause computes a recommendation score for each restaurant that is located within a distance d of a querying user based on his or her friends' opinions, and the `LIMIT` clause indicates at most k restaurants with the highest recommendation scores are returned to the user.

There are many feasible recommendation techniques to implement the function *RecScore* [2]. Among these techniques, we consider *collaborative filtering*, that is, by far, the most popular method used in recommender systems. In particular, GeoSocialDB employs the item-based collaborative filtering algorithm to implement the function *RecScore* (e.g., [11, 18, 25, 34, 35]). In general, the item-based collaborative filtering algorithm has three main steps. (1) A similarity step, performed *offline*, pre-computes the item-to-item similarity for each pair of items (objects). (2) A prediction step that is an *online* operation to determine a recommendation score for each object that has not been rated by a querying user based on the pre-computed item-to-item similarities and the querying user's object ratings. (3) Finally, a selection step selects the top-k objects in terms of the recommendation scores computed in the prediction step and recommends them to the querying user. Although the item-based collaborative filtering technique has been widely studied, it has not been incorporated into neither location-based services nor location-based social networking services, and thus, it is important to discuss the research challenges and directions for the location-based recommendation service.

5.1 Challenge I: Location-Aware Query Operators

The most challenging part in query *Q3* is the use of the recommendation score function *RecScore* in the `ORDER BY` clause. Basically, if this was not a function, but just an attribute within the restaurant table *Restaurants*, the whole query *Q3* would be easily evaluated as any nearest-neighbor query that is based on some ranking function. However, the challenging part here is that the recommendation score cannot be an attribute. Instead, the recommendation score has to be computed for each user profile. For example, the recommendation score of a restaurant x for user Alice is different from the recommendation score of the same restaurant x to user Bob. This is mainly because Alice and Bob have different profiles, i.e., they have different preferences and tastes. Similar to queries *Q1* and *Q2* discussed in Sections 3 and 4, respectively, it is extremely inefficient to compute the recommendation scores for all the restaurants in the system to process query *Q3*. Thus, GeoSocialDB implements a location-based selection operator and pushes it below the ranking evaluation to select a set of restaurants within a distance d of a querying user first, and then use the function *RecScore* to compute a recommendation score for each selected restaurant. Finally, the top-k restaurants with the highest recommendation scores are recommended to the querying user.

5.2 Challenge II: Answer Materialization

The user of GeoSocialDB can not only issue a location-based recommendation query explicitly to the system through the web-based user interface, but the user can also add the location-based recommendation service for a specific type of objects of interest as an application that shows the recommended objects on the user profile page. The materialized answer of a location-based recommender query of a user is updated when the user rates a new object or there are some relevant changes in the object-to-object similarities. Since the object-to-object similarities in the item-based collaborative filtering are only updated periodically, e.g., a day or a week, the update rate of the materialized answer is much smaller than the location-based news feed service. According to Equation 1 presented in Section 3.2, GeoSocialDB should maintain a materialized answer for each location-based recommendation query, which is registered as an application. Thus, whenever the user views his or her profile, the personalized recommendation is immediately available to be displayed on the user profile page.

5.3 Challenge III: Continuous Query Processing

In GeoSocialDB, the user can issue the location-based recommendation query like query *Q3* through his or her mobile device. When driving a vehicle, the user can issue a continuous location-based recommender query to get answers with respect to the user's current location. Similar to the location-based news ranking service, the location-based recommender query is *order-sensitive*. We will discuss how to use the two continuous query processing optimizations, *incremental query processing* and *safe region*, to improve the performance of the continuous location-based recommendation service.

