Personalization, Socialization, and Recommendations in Location-based Services 2.0

Mohamed F. Mokbel Jie Bao Ahmed Eldawy
Justin J. Levandoski Mohamed Sarwat

Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, USA
{mokbel,baojie,eldawy,justin,sarwat}@cs.umn.edu

ABSTRACT

The wide increase of web-based user-generated content and social networking technologies have led to the wide popularity of the term Web 2.0, in which the World Wide Web has moved from being an interface for information retrieval to an interactive medium. Following Web 2.0, a flurry of 2.0s have appeared including Library 2.0, Travel 2.0, Government 2.0, and even Revolution 2.0. In this paper, we present our vision of Location-based Services (LBS) 2.0, where users generate significant location-based content, and have meaningful location-aware interactions with both the system and other users. We address three main aspects of LBS 2.0, namely, personalization, socialization, and recommendations. In terms of personalization, LBS 2.0 looks beyond the use of rigid nearest-neighbor queries to more personalized best-neighbor queries where user preferences and context are taken into account when answering queries. In terms of socialization, LBS 2.0 goes beyond using location information as extra attributes in existing social networks to injecting location-awareness into the core functionality of social networks. For recommendations, LBS 2.0 aims to adapt existing commercially successful recommendation techniques to consider the spatial aspects of users and/or items when making its decisions. Overall, we discuss how the confluence of these topics form our vision of LBS 2.0.

1. INTRODUCTION

The term Web 2.0 is associated with web applications that facilitate participatory information sharing, crowd-sourcing, interoperability, user-centered design, and collaboration on the World Wide Web [28]. The popularity of Web 2.0 came about as a direct result of the wide increase of web-based user-generated content and social networking technologies. In Web 2.0, the World Wide Web has moved from being an interface for information retrieval to an interactive medium where users can share information, upload user-generated content, and interact with other users. Following the success of Web 2.0, a flurry of 2.0s have appeared including Library 2.0 [16], Travel 2.0 [29], Government 2.0 [9], and even Revolution 2.0 [24]. All application of 2.0s rely mainly on social interaction among participants, where the knowledge of one person helps others in an information participatory media.

In this paper, we present our vision of Location-based Services (LBS) 2.0, where users can generate significant location-based content and have meaningful location-aware interaction with both the system and other users. There are two ways to look at LBS 2.0, either as embedding location-awareness into existing Web 2.0 infrastructures, or embedding Web 2.0 functionality inside the core of existing location-based services. We take the former approach, which makes use of the existing Web 2.0 model, thus building upon an already-successful infrastructure. This approach is in contrast to reinventing Web 2.0 modules inside a location-based service environment. In other words, our approach is similar to the story of spatial databases over the last two decades, where the spatial functionalities were embedded inside existing database systems, making use of the existing infrastructure including query operators, optimizers, indexing, and transaction processing.

We address three main aspects of LBS 2.0, namely, personalization, socialization, and recommendations, which have been successfully realized in the world of Web 2.0. Below is a brief description of these aspects:

- **Personalization.** Nearest-neighbor queries are widely used in location-based services to locate the closest restaurant, gas station, or store. However, in the world of Web 2.0, and the proliferation of social networks, using rigid nearest-neighbor queries may not be favorable. LBS 2.0 needs to go beyond the use of nearest-neighbor queries to the more personalized best-neighbor queries where user preferences and context are taken into account evaluating location-based queries.
- **Socialization.** Location information are used in existing social networks as simple additional attributes used within a "Check-in" procedure. LBS 2.0 needs to go beyond this idea to injecting location-awareness into (a) the core functionality of social networks, and (b) the decision of what is the most related social news items given to each user.
- **Recommendation.** Recommender systems have made very successful commercial applications. However, to a large extent, they ignore the location properties of both their users and recommended items. LBS 2.0 needs to adapt existing commercially successful recommendation techniques to consider the spatial aspects when making its decisions.

