

# Geo-social Recommendations based on Incremental Tensor Reduction and Local Path Traversal

Panagiotis Symeonidis  
Aristotle University  
Thessaloniki, Greece  
symeon@csd.auth.gr

Alexis Papadimitriou  
Aristotle University  
Thessaloniki, Greece  
apapadi@csd.auth.gr

Yannis Manolopoulos  
Aristotle University  
Thessaloniki, Greece  
manolopo@csd.auth.gr

Pinar Senkul  
Middle East Technical  
University  
Ankara, Turkey  
senkul@ceng.metu.edu.tr

Ismail Toroslu  
Middle East Technical  
University  
Ankara, Turkey  
toroslu@ceng.metu.edu.tr

## ABSTRACT

Social networks have evolved with the combination of geographical data, into Geo-social networks (GSNs). GSNs give users the opportunity, not only to communicate with each other, but also to share images, videos, locations, and activities. The latest developments in GSNs incorporate the usage of location tracking services, such as GPS to allow users to “check in” at various locations and record their experience. In particular, users submit ratings or personal comments for their location/activity. The vast amount of data that is being generated by users with GPS devices, such as mobile phones, needs efficient methods for its effective management. In this paper, we have implemented an online prototype system, called Geo-social recommender system, where users can get recommendations on friends, locations and activities. For the friend recommendation task, we apply the FriendLink algorithm, which performs a local path traversal on the friendship network. In order to provide location/activity recommendations, we represent data by a 3-order tensor, on which latent semantic analysis and dimensionality reduction is performed using the Higher Order Singular Value Decomposition (HOSVD) technique. As more data is accumulated to the system, we use incremental solutions to update our tensor. We perform an experimental evaluation of our method with two real data sets and measure its effectiveness through recall/precision.

## Keywords

tensor, geographical, social, geo-social, recommendations<sup>1</sup>

<sup>1</sup>This work has been partially funded by the Greek GSRT (project number 10TUR/4-3-3) and the Turkish TUBITAK (project number 109E282) national agencies as part of Greek-Turkey 2011-2012 bilateral scientific cooperation.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ACM LBSN '11, November 1, 2011, Chicago, IL, USA

Copyright ©2011 ACM 978-1-4503-1033-8/11/11 ...\$10.00.

## 1. INTRODUCTION

Over the past few years, social networks have attracted a huge attention after the widespread adoption of Web 2.0 technology. Social networks combined with geographical data, have evolved into Geo-social networks (GSNs). GSNs such as Facebook Places, Google Places, Foursquare.com, etc., which allow users with mobile phones to contribute valuable information, have increased both in popularity and size. These systems are considered to be the next big thing on the web [4]. An interesting statistic is that more than 250 million users are daily accessing Facebook through their mobile devices and they are twice as active than non-mobile users.

GSNs allow users to use their GPS-enabled device, to “check in” at various locations and record their experience. In particular, users submit ratings or personal comments for the location/activity they visited/performed. That is, they “check in” at various places, to publish their location online, and see where their friends are. Moreover, they can either comment on a friend’s location or comment on their own. These GSN systems, based on a user’s “check in” profile, can also provide activity and location recommendations. For an activity recommendation, if a user plans to visit some place, the GSN system can recommend an activity (i.e. dance, eat, etc.). For a location recommendation, if a user wants to do something, the GSN system can recommend a place to go. Recently, Zheng et al. [12] proposed a User Collaborative Location and Activity Filtering (UCLAF) system, which is based on Tensor decomposition. However, as the authors claim, they do not update their system online as more users accumulate data continuously over time. Moreover, even though their system provides location and activity recommendations to users, it does not consider the case of providing also friend recommendations.

Our prototype system GeoSocial is an online recommender system that relies on user check-ins to provide friend, location and activity recommendations. Every registered user is presented with the option of checking in. The procedure involves selecting the location he is currently at, the activity he is performing there, and finally rating that activity. Based on the users’ “check in” history and friendship network, GeoSocial provides friend, location and activity recommendations. Friends are recommended based on

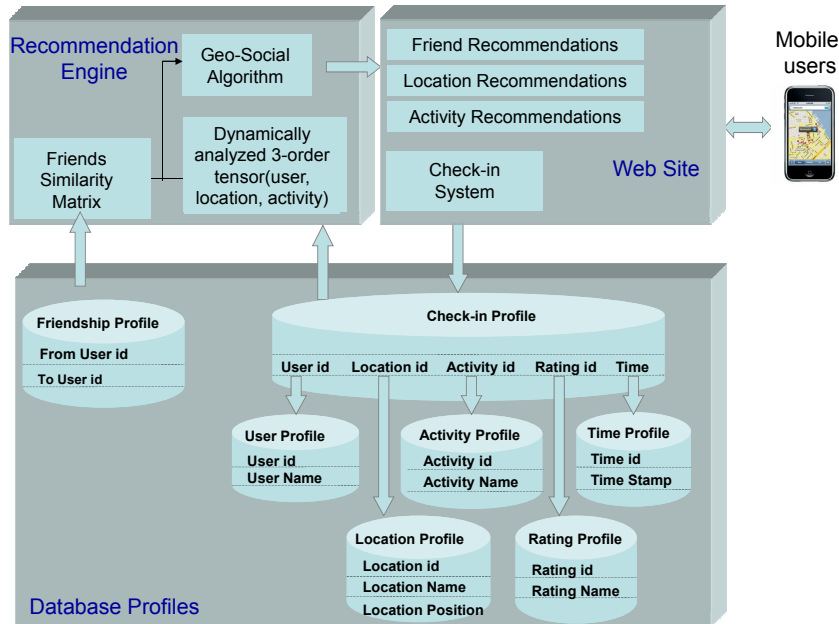


Figure 1: Components of the Geo-social recommender system.

the FriendLink algorithm [6] and the average geographical distances between users’ “check-ins”, which are used as link weights. Users, locations and activities are also inserted into a 3-order tensor, which is then used to provide location and activity recommendations.

