Joint Event-Partner Recommendation in Event-based Social Networks

Hongzhi Yin[†] Lei Zou[§] Quoc Viet Hung Nguyen[‡] Zi Huang[†] Xiaofang Zhou[†]
[†]*The University of Queensland, School of Information Technology and Electrical Engineering*
[§]*Institute of Computer Science and Technology, Peking University*
[‡]*School of Information and Communication Technology, Griffith University*
[†]db.hongzhi@gmail.com [§]zoulei@pku.edu.cn
[†]{huang,zxf}@itee.uq.edu.au [‡]quocviethung.nguyen@griffith.edu.au

***Abstract*—With the prevalent trend of combining online and offline interactions among users in event-based social networks (EBSNs), event recommendation has become an essential means to help people discover new interesting events to attend. However, existing literatures on event recommendations ignore the social attribute of events: people prefer to attend events with their friends or family rather than alone. Therefore, we propose a new recommendation paradigm: joint event-partner recommendation that focuses on recommending event-partner pairs to users.**

In this paper, we focus on the new problem of joint event-partner recommendation in EBSNs, which is extremely challenging due to the intrinsic cold-start property of events, the complex decision-making process for choosing event-partner pairs and the huge prediction space of event-partner combinations. We propose a generic graph-based embedding model (GEM) to collectively embed all the observed relations among users, events, locations, time and text content in a shared low-dimension space, which is able to leverage the correlation between events and their associated content and contextual information to address the cold-start issue effectively. To accelerate the convergence of GEM and improve its modeling accuracy, an adaptive noise sampler is developed to generate adversarial negative samples in the model optimization. Besides, to speed up the online recommendation, we propose a novel space transformation method to project each event-partner pair to one point in a new space and then develop effective space pruning and efficient online recommendation techniques. We conduct comprehensive experiments on our created real benchmark datasets, and the experimental results demonstrate the superiority of our proposals in terms of recommendation effectiveness, efficiency and scalability.

I. Introduction

With the prevalent trend of combining online to offline (O2O) interactions among users in the mobile Internet era, event-based social networks (EBSNs) [8], such as Meetup (www.meetup.com) and Plancast (www.plancast.com), have recently emerged as the convenient online platforms for people to create, distribute, organize and register social events. On these web services, people may propose social events, ranging from informal get-togethers (e.g. movie night and dining out) to formal activities (e.g., technical conferences and business meetings). These event-based online services promote face-to-face social interactions in the physical world. However, the sheer volume of available events, particularly in large and tourist cities, often overwhelm users to find the ones that best match their interests. It is quite valuable to develop the event recommendation service as an essential function of EBSNs.

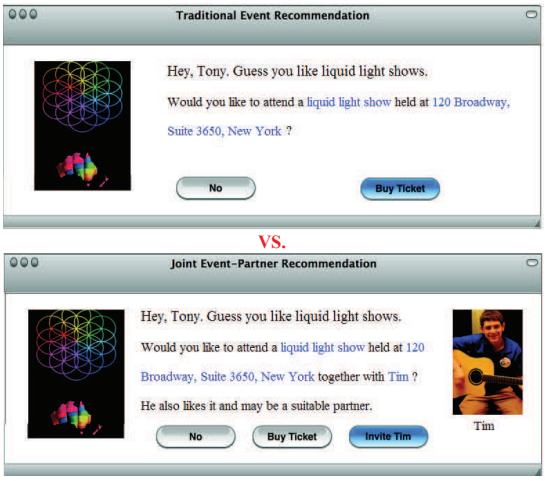


Fig. 1. Illustration of Event-Partner Recommendation

However, most existing work on event recommendation focused on utilizing users' attendance history, social links between users, geographical location, time and content information of events [36], [9], [3], [13] to predict the preferences of users, and ignored the special social attribute of events: people tend to attend events with their partners rather than alone. As analyzed in [22], humans are by nature social animals, and they prefer to find partners before attending social events, which means that if a system only recommends an event alone, the user is most likely to refuse the recommendation if she cannot think of a partner to attend it with. Thus, we extend the functionality of event recommendation into a new recommendation paradigm – joint event-partner recommendation, to help users simultaneously find their interested events and suitable partners, as shown in Figure 1.