This paper also presents GeoSocialDB, a location-based social networking system we are building to demonstrate our vision of LBS 2.0. The functionality of GeoSocialDB is fundamentally different from current incarnations of location-based social networks (e.g., Facebook Places [6], Foursquare [8]). These existing

PersDB 2011 Workshop, September 2, 2011, Seattle, Washington, USA

location-based networks are strictly built for mobile devices, and only allow users to receive messages about the whereabouts of their friends (e.g., Foursquare "check-ins" that give an alert that "your friend Alice has checked in at restaurant A"). GeoSocialDB, on the other hand, takes a broader approach that dovetails functionality of traditional social-networks with location-based social scenarios (e.g., friend news posts with spatial extents, location-influenced recommendations). Thus, GeoSocialDB is appropriate for both traditional social networking scenarios (e.g., desktop-based applications) as well as location-based scenarios (e.g., mobile-based applications).

The term LBS 2.0 has recently been used to refer to the new location-based functionalities that take place in the context of social networks [14]. Examples of these functionalities include Facebook places [6] and Foursquare [8] that are mainly used through mobile devices. Users of Facebook places and Foursquare get messages about the whereabouts of their friends in terms of *Check-in* messages. For example, "your friend Alice has checked in at restaurant A". In addition, users can receive notifications about friends who have checked in at a nearby place. In such systems, users' locations are treated simply as an additional attribute within the user profile. Yet, the location attribute is not really used in the core functionality of social networks. In an analogy to the spatial database era, this is similar to using the location attribute as an an additional table column queried through SQL. Ultimately, however, the treatment of location as a first-class citizen in databases led to the creation of specific spatial database operations (e.g., spatial joins) that in turn led to orders of magnitude performance improvements and the success of commercial spatial databases. This matches our vision for LBS 2.0, where location information will not only be pervasive to the users, but will also be pervasive to the underlying computational model, i.e., the location is used in the core functionality of social networks. This also relaxes the restriction that LBS 2.0 should only be applicable for mobile applications that generate "check-in" information from the GPS attached to the mobile device. In fact, LBS 2.0 should be also valid with traditional desktop computers, where a user of LBS 2.0 has full control on its location of interest.

The rest of this paper is organized as follows. Sections 2, 3, and 4 discuss the personalization, socialization, and recommendation aspects of LBS 2.0, respectively. Section 5 presents the GeoSocialDB project, under development at University of Minnesota that targets our vision of LBS 2.0. Finally, Section 6 concludes the paper.

2. PERSONALIZATION IN LBS 2.0

In this section, we discuss our vision for personalization in LBS 2.0. We start by introducing traditional preference queries and K-nearest-neighbor queries. Then, we present our proposal of K-best-neighbor queries that we envision as a marriage of these two existing query types. Finally, we present the challenges and approaches that need to be considered when designing K-best-neighbor queries.

2.1 Preference Queries

The idea of personalization is to allow users to express their personal preferences, which will be taken into account whenever a user requests a certain item (e.g., hotel or movie). The heart of personalization is preference query processing that aims to find the best answer according to a certain preference method. In recent years, a number of preference methods have been proposed, including top-k [3], skylines [1] k-dominance [2], and top-k dominance [32], among others. Each of these methods have semantics that challenge the notion of "best" answers. Since "best" is subjective, we will likely see the proposal of a number of new preference methods in the future.

Unfortunately, the *spatial* attributes in personalization are dealt with as any other non-spatial attribute. For example, in a typical *skyline* query that finds best hotels based on price and distance, the spatial distance attribute is used in the same way as the non-spatial price attribute. Very few works have exploited the special properties of spatial attributes, e.g., continuous movement [11, 17] and road network distance [4, 13]. However, such work is limited only to the case of *skyline* preference queries, and cannot be extended to support the wide variety of other preference queries.

