MobiFeed: A Location-Aware News Feed System for Mobile Users*

Wenjian Xu
Department of Computer Science
City University of Hong Kong
Hong Kong
wenjianxu2@cityu.edu.hk

Chi-Yin Chow
Department of Computer Science
City University of Hong Kong
Hong Kong
chiychow@cityu.edu.hk

Man Lung Yiu
Department of Computing
Hong Kong Polytechnic University
Hong Kong
csmlyiu@comp.polyu.edu.hk

Qing Li
Department of Computer Science
City University of Hong Kong
Hong Kong
itqli@cityu.edu.hk

Chung Keung Poon
Department of Computer Science
City University of Hong Kong
Hong Kong
csckpoon@cityu.edu.hk

ABSTRACT

A location-aware news feed system enables mobile users to share geo-tagged user-generated messages, e.g., a user can receive nearby messages that are the most relevant to her. In this paper, we present MobiFeed that is a framework designed for scheduling news feeds for mobile users. MobiFeed consists of three key functions, *location prediction*, *relevance measure*, and *news feed scheduler*. The *location prediction* function is designed to predict a mobile user's locations based on an existing path prediction algorithm. The *relevance measure* function is implemented by combining the vector space model with non-spatial and spatial factors to determine the relevance of a message to a user. The *news feed scheduler* works with the other two functions to generate news feeds for a mobile user at her current and predicted locations with the best overall quality. To ensure that MobiFeed can scale up to a larger number of messages, we design a heuristic *news feed scheduler*.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems—*Query processing*; H.2.8 [**Database Management**]: Database Applications—*Spatial databases and GIS*

General Terms

Algorithms, Design, Performance

*The work described in this paper was partially supported by grants from City University of Hong Kong (Project No. 7200216, 7002686 and 7002722).

ACM SIGSPATIAL GIS '12, November 6-9, 2012. Redondo Beach, CA, USA

Keywords

Location-aware social networks, news feed, mobile computing, online scheduling

1. INTRODUCTION

A news feed is a common functionality of existing location-aware social network systems. It enables mobile users to post geo-tagged messages and receive nearby user-generated messages, e.g., "*Alice can receive 4 messages that are the most relevant to her among the messages within 1 km from her location every 10 seconds*". Since a location-aware social network system usually possesses a huge number of messages, there are many messages in a querying user's vicinity. Coupled with user mobility, a key challenge for the location-aware news feed system is how to efficiently schedule the k most relevant messages for a user and display them on the user's mobile device. Although location-aware news feed and social network systems have attracted a lot of attention from different research communities, none of these applications has focused on how to schedule news feeds for mobile users. The state-of-the-art research prototype of a location-aware news feed system is GeoFeed [2]. In contrast to GeoFeed, MobiFeed focuses on challenges in providing location-aware news feeds for mobile users. We design a location-aware news feed scheduler that works with our *location prediction* and *message relevance measure* functions to provide news feeds for mobile users.

In this paper, we present MobiFeed that is a location-aware news feed framework designed for social network systems to schedule news feeds for mobile users. Figure 1 depicts an application scenario. A MobiFeed user, Alice, can generate a message and tag a point (e.g., m_1), a spatial extent (e.g., m_{14} is associated with a circular spatial area), or a venue (e.g., m_6 and m_7 are spatially associated with restaurant R_1) as its geo-location. Alice can also issue a location-aware news feed query to retrieve the k most relevant messages within her specified range distance D from her location. MobiFeed consists of three key functions: *location prediction*, *relevance measure*, and *news feed scheduler*. Given a user u's location $u.location$ at the current time t_0,

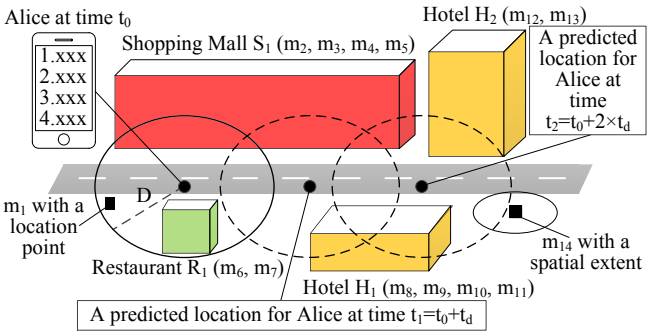

Figure 1: Location-aware news feed scheduling.