For the incremental query processing optimization, GeoSocialDB stores the previous answer with the searched area A_{old} of query *Q3* for a querying user, i.e., a circular area centered at the user's reported location *MyLocation* with a radius of distance d. When the user reports a new location to GeoSocialDB, if the new search area A_{new} overlaps with the previously searched area A_{old}, GeoSocialDB only needs to compute the recommendation scores for the restaurants located in the area of A_{new} that does not overlap with A_{old}, i.e., the area in $A_{new} \setminus (A_{new} \cap A_{old})$. To find a result answer for the newly reported location, GeoSocialDB ranks the restaurants in the previous and new answers based on their recommendation scores, and selects the k restaurants with the highest recommendation scores. GeoSocialDB sends the new restaurants, which are not in the previous answer, and an eviction list, which includes the restaurant in the previous answer but not in the new answer, to the user. Finally, the user updates his or her cached answer accordingly.

As the location-based recommender query is order-sensitive, the safe region optimization selects a restaurant that is nearest to a querying user, outside the searched area of a query answer, and has a higher recommendation score than the restaurants in the query answer. Hence, the safe region is a circular area centered at the user's reported location with a radius of the distance between the user and the selected restaurant. The returned query answer remains

the same as long as the required search area of a new user location is within the safe region. It is worthy to note that if GeoSocialDB materializes query answers for continuous location-based recommender queries, it needs a more sophisticated method to model the query cost *QueryCost* and the update cost *UpdateCost* in Equation 1.

5.4 Challenge IV: Privacy-Aware Query Processing

The location-based recommendation service is different from the location-based news feed service that the data for the location-based recommendation service, e.g., restaurants, are assumed to be provided by third parties (e.g., the information and ratings of restaurants from Yelp.com [43]), while the geo-tagged messages are submitted by users. For the location-based recommendation service, the data provider can use the privacy-preserving data transformation scheme [44] to provide the transformed locations and information of objects for GeoSocialDB and give the required information for query processing and data decryption for the user to subscribe the privacy-preserving location-based recommendation service.

6. RELATED WORK

In this section, we first highlight the related work regarding the location-based news feed and location-based news ranking services in GeoSocialDB, and then discuss the related work for the location-based recommendation service in GeoSocialDB.

Social networking systems. Existing location-based social networking systems focus on very specific services, which include sharing geo-tagged messages [7] and supporting privacy-preserving buddy search [24, 36]. The work [7] allows users to submit geo-tagged messages to the system and enables the users to get the geo-tagged messages within their proximity, where the proximity is determined by the system based on the capacity of their mobile devices. The privacy-preserving buddy search systems allow users to find their friends within a certain area without revealing their locations to the system.

GeoSocialDB provides the first attempt to equip database management systems to support scalable location-based news feed and location-based news ranking services by overcoming the research challenges in (1) designing location- and/or rank-aware query operators inside the database engine to optimize query processing performance, (2) materializing query answers to reduce system overhead and query response time, (3) designing query optimizations for continuous query processing, and (4) providing privacy-aware query processing to preserve the user location privacy.

Recommender systems. Collaborative filtering is, by far, the most popular method used in recommender systems. In general, algorithms for collaborative filtering recommender systems can be categorized into two classes: (1) *User-based collaborative filtering* (e.g.,[18, 25, 34]), where the idea is to predict a recommendation score for each item that has not been rated by a querying user, based on the opinions given by other *similar* users. Then, the top-k items with the highest recommendation scores are returned to the user. (2) *Model-based collaborative filtering* (e.g., [11, 22, 18, 25, 28, 34, 35]), where the idea is to use an offline-built model to predict recommendation scores for those items not rated by the querying user. The most popular method of model-based collaborative filtering is the item-based collaborative filtering method where the offline model is built based on item-to-item similarities. Unfortunately, none of these collaborative filtering methods have been used to support database queries for spatial objects.

Recommender systems have been extended to be location-aware. However, existing location-aware recommenders either rely on mining users' location and activity histories to recommend new places or activities [38, 47, 46, 45], or utilize the user profile that includes age, income, and preferred cuisine to recommend new restaurants [33]. Our location-based recommendation service in GeoSocialDB distinguishes itself from these works as it is the first attempt to (1) incorporate the collaborative filtering recommender systems into location-based social networking services, in order to use the community opinions to enhance the quality of answers of location-based recommendation queries, (2) design recommendation query operators inside the database engine to make the recommender systems aware of the spatial functions to improve query processing efficiency, (3) propose optimizations for continuous query processing of location-based recommendation queries for mobile users, and (4) provide privacy-aware query processing of the location-based recommendation queries.

