Recommending friends and locations over a Heterogeneous Spatio-Temporal graph

Pavlos Kefalas and Panagiotis Symeonidis

Department of Informatics, Aristotle University
of Thessaloniki, 54124, Greece
{kefalasp,symeon}@csd.auth.gr
http://delab.csd.auth.gr

Abstract. Recommender systems in location-based social networks (LBSNs), such as Facebook Places and Foursquare, have focused on recommending friends or locations to registered users by combining information derived from *explicit* (i.e. friendship network) and *implicit* (i.e. user-item rating network, user-location network, etc.) sub-networks. However, previous's work models were static, failing to capture adequately user preferences as they change over time. In this paper, we provide a novel recommendation method by incorporating the time dimension into our model through an auxiliary artificial node (i.e. session). In particular, we construct a hybrid tripartite (i.e., user, location, session) graph, which incorporates 7 different unipartite and bipartite graphs. Then, we run on it the well known Random Walk with Restart (RWR) algorithm, which randomly propagate through the network structure which has 7 differently weighted edge types (i.e., user-location, user-session, user-user, etc.) among its entities. We evaluate experimentally how RWR improve the procession of the recommendations during different time-windows against one state-of-the-art algorithm over the GeoSocialRec and the Foursquare datasets.

Keywords: Algorithms, link prediction, location recommendation, social networks

1 Introduction

Users utilize location-based social networks (LBSNs) to share their location with their friends, by incorporating in their posts the longitude and latitude information of their location. In LBSNs, users *explicitly* build a friendship network by adding each other as friends. In addition, users form *implicit* sub-networks through their daily interactions, like commenting on same posts or rating similarly same products/services in places they have co-visited.

Previous works have focused on recommending friends or locations [5, 11] to users by combining information derived from multi-modal and heterogeneous explicit and implicit networks. In particular, there has been extensive research in this area, which mainly focuses on information derived from users' interaction

with locations over user-location bipartite network ties. However, such models are static, failing to capture adequately users' preferences as they change over time. That is, time is an important factor in LBNSs, which affects the accuracy of recommendations. For example, users periodically perform daily activities in specific locations (e.g. home, work, etc.).

To incorporate the time dimension into their model, Xiang et al. [12] proposed the construction of tripartite graphs (i.e., users, locations, sessions) known as *Session-based Temporal Graph (STG)*. But, STG graph do not incorporate edges among nodes of the same set, i.e. failing to exploit information from all three unipartite networks (user-user, location-location and session-session). For instance, STG do not have links among user nodes. But, intuitively friends tend to visit similar locations at close time points, which means that friendship links could leverage the accuracy of location recommendations. A second problem of STG stems from their own structure. That is, STG do not connect directly users either with locations or sessions, which results to lower recommendation accuracy when data (i.e., session/location nodes) are sparse.

In this paper, we provide recommendations based on a Heterogeneous Spatio-Temporal graph (HST graph) by incorporating time dimension into our model. To build this HST graph, we create a new type of an artificial node, denoted as session node, which is associated with the co-location of two or more users in a location at a specific time period. Our HST graph incorporates 7 different unipartite or bipartite graphs, which makes it more informative in comparison to STG. Moreover, we follow a star-schema structure, where users are directly connected with locations and sessions. This structure can be more resistant in cases of sparsity (e.g. when there are not enough session nodes as a result of the fact that users check-into locations at different time periods).

Based on our HST graph that incorporates user, location and session nodes, we run the well known Random Walk with Restart algorithm (RWR) to provide spatio-temporal recommendations. RWR has properties, which can adequately capture the notion of user-user similarity or the user-location relevance in our HST graph. That is, social drivers which influence the ties formation in communities like homophily, social influence, common friendship, etc. are incorporated by nature in RWR algorithm, as it will be shown later.

The contributions of this paper are summarized as follows: *i*) We propose the construction of HST graph, which is a tripartite graph that consists of 3-disjoint sets of nodes (i.e. sessions, users, locations), and incorporates edges among nodes of the same set, including also three unipartite graphs *ii*) We use the Random Walk with Restart algorithm (RWR) on this new graph to examine how spatial and temporal features can leverage the recommendations according to proximity and time distance. *iii*) We have compared our method with one state-of-the-art algorithm over two real world datasets.

The rest of this paper is organized as follows. Section 2 summarizes the related work, whereas Section 3 describes the construction of our HST graph, its edge weighting and our proposed algorithm. Experimental results are given in Section 4. Finally, Section 5 concludes this paper.

2 Related Work

Time is a crucial factor in LBSNs, since it could leverage the accuracy of friend, location and activity recommendations. Recently, Yuan et al. [14] exploited spatio-temporal characteristics of POIs by using a unified framework consisting of spatial and temporal dimensions.