2.2 K-Nearest-Neighbor Queries

A primary use of location-based services has been to help users find interesting nearby destinations (e.g., restaurant, stores) through k-nearest-neighbor (kNN) queries [10, 18, 25, 31]. Though widely used, kNN queries are rigid as they provide answers based solely on distance-based measures, e.g., Euclidean or road network distance. For example, when a user looks for a restaurant, the user actually wants to find the "best" restaurant, which can be interpreted differently from one user to another. Existing kNN queries reduce the meaning of "best" to be the "closest". The rigidness of kNN queries can be shown with a simple example where a user asks for four restaurants. After retrieving the answer, the user discovers that she does not like the style or the food of the first two restaurants, while the other two restaurants do not match her dietary and budget restrictions. Location-based services should be useful, and a more useful set of answers could have been given in the previous example had we considered the user's *personal preferences*.

2.3 LBS 2.0: K-Best-Neighbor Queries

In LBS 2.0, location-awareness should be inherent in the computation of *all* preference queries as a means of personalization. LBS 2.0 needs to go beyond the rigid nearest-neighbor queries to the more personalized best-neighbor queries where the user preference and context are taken into account when deciding on the query answer. We envision best-neighbor queries as a marriage of both preference queries and nearest-neighbor queries. The term "best" is inherited from preference queries that aim to find the best answer subject to a certain preference function and user-provided constraints, while the term "neighbor" refers to the spatial nature of the results, which require special handling when combined with other non-spatial attributes. In general, *k-best-neighbor* queries aim to find the k best objects of interest according to a set of user preference constraints, a preference function, and surrounding context, while considering the location aspects of both the user and objects.

It is important to note here that we are not aiming to propose or design new preference methods. Instead, we aim to make *existing* preference methods location-aware. There is already a mature research field aimed at inventing new preference functions, either within the database research community, HCI community, or even among economists. Starting from there, we do not need to reinvent the wheel and think of new preference functions. Instead, we plan to work with existing methods to make them location-aware.

2.3.1 Location-Aware Preference Queries

One approach to inject the location-awareness inside existing preference functions is to redesign the algorithms of *each* preference query type (e.g., top-k, skyline, k-dominance) to include location-awareness. This is the approach applied so far for *skyline* queries [4, 11, 13, 17]. However, with the plethora of new preference functions and the likeliness to see new preference functions in the future, this approach is not practical. It requires a huge effort applying spatial properties to *each* single preference method.

Our approach is to approach the problem from another angle. *Instead of augmenting each preference method with location-awareness, we plan to make existing spatial operations preference-aware*. We believe that this approach is more practical as the number of *basic* spatial operations is much less than the number of current, or future, preference methods. One way to realize our approach is to utilize extensible architectures like the FlexPref [15] extensible preference evaluation framework. In FlexPref, traditional query operators (e.g., selection and join) are extensible to *any* registered preference method. Registering a preference method in FlexPref is as easy as designing a simple procedural interface of a few short functions. Along the same line, we envision redesigning basic spatial operations (e.g., spatial selection and spatial join) to be extensible to a number of preference methods. This means that the code for a spatial selection/join operator will be written in terms of certain extensible functions that can be defined for a number of preference methods. As with FlexPref, the promise of our approach is that the effort required to design these extensible functions for each preference method is orders of magnitude less than the effort to make each preference function location-aware.

2.3.2 Exploiting the Properties of Spatial Attributes

Preference functions need to be aware of the special properties of spatial attributes. For example, spatial attributes are hard to compute compared to other attributes. For example, unlikely that a preference function (e.g., a skyline) optimizing over *price* and *distance* will deal with the the two attributes in the same way. A *price* attribute is easy to retrieve, while a *distance* attribute may need to be computed on-the-fly considering some constraints, e.g., road network and traffic. One way to process such a query would be to first compute the *distance* attribute for all objects of interest, then apply the preference function. Needless to say, that would be an inefficient way of computing preference queries. Preference functions should be aware of the cost of computing spatial attributes, and avoid unnecessary spatially related computations for those objects that will never contribute to the answer of the preference query.

A major advantage in our proposed approach mentioned in the previous section is its extensibility in supporting a variety of functions over all supported preference methods. For example, instead of proposing one algorithm for the spatial skyline query that exploits expensive computations of spatial attributes, and another one for spatial k-dominant queries, spatial top-k queries, and so forth, we need to introduce only one algorithm for handling spatial attributes for *all* preference queries.