u's required minimum message display time t_d, u's specified range distance D, u's requested number of messages per news feed, and a look-ahead steps n, the *location prediction* function estimates n future locations for u at times $t_1 = t_0 + t_d$, $t_2 = t_0 + 2 \times t_d$, ..., and $t_n = t_0 + n \times t_d$, the *relevance measure* function calculates the relevance score of each candidate message with a geo-location intersecting any u's query region (i.e., a circular area centered at $u.location$ or a predicted location with a radius D), and the *news feed scheduler* generates news feeds from the candidate messages for u's query regions at $t_0, t_1, \ldots, t_n$ with the best total relevance score.

2. SYSTEM OVERVIEW

Figure 2 depicts an overview of the MobiFeed framework. MobiFeed stores geo-tagged user-generated messages in a database. It interacts with the *location prediction* and *relevance measure* functions to select a collection of messages from the database as a news feed for a mobile user at a particular location.

Geo-tagged messages. A geo-tagged user-generated message is defined as a tuple (*MessageID*, *SenderID*, *Content*, *Timestamp*, *Spatial*), where *MessageID* and *SenderID* are the message identifier and its sender's identifier, respectively. *Content* is the message content. *Timestamp* is the message submission time, and *Spatial* specifies the message's spatial extent. As depicted in Figure 1, the spatial extent of a message can be a point location (e.g., m_1), a user-specified spatial region (e.g., m_{14}), or the spatial region of a venue (e.g., the spatial extent of m_2 is the shopping mall S_1).

System users. A mobile user u at location $u.location$ equipped with a GPS-enabled mobile device is able to (a) post a new message with a spatial extent, and (b) receive at most $u.k$ messages within u's specified range distance $u.D$ (i.e., the query region of a news feed) at a particular time as a news feed. MobiFeed computes a news feed for u by

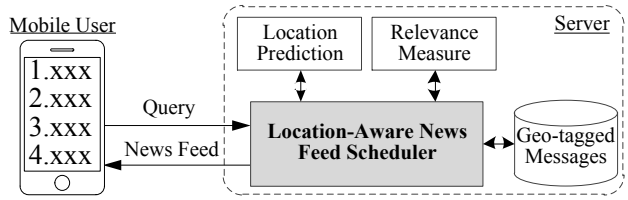

Figure 2: MobiFeed framework.

selecting messages based on their relevance to u and u's movement. Each selected message must be displayed on u's mobile device without any interruption for at least u's specified minimum display time $u.t_d$. u reports its location to the system at every time period $u.t_u$. After receiving u's location update, a news feed is computed for u. In MobiFeed, we set $u.t_u = u.t_d$. If $u.t_u < u.t_d$, newly selected messages cannot be displayed until previously selected messages have been displayed for $u.t_d$. On the other hand, if $u.t_u > u.t_d$, the system may not be able to provide an accurate news feed for u as it does not know u's exact location. Altogether, $u.k$, $u.D$ and $u.t_d$ constitute u's *preferences* for MobiFeed.

Quality measure. Given a user u_i and a message m_j, the relevance measure function returns a relevance score $relevanceScore(u_i, m_j)$. Without loss of generality, we assume the relevance score is between zero and one. In information retrieval, query-relevance ranking algorithms usually display a document that is more relevant to a user's query at a higher position in a result list [1]. To this end, MobiFeed supports different weights for different slots in a news feed result list, i.e., a higher weight is given to a message displayed at a higher slot because it would be easier to draw a user's attention. In this paper, we use a simple weighting scheme. Given a result list with k slots, the weight of the first slot is k, the weight of the second slot is $k-1$, and so on. In general, the weight of a message m_j at the top j-th slot ($1 \le j \le k$) is $displayWeight(j,k) = k - (j-1)$. Thus, the relevance score of a news feed f_i with k messages $m_1, m_2, \ldots, m_k$ displayed at the j-th position in a result list for a user u_i is calculated as:

$$relevanceScore(f_i) = \sum_{j=1}^{k} relevanceScore(u_i, m_j) \times displayWeight(j,k). \quad (1)$$