Gao et al. [2] proposed the *Location Recommendation with Temporal effects* (LRT) algorithm. They argue that time dimension is crucial in recommendation and introduce a framework to make time-aware recommendations. In the same direction, a time-aware method was proposed by Marinho et al. [6] to improve location recommendations in LBSNs. Ho et al. [4] extract spatio-temporal information for future events from news articles. Furthermore, Raymond et al. [8] proposed a method to provide location recommendations for users that use buses. Their method is based on users' location histories and spatio-temporal correlations among the locations. By combining collaborative filtering algorithms with link propagation, they are able to predict origins, destinations and arrival times of buses.

The creation of artificial session nodes has been originally proposed by Xiang et al. [12], who designed a framework that models users' long-term and short-term preferences over time. Their model is based on a Session-based Temporal Graph (STG), which incorporates user, location and session information. In addition, Xiang et al. [12] proposed a novel recommendation algorithm named Injected Preference Fusion (IPF) and extended the personalized Random Walk for temporal recommendation.

Assume that, there are 2 users, 4 locations and 3 session nodes. User U1 has visited locations L1, L2 and L3, whereas user U2 has visited locations L3 and L4. Notice also that locations L1 and L2 are linked to Session 1 node. This means, both locations (L1 and L2) were co-visited by U1 at the same period T1 (e.g. during the morning of Thursday 19 September 2013). Based on the aforementioned graph, the user-location bipartite graph denotes the long term preferences of a user, whereas the location-session bipartite graph denotes the short term preferences of a user.

Our work is inspired by the work of Xiang et al. [12]. However, our HST graph has two main differences in comparison with STG. Firstly, in our case we create session nodes that connect users and not locations. That is, user nodes are the heart of a star schema graph and, thus, they are connected with direct links with both location and session nodes. The second difference is the graph structure per se. It is not only a 3-partite graph that consists of 3-disjoint sets of nodes (i.e. sessions, users, locations). In contrast, it incorporates also edges among nodes of the same set, i.e. three unipartite graphs, which makes it even richer in information.

3 Background and Preliminaries

In this section, we introduce the most important notations with the necessary definitions and a motivating example depicted in Figure 1. Also, we provide an

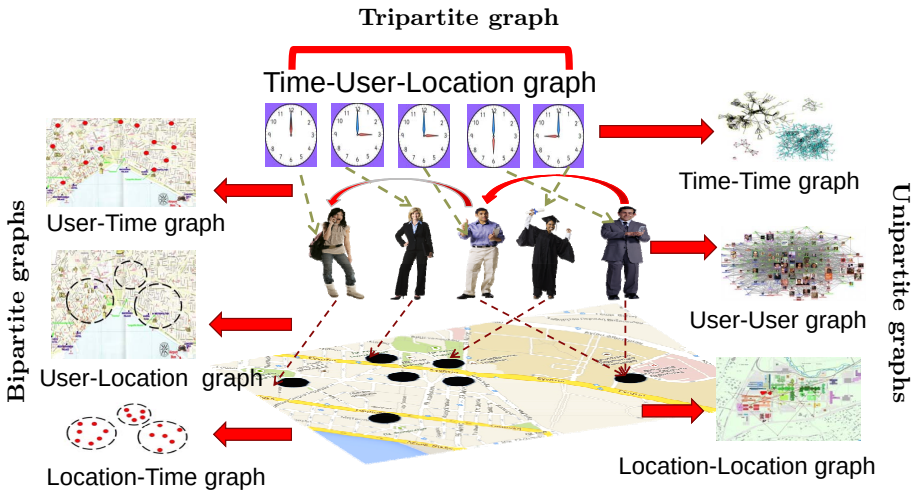

Fig. 1: Latent Relations among Time, Users, Locations dimensions of an LBSN and the generated k-partite graphs

analytical description of the basic entities that interact in a LBSN, i.e. users, locations, and time dimension and discuss the information types that can be extracted from the connections among them. Figure 1 shows the relations among the aforementioned entities. As shown in Figure 1, we have 3 layers (one layer for each entity) and 5 users who have visited some places. For each visit we keep the time of the user's check-in. As also shown in Figure 1, there are 7 graphs of different participating entities (i.e. three unipartite, three bipartite and one tripartite). On the right side of Figure 1, we can see the 3 generated unipartite graphs (Time-Time graph, User-User graph, and Location-Location graph). On the left side of Figure 1, we observe 3 bipartite graphs (User-Time graph, User-Location graph, and Location-Time graph). Finally, on top of Figure 1, we can see the tripartite graph (Time-User-Location graph).