3. SOCIALIZATION IN LBS 2.0

In terms of socialization, LBS 2.0 needs to go beyond using the location information as extra attributes in existing social networks to injecting the location-awareness into the core functionality of social networks.

3.1 News Feed Functionality

The news feed functionality [27] is, by far, the most common functionality in all social networking systems (e.g., Facebook [5] and Twitter [30]) and news aggregators (e.g., My Yahoo! [20] and iGoogle [12]). The main idea of the news feed functionality is that users of social networks and news aggregators receive a set of news from their friends and favorite news sources, respectively. As the number of related news for any user could be overwhelming, existing news feeds limit their output to only a set of k news. This set is either selected as the most recent k news items or some weighting criteria is applied to get only the most (expected to be) interesting items to the users.

3.2 Location-Aware News Feed

This section presents the motivation and operation of the location-aware news feed functionality, needed to realize socialization in LBS 2.0.

3.2.1 Motivation

Although the news feed functionality is widely available in all social network and news aggregator systems, unfortunately, it ignores the spatial aspect of posted messages, hence, users may miss several important messages that are spatially related to them. For example, when a traveling user logs on to a social network site, the user would like to get the news feed that match his/her new location, rather than sticking to the most recent news feed. The same concept can also be applied for users who keep logging on to the system from the same location, yet, they have a large number of friends. It is of essence for such users to limit their news feed to the ones related to their locations. Examples of the location-aware news feed include a message about local news, a comment about a local store, or a status message targeting friends in a certain area.

3.2.2 Operation

In order to make the news feed functionality location-aware, each user-posted message is associated with a spatial extent that determines a message's range of effectiveness. Meanwhile, each user is associated with an inherent location. When retrieving the location-aware news feed, the user location and message's spatial extents are used to retrieve the spatially relevant news feed.

The main idea of the *location-aware news feed* is to abstract the location-aware news feed problem to evaluating a set of location-based point queries where each query is posed to a friend to retrieve the set of messages that are issued by that friend and overlap with the user location. The *location-aware news feed* is equipped with three different approaches for evaluating each location-based query, namely, (1) *spatial pull approach*, in which the query is answered through exploiting a spatial index over the messages posted by the friend, (2) *spatial push approach*, in which the query simply retrieves the answer from a pre-computed materialized view maintained by the friend, and (3) *shared push approach*, in which the pre-computation and materialized view maintenance are shared among multiple users. Then, the main challenge of the *location-aware news feed* module is to decide when to use each of these three approaches to which queries. Our system, GeoSocialDB, implements an elegant decision model that decides upon using an approach such that it: (a) minimizes the system overhead for delivering the news feed, and (b) guarantees a certain response time for each user to obtain the requested location-aware news feed.

A better response time will call for using the *spatial push* approach for all queries, where all location-aware news feeds are pre-computed. However, this results in a huge system overhead to maintain a massive number of materialized views. On the other hand, favoring system overhead may result in executing more queries using the *spatial pull* approach as no views need to be maintained. However, this may be over killing for users who have a large number of friends as they will suffer a long delay when retrieving their news feed. GeoSocialDB takes these factors into account when deciding on which approach to use to evaluate each query in a way that minimizes the system overhead and guarantees a certain user response time.

3.3 Location-Aware Ranking

This section presents the motivation and operation of the location-aware ranking functionality, needed to realize socialization in LBS 2.0.

3.3.1 Motivation

Social network and news aggregator users may have different preferences over the received messages in the news feed. For example, a traveling user may be more interested in the messages whose issuing locations are close to his/her current location. On the other hand, a stationary user may be more interested in the messages whose issuing time is more recent to the log-on query issuing time. With the large volume of messages generated in a social networking system (e.g., more than 30 billion pieces of content shared per month in Facebook [7]) and the user's limited viewing capability (e.g., 40 messages for the web page and 20 messages for an iPhone app), it is of essence to rank the related news feed, and only show the most related k news items.