The remainder of this paper is organized as follows. Section 2 summarizes the related work, whereas Section 3 describes the GeoSocial recommender system and its components. Section 4 describes the FriendLink algorithm, which performs a local path traversal on the friendship network to provide friend recommendations. Section 5 explains the main steps that are followed when performing the tensor reduction to detect latent associations between the user, location and activity dimensions and also the way we update the tensor data by implementing the Incremental Tensor Reduction (ITR) algorithm. In Section 6 we study the performances of ITR and FriendLink in terms of friend, location and activity recommendations. Finally, Section 7 concludes the paper and proposes possible future work.

## 2. RELATED WORK

Recently emerged GSNs (i.e. Gowalla.com, Foursquare.com, Facebook Places etc.) provide to users activity or location recommendation. For example, in Gowalla.com a target user can provide to the system the activity he wants to do and the place he is (e.g. coffee in New York). Then, the system provides a map with coffee places which are nearby the user’s location and were visited many times from people he knows. Moreover, Facebook Places allows users to see where their friends are and share their location in the real world.

There is a little research on the scientific field of GSNs. Backstrom et al. [1] use user-supplied address data and the network of associations between members of the Facebook social network to measure the relationship between geography and friendship. Using these measurements, they can predict the location of an individual. Scellato et al. [10] pro-

posed a graph analysis based approach to study social networks with geographic information. They also applied new geo-social metrics to four large-scale Online Social Network data sets (i.e. Liveljournal, Twitter, FourSquare, BrightKite). Quercia et al. [7] address the mobile cold-start problem when recommending social events to users without any location history.

Leung et al. [5] propose the Collaborative Location Recommendation (CLR) framework for location recommendation. The framework considers activities and different user classes to generate more precise and refined recommendations. The authors also incorporate a dynamic clustering algorithm, namely Community-based Agglomerative-Divisive Clustering (CADC), into the framework in order to cluster the trajectory data into groups of similar users, similar activities and similar locations. The algorithm can also be updated incrementally when new GPS trajectory data is available.

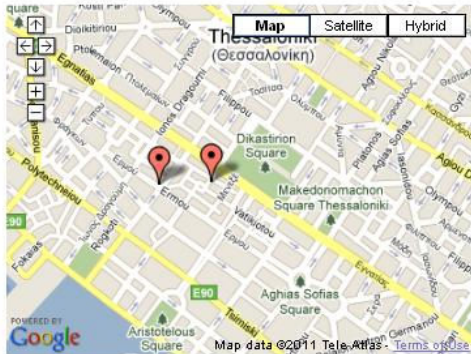
Ye et al. [11] believe that user preferences, social influence and geographical influence should be considered when providing Point of Interest recommendations. They study the geographical clustering phenomenon and propose a power-law probabilistic model to capture the geographical influence among Points of Interest. Finally, the authors evaluate their proposed method over the Foursquare and Whrrl datasets and discover, among others, that geographical influence is more important than social influence and that item similarity is not as accurate as user similarity due to a lack of user check-ins.

Moreover, there are tensor-based approaches. For example, Biancalana et al. [2] implemented a social recommender system based on a tensor that is able to identify user preferences and information needs and suggests personalized recommendations for possible points of interest (POI). Furthermore, Zheng et al. [13] proposed a method, where geographical data is combined with social data to provide location

Please choose an activity you would like to perform:

working

**We recommend the following location(s)  
for the selected activity:**



Place	Times visited by your friends and similar users	Average Rating
Auth Library	1	1/3
Starbucks	1	3/3

(a)

Please choose a nearby location:

Auth Library

**We recommend the following activities  
for the selected location:**



Place	Activity	Times visited by your friends and similar users
Trendy Bar	clubbing	1
Picadilly	clubbing	1

(b)

**Figure 2: Location and activity recommendations made by the Geo-social recommender system.**

and activity recommendations. The authors used GPS location data, user ratings and user activities to propose location and activity recommendations to interested users and explain them accordingly. Moreover, Zheng et al. [12] proposed a User Collaborative Location and Activity Filtering (UCLAF) system, which is based on Tensor decomposition.

In contrast to the aforementioned tensor-based methods, our Geosocial recommender system provides (i) location and activity recommendations (ii) friend recommendations by combining FriendLink algorithm [6] with the geographical distance between users. Moreover, our tensor method includes an incremental stage, where newly created data is inserted into the tensor by incremental solutions. [9, 3].