It is necessary to emphasize that the above graph is not a k-partite graph, since there can exist also edges among nodes of the same set (e.g. an edge between a user and another user, i.e. friendship). We denote this special case of a graph henceforth, as hybrid k-partite graph because it is a k-partite graph that consists of k-disjoint sets of nodes (i.e. time, users, locations), incorporating edges among nodes of the same set as well.

As depicted in Figure 1, our data are in the form of $\langle time, user, location \rangle$ triplets, which are usually modeled by a tripartite graph or a tensor. However, if we had to use a tripartite graph or a tensor for capturing the time dimension as well, then we should create a new node for each different timestamp. This would create a huge tensor or a temporal graph with an enormous number of time nodes creating severe noise in the model.

Based on the above considerations, we choose to create a new type of an artificial node, denoted as session node, which is associated with the co-location of two or more users in a location at a specific time period. This co-location of two or more users reflects their interest for a place at a specific time. For example, two users can visit a music band at a bar every Thursday night. Thus, the possibility of having both common music interests is very high. That is, two users who visit a location at a common time period have a higher possibility to become friends than those users who visit a location but not on the same timestamp.

To create a new session node, in the same direction as [12], we transform the $\langle user, location, time \rangle$ into $\langle user, location \rangle$ and $\langle user, session \rangle$ by dividing the time into discrete intervals (bins). Then, we associate each bin with corresponding users who visited a place at the same time slot. Notice that a session node combines a number of users, with a location at a specific time interval. The length of a session can last from one hour, to six hours, or even one day etc. Based on the $\langle user, location \rangle$ and $\langle user, session \rangle$ we create our temporal graph, i.e. HST graph.

3.1 Session Node Extraction

Users may visit locations all day long. The huge amount of these check-ins, prevent us from understanding their trends and their likes, without before preprocessing the time dimension of this information. To have an abstract and thorough understanding of the users' behavior, we create artificial session nodes based on SQL statements. For our running example, let's assume that we create a table, which holds information about users, locations, and the time of their check-ins.

We extract the artificial session nodes, by using an SQL statement as shown in SQL Statement 1. This SQL statement finds co-locations between two or more users during the same time period, i.e. a session. In our running example, we set the time window for a co-location of two or more users equal to 6 hours. It is obvious that we can also use other lengths of a session's time window (i.e. we can split time into bins of an hour, a day, a month or a year, depending on the desired session for extraction).

Sql Statement 1 SQL query for session nodes extraction

```
SELECT A.userID, B.userID, A.Locationid
FROM ultime as A,ultime as B
WHERE A.LocationID = B.LocationID AND A.userID <> B.userID AND
(DATEDIFF(HOUR, A.tmp, B.tmp) / 24=0) AND (DATEDIFF(HOUR, A.tmp,
B.tmp) \% 24 between 0 and 6)
```

3.2 Constructing the Heterogeneous Spatio-Temporal Graph

We define a hybrid 3-partite graph as $\mathcal{G}(\mathcal{S},\mathcal{U},\mathcal{L},\mathcal{E}^{(\mathcal{US})},\mathcal{E}^{(\mathcal{SU})},\mathcal{E}^{(\mathcal{UL})},\mathcal{E}^{(\mathcal{LU})},\mathcal{E}^{(\mathcal{SS})},\mathcal{E}^{(\mathcal{UU})},\mathcal{E}^{(\mathcal{LL})})$, which consists of 3-disjoint sets of nodes ($\mathcal{S}$ for session, $\mathcal{U}$ for user, $\mathcal{L}$ for location). $\mathcal{G}$ is called "hybrid" because it has also edges among nodes of the same set. Similarly, there are edges among sessions and edges among locations. $\mathcal{E}^{(\mathcal{US})}$ represents the edges between nodes in $\mathcal{U}$ and $\mathcal{S}$. Vice versa, $\mathcal{E}^{(\mathcal{SU})}$ represents edges between nodes in $\mathcal{S}$ and $\mathcal{U}$. $\mathcal{E}^{(\mathcal{UL})}$ represents edges between nodes in $\mathcal{U}$ and L, whereas on the other hand $\mathcal{E}^{(\mathcal{LU})}$ represents edges between the nodes in $\mathcal{L}$ and U. $\mathcal{E}^{(\mathcal{SS})}$ represents the edge set linking the nodes in $\mathcal{S}$. $\mathcal{E}^{(\mathcal{UU})}$ represents the edge set linking the nodes in $\mathcal{U}$. Finally, $\mathcal{E}^{(\mathcal{LL})}$ represents the edge set linking the nodes in $\mathcal{L}$. For clarity, in Table 1 we provide a list of all used symbols notations and descriptions. We assume that graph $\mathcal{G}$ is directed and weighted. We also assume that graph $\mathcal{G}$ may have multiple edges connecting two nodes s and u.