7. CONCLUSION

In this paper, we introduced the system architecture of GeoSocialDB; a holistic location-based social networking database system, currently under joint development at City University of Hong Kong and University of Minnesota, that delivers location-based news feed, location-based news ranking, and location-based recommendation services to its users based on their personalized spatial and social preferences. Within the framework of GeoSocialDB, we have discussed four main research challenges that are needed to be addressed by the research community in order to have a practical and scalable realization of location-based social networking services. The four research challenges are: (1) Designing location-based and/or rank-aware query operators inside the database engine to optimize query processing performance, (2) Utilizing materialization techniques to reduce computational overhead and query response time, (3) Enabling efficient and scalable execution of location-based social networking queries that takes into consideration of the frequent location updates of mobile users, and (4) Providing privacy-aware query processing for location-based social networking queries to preserve the user location privacy.

8. REFERENCES

[1] Facebook Statistics. `http://www.facebook.com/press/info.php?statistics`.

[2] G. Adomavicius and A. Tuzhilin. Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE TKDE*, 17(6):734–749, 2005.

[3] S. Agrawal, S. Chaudhuri, and V. R. Narasayya. Automated Selection of Materialized Views and Indexes in SQL Databases. In *Proc. of VLDB*, 2000.

[4] B. Bamba, L. Liu, P. Pesti, and T. Wang. Supporting Anonymous Location Queries in Mobile Environments with PrivacyGrid. In *Proc. of WWW*, 2008.

[5] Microsoft Bing Maps. `http://www.bing.com/maps`.

[6] Google Buzz. `http://www.google.com/buzz`.

[7] Y. Cai and T. Xu. Design, Analysis, and Implementation of a Large-Scale Real-Time Location-Based Information Sharing System. In *Proc. of ACM MobiSys*, 2008.

[8] J. Chen, D. J. DeWitt, F. Tian, and Y. Wang. NiagaraCQ: A Scalable Continuous Query System for Internet Databases. In *Proc. of ACM SIGMOD*, 2000.

[9] C.-Y. Chow and M. F. Mokbel. Enabling Private Continuous Queries For Revealed User Locations. In *Proc. of SSTD*, 2007.

[10] C.-Y. Chow, M. F. Mokbel, and W. G. Aref. Casper*: Query Processing for Location Services Without Compromising Privacy. *ACM TODS*, 34(4):1–48, 2009.

[11] M. Deshpande and G. Karypis. Item-Based Top-N Recommendation Algorithms. *ACM TOIS*, 22(1):143–177, 2004.

[12] Facebook. `http://www.facebook.com`.

[13] B. Gedik and L. Liu. Protecting Location Privacy with Personalized k-Anonymity: Architecture and Algorithms. *IEEE TMC*, 7(1):1–18, 2008.

[14] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan. Private Queries in Location Based Services: Anonymizers are not Necessary. In *Proc. of ACM SIGMOD*, 2008.

[15] Google Maps. `http://maps.google.com`.

[16] M. Gruteser and D. Grunwald. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In *Proc. of ACM MobiSys*, 2003.

[17] A. Y. Halevy. Answering Queries Using Views: A Survey. *VLDB Journal*, 10(4):270–294, 2001.

[18] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An Algorithmic Framework for Performing Collaborative Filtering. In *Proc. of ACM SIGIR*, 1999.

[19] H. Hu, J. Xu, and D. L. Lee. A Generic Framework for Monitoring Continuous Spatial Queries over Moving Objects. In *Proc. of ACM SIGMOD*, 2005.

[20] I. F. Ilyas, R. Shah, W. G. Aref, J. S. Vitter, and A. K. Elmagarmid. Rank-Aware Query Optimization. In *Proc. of ACM SIGMOD*, 2004.