### 3. GEOSOCIAL SYSTEM DESCRIPTION

Our GeoSocial system consists of several components. The system’s architecture is illustrated in Figure 1, where three main sub-systems are described: (i) the Web Site, (ii) the Database Profiles and (iii) the Recommendation Engine. In the following sections, we describe each sub-system of GeoSocial in detail.

#### 3.1 GeoSocial Web Site

The GeoSocial system uses a web site <sup>2</sup> to interact with the users. The web site consists of four sub-systems: (i) the friend recommendation, (ii) the location recommendation, (iii) the activity recommendation and (iv) the check-in system. The friend recommendation sub-system is responsible for evaluating incoming data from the Recommendation Engine of GeoSocial and providing updated friend recommen-




<sup>2</sup><http://delab.csd.auth.gr/geosocial>

dations. In order to provide such recommendations, the web site sub-system implements the FriendLink algorithm [6] and also considers the geographical distance between users and “check in” points. The same applies to the location and activity recommendation sub-systems where new and updated location and activity recommendations are presented to the user as new “check-ins” are stored in the Database profiles. Finally, the check in system is responsible for passing the data inserted by the users to the respective Database profiles.


Figure 2a shows a location recommendation while Figure 2b depicts an activity recommendation. As shown in Figure 2a, the user selects an activity that he would like to perform, in this case working, and the system provides location recommendations where he could perform his selected activity, in this case either Starbucks or the Auth Library. As shown in Figure 2b, the user selects a nearby location, i.e. Auth Library and the system provides activities that he could perform. In this case the user’s location is near the Auth Library and the system proposes clubbing at the “Trendy bar” or the “Picadilly” as possible activities.

Figure 3a presents a scenario where the GeoSocial system recommends 4 possible friends to a target user. As shown, the first table includes 3 users, namely Maria Kontaki, Nikos Dimokas and Panagiotis Symeonidis, who are connected to the target user via 2-hop paths. The results are ordered based on the second to last column of the table, which indicates the number of common friends that the target user shares with each possible friend. As shown in Figure 3a, Maria Kontaki is the top recommendation because she shares 3 common friends with the target user. The common friends are then presented in the last column

We recommend the following users as possible 2-hop friends!

Name	Last Name	E-mail	Add as a friend	Picture	Number of common friends	Names of common friends
Maria	Kontaki	kontaki@csd.auth.gr	<input type="button" value="Add"/>		3	Dimitris Katsaros Yannis Manolopoulos Efi Tsamoura
Nikos	Dimokas	dimokas@csd.auth.gr	<input type="button" value="Add"/>		2	Yannis Manolopoulos Efi Tsamoura
Panagiotis	Symeonidis	symeon@csd.auth.gr	<input type="button" value="Add"/>		1	Dimitris Katsaros

We recommend the following users as possible 3-hop friends!

Name	Last Name	E-mail	Add as a friend	Picture	Number of pairs of users	Paths
Tasos	Gounaris	tasos@csd.auth.gr	<input type="button" value="Add"/>		1	Yannis Manolopoulos -- Nikos Dimokas -- Tasos Gounaris

(a)

Checkins Table

nodeID	placeID	ratingID	activityID	date	comment	cityID
35	1	1	1	2011-05-03 11:20:04	great coffee!	2
35	1	2	5	2011-05-03 11:20:09	not too bad!	2
27	1	1	1	2011-05-03 11:31:03	super run!	2
27	5	3	5	2011-05-03 11:31:08	the food was awful!	2
27	7	1	6	2011-05-04 10:02:45	could not be happier with it!	2

Activities Table

actID	activityName
1	dancing
2	food
3	drinks

Places Table

placeID	placeName	placeX	placeY	placeAddress	poBox
1	Starbucks	22.952090	40.631750	Egnatias 123	54624
2	Goody's	22.936381	40.639181	Dodekanisou 7	54626
3	McDonalds	23.733690	37.975740	Ermou 2	10563
4	Hard Rock Cafe	23.733640	37.973150	Filelimn 18	10557

Users Table

nodeID	nodename	password	email	lastname
1	Alex	alex	apapadi@csd.auth.gr	Papadimitriou
2	Nikos	nikos	dimokas@csd.auth.gr	Dimokas
3	Maria	maria	kontaki@csd.auth.gr	Kontaki

(b)

Figure 3: (a) Friend recommendations provided by the GeoSocial system. (b) Database check-in profile.

of the table. The second table contains one user, namely Tasos Gounaris, who is connected to the target user via a 3-hop path. The last column of the table indicates the number of pairs that connect the target user with the possible friend. As shown in Figure 3a, Tasos Gounaris is connected to the target user via 1 3-hop path. This path is presented in the last column of the table.

### 3.2 GeoSocial Database Profiles

The database that supports the GeoSocial system is a MySQL(v.5.5.8)<sup>3</sup> database. MySQL is an established Database Management System (DBMS), which is widely used in on-line, dynamic, database driven websites.

The database profile sub-system contains five profiles where data about the users, locations, activities and their corresponding ratings is stored. As shown in Figure 3b, this data is received by the Check-In profile and along with the Friendship profile, they provide the input for the Recommendation Engine sub-system.

