Event Recommendation based on Graph Random Walking and History Preference Reranking

Shenghao Liu, Bang Wang and Minghua Xu*
Huzhong University of Science and Technology (HUST), Wuhan, China
{shenghao_liu,wangbang,xuminghua}@hust.edu.cn

ABSTRACT

Event recommendation has become an important issue in *event-based social networks* (EBSN). In this paper, we study how to exploit diverse relations in an EBSN as well as individual history preferences to recommend preferred events. We first construct a hybrid graph consisting of different types of nodes to represent available entities in an EBSN. The graph uses explicit relations as edges to connect nodes of different types; while transferring implicit relations of event attributes to interconnect the event nodes. After executing the graph random walking, we obtain the candidate events with high convergency probabilities. We next extract a user preference from his attended events to further compute his interest similarities to his candidate events. The recommended event list is then obtained by combining the two similarity scores. Data sets from a real EBSN are used to examine the proposed scheme, and experiment results validate its superiority over peer schemes.

KEYWORDS

Event recommendation, cold-start problem, graph-based random walking, event-based social networks

1 INTRODUCTION

Recently, *event-based social networks* (EBSNs), such as Meetup and Douban Event [1], have been becoming widely developed, which not only offer a convenient platform to announce various social events, but also construct a complicated social network among users. How to efficiently recommend appropriate events to users in EBSNs has become a hot topic in both academia and industry [2, 4]. Unlike item recommendation such as recommending books, movies, which normally has not been embedded into a social network framework, recommending events in EBSNs has posed many new challenges.

In a typical EBSN, a user can join multiple online groups and attend many offline events, showing different online and offline relations. On the other hand, a social event has its unique temporal and spatial characteristics in terms of its time and location attributes. Furthermore, an event cannot be actually *consumed or evaluated* before its commencement, which raises the issue of new event *cold-start* problem. To address the challenges, some event recommendation algorithms have been proposed based on impacting factor analysis for content-based filtering [3, 5, 7, 9]. Recently, some graph-based event recommendation algorithms have drawn a lot of research interests[1, 4, 6, 8].

In graph-based event recommendation, entities in a EBSN, such as users, groups, events, hosts and etc., are represented as graph nodes, while their relations are used as edges. A *heterogeneous graph* structure is widely adopted: Entities are first grouped according to their types, such as user and event groups. Two nodes in two different groups may be connected through an edge, if some relation exists in between them. For example, a user attended an event, then an edge is drawn in between them. However, no edges are drawn in between nodes of the same group [1, 6, 8]. Random walking is performed on a constructed graph to obtain converged probabilities of each node, and the event probabilities are used to rank and recommend events to users. Due to its better capability to describe the diverse relations in EBSNs, graph-based event recommendation has been shown to achieve better performance, compared with the traditional content and collaborative filtering recommendation algorithms.

In this paper, we first propose to construct a *hybrid graph* structure for event recommendation. Similar to the heterogeneous graph construction, we also apply node grouping for some entities to first form node groups. We note that the heterogeneous graph structure [1, 6, 8] only exploits explicit relations in an EBSN, which may produce many *dangling nodes*. Although a dangling node is not disconnected to the whole graph, it only contains edges to only one type of nodes. Such dangling nodes impact on the graph connectivity property as well as the random walk. For example, the nodes of event attributes including its type, cost, time and location only connect to the event nodes, never connecting to other group nodes, which may deviate the random walk to undesirable routes. Therefore, we propose to use these attributes to first compute the similarities among events and further interconnect the event nodes in its group based on

---

*Shenghao Liu and Bang Wang are with the School of Electronic Information and Communications, HUST; Minghua Xu is with the School of Journalism and Information Communication, HUST. This work is supported in part by National Natural Science Foundation of China (Grant No: 61371141) and National Social Science Foundation of China (Grant No: 14CXW018).