3.3.2 Operation

One way to do the location-aware ranking is to first retrieve all the related news feed through a location-aware news feed system as described in Section 3.2, then, apply a certain ranking function that combines both the spatial and temporal aspects of each message, and only show the messages with top-k ranks. However, doing so may result in significant redundant computations of retrieving a lot of news feed messages that have no chance of making it to the top-k list. Our idea is that instead of ranking all messages after being retrieved from the user's friends, we encapsulate the user's ranking preferences within the query processing. The main idea is to reduce the number of friends the system queries for the relevant messages.

We can do so by tracking the highest ranked message scores from all the user friends to select a candidate set of friends to retrieve the relevant messages from. This immediately saves from the need of retrieving all related messages from all friends, as we limit the message retrieval from a candidate set of friends. Then, we query each candidate friend by their highest ranked message scores and keeps updating the ranking boundary for top-k candidate messages. The query processing terminates early once the highest ranking score of the next querying friend is less then the ranking boundary to save a significant amount of redundant computations. Moreover, these highest ranked message scores are updated as the messages are updated from the user's friends, when necessary.

4. RECOMMENDATION IN LBS 2.0

In this section, we present our vision for *location-based recommender systems*: a synergy between location-based services and recommender system functionality with the goal of providing *interesting* objects to users. So far, we have explored two primary methods to provide *interesting* objects in location-based services: (1) kNN techniques (Section 2.2) simply retrieve the k objects nearest to a user and are completely removed from any notion of *personalization*. (2) k-Best-Neighbor queries (Section 2.3) require users to provide *explicit* preferences in order to retrieve interesting objects. We now make the case that LBS 2.0 systems should also provide interesting objects by dovetailing recommender system functionality, which exploits *implicit* user preferences based on past behavior and opinions, with spatial properties of users and objects inherent in location-based services. We first review how recommender systems work. Second, we highlight a taxonomy of three novel types of *location-based ratings* capable of being generated from many existing location-based applications. Finally, we provide an analysis of real location-based data taken from the Foursquare [8] location-based social network. Our analysis shows that user behavior and opinions are spatially correlated and motivates the need for *location-based recommender systems*.

4.1 How Existing Recommender Systems Function

Recommender systems make use of community opinions and user behavior in an application to help users identify useful, interesting items from a considerably large search space (e.g., inventory from Amazon, news from Google, or movies from Netflix). Many successful and widely deployed systems mainly rely on the *collaborative filtering* (CF) recommendation technique[1]. CF assumes that users provide opinions about a set of items (e.g., movies, books, restaurants), expressed through ratings represented by the triple (*user*, *rating*, *item*) that represents a *user* providing a numeric *rating* for an *item* (e.g., movie). These rating triples are then analyzed to find correlations between similar items (e.g., item-based CF [26]) or similar users (e.g., user-based CF [23]) in order to provide a querying user with interesting recommendations.

4.2 Location-Aware Recommender Systems

Unfortunately, many popular recommendation techniques do not consider the spatial properties of users and/or objects inherent in location-based services. Meanwhile, location-based applications have not considered the use of recommendation techniques. We propose to rectify this situation by highlighting the need for *location-based recommender systems* that exploit the synergy between recommender systems and location-based services in LBS 2.0. For example, whenever a user requests a restaurant in an LBS 2.0 application, the answer should be based on ratings left by users when they were spatially close to the querying user location. Or, when a user requests a movie, the answer could be linked to the user location by suggesting a movie liked by others who live in the same neighborhood as the querying user.

4.2.1 Location-Aware Ratings

The key to creating location-aware recommender systems is to exploit *location-aware ratings*. Recall that existing recommender systems assume that user opinions are expressed through ratings represented by the triple (*user*, *rating*, *item*) where a *user* provides a numeric *rating* for an *item*. However, many existing applications are capable of producing location-aware ratings that augment this traditional rating triple with spatial attributes of users and/or items. We now provide a taxonomy containing three types of location-aware ratings, along with examples of applications that can produce each type of rating.

- **Spatial User Ratings for Non-Spatial Items** associate a location with the user that issued the rating, represented as a four-tuple (*user*, *ulocation*, *rating*, *item*) where *ulocation* indicates the user location. Items are non-spatial in nature (e.g., movies), thus do not have an associated location. As an example, traditional e-commerce applications (e.g., Netflix) may use a users home address as *ulocation*, while mobile applications may associate the location where the user rated the item as the *ulocation*.