The collected data is stored in a database table, called “checkins”, which is shown in Figure 3b. Each table field represents the respective data that is collected by the Check-In profile. NodeID, placeID and activityID refer to specific IDs given to users, locations and activities respectively. We also store information about the time of the check-in inside the date field and an optional user comment inside the comment field.

### 3.3 GeoSocial Recommendation Engine

The recommendation engine is responsible for collecting the data from the database and producing the recommendations which will then be displayed on the web site. As shown in Figure 1, the recommendation engine constructs a friends similarity matrix by implementing the FriendLink algorithm

<sup>3</sup><http://www.mysql.com>

proposed in [6]. The average geographical distances between users’ “check-ins” are used as link weights. To obtain the weights, we calculate the average distance between all pairs of POIs that two users have checked-in. The recommendation engine also produces a dynamically analyzed 3-order tensor, which is firstly constructed by the HOSVD algorithm and is then updated using incremental methods [9, 3], both of which are explained in later sections.

## 4. THE FRIENDLINK ALGORITHM

In this section, we describe our FriendLink [6] algorithm, which can be used for the task of friend recommendations. Here, we describe how FriendLink is applied on GSNs and how the recommendation of friends is performed according to the detected associations.

When using an GSN, users explicitly declare their friends so that they are able to share information location with them (i.e. photos etc.) After some time, the geo-social network accumulates a set of connection data (graph of friendships), which can be represented by an undirected graph.

Our method assumes that persons in an GSN can use all the pathways connecting them, proportionally to the pathway lengths. Thus, two persons who are connected with many unique pathways have a high possibility to know each, proportionally to the length of the pathways they are connected with.

**Definition 1.** The similarity  $sim(v_x, v_y)$  between two graph nodes  $v_x$  and  $v_y$  is defined as the counts of paths of varying length  $\ell$  from  $v_x$  to  $v_y$ :

$$sim(v_x, v_y) = \sum_{i=2}^{\ell} \frac{1}{i-1} \cdot \frac{|paths_{v_x, v_y}^i|}{\prod_{j=2}^i (n-j)} \quad (1)$$

where

- $n$  is the number of vertices in a graph  $G$ ,
  - $\ell$  is the maximum length of a path between the graph nodes  $v_x$  and  $v_y$  (excluding paths with cycles). By the term “paths with cycles” we mean that a path can not be closed (cyclic). Thus, a node can exist only one time in a path (e.g. path  $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_1 \rightarrow v_5$  is not acceptable because  $v_1$  is traversed twice),
  - $\frac{1}{i-1}$  is an “attenuation” factor that weights paths according to their length  $\ell$ . Thus, a 2-step path measures the non-attenuation of a link with value equals to 1 ( $\frac{1}{2-1} = 1$ ). A 3-step path measures the attenuation of a link with value equals to  $\frac{1}{2}$  ( $\frac{1}{3-1} = \frac{1}{2}$ ) etc. In this sense, we use appropriate weights to allow the lower effectiveness of longer path chains.
  - $|paths_{v_x, v_y}^\ell|$  is the count of all length- $\ell$  paths from  $v_x$  to  $v_y$ ,
  - $\prod_{j=2}^i (n-j)$  is the count of all possible length- $\ell$  paths from  $v_x$  to  $v_y$ , if each vertex in graph  $G$  was linked with all other vertices. By using the fraction  $\frac{|paths_{v_x, v_y}^\ell|}{\prod_{j=2}^i (n-j)}$ ,
- our similarity measure is normalized and takes values in  $[0,1]$ . If two nodes are similar we expect the value  $sim(v_x, v_y)$  to be close to 1. On the other hand, if the two nodes are dissimilar, we expect the value  $sim(v_i, v_j)$  to be close to 0.

Our FriendLink approach finds similarities between nodes in an undirected graph constructed from these connection data. The FriendLink algorithm uses as input the connections of a graph  $\mathcal{G}$  and outputs a similarity matrix between any two nodes in  $\mathcal{G}$ . Therefore, friends can be recommended to a target user  $u$  according to their weights in the similarity matrix.

Friendlink computes node similarity between any two nodes in a graph  $\mathcal{G}$ . The initial input of Friendlink is the number  $n$  of nodes of  $\mathcal{G}$ , the adjacency matrix  $A$ , and the length  $\ell$  of paths that will be explored in  $\mathcal{G}$ . To enumerate all simple paths in  $\mathcal{G}$ , Rubin’s algorithm [8] can be employed. However, Rubin’s algorithm uses  $O(n^3)$  matrix operations to find all paths of different length between any pair of nodes, where  $n$  is the number of nodes in  $\mathcal{G}$ . In the following, we customize Rubin’s algorithm to create only paths of length up to  $\ell$  for our purpose.

As shown in Figure 4, FriendLink consists of a main program and two functions. In the main program, we modify the adjacency matrix so instead of holding 0/1 values, the  $(i, j)$  entry of the matrix  $A$  is a list of paths from  $i$  to  $j$ . Then, in the function Combine Paths(), we perform the matrix multiplication algorithm. However, instead of multiplying and adding entries, we concatenate pairs of paths together. Finally, in the function Compute Similarity(), we update the similarity between nodes  $i$  and  $j$ , for each length- $\ell$  path we find, where  $i$  is the start node and  $j$  is the destination node (i.e all paths of length  $[\dots, \ell]$ ). For the update of the similarity value between nodes  $i$  and  $j$  we use Equation 1. Notice that, we do not take into account cyclic paths in our similarity measure.