[1]Meetup: www.meetup.com; Douban Event: www.douban.com

SIGIR'17, August 7–11, 2017, Shinjuku, Tokyo, Japan.
© 2017 ACM. 978-1-4503-5022-8/17/08...$15.00
DOI: http://dx.doi.org/10.1145/3077136.3080663

their similarities. As such, we construct a *hybrid graph* that exploits both explicit and implicit relations among entities to perform random walking. Furthermore, we notice that although the graph approach is able to describe the complicated relations among different entities, it may not well prioritize different types of relations. We next propose to use a content-based reranking algorithm to obtain the final event recommendation list from the candidate events selected by the graph random walking. We have crawled data sets from the famous Chinese EBSN, Douban Event, for two typical cities, Beijing and Shanghai. Our experiments show that the proposed scheme can achieve better recommendation results compared with the peer schemes.

2 RESEARCH METHOD

2.1 Hybrid Graph Construction

The essential entities of an EBSN include the users, hosts and events, yet the total available entities are actually platform dependent. In this paper, we focus on the event recommendation for a very popular Chinese EBSN, Douban Event, where the available entities include: users U, events E, groups G, hosts H, tags T. Furthermore, an event is also described by the following attributes: event time E_m, event location E_l, event cost E_c and event type E_t. These in total nine entities can be used as nodes for graph construction. However, some entities take real numbers, e.g., E_m and E_c; while some entities may take too many discrete values, e.g., T and E_l. Therefore, we need to first preprocess these entities for reducing graph complexity.

The basic idea of our preprocessing is to use segmentation or aggregation to reduce the parameter value space. For entity event time E_m, we divide the continuous time line into seven week days plus one another 'Everyday', i.e., from Monday to Sunday and Everyday, as we argue that people daily life often takes some periodic feature. For entity event cost E_c, we partition its value into five ranges for normal expense habits, i.e., free charge, $1 \sim 200$, $201 \sim 500$, $501 \sim 1000$, and above 1000 Chinese Yuan. For event locations E_l, we use fewer *administrative regions* E_r each to represent for one event location. Most of event locations also include the administrative region. If an event location does not contain the region information, we use the *nearest neighbor* algorithm to include it into the region with the shortest Euclidean distance to the region center. For tags T, we cluster them into fewer *subjects* by using the *unweighted pair-group method with arithmetic mean* (UPGMA). In each iteration, we group the most similar two clusters or tags into a new one. The iteration terminates, until the required number of clusters has achieved. Note that the intersection of any two tag clusters (i.e., subjects) is an empty set.

After the preprocessing, we have the following entities as graph nodes, namely, U, E, G, H, subjects S, E_c, E_m, E_t and event regions E_r. Like those traditional heterogeneous graph construction, we first use explicit relations to obtain *explicit edges* in between two nodes. Note that all explicit edges are undirected. For example, a user $U1$ joins an online group $G1$, then an undirected edge links $U1$ and $G1$; $U1$ attended an offline event $E1$, an edge exists between them. Note that after tag clustering, each group, host and event can be stamped by one or more subjects, given its tags appearing in how many subjects.

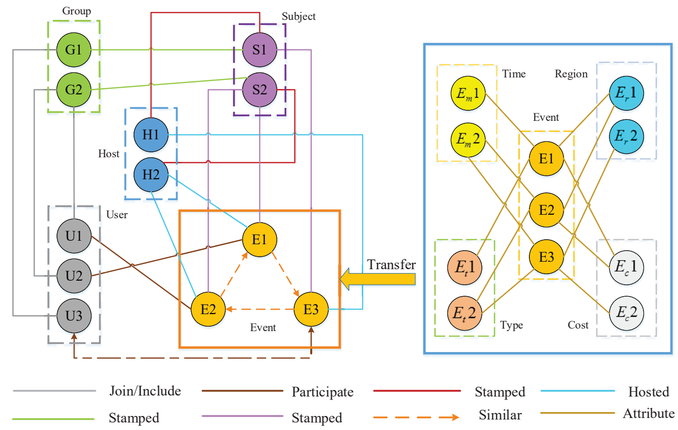


Figure 1: Illustration of hybrid graph construction.