Symbol	Description
S	Set of sessions, $S = \{s_1, s_2, ..., s_n\}$
S_u	Set of sessions a user participated
S_l	Set of sessions a location shown
s, s'	Some sessions
U	Set of users, $U = \{u_1, u_2, ..., u_n\}$
U_u	Set of users who are friends with user u
U_s	Set of users who participated in a session s
U_l	Set of users who visited a location l
u, u'	Some users
L	Set of locations, $L = \{l_1, l_2, ..., l_n\}$
L_u	Set of locations visited by a user u
l, l'	Some locations
$d_{l,l'}$	Distance between locations l and l'
$\mathcal{E}^{(\mathcal{US})}$	Set of edges linking nodes of U to nodes of S
$\mathcal{E}^{(\mathcal{SU})}$	Set of edges linking nodes of S to nodes of U
$\mathcal{E}^{(\mathcal{UL})}$	Set of edges linking nodes of U to nodes of L
$\mathcal{E}^{(\mathcal{LU})}$	Set of edges linking nodes of L to nodes of U
$\mathcal{E}^{(\mathcal{SS})}$	Set of edges linking the nodes of S
$\mathcal{E}^{(\mathcal{UU})}$	Set of edges linking the nodes of U
$\mathcal{E}^{(\mathcal{LL})}$	Set of edges linking the nodes of L

Table 1: Symbols notations and descriptions

3.3 Edge Weighting

In this section, we define the weights between nodes in our HST graph. By incorporating the artificial session nodes into our HST graph, we have the following 7 types of edges, which have to be weighted differently: *a*) an edge from a session node s to a user node u, *b*) an edge from a user u to a session s, *c*) an edge from a user u to a location l, *d*) an edge from location l to a user u, *e*) an edge from

a user u to another user u', *f*) an edge from a location l to another location l', and *g*) an edge from a session s to another session s'.

In the following, we define the edge weights for the 7 different edge types, starting from the edges of the bipartite graphs (session-user and user-location). Firstly, we set the weight $w(s,u)$ of the edge from a session node s to a user node u as:

$$w(s,u) = \frac{1}{|U_s|}, \tag{1}$$

where $(|U_s|)$ is the number of users who participated in a session s. Notice that we weight differently an edge that starts from a user u and ends to a session s. Specifically, $w(u,s)$ is:

$$w(u,s) = \frac{1}{|S_u|}, \tag{2}$$

where $|S_u|$ is the number of sessions in which a user u has participated. That is, the probability of a user to join a session is equally divided on all sessions he has participated.

Next, we define the edge weight $w(u,l)$ of the edge from a user node u to a location node l as:

$$w(u,l) = \frac{n_{u,l}}{\sum\limits_{\forall l \in L} n_{u,l}}, \tag{3}$$

where $n_{u,l}$ is the number of times a user u visited a location l and $\sum_{\forall l \in L} n_{u,l}$ is the total number of check-ins in all locations by user u. For define the edge weight $w(l,u)$, that starts from location l and ends at a user u as:

$$w(l,u) = \frac{n_{l,u}}{\sum\limits_{\forall u \in U} n_{l,u}}, \tag{4}$$

where the $n_{l,u}$ is the number of times a location l is visited by a user u and $\sum_{\forall u \in U} n_{l,u}$ is the total number of check-ins of all users in location l.

We proceed with the edge weighting of the unipartite graphs (user-user, location-location, session-session). First, the edge weight $w(u,u')$ between two user nodes u and u' is defined as the fraction of 1 to the number of users (U_u), who are friends with a user u:

$$w(u,u') = \frac{1}{|U_u|}, \tag{5}$$

The edge weight between two location nodes l and l' is defined as:

$$w(l,l') = \left(1 - \frac{(geodist_{l,l'})}{\sum\limits_{\forall l,l' \in L} (geodist_{l,l'})}\right), \tag{6}$$

In this case, we set as link weight the geographical distance between two location nodes l and l'. To obtain all weights, we calculate the distance between all pairs of locations.

Finally, for the edge weighting between two session nodes s and s', we take under consideration both the location and the time dimensions of each session nodes after normalizing both dimensions, by using the following equation:

$$w(s,s') = \left(1 - \frac{(geodist_{s,s'}}{\sum\limits_{\forall s,s' \in S} (geodist_{s,s'})}\right) + \left(1 - \frac{(timediff_{s,s'})}{\sum\limits_{\forall s,s' \in S} (timediff_{s,s'})}\right), \quad (7)$$

where $geodist_{s,s'}$ and $timediff_{s,s'}$ are the geographical distance and the time difference between two session nodes s and s', respectively.