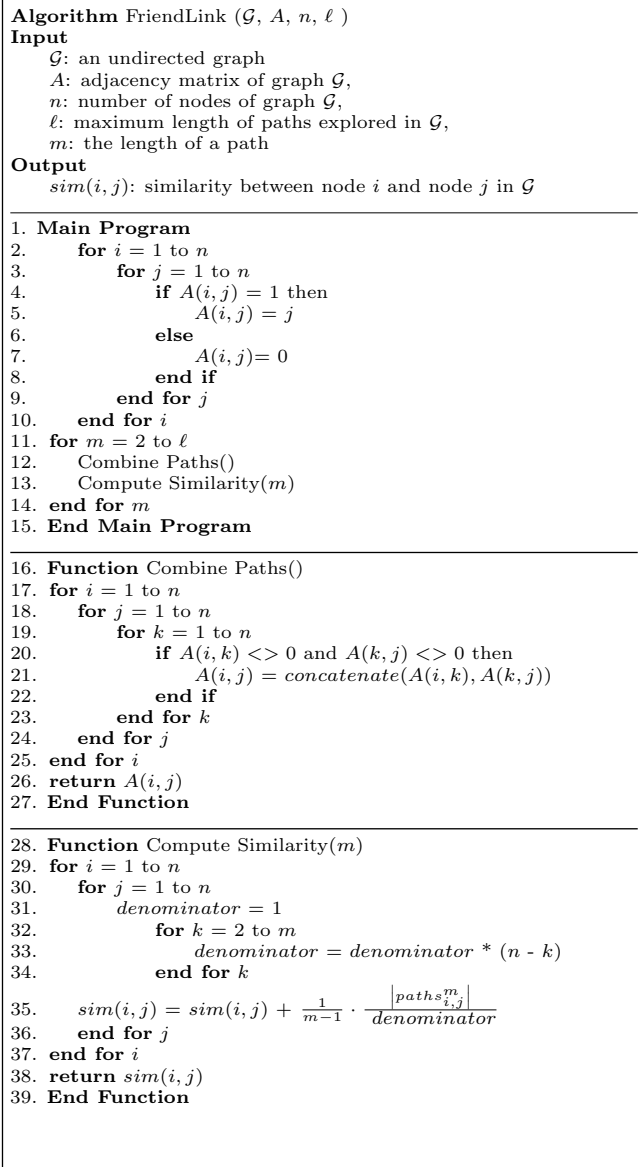


Figure 4: The FriendLink algorithm.

## 5. OUR INCREMENTAL TENSOR REDUCTION APPROACH

In this section we provide details on how HOSVD is applied on tensors and how location/activity recommendation is performed based on the detected latent associations.

Our Tensor Reduction approach initially constructs a tensor, based on usage data triplets  $\{u, l, a\}$  of users, location and activity. The motivation is to use all three objects that interact inside a location-based social network. Consequently, we proceed to the unfolding of  $\mathcal{A}$ , where we build three new matrices. Then, we apply SVD in each new matrix. Finally, we build the core tensor  $\mathcal{S}$  and the resulting tensor  $\hat{\mathcal{A}}$ . The 6 steps of the proposed approach are summarized as follows:

- *Step 1:* The initial tensor  $\mathcal{A}$  construction, which is based on usage data triplets (user, location, activity).

- *Step 2:* The matrix unfoldings of tensor  $\mathcal{A}$ , where we matricize the tensor in all three modes, creating three new matrices (one for each mode).
- *Step 3:* The application of SVD in all three new matrices, where we keep the  $c$ -most important singular values for each matrix.
- *Step 4:* The construction of the core tensor  $\mathcal{S}$ , that reduces the dimensionality.
- *Step 5:* The construction of the  $\hat{\mathcal{A}}$  tensor, that is an approximation of tensor  $\mathcal{A}$ .
- *Step 6:* Based on the weights of the elements of the reconstructed tensor  $\hat{\mathcal{A}}$ , we recommend location/activity to the target user  $u$ .

Steps 1 – 5 build a model and can be performed off-line. The recommendation in Step 5 is performed on-line, i.e., each time we have to recommend a location/activity to a user, based on the built model. In the following, we provide more details on each step.

## 5.1 The initial construction of tensor $\mathcal{A}$

From the usage data triplets (user, location, activity), we construct an initial 3-order tensor  $\mathcal{A} \in R^{I_u \times I_l \times I_a}$ , where  $I_u, I_l, I_a$  are the numbers of users, locations and activities, respectively. Each tensor element measures the number of times that a user  $u$  checked in a location  $l$  and made an activity  $a$ .