To further promote and enhance the online social friendship, the recommended partners are not limited to existing online friends in our proposed recommendation paradigm, as the potential friends (e.g., users who do not have direct social connections, but share many common friends) may also be suitable partners. Our proposed new recommendation paradigm will bring people's friendship from the online virtual world into their real lives in the physical world: turning the online virtual friendship into the true friendship in the reality by providing them opportunities to participate in the real-

life activities together. Most of young people boast Facebook friends that number in the hundreds but in reality they often stay alone as they have nobody to hang out with. Our proposed new recommendation paradigm has the great potential to alleviate this serious social issue.

However, the event-partner recommendation is a highly challenging problem due to the following three main reasons. (1) **Cold Start**. As events published in EBSNs are typically short-lived and, by definition, are always in the future, they have little or no trace of historical attendance. The events with enough historical attendance or registration records are always those outdated ones, and their starting times were in the past. The item cold-start problem arises naturally in this setting. The classic recommendation approaches [3], [22], [13] that assume events have ample historical records of users (e.g., attendance or registration) would underperform in this scenario. (2) **Complex Decision Process**. A successful event-partner recommendation depends on not only whether the target user will adopt the recommended events, but also whether the recommended partner will accept the invitation from the target user. Thus, we need to consider not only the preferences of the target user, but also the recommended partner's preferences and the social proximity of the two users. (3) **Huge Prediction Space**. Given U users and V events, the number of possible event-partner combinations would be $O(U \times V)$. If we compute a score for each event-partner combination and then select top-n ones with highest scores as recommendations, it would not be feasible due to the huge online computation cost.

In this paper, we propose a generic graph-based embedding model (GEM) to learn the embedding (i.e., vector representation) of both users and events in the same low-dimension latent space for event-partner recommendation. **To overcome the cold start issue**, we exploit both content and contextual information of events: 1) content, which provides the semantic description of the event theme, 2) geographical location, where the event will be held, and 3) time, when the event will be held. Recent research has shown that personal interests exhibit strong regularity [27], and events that a user has attended and will attend have similar or nearly identical content. Besides, users tend to attend events that are geographically close to the ones they have already attended before [36]. As different users have different working schedules and life styles, they have different temporal preferences [9], [26]. Some users might prefer to attend events on the mornings of weekends, while others prefer events held on Friday nights. Specifically, we first use a bipartite graph model to represent the interactions between users and events, events and content words, events and locations, and events and time, respectively. Then, we embed all the heterogeneous information networks into a shared low-dimension space, in which the vector representations of cold-start events are learned from their associated content and contextual information captured by event-word, event-location and event-time bipartite graphs. The user-event relation graph plays the role of bridge to link users and these content and contextual information together, enabling the learned users' vectors to capture users' personal interests, spatial and temporal preferences.

To efficiently model the complex decision process, we decompose the high order interrelationships among users, events and partners into a set of pair-wise interaction relationships, inspired by [16], i.e., the triple (user, partner, event) is decomposed to (user, event), (partner, event) and (user, partner) pairs. Thus, we not only consider the target user's preference for the recommended event, but also the partner's preference for the event and the social proximity between the target user and the recommended partner when producing event-partner recommendations. To make the learned users' vectors also capture the social dimension, we embed the social network graph into the same latent space with user-event, event-word, event-location and event-time relation graphs.

To address the challenge of the huge prediction space, we first propose a novel space transformation method to map each event-partner pair to one point in a new space. Then, our proposed joint event-partner recommendation problem is transformed to the traditional item recommendation problem in the new space. Based on the new space, we propose an effective pruning strategy to significantly reduce the number of candidate event-partner pairs and then adopt the efficient TA-based retrieval algorithm [32] to further speed up online top-n recommendation. The TA-based algorithm has the nice property of returning top-n recommendations by scanning or examining the minimum number of event-partner pairs.

To summarize, we make the following contributions:

1) **Novel Problem**. We propose a novel recommendation paradigm: joint event-partner recommendation.
2) **Comprehensive Model**. We develop a generic graph-based embedding model to embed multiple heterogenous bipartite graphs into a shared low-dimensional space so that the representations of cold-start events can be learned from their associated content and contextual information. To speed up the convergence of our model and improve its modeling accuracy, we propose a bidirectional negative sampling method and a novel adaptive noise sampler to generate adversarial negative samples.
3) **Efficient Online Recommendation Techniques**. We propose a new space transformation method to project each event-partner pair to one point in a new space, in which an effective pruning strategy is developed and the TA-based query processing technique is employed to speed up the online recommendation process.
4) **Extensive Experiments**. We conduct extensive experiments on a large-scale EBSN dataset to evaluate the performance of our recommender model in terms of recommendation effectiveness and efficiency. The results show the superiority of our proposals by comparing with the state-of-the-art techniques.