- **Non-Spatial User Ratings for Spatial Items** assume items are spatial in nature (e.g., restaurants) and associate the items location with the rating, represented as a four tuple (*user*, *rating*, *item*, *ilocation*), where *ilocation* is the item location. It is assumed that user location is not collected (or available) with the rating. Examples of applications that produce such ratings are e-commerce applications that gather user opinions

[1]We focus on collaborative filtering due to its popularity, though other techniques may be used in practice.

U.S. State	Top Movie Genres	Avg. Rating
Minnesota	Film-Noir	3.8
	War	3.7
	Drama	3.6
	Documentary	3.6
Wisconsin	War	4.0
	Film-Noir	4.0
	Mystery	3.9
	Romance	3.8
Florida	Fantansy	4.3
	Animation	4.1
	War	4.0
	Musical	4.0

Figure 1: MovieLens spatially localized movie ratings

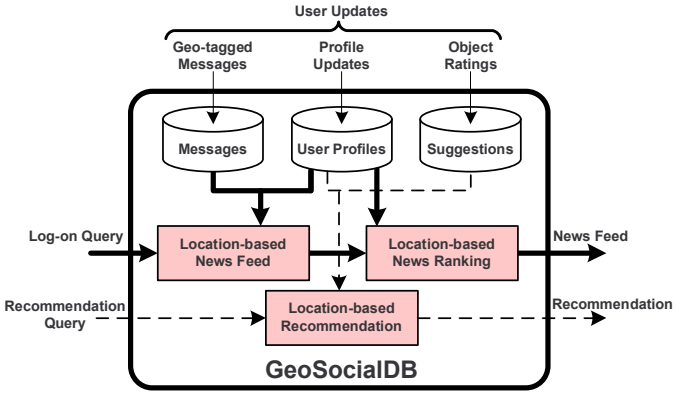

Figure 2: The system architecture of GeoSocialDB.

on venues/destinations (e.g., restaurant review websites), but do not collect user locations.

- **Spatial User Ratings for Spatial Items** associate both user and item locations with the rating, represented as the five-tuple (*user*, *ulocation*, *rating*, *item*, *ilocation*) where *ulocation* and *ilocation* indicate the user and item locations, respectively. Examples of applications that produce such ratings are location-based social networks (e.g., Foursquare [8], Facebook Places [6]).

Using this taxonomy, traditional rating triples can be classified as non-spatial ratings for non-spatial items. Our vision of location-aware recommender systems in LBS 2.0 applications calls for supporting the above taxonomy of location-based ratings in order to provide k interesting items/destinations to querying users.

4.2.2 Motivation

As a testimony to the need for location-aware recommender systems, we analyzed movie ratings from the well-known MovieLens movie recommender system built at the University of Minnesota [19]. We extracted approximately 90K ratings from the system where each rating was associated with the zip code of the user rating the movie, which gave us a real set of spatial user ratings for non-spatial items. We analyzed these ratings to explore whether movie tastes are spatially localized. Figure 1 lists the top-4 movie genres using average ratings of users from the U.S. states Minnesota, Wisconsin, and Florida. While Minnesota and Wisconsin both list "Film-Noir" and "War" movies in their top-2 genres (though in opposite order), the rest of their lists differ, with Minnesota having "Drama" and "Documentary" as their next popular genre, while Wisconsin lists "Mystery" and "Romance" as the next most popular genre. Meanwhile, the top-4 movie genres from Florida differ vastly, as "Fantasy" and "Animation" are the most popular movie genres, followed by the "War" and "Musical" genres. This analysis suggests that movie preferences are unique to specific spatial regions. Furthermore, our analysis verifies earlier work from the New York Times [21] that analyzed Netflix user queues in major U.S. cities to determine movie tastes tend to be localized even at the granularity of zip codes.