## 5.2 Matrix unfolding of tensor $\mathcal{A}$

As described, a tensor  $\mathcal{A}$  can be unfolded (matricized), i.e., we build matrix representations of tensor  $\mathcal{A}$  in which all the column (row) vectors are stacked one after the other. The initial tensor  $\mathcal{A}$  is matricized in all three modes. Thus, after the unfolding of tensor  $\mathcal{A}$  for all three modes, we create 3 new matrices  $A_1, A_2, A_3$ , as follows:

$$A_1 \in R^{I_u \times I_l I_a},$$

$$A_2 \in R^{I_l \times I_u I_a},$$

$$A_3 \in R^{I_a \times I_u I_l}$$

## 5.3 Application of SVD on each matrix

We apply SVD on the three matrix unfoldings  $A_1, A_2, A_3$ . We result, in total, to 9 new matrices.

$$A_1 = U^{(1)} \cdot S_1 \cdot V_1^T \quad (2)$$

$$A_2 = U^{(2)} \cdot S_2 \cdot V_2^T \quad (3)$$

$$A_3 = U^{(3)} \cdot S_3 \cdot V_3^T \quad (4)$$

For Tensor Dimensionality Reduction, there are three dimensional parameters to be determined. The numbers  $c_1, c_2$

and  $c_3$  of left singular vectors of matrices  $U^{(1)}, U^{(2)}, U^{(3)}$ , respectively, that are preserved. They will determine the final dimensionality of the core tensor  $\mathcal{S}$ . Since each of the three diagonal singular matrices  $S_1, S_2$  and  $S_3$  are calculated by applying SVD on matrices  $A_1, A_2$  and  $A_3$ , respectively, we use different  $c_1, c_2$  and  $c_3$  numbers of principal components for each matrix  $U^{(1)}, U^{(2)}, U^{(3)}$ . The numbers  $c_1, c_2$  and  $c_3$  of singular vectors are chosen by preserving a percentage of information of the original  $S_1, S_2, S_3$  matrices after appropriate tuning (the default percentage is set to 50% of the original matrix).

## 5.4 The construction of the core tensor $\mathcal{S}$

The core tensor  $\mathcal{S}$  governs the interactions among users, locations and activities. Since we have selected the dimensions of  $U^{(1)}, U^{(2)}$  and  $U^{(3)}$  matrices, we proceed to the construction of  $\mathcal{S}$ , as follows:

$$\mathcal{S} = \mathcal{A} \times_1 U_{c_1}^{(1)T} \times_2 U_{c_2}^{(2)T} \times_3 U_{c_3}^{(3)T}, \quad (5)$$

where  $\mathcal{A}$  is the initial tensor,  $U_{c_1}^{(1)T}$  is the tranpose of the  $c_1$ -dimensionally reduced  $U^{(1)}$  matrix,  $U_{c_2}^{(2)T}$  is the tranpose of the  $c_2$ -dimensionally reduced  $U^{(2)}$  matrix,  $U_{c_3}^{(3)T}$  is the tranpose of the  $c_3$ -dimensionally reduced  $U^{(3)}$  matrix.

## 5.5 The construction of tensor $\hat{\mathcal{A}}$

Finally, tensor  $\hat{\mathcal{A}}$  is build as the product of the core tensor  $\mathcal{S}$  and the mode products of the three matrices  $U^{(1)}, U^{(2)}$  and  $U^{(3)}$  as follows:

$$\hat{\mathcal{A}} = \mathcal{S} \times_1 U_{c_1}^{(1)} \times_2 U_{c_2}^{(2)} \times_3 U_{c_3}^{(3)}, \quad (6)$$

$\mathcal{S}$  is the reduced core tensor,  $U_{c_1}^{(1)}$  is the  $c_1$ -dimensionally reduced  $U^{(1)}$  matrix,  $U_{c_2}^{(2)}$  is the  $c_2$ -dimensionally reduced  $U^{(2)}$  matrix,  $U_{c_3}^{(3)}$  is the  $c_3$ -dimensionally reduced  $U^{(3)}$  matrix.

## 5.6 The generation of the location/activity Recommendation list

Tensor  $\hat{\mathcal{A}}$  measures the associations among the users, locations and activities and acts as a model that is used during the recommendation.

Each element of  $\hat{\mathcal{A}}$  represents a quadruplet  $\{u, l, a, p\}$ , where  $p$  is the likeliness that user  $u$  will visit location  $l$  and perform activity  $a$ . Therefore, locations/activities can be recommended to  $u$  according to their weights associated with  $\{u, a\}$  and  $\{u, l\}$  pairs, respectively. If we want to recommend to  $u$   $N$  activities for location  $l$ , then we select the  $N$  corresponding activities with the highest weights.

## 5.7 Inserting new users, locations, or activities over time

As new users, locations, or activities are being introduced to the system, the  $\hat{\mathcal{A}}$  tensor, which provides the recommendations, has to be updated. The most demanding operation for this task is the updating of the SVD of the corresponding unfoldings. We can avoid the costly batch recomputation of the corresponding SVD, by considering incremental solutions [9, 3]. Depending on the size of the update (i.e.,

number of new users, locations, or activities), different techniques have been followed in related research. For small update sizes we can consider the *folding-in* technique [9], whereas for larger update sizes we can consider Incremental SVD techniques [3].

### 5.7.1 Update by Incremental SVD

Folding-in incrementally updates SVD, but the resulting model is not a perfect SVD model, because the space is not orthogonal [9]. When the update size is not big, loss of orthogonality may not be a severe problem in practice. Nevertheless, for larger update sizes the loss of orthogonality may result to an inaccurate SVD model. In this case we need to incrementally update SVD so as to ensure orthogonality. This can be attained in several ways. Next we describe how to use the approach proposed by Brand [3].