Roadmap of this paper. Section 2 details our proposed graph-based embedding model. We present the efficient online recommendation techniques in Section 3, and report the experimental results in Section 4. Section 5 reviews the related work and Section 6 concludes the paper.

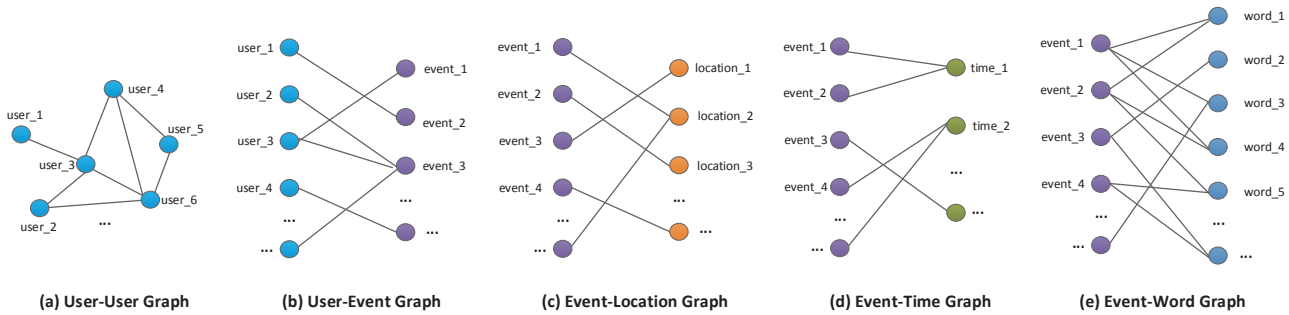


Fig. 2. User-event graph, event-location graph, event-time graph and event-word graph are all bipartite graphs, while user-user graph is a general graph which can also be treated as a bipartite graph when one user is on one side and others are on the other side.

II. Preliminary and Problem Formulation

In this section, we first introduce some necessary definitions and notations used in this paper.

Definition 1: **(Event-based Social Network)**. An event-based social network is represented by a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and it mainly contains five types of nodes $\mathcal{V} = \mathcal{V}_X \bigcup \mathcal{V}_U \bigcup \mathcal{V}_T \bigcup \mathcal{V}_C \bigcup \mathcal{V}_L$. Among all types of nodes, the social event set $\mathcal{V}_X$ is the most significant one that associates other types of nodes together. For an event $x \in \mathcal{V}_X$, its location is $l \in \mathcal{V}_L$ and its time is $t \in \mathcal{V}_T$. There is also a textual document $\mathcal{D}_x$ to describe the event x, and $\mathcal{V}_C$ is a vocabulary word set. For a user $u \in \mathcal{V}_U$, if she is interested in the event x, she will register online for attending. We use $\mathcal{X}_u$ to denote the set of events attended by u, and $\mathcal{U}_x$ to denote the set of users who attend event x.

Based on the heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we define the following 5 bipartite graphs, as shown in Figure 2.

Definition 2: **(User-User Graph)** A User-User graph, denoted as $\mathcal{G}_{UU} = (\mathcal{V}_U \bigcup \mathcal{V}_U, \mathcal{E}_{UU})$, captures the social relationship between users, where $\mathcal{E}_{UU}$ is a set of edges between users. Given two users u and u', if they are friends in the social network, there is an edge $e_{uu'}$ between them. The weight $w_{uu'}$ on edge $e_{uu'}$ is defined as $1 + |\mathcal{X}_u \bigcap \mathcal{X}_{u'}|$, where $|\mathcal{X}_u \bigcap \mathcal{X}_{u'}|$ denotes the number of common events u and u' have attended.

Definition 3: **(User-Event Graph)** An User-Event graph, denoted as $\mathcal{G}_{UX} = (\mathcal{V}_U \bigcup \mathcal{V}_X, \mathcal{E}_{UX})$, is a bipartite graph where $\mathcal{V}_X$ is a set of events and $\mathcal{V}_U$ is a set of users. $\mathcal{E}_{UX}$ is a set of edges between users and events. If user u attends event x, there is an edge e_{ux} between them; otherwise, none. If the user rating information is available, the edge weight w_{ux} is defined as the rating; otherwise, the weight w_{ux} is set to 1.