In this paper, we proposed a new hybrid graph type which differs from the heterogeneous graph in that we convert the explicit relations between an event to its attributes into some *implicit relations* in between events. As shown in Fig. 1, the right big box illustrates the explicit edges between event node and its attribute nodes, where no edge exists in between event nodes and also no edge exists in between each type of attribute nodes. If we use this box to replace the small organge box, then a heterogeneous graph is constructed. However, such a heterogeneous graph contains too many dangling event attribute nodes, each of which only connects to event nodes. We observe from our experiments that such dangling nodes not only impact on the graph connectivity, but also often lead to deviated routes from event nodes to its attribute nodes in the random walking.

For our hybrid construction, we propose the following relation conversion: Let $\mathbf{A}_{EE_c}$, $\mathbf{A}_{EE_m}$, $\mathbf{A}_{EE_r}$ and $\mathbf{A}_{EE_t}$ denote the adjacency matrices of events and event attribute nodes. Let $\mathbf{A}_E$ denote the concatenation matrix of these matrices, where each row represents the attribute vector of one event node. We compute the cosine similarity between two event nodes by $sim(E_i, E_j) = \cos(\overrightarrow{A}_i, \overrightarrow{A}_j)$, where $\overrightarrow{A}_i$ is the ith row vector of $\mathbf{A}_E$. For each event E_i, we select its top K most similar events to establish K directed implicit edges each from E_i to one of its these similar event nodes. In this paper, we set $K = 100$. Note that since the sets of similar events may be different of two different events, so we use directed implicit edges. Fig. 1 illustrates the constructed hybrid graph, where only event nodes contain directed implicit edges in between the event nodes.

2.2 Random Walking with Restart

We use a multivariate Markov chain to transform event recommendation task into a node convergency probability computation problem. Let $\boldsymbol{A}_{MN}$ be the adjacency matrix of type M nodes and type N nodes, where $\boldsymbol{A}_{MN}(m,n)=1$ indicates that an explicit or implicit relation exists between the node n and node m; Otherwise, $\boldsymbol{A}_{MN}(m,n)=0$. A transition matrix $\boldsymbol{P}_{MN}$ is then obtained by row-normalizing the adjacency matrix $\boldsymbol{A}_{MN}$. We define the *user query vector* as $\boldsymbol{q}_u$. For each user u_j, $\boldsymbol{q}_u(i)=1$, if $i=j$, Otherwise, $\boldsymbol{q}_u(i)=0$.

We randomly initialize the probability vector of users, events, groups, hosts and subjects as $\boldsymbol{u}^{(0)}, \boldsymbol{e}^{(0)}, \boldsymbol{g}^{(0)}, \boldsymbol{h}^{(0)}, \boldsymbol{s}^{(0)}$. To obtain the convergency probabilities, the *random walking with restart* (RWR) algorithm is to iteratively compute the following equations:

$$\begin{aligned}\boldsymbol{u}^{(t+1)} &= \alpha_{EU}\boldsymbol{e}^{(t)}\boldsymbol{P}_{EU} + \alpha_{GU}\boldsymbol{g}^{(t)}\boldsymbol{P}_{GU} \\ &\quad + (1-\alpha_{EU}-\alpha_{GU})\boldsymbol{q}_u \end{aligned} \tag{1}$$

$$\begin{aligned}\boldsymbol{e}^{(t+1)} &= \alpha_{UE}\boldsymbol{u}^{(t)}\boldsymbol{P}_{UE} + \alpha_{HE}\boldsymbol{h}^{(t)}\boldsymbol{P}_{HE} + \alpha_{SE}\boldsymbol{s}^{(t)}\boldsymbol{P}_{SE} \\ &\quad + (1-\alpha_{UE}-\alpha_{HE}-\alpha_{SE})\boldsymbol{e}^{(t)}\boldsymbol{P}_{EE} \end{aligned} \tag{2}$$