Let  $M_{p \times q}$  be a matrix, upon which we apply SVD and maintain the first  $r$  singular values, i.e.,

$$M_{p \times q} = U_{p \times r} S_{r \times r} V_{r \times q}^T$$

Assume that each column of matrix  $C_{p \times c}$  contains the additional elements. Let  $L = U^T C = U^T C$  be the projection of  $C$  onto the orthogonal basis of  $U$ . Let also  $H = (I - UU^T)C = C - UL$  be the component of  $C$  orthogonal to the subspace spanned by  $U$  ( $I$  is the identity matrix). Finally, let  $J$  be an orthogonal basis of  $H$  and let  $K = J^T H = J^T H$  be the projection of  $C$  onto the subspace orthogonal to  $U$ . Consider the following identity:

$$\begin{aligned} [U \ J] \begin{bmatrix} S & L \\ 0 & K \end{bmatrix} \begin{bmatrix} V & 0 \\ 0 & I \end{bmatrix}^T &= \\ [U(I - UU^T)C/K] \begin{bmatrix} S & U^T C \\ 0 & K \end{bmatrix} \begin{bmatrix} V & 0 \\ 0 & I \end{bmatrix}^T &= \\ [USV^T \ C] &= [M \ C] \end{aligned}$$

Like an SVD, the left and right matrixes in the product are unitary and orthogonal. The middle matrix, denoted as  $Q$ , is diagonal. To incrementally update the SVD,  $Q$  must be diagonalized. If we apply SVD on  $Q$  we get:

$$Q = U' S' (V')^T$$

Additionally, define  $U'', S'', V''$  as follows:

$$U'' = [U \ J]U', \quad S'' = S', \quad V'' = \begin{bmatrix} V & 0 \\ 0 & I \end{bmatrix} V'$$

Then, the updated SVD of matrix  $[M \ C]$  is:

$$[M \ C] = [USV^T \ C] = U'' S'' (V'')^T$$

This incremental update procedure takes  $O((p+q)r^2 + pc^2)$  time [3].

Returning to the application of incremental update for new users, locations, or activities, in each case we result with a number of new rows that are appended in the end of the unfolded matrix of the corresponding mode. Therefore, we need an incremental SVD procedure in the case where we add new rows, whereas the aforementioned method works in the case where we add new columns. In this case we simply swap  $U$  for  $V$  and  $U''$  for  $V''$ .

## 6. EXPERIMENTAL CONFIGURATION

In this section, we study the performance of our approach in terms of friend, location and activity recommendations. To evaluate the aforementioned recommendations we have chosen two real data sets. The first one, denoted as geosocial data set is extracted from our newly developed site. There are 1,173 triplets in the form user–location–activity. To these triplets correspond 102 users, 46 locations and 18 activities. The second data set, denoted as UCLAF<sup>4</sup> [12] data set consists of 164 users, 168 locations and 5 different types of activities, including “Food and Drink”, “Shopping”, “Movies and Shows”, “Sports and Exercise”, and “Tourism and Amusement”.

The numbers  $c_1$ ,  $c_2$ , and  $c_3$  of left singular vectors of matrixes  $U^{(1)}$ ,  $U^{(2)}$ ,  $U^{(3)}$  for our approach, after appropriate tuning, are set to 25, 12 and 8 for the geosocial dataset, and to 40, 35, 5 for the UCLAF data set. Due to lack of space we do not present experiments for the tuning of  $c_1$ ,  $c_2$ , and  $c_3$  parameters. The core tensor dimensions are fixed, based on the aforementioned  $c_1$ ,  $c_2$ , and  $c_3$  values.

### 6.1 Evaluation Metrics

We perform 4-fold cross validation and the default size of the training set is 75% – we pick, for each user, 75% of his check-ins and friends randomly. The task of all three recommendation types (i.e. friend, location, activity) is to predict the friends/locations/activities of the user’s 25% remaining check-ins and friends, respectively. As performance measures we use precision and recall, which are standard in such scenarios. For a test user that receives a list of  $N$  recommended friends/locations/activities (top- $N$  list), the following are defined:

- **Precision**: is the ratio of the number of relevant friends/locations/activities in the top- $N$  list relative to  $N$ .
- **Recall**: is the ratio of the number of relevant friends/locations/activities in the top- $N$  list relative to the total number of relevant friends/locations/activities, respectively.

### 6.2 Comparison Results

In this section, we study the accuracy performance of our method in terms of precision and recall. This reveals the robustness of our method in attaining high recall with minimal losses in terms of precision. We examine the top- $N$  ranked list, which is recommended to a test user, starting from the top friend/location/activity. In this situation, the recall and precision vary as we proceed with the examination of the top- $N$  list. In Figure 5, we plot a precision versus recall curve. As it can be seen, our ITR approach presents high accuracy. The reason is that we exploit altogether the information that concerns the three entities (friends, locations, and activities) and thus, we are able to provide accurate location/activity recommendations. Notice that activity recommendations are more accurate than location recommendations. A possible explanation could be the fact that the number of locations is bigger than the number of activities. That is, it is easier to predict accurately an activity than a location. Notice that for the task of friend recommendation, the performance of Friendlink is not so high. The main reason is data sparsity. In particular, the friendship network has average nodes’ degree equal to 2.7 and average shortest

<sup>4</sup><http://www.cse.ust.hk/vincentz/aaai10.uclaf.data.mat>

distance between nodes 4.7, which means that the friendship network can not be consider as a “small world” network and friend recommendations can not be so accurate.