To overcome the cold-start issue and learn the events' representations from their contextual and content information, three bipartite graphs, *event-location*, *event-time* and *event-word*, are introduced below, as each of the three signals, content, location and time of an event, has a significant influence on the user's decision. As nodes are discrete in graphs, we need first to transform the continuous spatial and temporal information into discrete representations. Specifically, we divide all events into a set of regions $\mathcal{V}_L$ using DBSCAN based on their geographic coordinates. All events' time is divided into a set of time slots $\mathcal{V}_T$. The time of users' attending events has temporal periodicity, and the periodicity exists in multiple scales, such as hours of a day, different days of a week and different weekday types (i.e., weekdays vs. weekends). Thus, we define 33 time slots across three scales in $\mathcal{V}_L$, including 24 hour slots, 7 day slots, and two other time slots to indicate weekday and weekend. For example, "2017-06-29 18:00" corresponds to three time slots in $\mathcal{V}_L$: 18:00, Thursday, and weekday.

Definition 4: **(Event-Location Graph)** An Event-Location graph, denoted as $\mathcal{G}_{XL} = (\mathcal{V}_X \bigcup \mathcal{V}_L, \mathcal{E}_{XL})$, is a bipartite graph where $\mathcal{V}_L$ is a set of region nodes, and $\mathcal{E}_{XL}$ is a set of edges between event nodes and region nodes. If event x is held in region l, there is an edge between x and l; otherwise, none. The weight w_{xl} is set to 1 if the edge e_{xl} exists.

Definition 5: **(Event-Time Graph)** An Event-Time graph, denoted as $\mathcal{G}_{XT} = (\mathcal{V}_X \bigcup \mathcal{V}_T, \mathcal{E}_{XT})$, is a bipartite graph where $\mathcal{V}_T$ is a set of time nodes, and $\mathcal{E}_{XT}$ is a set of edges between event nodes and time nodes. If event x is held at t, there is an edge e_{xt} between them; otherwise, none. The weight w_{xt} is set to 1 if the edge e_{xt} exists. Each event node is linked to three time nodes, as we adopt three different time granularities.

Definition 6: **(Event-Content Graph)** An Event-Content graph, denoted as $\mathcal{G}_{XC} = (\mathcal{V}_X \bigcup \mathcal{V}_C, \mathcal{E}_{XC})$, is a bipartite graph where $\mathcal{V}_C$ is a set of vocabulary nodes, and $\mathcal{E}_{XC}$ is a set of edges between event nodes and word nodes. For each content word $c \in \mathcal{D}_x$, there is an edge e_{xc} between nodes x and c. As $\mathcal{D}_x$ is a bag of words associated with event x, we employ the standard $tf.idf$ to compute the edge weight w_{xc}.

In the next section, we will develop a graph-based embedding model (GEM) to embed the above five bipartite graphs into a shared K-dimension latent space $\mathbb{R}^K$, in which each node is represented by a K-dimension vector. Through the *Event-Content*, *Event-Location* and *Event-Time* graphs, the learned event vector $\vec{x}$ simultaneously captures the event's semantic, spatial and temporal information. Thus, the *User-Event* graph enables user vector $\vec{u}$ to capture u's personal interests, spatial and temporal preferences that determine u's preferences for events. For the success of event-partner recommendation, the social affinity between the target user and the recommended partner is another key factor. Therefore, we leverage *User-User* graph to model the social proximity.

Finally, we formally define the problem investigated in our work. Given an event-based social network $\mathcal{G}$, we aim to provide joint event-partner recommendations stated as follows.

Problem 1: **(Joint Event-Partner Recommendation)** Giv-

en an event-based social network $\mathcal{G}$ and a target user u, our goal is to recommend top-n event-partner pairs so that u and the recommended partners would like to attend the recommended events together.

III. Graph-based Embedding Model

In this section, we first propose a bipartite graph embedding model and its optimization, and then present a novel approach to perform joint embedding of multiple bipartite graphs.