This behavior we observe here suggests that recommendations should be *localized* to a querying user location, i.e., recommendations should be influenced by ratings with embedded user locations spatially close to the querying user. The intuition is that localized recommendations will provide a sense of what other system users prefer to do when in the same location as the querying user. We are currently crawling for and analyzing location-based data from the Foursquare location-based social network [8]. Our initial analysis of the Foursquare data further suggests that user behavior and preferences are spatially localized.

5. THE GEOSOCIALDB SYSTEM

This section presents the GeoSocialDB system, still under development at University of Minnesota. We built GeoSocialDB to demonstrate the synergy of all aspects of LBS 2.0. The components of the system are divided into modules built into the PostgreSQL open-source database engine [22] and modules built on top of PostgreSQL. In particular, the personalization module is built into PostgreSQL through the FlexPref engine [15]. The socialization modules (news feed and news ranking) are currently built on top of the engine (we have plans to migrate parts of them inside later). Finally, part of the recommender system, namely, generating the collaborative filtering model is built inside PostgreSQL, while the location-aware recommendation module is outside the engine.

Figure 2 depicts the GeoSocialDB system architecture that consists of three main modules: *location-aware news feed*, *location-aware news ranking*, and *location-aware recommender*. In addition, GeoSocialDB stores three types of data: spatial messages, user profiles, and spatial ratings. Note that the personalization components is built into in the PostgreSQL engine we are using. GeoSocialDB can take five different types of input as follows: (1) A profile update, where system users can update their personal information and/or friend list. (2) A new (spatial) message, whereby a system user posts a new message and specifies the spatial extents of that message, indicating the spatial range for which the message is effective. The message is deemed relevant to only those users who are located within its spatial extent. (3) A new (spatial) rating, whereby a system user rates an item and the user location and/or item item location is associated with the rating, as described in Section 4. (4) A location-aware news feed query, i.e., log-on query. Once a GeoSocialDB user logs on to the system, a location-aware news feed query is fired to retrieve the relevant news feed, i.e., messages posted from the user friends that have spatial extents that include the location of the querying user. The processing of this query type was discussed in Section 3. (5) A Location-aware recommendation queries. GeoSocialDB users can request recommendations of either spatial items (e.g., restaurants, stores) or non-spatial items (e.g., movies, books) through explicitly submitting a location-aware recommender query. Then, the *location-aware recommender* module suggests a set of items based on: (a) the user location (if available), (b) the item location (if available), and (c) pre-

Figure 3: GeoSocialDB Web Application ScreenShot

viously posted ratings by either the user or the friends of the user, as was briefly described in Section 4.

Figure 3 depicts the user interface of GeoSocialDB. The left-hand side of the user interface shows the user's personal information and the navigation menu, where the location-based news feed service is shown as selected by the user. The center area of the user interface has a form for the user to share location-based news with friends and displays the relevant news for the user. For example, the user may share a message "The pizza at ABC restaurant is awesome" with a range distance of one mile. The right-hand side of the user interface is integrated with Google Maps, where the arrow indicates the user's current location, the circle indicates the range distance specified for the message being edited by the user (i.e.,one mile), and the markers "A" and "B" indicate the locations of the user's relevant news, as determined by the GeoRank module (Section 3.3). A user can also select the recommendation functionality to get a set of recommended restaurants shown on the map.

6. CONCLUSION

We call for a new era of Location-based Services (LBS) 2.0, in which personalization, socialization, and recommendations play a major role in location-based services. This is much needed for location-based services to cope with the ubiquitous Web 2.0 environment where content sharing, social networks, and interoperability became common for all web services. For personalization in LBS 2.0, we advocate for going beyond the rigid *nearest*-neighbor queries to the more personalized *best*-neighbor queries where the user preference and context are taken into account when deciding on the query answer. For socialization in LBS 2.0, we advocate for going beyond using the location information as extra attributes to injecting the location-awareness into the core functionality of social networks and news aggregators. For recommendations in LBS 2.0, we advocate for adapting existing commercially successful recommender systems to consider the spatial aspects when making any of its decisions. Then, we presented an early prototype of the GeoSocialDB system, under development at University of Minnesota, with the goal of realizing LBS 2.0 with its three aspects, personalization, socialization, and recommendation.