3.4 Construction of the Transition Probability Matrix

Random walk processes on graphs have been extensively used in social network analysis [7, 13]. To apply a random walk on a heterogeneous spatio-temporal graph, we have to construct a transition probability P matrix to configure and set all transition probabilities among the nodes of our HST graph. To represent all possible transitions on the HST graph, the size of the P matrix should be $(|\mathcal{S}| + |\mathcal{U}| + |\mathcal{L}|) \times (|\mathcal{S}| + |\mathcal{U}| + |\mathcal{L}|)$. By combining Equations 1-7, we compute the transition probability matrix P which comprises of several sub-matrices that correspond to our HST graph, as follows:

$$\mathbf{P} = \begin{bmatrix} \mathcal{SS} & \mathcal{SU} & 0 \\ \mathcal{US} & \mathcal{UU} & \mathcal{UL} \\ 0 & \mathcal{LU} & \mathcal{LL} \end{bmatrix} \quad (8)$$

where $\mathcal{SS}$ is a $|\mathcal{S}| \times |\mathcal{S}|$ sub-matrix representing the transition probability between session nodes to session nodes, as defined in Equation 7. $\mathcal{UU}$ is a $|\mathcal{U}| \times |\mathcal{U}|$ sub-matrix, which is not symmetric because transition probabilities between two user nodes are defined based on the number of neighbors of each user node (see Equation 5). $\mathcal{LL}$ is a $|\mathcal{L}| \times |\mathcal{L}|$ sub-matrix representing the transition probability from location nodes to location nodes, as defined in Equation 6. $\mathcal{US}$ sub-matrix holds the transition probabilities from user nodes to session nodes, whereas $\mathcal{SU}$ sub-matrix holds the transition probabilities from session nodes to user nodes. Similarly, $\mathcal{UL}$ sub-matrix holds the transition probabilities from user nodes to location nodes, whereas $\mathcal{LU}$ sub-matrix holds the transition probabilities from location nodes to user nodes.

3.5 Normalization

In Section 3.3, we described the edge weighting among nodes of our HST graph in both unipartite and bipartite sub-networks. We aimed to assign weights on edges in the interval [0,1]. All these weights will be inserted in a probability

transition matrix P, and then we will run our method for capturing the notion of similarity between the nodes of the HST graph. However, in several cases the distribution of the weight values in the interval [0,1] between the 7 edge types (i.e. session-user, user-user, etc.) differs significantly. For example, consider the case that the most weights in $\mathcal{E}^{(\mathcal{US})}$ are normally distributed between 0 and 0.1, whereas most similarity values of $\mathcal{E}^{(\mathcal{LL})}$ are normally distributed between 0.9 and 1. That is, the weighting values of $\mathcal{E}^{(\mathcal{US})}$ will always be dominated by those of $\mathcal{E}^{(\mathcal{LL})}$.

To avoid this problem, we present a normalization step for the construction of the final transition probability P matrix: *a*) We compute the mean similarity value m_P of the matrix P. *b*) We compute the standard deviation value s_P of the matrix P. *c*) For each (i, j) cell of the P matrix, where $i \neq j$, we apply the transformation:

$$P(i,j) = \frac{P(i,j) - m_P}{s_P} \tag{9}$$

d) Finally, we scale and translate the derived values back in the interval [0,1]:

$$P(i,j) = \frac{P(i,j) - min_P}{max_P - min_P} \tag{10}$$

where min_P, max_P are the minimum and maximum values of matrix P after the transformation by Equation 9, respectively. Please notice, that by adding the probabilities of propagation through the nodes of a each column, we gain the maximum probability. Thus, after normalization step, each column of our transition probability matrix P cast up to 1.

3.6 Random Walk on the Normalized Transition Probability Matrix

Random walk with restart (RWR) algorithm [10] is a variation of the well-known PageRank algorithm. RWR has properties, which can adequately capture the notion of user-user similarity or the user-location relevance for a specific user u of our HST graph. The main advantage of RWR over PageRank is its teleporting characteristic, which obliges the random walker to re-start his walk from the initial node u. As expected, RWR assigns more importance/similarity to the nearby nodes of u. Thus, if two users are close to each other, the probability of becoming friends is larger. Moreover, RWR can capture the notion of similarity among users who share a large number of common friends. For the user-location graph, if two users visit the same locations, then the overall probability for connecting them (via a location node) increases. The same holds for two users via a session node.