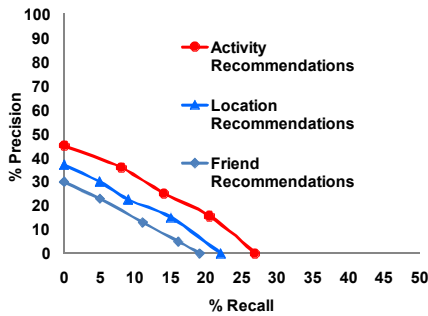


Figure 5: Precision Recall diagram of ITR and FriendLink for activity, location and friend recommendations on the Geosocial data set

For the UCLAF data set, as shown in Figure 6, our ITR algorithm attains analogous results. Notice that the recall for the activity recommendations, reaches 100% because the total number of activities is 5. Moreover, notice that in this diagram, we do not present results for the friend recommendation task, since there is no friendship network in the corresponding UCLAF data set.

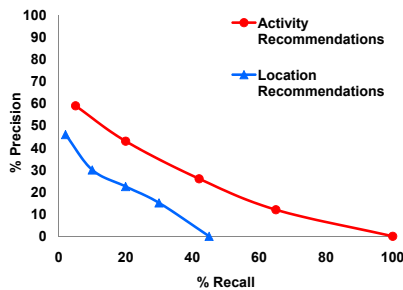


Figure 6: Precision Recall diagram of ITR for activity and location recommendations on the UCLAF data set

## 7. CONCLUSION AND FUTURE WORK

In this paper we have proposed a Geo-social recommender system, which is capable of recommending friends, locations and activities. For the location/activity recommendation task, we used a tensor, which is updated by incremental tensor approaches, as new users, locations, or activities are being inserted into the system. For the friend recommendation task, we apply the FriendLink algorithm, which performs a local path traversal on the friendship network.

As future work, we plan on conducting a user study concerning the recommendations in our Geo-social web site to measure user satisfaction. We are also planning on comparing our method to other state-of-the-art methods in terms of effectiveness and efficiency. Moreover, we want to extend our Geo-social recommender system by taking also into account the geographical influence and semantics. That is, the geographical proximities of locations and their semantics could play a determinant role on users’ check-in behavior.

## References

- [1] L. Backstrom, E. Sun, and C. Marlow. Find me if you can: improving geographical prediction with social and spatial proximity. In *WWW '10: Proceedings of the 19th international conference on World Wide Web*, pages 61–70, New York, NY, USA, 2010. ACM.
- [2] C. Biancalana, F. Gasparetti, A. Micarelli, and G. Sansonetti. Social tagging for personalized location-based services. 2011.
- [3] M. Brand. Incremental singular value decomposition of uncertain data with missing values. In *European Conference on Computer Vision (ECCV2002)*, 2002.
- [4] Economist. A world of connections: A special report on networking. *The Economist: Editorial Team*, 2010.
- [5] K. W.-T. Leung, D. L. Lee, and W.-C. Lee. Clr: a collaborative location recommendation framework based on co-clustering. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information, SIGIR '11*, pages 305–314, New York, NY, USA, 2011. ACM.
- [6] A. Papadimitriou, P. Symeonidis, and Y. Manolopoulos. Friendlink: Predicting links in social networks via bounded local path traversal. In *Proceedings of the 3rd Conference on Computational Aspects of Social Networks (CASON'2011)*, Salamanca, Spain (to appear), 2011.
- [7] D. Quercia, N. Lathia, F. Calabrese, G. Di Lorenzo, and J. Crowcroft. Recommending Social Events from Mobile Phone Location Data. In *Proceedings of IEEE ICDM 2010*, Dec. 2010.
- [8] F. Rubin. Enumerating all simple paths in a graph. *IEEE Transactions on Circuits and Systems*, 25(8):641–642, 1978.
- [9] B. Sarwar, J. Konstan, and J. Riedl. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *International Conference on Computer and Information Science*, 2002.
- [10] S. Scellato, C. Mascolo, M. Musolesi, and V. Latora. Distance Matters: Geo-social Metrics for Online Social Networks. In *Proceedings of the 3rd Workshop on Online Social Networks (WOSN 2010)*, June 2010.
- [11] M. Ye, P. Yin, W.-C. Lee, and D.-L. Lee. Exploiting geographical influence for collaborative point-of-interest recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information, SIGIR '11*, pages 325–334, New York, NY, USA, 2011. ACM.
- [12] V. W. Zheng, B. Cao, Y. Zheng, X. Xie, and Q. Yang. Collaborative filtering meets mobile recommendation: A user-centered approach. *AAAI'10*, pages 236–241, 2010.
- [13] W. Zheng, Y. Zheng, X. Xie, and Q. Yang. Collaborative location and activity recommendations with gps history data. In *WWW '10: Proceedings of the 19th international conference on World Wide Web*, pages 1029–1038, New York, NY, USA, 2010. ACM.