Problem definition. Figure 3 depicts an example of location-aware news feed scheduling, where a user u sends a query with her location to MobiFeed at the current time t_0. The *location prediction* function predicts u's location at each of the next two (i.e., $n = 2$) minimum display times t_d, i.e., $t_1 = t_0 + t_d$ and $t_2 = t_0 + 2 \times t_d$. There are totally 11 candidate messages for the three news feeds at the times t_0, t_1, and t_2. m_4 and m_5 are tagged with a spatial region and a point location, respectively. $\{m_1, m_2, m_3\}$, $\{m_6, m_7, m_8, m_9\}$, and $\{m_{10}, m_{11}\}$ are associated with venues A, B, and C (represented by rectangles), respectively. The lifetime of each message with its relevance score for u at t_0, t_1, and t_2 is shown on a timeline chart. Note that the lifetime of m_5 is broken from t_1 to t_2; however, most existing scheduling algorithms assume tasks with a continuous lifetime. Our scheduling problem can be formulated as follows: *Given a user u's location-aware news feed query and a look-ahead step n, MobiFeed predicts u's locations at each of the next n minimum display times, and schedules at most k messages for the news feed at each location (i.e., one reported and n predicted locations), such that the total relevance score of the generated news feeds is maximized. Since our problem focuses on online scheduling, it requires efficient query processing.*

3. LOCATION PREDICTION AND MESSAGE RELEVANCE MEASURE

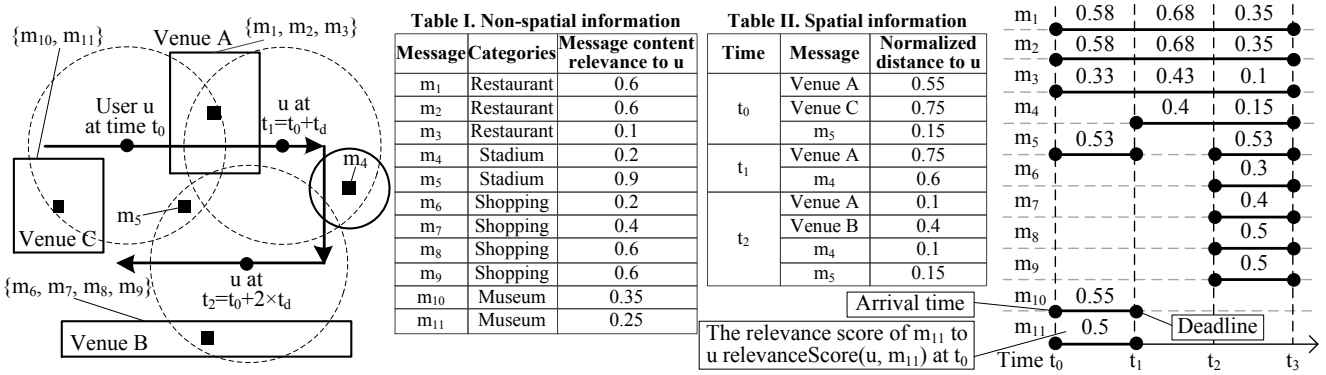

Table I. Non-spatial information

Message	Categories	Message content relevance to u
m_1	Restaurant	0.6
m_2	Restaurant	0.6
m_3	Restaurant	0.1
m_4	Stadium	0.2
m_5	Stadium	0.9
m_6	Shopping	0.2
m_7	Shopping	0.4
m_8	Shopping	0.6
m_9	Shopping	0.6
m_{10}	Museum	0.35
m_{11}	Museum	0.25

Table II. Spatial information

Time	Message	Normalized distance to u
t_0	Venue A	0.55
	Venue C	0.75
	m_5	0.15
t_1	Venue A	0.75
	m_4	0.6
t_2	Venue A	0.1
	Venue B	0.4
	m_4	0.1
	m_5	0.15

Figure 3: Location-aware news feed scheduling ($w = 0.5$).