[21] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias. Preventing Location-Based Identity Inference in Anonymous Spatial Queries. *IEEE TKDE*, 19(12):1719–1733, 2007.

[22] G. Karypis. Evaluation of Item-Based Top-N Recommendation Algorithms. In *Proc. of ACM CIKM*, 2001.

[23] A. Khoshgozaran and C. Shahabi. Blind Evaluation of Nearest Neighbor Queries Using Space Transformation to Preserve Location Privacy. In *Proc. of SSTD*, 2007.

[24] A. Khoshgozaran and C. Shahabi. Private Buddy Search: Enabling Private Spatial Queries in Social Networks. In *Proc. of SIN*, 2009.

[25] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. GroupLens: Applying Collaborative Filtering to Usenet News. *Communications of the ACM*, 40(3):77–87, 1997.

[26] Google Latitude. `http://www.google.com/mobile/latitude`.

[27] C. Li, K. C.-C. Chang, I. F. Ilyas, and S. Song. RankSQL: Query Algebra and Optimization for Relational Top-k Queries. In *Proc. of ACM SIGMOD*, 2005.

[28] G. Linden, B. Smith, and J. York. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[29] Loopt. `http://www.loopt.com`.

[30] M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The New Casper: Query Processing for Location Services without Compromising Privacy. In *Proc. of VLDB*, 2006.

[31] M. F. Mokbel, X. Xiong, and W. G. Aref. SINA: Scalable Incremental Processing of Continuous Queries in Spatio-temporal Databases. In *Proc. of ACM SIGMOD*, 2004.

[32] K. Mouratidis, D. Papadias, and M. Hadjieleftheriou. Conceptual Partitioning: An Efficient Method for Continuous Nearest Neighbor Monitoring. In *Proc. of ACM SIGMOD*, 2005.

[33] M.-H. Park, J.-H. Hong, and S.-B. Cho. Location-Based Recommendation System Using Bayesian User's Preference Model in Mobile Devices. In *Proc. of UIC*, 2007.

[34] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proc. of CSCW*, 1994.

[35] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-Based Collaborative Filtering Recommendation Algorithms. In *Proc. of WWW*, 2001.

[36] L. Siksnys, J. Thomsen, S. Saltenis, and M. L. Yiu. Private and Flexible Proximity Detection In Mobile Social Networks. In *Proc. of MDM*, 2010.

[37] A. Silberstein, J. Terrace, B. F. Cooper, and R. Ramakrishnan. Feeding Frenzy: Selectively Materializing Users' Event Feeds. In *Proc. of ACM SIGMOD*, 2010.

[38] Y. Takeuchi and M. Sugimoto. CityVoyager: An Outdoor Recommendation System Based on User Location History. In *Proc. of UIC*, 2006.

[39] Twitter. `http://www.twitter.com`.

[40] W. K. Wong, D. W.-L. Cheung, B. Kao, and N. Mamoulis. Secure kNN Computation on Encrypted Databases. In *Proc. of ACM SIGMOD*, 2009.

[41] T. Xu and Y. Cai. Exploring Historical Location Data for Anonymity Preservation in Location-based Services. In *Proc. of IEEE INFOCOM*, 2008.

[42] Yahoo! Maps. `http://maps.yahoo.com`.

[43] Yelp, Inc. `http://www.yelp.com`.

[44] M. L. Yiu, G. Ghinita, C. S. Jensen, and P. Kalnis. Enabling Search Services on Outsourced Private Spatial Data. *VLDB Journal*, 19(3):363–384, 2010.

[45] V. W. Zheng, Y. Zheng, X. Xie, and Q. Yang. Collaborative Location and Activity Recommendations with GPS History Data. In *Proc. of WWW*, 2010.

[46] Y. Zheng and X. Xie. Learning Travel Recommendation from User-Generated GPS Trajectories. *ACM TIST*, To appear.

[47] Y. Zheng, X. Xie, and W.-Y. Ma. GeoLife: A Collaborative Social Networking Service among User, Location and Trajectory. *IEEE Data Eng. Bull.*, 33(2):32–40, 2010.