RWR considers one random walker starting from an initial user node u and randomly choosing among the available edges with a probability α. In addition, each time the random walker may return back to the initial node with a probability $1 - \alpha$. Therefore, the random walk process can be represented as:

$$S^{(UU)}(t+1) = \alpha \times P \times S^{(UU)}(t) + (1-\alpha) \times I \tag{11}$$

where $S^{(UU)}(t)$ and $S^{(UU)}(t+1)$ are the state probability matrices at time t and $t+1$, respectively. $S^{(UU)}$ is a matrix that represents the link relevance from all HST graph nodes to the target user u. Parameter a is the prior probability that the random walker will leave its current state. Moreover, I is the identity and P the transition-probability matrix.

4 Experimental Evaluation

In this section, we compare experimentally RWR [10] with a fast version of the classic Katz algorithm, denoted as Fast-Katz [1] The parameters used to evaluate the performance of this algorithm are identical to those reported in the original paper. However, for datasets that were not used in these papers, we tuned the parameters so as to get the best results possible.

4.1 Data Sets

We performed our experiments using two real-world datasets, i.e., Foursquare[1] and GeoSocialRec[2]. Foursquare [3] dataset contains 18,107 users 2,073,740 check-ins, 847,081 locations and 231,148 social ties among users. This dataset is collected between March 2010 and January 2011. Please notice that we did not use the dataset of our main competitor [15] because it does not incorporate the friendship network. GeoSocialRec [9] dataset concerns 149 users who have 595 social ties among them (i.e. friendship network). Also, they have performed 853 check-ins to 438 locations. This dataset is collected between August 2011 and January 2012.

Detailed information about both networks is illustrated in Table 2. In particular, information about friendship networks can be seen in Table 2(a), where we present the type of each network (i.e. directed or undirected), the number of users, the number of links among users, the nodes' Average Degree (ADG) and the Local Clustering Coefficient (LLC). As expected, the sparsity of the user-user matrix is very big, i.e., 97.31% and 99.92% for the GeoSocialRec and the Foursquare datasets, respectively.

Furthermore, Table 2(b) contains information about the bipartite user-location network. In this table, we present the number of users, the number of locations, and the number of check-ins. Moreover, parameter AVG_u denotes the average number of check-ins per user, whereas parameter AVG_l denotes the average number of check-ins per location. Please notice that the average number of check-ins per user is 11.08 and 101.00 for the GeoSocialRec and the Foursquare dataset, respectively. This is a huge difference in terms of density between the two datasets. It is inevitable that the accuracy of recommendations for the GeoSocialRec data set will be low for both algorithms.

In Figure 2 we show statistics on the GeoSocialRec and the Foursquare datasets. Notice that both x-axis and y-axis are in the log scale. As shown,

[1] `http://www.public.asu.edu/~hgao16/dataset.html`
[2] `http://delab.csd.auth.gr/~symeon/GeoSocialRec_Dataset.rar`

(a) Friendship Network

Dataset	Type	Users	Edges	ADG	LCC
GeoSocialRec	undirected	149	595	6.3013	0.0342
Foursquare	undirected	18107	231148	10.5800	0.1841

(b) User-Location Network

Dataset	User	Location	Check-ins	AVG_u	AVG_l
GeoSocialRec	149	438	853	11.08	2.08
Foursquare	18107	847081	2073740	101	48.16

(c) User-Location-Session Network

			Session nodes				
Dataset	Users	POIs	3 Hour	6 Hour	9 Hour	12 Hour	24 Hours
GeoSocialRec	149	438	16	27	35	35	47
Foursquare	18107	847081	36606	78402	89079	90012	93204

Table 2: Datasets specifications

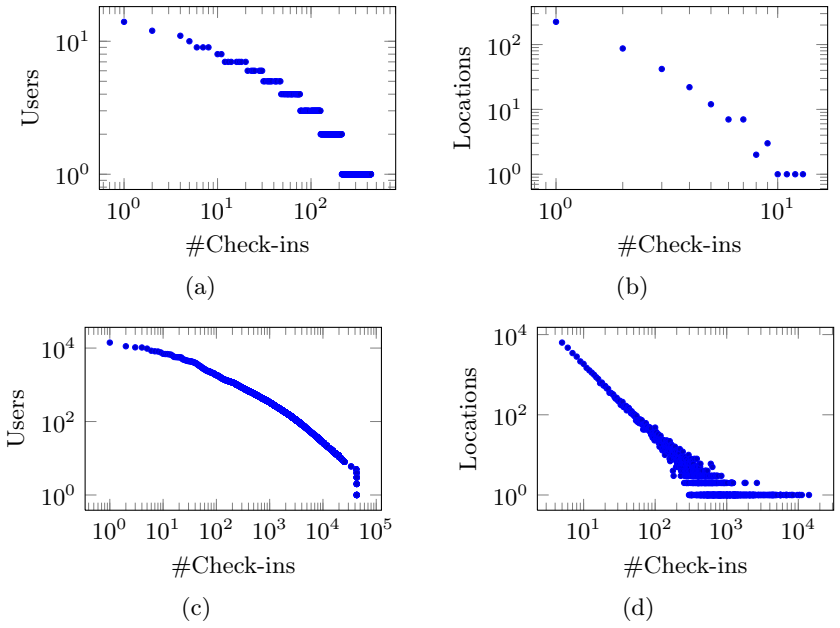

Fig. 2: Power Law distribution diagrams for GeoSocialRec [(a) and (b)] and Foursquare [(c) and (d)] datasets