$$\boldsymbol{h}^{(t+1)} = \alpha_{EH}\boldsymbol{e}^{(t)}\boldsymbol{P}_{EH} + (1-\alpha_{EH})\boldsymbol{s}^{(t)}\boldsymbol{P}_{SH} \tag{3}$$

$$\boldsymbol{g}^{(t+1)} = \alpha_{UG}\boldsymbol{u}^{(t)}\boldsymbol{P}_{UG} + (1-\alpha_{UG})\boldsymbol{s}^{(t)}\boldsymbol{P}_{SG} \tag{4}$$

$$\begin{aligned}\boldsymbol{s}^{(t+1)} &= \alpha_{GS}\boldsymbol{g}^{(t)}\boldsymbol{P}_{GS} + \alpha_{HS}\boldsymbol{h}^{(t)}\boldsymbol{P}_{HS} \\ &\quad + (1-\alpha_{HS}-\alpha_{GS})\boldsymbol{e}^{(t)}\boldsymbol{P}_{ES} \end{aligned} \tag{5}$$

where $\boldsymbol{u}^{(t+1)}, \boldsymbol{e}^{(t+1)}, \boldsymbol{g}^{(t+1)}, \boldsymbol{h}^{(t+1)}, \boldsymbol{s}^{(t+1)}$ are probability vectors representing the probability that user, event, group, host and subject nodes are visited in the tth iteration, respectively. α_{MN} denotes the transition weight from one type node to another type node. For example, in Eq. (1) user nodes get α_{EU} probability from event nodes, α_{GU} probability from group nodes, and return to the candidate user node with $(1-\alpha_{EU}-\alpha_{GU})$ probability. Since the hybrid graph is of large scale, we do not try to train the weights for computation complexity considerations. Instead, we set that the weights of each affecting factor of transition probability are equal. For example, in Eq. (1) we set $\alpha_{EU}=\alpha_{GU}=(1-\alpha_{EU}-\alpha_{GU})=1/3$. The iteration terminates until the pairwise difference in between two iteration probability vectors is small than a predefined threshold. It has been proven in [8] that if the constructed graph is a connected one, then the iterations can converge. After the iteration termination, each user u obtains a vector of event convergency probabilities. We use $sim_g(u,e)$ to denote the convergency event probability of event e by a user u.

2.3 History Preference Reranking

The random walking on a graph is an efficient approach to embed an event recommendation task into a social network. However, such a graph might have ignored the history preference of individual user to events, if the transition weights are not individually set and trained for each user. On the other hand, as new events have not been consumed and evaluated by any user, they are directly included in the constructed graph without considering potential relations to some users, which may lead to the new event *cold-start* problem. Considering these, we next rerank events based on the user history preference to obtain the final recommendation list.

For each event e, we use $\vec{e}$ to denote its concatenated feature vector consisting of its four attribute vectors plus its subject vector, i.e., $\vec{e}=(\vec{e}_c,\vec{e}_m,\vec{e}_r,\vec{e}_t,\vec{e}_s)$. Let $\vec{u}$ denote the preference vector of user u, which has the same structure as $\vec{e}$. Let $\mathcal{E}_u^{old}$ denote the set of old events that a user u has attended. We use the pairwise sum of the attended event features to obtain a user history preference vector, that is, $\vec{u}=\sum_{j\in\mathcal{E}_u^{old}}\vec{e}_{(j)}$.