7. REFERENCES

[1] S. Börzsönyi, D. Kossmann, and K. Stocker. The Skyline Operator. In *ICDE*, 2001.

[2] C.-Y. Chan, H. Jagadish, K.-L. Tan, A. K. Tung, and Z. Zhang. Finding k-Dominant Skylines in High Dimensional Space. In *SIGMOD*, 2006.

[3] S. Chaudhuri and L. Gravano. Evaluating Top-K Selection Queries. In *VLDB*, 1999.

[4] K. Deng, X. Zhou, and H. T. Shen. Multi-source Skyline Query Processing in Road Networks. In *ICDE*, 2007.

[5] Facebook. http://www.facebook.com/.

[6] The Facebook Blog, "Facebook Places": http://tinyurl.com/3aetfs3.

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

[8] Foursquare: http://foursquare.com.

[9] Government 2.0 Summit. Washington D.C., USA, September, 2010. http://www.gov2summit.com/gov2010.

[10] G. R. Hjaltason and H. Samet. Distance Browsing in Spatial Databases. *TODS*, 24(2):265–318, 1999.

[11] Z. Huang, H. Lu, B. C. Ooi, and A. K. H. Tung. Continuous Skyline Queries for Moving Objects. *TKDE*, 18(12):1645–1658, 2006.

[12] iGoogle. http://www.google.com/ig.

[13] H.-P. Kriegel, M. Renz, and M. Schubert. Route Skyline Queries: A Multi-Preference Path Planning Approach. In *ICDE*, 2010.

[14] Di-Ann Eisnor. LBS 2.0: Don't Just Map It, Manipulate It. April, 2010. http://gigaom.com/2010/04/05/lbs-2-0-dont-just-map-it-manipulate-it/.

[15] J. J. Levandoski, M. F. Mokbel, and M. E. Khalefa. FlexPref: A Framework for Extensible Preference Evaluation in Database Systems. In *ICDE*, 2010.

[16] Library 2.0. http://www.library20.org/.

[17] M. D. Morse, J. M. Patel, and W. I. Grosky. Efficient Continuous Skyline Computation. *Information Science*, 177(17):3411–3437, 2007.

[18] K. Mouratidis, M. L. Yiu, D. Papadias, and N. Mamoulis. Continuous nearest neighbor monitoring in road networks. In *VLDB*, 2006.

[19] MovieLens: http://www.movielens.org/.

[20] MyYahoo! http://my.yahoo.com/.

[21] New York Times - A Peek Into Netflix Queues: http://www.nytimes.com/interactive/2010/01/10/nyregion/20100110-netflix-map.html.

[22] PostgreSQL: http://www.postgresql.org/.

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

[24] Mark Sedra. "Revolution 2.0: democracy promotion in the age of social media". The Global and Mail. February, 2011. http://www.theglobeandmail.com/news/opinions/opinion/revolution-20-democracy-promotion-in-the-age-of-social-media/article1891015/.

[25] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest Neighbor Queries. In *SIGMOD*, 1995.

[26] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-Based Collaborative Filtering Recommendation Algorithms. In *WWW*, 2001.

[27] A. Silberstein, J. Terrace, B. F. Cooper, and R. Ramakrishnan. Feeding Frenzy: Selectively Materializing User's Event Feed. In *SIGMOD*, pages 831–842, June 2010.

[28] Tech Pluto. Core Characteristics of Web 2.0 Services. http://www.techpluto.com/web-20-services/.

[29] Travel 2.0. http://travel2dot0.com/.

[30] Twitter. http://www.twitter.com/.

[31] X. Xiong, M. F. Mokbel, and W. G. Aref. SEA-CNN: Scalable Processing of Continuous K-Nearest Neighbor Queries in Spatio-temporal Databases. In *ICDE*, Apr. 2005.

[32] M. L. Yiu and N. Mamoulis. Efficient Processing of Top-k Dominating Queries on Multi-Dimensional Data. In *VLDB*, 2007.