the datasets follow a power law distribution for both the number of users' check-ins (Figures 2(a) and 2(c)) and the number of visits to a particular location (Figures 2(b) and 2(d)). As shown in Figures 2(a) and 2(c), there is a small number of users who have checked-in to many locations (short head) and many

users that have only checked-in a small number of locations (long tail). Similarly, as shown in Figures 2(b) and 2(d), few locations have many visits, whereas many locations have a small number of visits. Notice that, it is very difficult for all algorithms to recommend accurately locations, which have not been visited from many users (i.e., recommendation in the long tail of the distribution).

In addition, as shown in Table 2(c) we have created artificial session nodes to study the effect of the length of a session slot. We have created session nodes based on 3 hours, 6 hours, 9 hours, 12 hours and 24 hours. Please notice that the average session per user (when session is set to 3 Hours) is 0.10 and 2.02 for the GeoSocialRec and the Foursquare dataset, respectively. This means that it is very difficult to find co-locations among users in the first dataset, which will affect the recommendation accuracy of all algorithms, as will be shown experimentally later.

4.2 Comparison with Other Methods

In this section, we compare the well known RWR algorithm with other one state-of-the art comparison partner i.e. Fast-Katz, in terms of precision and recall. As the number N of the recommended users/locations varies starting from the top-1 to top-N, we examine the precision and recall scores. Achieving high recall scores while precision follows with the minimum decline indicates the robustness of the examined algorithm.

For the friend recommendation task, in Figures 3(a) and 3(c), we visualise the precision versus recall curve for the GeoSocialRec and Foursquare datasets, respectively. As N increases, precision falls, while recall increases as expected for both algorithms. RWR attains the best results achieving the highest precision, against Fast-Katz algorithm. The reason is that RWR exploits effectively information from all sub-networks (i.e., friendship, user-location, etc.) in contrast to Fast katz algorithm which exploits the relations of the nodes with the target node in depth of 4 hops. Please notice, that hidden relation may exist in greater depth than 4 hops. Thus, it is obvious why our approach gains higher values of precision versus recall in contrast to our competitor. Also, notice that while experimenting with GeoSocialRec dataset, the precision and the recall values are low. Thats is because there are only few relations among the nodes of constructed HST graph which tackles the performance of both algorithms.

For location recommendations, we get similar results as shown in Figures 3(b) and 3(d), for the GeoSocialRec and Foursquare datasets, respectively. Notice, that RWR outperforms again the other algorithm. The reason is that RWR exploits information from more sub-networks than Fast-Katz. Thus, RWR has more options to walk through the network structure using different paths and edge types. Moreover, RWR is more robust as we increase the number of top-N recommended locations because it achieves high recall scores while precision score drops smoothly. Please notice, that Fast Katz traverse globally the network, missing to capture adequately the local characteristics of the HST graph. Also, Fast Katz defines a measure that directly sums over all paths between any pair of nodes in the graph, exponentially damped by length to count short paths