We compute the cosine similarity between $\vec{u}$ and $\vec{e}$ as the preference similarity between a user u and an event e, i.e., $sim_p(u,e)=\frac{\vec{u}\times\vec{e}}{|\vec{u}|\cdot|\vec{e}|}$. Instead of considering all new events, we only select top N new events from random walking for reranking in order to reduce the computation complexity. Let $\mathcal{E}_u^{newN}$ denote the set of such top N new events for user u. Then for each event $e\in\mathcal{E}_u^{newN}$, our reranking is based on the following similarity computation:

$$sim(u,e)=sim_g(u,e)\times sim_p(u,e), e\in\mathcal{E}_u^{newN}, \tag{6}$$

which simply strikes a balance between two types of algorithms. The final recommendation list is then obtained based on the event similarity value $sim(u,e)$ in a decreasing order.

3 EXPERIMENT

We have crawled data sets from Douban Event for two main cities, Beijing and Shanghai, in China. For Beijing, we obtained 6982 events and 88963 users from Jul 1st, 2015 to Dec 31st, among which in total 80153 effective user-event pairs are used as data set. For Shanghai, we obtained 6427 events and 75829 users from Sep 1st, 2015 to Dec 31st, among which in total 67822 effective user-event pairs are used as data set. We use the five-fold cross validation to obtain the averaged results. Since most of users have not attended more than three events, so we mainly examine the top three recommended events. In our experiments, we compare the performance for the following algorithms:

CB: The traditional content-based recommendation, where we use the proposed user history preference feature.

HetG: The peer heterogeneous graph-based random walking, where we use the event attributes directly in the graph.

HetG+R: The HetG algorithm plus our proposed history preference reranking.

HybG: The proposed hybrid graph-based random walking.

HybG+R: The proposed HybG algorithm plus our reranking algorithm.

Tables 1 and 2 list all results for Bejing and Shanghai, respectively. We observe that the proposed HybG+R algorithm outperforms all the other peer algorithms in all performance metrics, except a slightly lower coverage in the Beijing data set. In particular, the *mean average precision* (MAP) improvements of Beijing are 65.17%, 183.94%, 63.63% and 9.91% over the CB, HetG, HetG+R and HybG algorithm, respectively. And they are 65.56%, 124.10%, 32.35% and 3.46% in Shanghai, respectively. Although all test users have attended at least three events, some do have attended more.

Table 1: Performance comparison of Beijing

%	CB	HetG	HetG+R	HybG	HybG+R
P@1	12.04	3.40	8.64	18.93	**21.66**
P@2	11.55	5.66	11.20	18.17	**20.32**
P@3	10.78	7.06	12.15	17.90	**19.25**
Precision	10.78	7.06	12.15	17.90	**19.25**
Recall	2.38	2.45	3.74	4.80	**5.07**
F1	3.90	3.64	5.71	7.55	**8.00**
MAP	15.44	8.82	15.30	22.77	**25.03**
AUC	24.32	48.48	48.48	51.29	**51.33**
Coverage	**18.44**	7.70	12.04	16.50	18.09

Table 2: Performance comparison of Shanghai

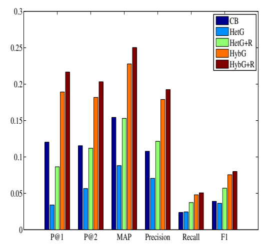
%	CB	HetG	HetG+R	HybG	HybG+R
P@1	23.87	10.09	27.54	44.05	**45.28**
P@2	24.14	16.91	32.95	44.54	**45.91**
P@3	23.60	21.80	35.27	44.48	**45.65**
Precision	23.60	21.80	35.27	44.48	**45.65**
Recall	5.76	7.98	11.42	13.08	**13.30**
F1	9.26	11.64	17.24	20.20	**20.58**
MAP	30.89	22.82	38.64	49.43	**51.14**
AUC	15.56	51.20	51.20	52.90	**53.90**
Coverage	17.49	9.32	15.14	18.39	**19.19**