In this section, we present the details of the *location prediction* and *relevance measure* functions in MobiFeed.

3.1 Location Prediction

The *location prediction* function can use any existing location prediction algorithm if it can predict a user's location at a specified future time in a road network. We here describe how to incorporate the path prediction algorithm [4] into MobiFeed. Given a user u's current location, u's historical trajectories, the road map, and a future time t, the path prediction algorithm estimates u's location at t. Let $G = (V, E)$ be a graph model of a road network, where E is a set of road segments and V is a set of intersections of road segments that are represented by circles and lines, respectively. The algorithm performs two steps to predict u's direction and speed.

Step 1. Direction prediction. When a user u is moving on an edge e_i, this step predicts which adjacent edge e_j of e_i u will go based on u's historical trajectory set $\mathcal{T}_u$. This step has three ways to predict a user's path [4]. (1) Given two edges e_i and e_j incident to a vertex v, the transition probability of u turning from e_i to e_j is defined as $P(e_i, v, e_j) = \frac{\tau(\mathcal{T}_u, e_i \to e_j)}{\sum_{e_k} \tau(\mathcal{T}_u, e_i \to e_k)}$, where $\tau(\mathcal{T}_u, e_i \to e_j)$ is the number of trajectories in $\mathcal{T}_u$ that turn from e_i to e_j and e_k is an adjacent edge of e_i incident to v. For each adjacent edge e_j of e_i incident to v, this step calculates $P(e_i, v, e_j)$ and predicts that u will turn to e_j with the largest probability. (b) However, if $\tau(\mathcal{T}_u, e_i \to e_j)$ is empty, the notion of reverse mobility statistics $P(e_i, v, e_j) = \frac{\tau(\mathcal{T}_u, e_j \to e_i)}{\sum_{e_k} \tau(\mathcal{T}_u, e_k \to e_i)}$ is used. (c) In case that both $\tau(\mathcal{T}_u, e_i \to e_j)$ and $\tau(\mathcal{T}_u, e_j \to e_i)$ are empty, we select the adjacent edge of e_i incident to v with the smallest deviation angle from u's current travel direction, which is derived from u's initial location at the query time and current location.

Step 2. Speed prediction. This step estimates u's travel speed $S(e)$ on an edge e. The basic idea is to compute $S(e)$ by the average historical travel speeds of e from u's $\mathcal{T}_u$ [4]. If e does not exist in $\mathcal{T}_u$, we use a heuristic method that computes $S(e) = A(e) \times \alpha$, where $A(e)$ is the speed limit of e and α is a system parameter.

In MobiFeed, after we find that u moving on an edge e_i will enter an edge e_j at t' from a vertex v_s and stay at e_j at t. Let (x_s, y_s) denote the location of v_s and (x_e, y_e) denote the the location of the other vertex of e_j. The predicted location of u at t is calculated as $(\frac{\lambda_1 \times x_e + \lambda_2 \times x_s}{\lambda_1 + \lambda_2}, \frac{\lambda_1 \times y_e + \lambda_2 \times y_s}{\lambda_1 + \lambda_2})$, where $\lambda_1 = (t - t') \times S(e_j)$ and $\lambda_2 = L(e_j) - \lambda_1$ (where $L(e_j)$ is the length of e_j).

3.2 Message Relevance Measure

MobiFeed only requires the *relevance measure* function to return a score to indicate the relevance of a message m_j to a user u_i, i.e., $relevanceScore(u_i, m_j)$. We present three relevance measure methods, and then describe how to combine them to implement the *relevance measure* function.