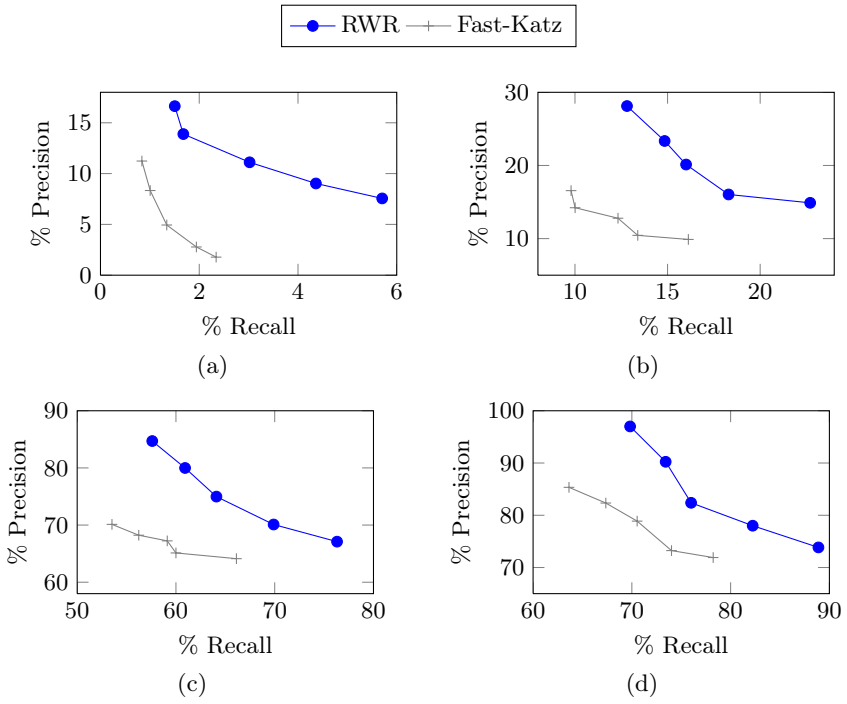

Fig. 3: Comparing RWR against Fast-Katz performance in term of Precision and Recall at top-N recommended [(a) users and (b) locations] on GeoSocialRec dataset, [(c) users and (d) locations] on Foursquare dataset.

more heavily. This way Fast Katz misses the relations existing in greater depth in contrast to our approach which takes them into account.

5 Conclusions

Since recommender systems incorporate information from *explicit* and *implicit* sub-networks to provide recommendation we argue that time is an important factor. Thus, we introduce the creation of an artificial node which captures the time dimension into our model. Moreover, we construct a novel hybrid k-partite graph which holds information from all participating networks. Then, we evaluate to what extent the well RWR algorithm improves its recommendation agains Fast-Katz algorithm in terms of precision and recall.

References

1. K. C. Foster, S. Q. Muth, J. J. Potterat, and R. B. Rothenberg. A faster katz status score algorithm. *Computational & Mathematical Organization Theory*, 7(4):275–285, 2001.

2. H. Gao, J. Tang, X. Hu, and H. Liu. Exploring temporal effects for location recommendation on location-based social networks. In *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys)*, pages 93–100, 2013.
3. H. Gao, J. Tang, and H. Liu. Exploring social-historical ties on location-based social networks. In *In Proceedings of the 6th International Conference on Weblogs and Social Media (ICWSM)*, pages 114–121, 2012.
4. S.-S. Ho, M. Lieberman, P. Wang, and H. Samet. Mining future spatiotemporal events and their sentiment from online news articles for location-aware recommendation system. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems (MobiGIS)*, pages 25–32, 2012.
5. Z. Lu, B. Savas, W. Tang, and I. S. Dhillon. Supervised link prediction using multiple sources. In *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM)*, pages 923–928, 2010.
6. L. B. Marinho, I. Nunes, T. Sandholm, C. Nóbrega, J. a. Araújo, and C. E. S. Pires. Improving location recommendations with temporal pattern extraction. In *Proceedings of the 18th Brazilian Symposium on Multimedia and the Web (WebMedia)*, pages 293–296, 2012.
7. A. Noulas, S. Scellato, N. Lathia, and C. Mascolo. A random walk around the city: New venue recommendation in location-based social networks. In *Proceedings of the International Conference on Privacy, Security, Risk and Trust (PASSAT), and International Conference on Social Computing (SocialCom)*, pages 144–153, 2012.
8. R. Raymond, T. Sugiura, and K. Tsubouchi. Location recommendation based on location history and spatio-temporal correlations for an on-demand bus system. In *Proceedings of the 19th ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*, pages 377–380, 2011.
9. M. Sattari, I. Toroslu, P. Karagoz, P. Symeonidis, and Y. Manolopoulos. Extended feature combination model for recommendations in location-based mobile services. *Knowledge and Information Systems*, 2014.
10. H. Tong, C. Faloutsos, and J. Pan. Fast random walk with restart and its applications. In *Proceedings of the 6th International Conference on Data Mining (ICDM)*, pages 613–622, 2006.
11. V. Vasuki, N. Natarajan, Z. Lu, B. Savas, and I. Dhillon. Scalable affiliation recommendation using auxiliary networks. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(1):3:1–3:20, 2011.
12. L. Xiang, Q. Yuan, S. Zhao, L. Chen, X. Zhang, Q. Yang, and J. Sun. Temporal recommendation on graphs via long- and short-term preference fusion. In *Proceedings of the 16th ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 723–732, 2010.
13. Z. Yin, M. Gupta, T. Weninger, and J. Han. A unified framework for link recommendation using random walks. In *Proceedings of the IEEE International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 152–159, 2010.
14. Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M. Thalmann. Time-aware point-of-interest recommendation. In *Proceedings of the 36th ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 363–372, 2013.
15. Q. Yuan, G. Cong, and A. Sun. Graph-based point-of-interest recommendation with geographical and temporal influences. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM)*, pages 659–668, 2014.