Therefore, using a recommendation list of three may not cover all the events been attended by the test users, which leads to low recall of all algorithms. However, our proposed algorithms still achieve higher recall than the peer ones. The results indicate that event recommendation in EBSNs should consider not only the online and offline social relations but also the individual user history preferences. The CB algorithm only focuses on the user preference while ignoring the social relations, which could result in its poorer performance. On the other hand, although the graph-based approach can well describe all relations in EBSNs, care must be taken for prioritizing different types of relations in the graph construction. The reasons of the poor HetG performance could be attributed to its inclusion of nonessential event attributes in the graph, which introduces many dangling nodes only connecting to one another type of nodes that often induce some deviated random walking routes. Comparing the results of with and without using history preference reranking, we notice the potential of boosting recommendation performance, if multiple algorithms could be appropriately integrated.

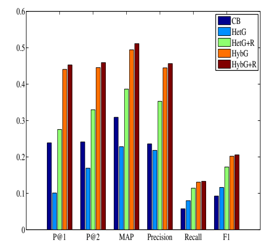
4 CONCLUDING REMARKS

In this paper, we have proposed an event recommendation scheme based on graph random walking and history preference reranking. A hybrid graph is constructed to exploit and prioritize diverse relations among the available entities of an EBSN. After the random walking on the constructed graph, we have proposed to boost the recommendation results via an individual history preference reranking. Experiments on real data sets have validated the superiority of the proposed recommendation scheme.

In our future work, we shall first consider how to further extract more history preferences from semantic analysis of the event announcements. When considering that recommendation needs not to be done for all users at the same time, so some feedbacks might be available if some users could have seen the recommended events ahead of others. As such, the cold-start problem may be alleviated from such feedbacks. Our future work shall also study how to obtain and exploit such feedbacks for event recommendation.



(a)



(b)

Figure 2: The results of $P@1$, $P@2$, MAP, $Precision$, $Recall$ and $F1$ in Beijing (a) and Shanghai (b).

REFERENCES

[1] Hakan Bagci and Pinar Karagoz. Random walk based context-aware activity recommendation for location based social networks. In *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–9. IEEE, 2015.

[2] Jie Bao, Yu Zheng, David Wilkie, and Mohamed F Mokbel. A survey on recommendations in location-based social networks. *ACM Transaction on Intelligent Systems and Technology*, 2013.

[3] Elizabeth M Daly and Werner Geyer. Effective event discovery: using location and social information for scoping event recommendations. In *The 5th ACM conference on Recommender systems*, pages 277–280. ACM, 2011.

[4] Pavlos Kefalas, Panagiotis Symeonidis, and Yannis Manolopoulos. A graph-based taxonomy of recommendation algorithms and systems in lbsns. *IEEE Transactions on Knowledge and Data Engineering*, 28(3):604–622, 2016.

[5] Houda Khrouf and Raphaël Troncy. Hybrid event recommendation using linked data and user diversity. In *The 7th ACM conference on Recommender systems*, pages 185–192. ACM, 2013.

[6] Bixi Li, Bang Wang, Yijun Mo, and Laurencet T. Yang. A novel random walk and scale control method for event recommendation. In *The 13th IEEE International Conference on Ubiquitous Intelligence and Computing (UIC)*, pages 567–578, 2016.

[7] Einat Minkov, Ben Charrow, Jonathan Ledlie, Seth Teller, and Tommi Jaakkola. Collaborative future event recommendation. In *The 19th ACM international conference on Information and knowledge management*, pages 819–828. ACM, 2010.

[8] Tuan-Anh Nguyen Pham, Xutao Li, Gao Cong, and Zhenjie Zhang. A general graph-based model for recommendation in event-based social networks. In *The 31st IEEE International Conference on Data Engineering*, pages 567–578. IEEE, 2015.

[9] Max Sklar, Blake Shaw, and Andrew Hogue. Recommending interesting events in real-time with foursquare check-ins. In *The 6th ACM conference on Recommender systems*, pages 311–312. ACM, 2012.