Message categories. We group messages into categories based on their geo-tagged locations or keywords. For example, in FourSquare, each message can be categorized by its one or more associated venues, e.g., restaurant, stadium, and museum. We maintain a *user-category matrix* where each entry c_{ij} is the ratio of the number of a user u_i's messages associated with the category c_j to the total number of u_i's messages. For example, if u_i has issued two messages, i.e., m_1 with categories "restaurant" and "stadium", and m_2 with category "stadium", the ratios of "restaurant" and "stadium" are $\frac{1}{2}$ and 1, respectively. c_{ij} indicates the relevance of a message category c_j to a user u_i, i.e., $categoryScore(u_i, c_j)$.

Message contents. We treat a message as a document and use an existing information retrieval technique to measure its relevance to a particular user. We here consider the vector space model, where each message m_j is considered as a vector of weighted terms, $m_j.V = \langle w_{j1}, w_{j2}, \ldots, w_{j|\mathcal{T}|} \rangle$, where $\mathcal{T}$ is the term set. In general, a term $T_k \in \mathcal{T}$ should be weighted higher for m_j if T_k occurs more frequently in m_j but rarely in other messages in the message set $\mathcal{M}$. The weight can be computed by the well-known $TF \times IDF$ scheme: $w_{jk} = tf_{jk} \cdot \log \frac{|\mathcal{M}|}{df_k}$, where tf_{jk} is the *term frequency* of T_k in m_j and df_k is the *document frequency* of T_k in $\mathcal{M}$. To incorporate the vector space model into MobiFeed, we maintain a query vector for each user based on her submitted messages. Given a querying user u_i with a query vector $u_i.V$ and a message m_j, we use the cosine similarity to compute $contentScore(u_i, m_j) = \frac{\sum_{k=1}^{|\mathcal{T}|} w_{ik} \cdot w_{jk}}{\sqrt{\sum_{k=1}^{|\mathcal{T}|} w_{ik}^2} \cdot \sqrt{\sum_{k=1}^{|\mathcal{T}|} w_{jk}^2}}$.

Distance. The relevance of a message m_j to a user u_i can be measured by their distance, i.e., $Dist(u_i, m_j)$. If m_j is associated with a spatial extent or a venue, $Dist(u_i, m_j)$ returns the *minimum* distance between u_i and the spatial extent or venue of m_j. To accommodate the difference in the value ranges of $Dist(u_i, m_j)$ and other relevance measures, we normalize $Dist(u_i, m_j)$ to be from zero to one, i.e.,

$NDist(u_i, m_j) = 1 - \frac{Dist(u_i, m_j)}{u.D}$, where $u.D$ is the query range distance of a news feed.

Relevance measure function. We employ two-level and linear combinations to integrate the aforementioned three methods into the *relevance measure* function. At the first level, we select a querying user u_i's top-δ categories using $categoryScore(u_i, c_j)$, where δ is a parameter to control the number of categories, and filter out candidate messages that do not belong to any top-δ categories. At the second level, we measure the relevance of each candidate message m_j to u_i using a linear combination of $contentScore(u_i, m_j)$ and $NDist(u_i, m_j)$ [3] as:

$$relevanceScore(u_i, m_j) = contentScore(u_i, m_j) \times (1 - w) + NDist(u_i, m_j) \times w, \quad (2)$$

where $0 \leq w \leq 1$ and w is a parameter that gives a weight for the importance of the distance factor with respect to the message content score.

4. LOCATION-AWARE NEWS FEED SCHEDULER

In this section, we present a n-look-ahead location-aware news feed scheduling algorithm for MobiFeed, where n is a system parameter to control the number of locations predicted for a mobile user. In general, n should be larger, if the location prediction algorithm can provide more accurate locations, and thus, different values of n would be assigned to different users and areas in a road network (e.g., $P(e_i, v, e_j)$ should be larger than a certain threshold). Since our online scheduling algorithm has to be efficient and scalable, we design a heuristic scheduling algorithm for MobiFeed.

Data structure. In MobiFeed, a spatial grid structure is used to index all geo-tagged messages. Given a user u's query, a range query is issued to the grid index to retrieve the geo-tagged messages, which are not generated by u, associated with a location point, a spatial extent, or a venue region intersecting the query region.

Algorithm. After a mobile user u issues a location-aware news feed query to MobiFeed, MobiFeed calls the *location prediction* function to return n future locations for u. Its scheduler then finds a set of candidate messages for each of $n + 1$ locations and calls the *relevance measure* function to filter out all candidate messages that do not belong to any top-δ categories and determine the relevance of each remaining candidate message to u. The scheduler finally returns a news feed for each location such that the total relevance score is maximized. Our heuristic scheduling algorithm consists of two main steps.

Step 1. Candidate message step. Given u's query at time t_0, the *location prediction* function predicts n locations for u at times t_1, t_2, ..., t_n, where $t_i = t_0 + u.t_d \times i$ and $u.t_d$ is u's specified message minimum display time. For each of $n+1$ locations, a range query with a circular region centered at the location with a radius of $u.D$ is issued to retrieve the messages intersecting the query region as a set of candidate messages $CandidateMsg_i$ $(0 \leq i \leq n)$. Then, the *relevance measure* function filters out all messages that do not belong to any top δ categories from each $CandidateMsg_i$. For each remaining candidate message m, a relevance score for m, i.e., $relevanceScore(u, m)$, is calculated to indicate the relevance of m to u. Finally, the messages in each $CandidateMsg_i$ are sorted by their relevance score in non-increasing order. To break ties, precedence will be given to a message with a more recent post time.

Step 2. Online scheduling step. As depicted in the running example (see Figure 3), some candidate messages are included in multiple sets of candidate messages. For example, m_1 is included in $CandidateMsg_0$, $CandidateMsg_1$ and $CandidateMsg_2$, so m_1 can be scheduled to one of these query regions or none. This step aims at scheduling at most $k \times (n + 1)$ candidate messages to the $n + 1$ query regions such that the total relevance score of these query regions is maximized. The input of this step is $n + 1$ sets of sorted candidate messages for $n + 1$ query regions. For each query region q_i, we calculate a score for its candidate message m_j with the highest relevance score by $relevanceScore(u, m_j) \times displayWeight(j, k)$ (see Equation 1), where u is the querying user and k is the highest available position in q_i's result list. The message with the highest score, denoted as *BestMsg*, is selected. *BestMsg* is assigned to the query region giving the highest score, and it is no longer a candidate message for any query region. To break a tie, *BestMsg* is assigned to the query region where the first message in its candidate message set has the smallest relevance score. The reason is that other query regions have a higher chance to put a message with a larger relevance score to the same slot in the result list. Candidate messages are repeatedly selected to appropriate query regions until each query region has k messages or its candidate message set becomes empty. Whenever k messages have been assigned to a query region, its corresponding candidate message set is discarded. The computed $n + 1$ news feeds are sent to u. u's mobile device immediately displays the first news feed, i.e., the query region at t_0, and then displays each of the remaining news feeds one by one for every t_d.

5. CONCLUSION

We presented MobiFeed that is a location-aware news feed framework designed for scheduling news feeds for mobile users. We described the three key functions of MobiFeed, namely, *location prediction*, *relevance measure*, and *news feed scheduler*. The *location prediction* function is designed to estimate a user's location based on the path prediction algorithm. The *relevance measure* function is implemented by integrating non-spatial and spatial factors into the vector space model to measure the relevance of a message to a user. We designed a heuristic *news feed scheduler* that works with the other two functions to generate news feeds for a user at her current and look-ahead locations.

6. REFERENCES

[1] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press/Addison-Wesley, 1999.

[2] J. Bao, M. F. Mokbel, and C.-Y. Chow. GeoFeed: A location-aware news feed system. In *IEEE ICDE*, 2012.

[3] C.-Y. Chow, J. Bao, and M. F. Mokbel. Towards location-based social networking services. In *ACM SIGSPATIAL LBSN*, 2010.

[4] H. Jeung, M. L. Yiu, X. Zhou, and C. S. Jensen. Path prediction and predictive range querying in road network databases. *VLDB Journal*, 19(4):585–602, 2010.