A. Bipartite Graph Embedding

In this subsection, we introduce a probabilistic model to learn embeddings of heterogeneous graph nodes, which is different from the *LINE* model [21] that is designed for homogeneous information networks. The basic idea is to preserve similarities between graph nodes in a low-dimensional latent space. Given a bipartite graph $\mathcal{G}_{AB} = (\mathcal{V}_A \cup \mathcal{V}_B, \mathcal{E}_{AB})$, where $\mathcal{V}_A$ and $\mathcal{V}_B$ are two disjoint sets of nodes with different types, and $\mathcal{E}_{AB}$ is a collection of edges between them. Given a pair of nodes (v_i, v_j) ($v_i \in \mathcal{V}_A, v_j \in \mathcal{V}_B$), we first define the probability of observing a binary edge e_{ij} between them as:

$$p(e_{ij} = 1) = f(\vec{v}_i^{\mathrm{T}} \cdot \vec{v}_j), \tag{1}$$

where $\vec{v}_i$ is the vector representation of node v_i in the latent space (i.e., the embedding of node v_i), and $f(\cdot)$ is a probabilistic function w.r.t. the inner product between vectors $\vec{v}_i$ and $\vec{v}_j$, i.e., $\vec{v}_i^{\mathrm{T}} \cdot \vec{v}_j$. When the similarity between v_i and v_j is high in the latent space (i.e., $\vec{v}_i^{\mathrm{T}} \cdot \vec{v}_j$ is large), the probability of observing an edge between the two nodes is also high. Following the recent advances in similarity computation [10], [21], we adopt the sigmoid function to compute the probability, i.e., $f(x) = 1/(1 + exp(-x))$. Eqn. (1) only defines the probability of observing a binary edge. To further extend it to general weighted edges, we define the likelihood of observing an edge e_{ij} with weight w_{ij} as $p(e_{ij} = 1)^{w_{ij}}$.

Based on the above definition, given a weighted bipartite graph $\mathcal{G}_{AB} = (\mathcal{V}_A \cup \mathcal{V}_B, \mathcal{E}_{AB})$, its likelihood can be computed:

$$L_{AB} = \prod_{(v_i,v_j)\in\mathcal{E}_{AB}} p(e_{ij} = 1)^{w_{ij}} \prod_{(v_i,v_j)\in\overline{\mathcal{E}}_{AB}} (1 - p(e_{ij} = 1))^{\gamma_{ij}},$$

where $\overline{\mathcal{E}}_{AB}$ is the set of edges that are not observed (i.e., negative edges), and γ_{ij} is a weight assigned to the negative edge. Our goal is to maximize the above objective function, which is equivalent to minimizing the negative log likelihood:

$$\begin{aligned} O_{AB} = &- \sum_{(v_i,v_j)\in\mathcal{E}_{AB}} w_{ij} \log p(e_{ij} = 1) \\ &- \sum_{(v_i,v_j)\in\overline{\mathcal{E}}_{AB}} \gamma_{ij} \log(1 - p(e_{ij} = 1)). \end{aligned} \tag{2}$$

Model Optimization. Directly minimizing Eqn. (2) is computationally expensive, as the number of negative edges is huge and approximately quadratic to the number of nodes. Inspired by the negative sampling techniques [10], rather than using all negative edges, we choose to sample some negative edges for each positive edge in the model optimization, and the weights on the negative edges are assumed to be equal to the weight on their corresponding positive edge. Given a positive edge $(v_i,\ v_j)$, all existing works on negative sampling [10], [21], [1] generate the negative edges from only one side, i.e., the negative sampling is unidirectional, resulting in the objective function in Eqn. (3). Specifically, they generally fix the left-side node v_i and sample some noise nodes v_k (i.e., nodes without any link to v_i) from $\mathcal{V}_B$ according to a noisy distribution $P_n(v)$, and treat (v_i, v_k) as the negative edges. Although this negative sampling method can achieve good performance in the homogeneous graphs [10], [21], [1], constructing negative edges from one side is problematic and insufficient in the heterogenous bipartite graphs and would lead to the slow convergence and low modeling accuracy. For example, in the *User-Event* graph, if we only sample noise nodes from the event side to generate negative edges, as done in Bayesian personalized ranking (BPR) [18], we cannot accurately learn latent vector representation for event node v_j, because only positive users (i.e., users who attend event v_j) are considered and thus the learned event vector $\vec{v_j}$ could not discriminate between its positive users and negative users.

$$\begin{aligned} O_{AB} = &- \sum_{(v_i,v_j)\in\mathcal{E}_{AB}} w_{ij} \Big(\log p(e_{ij} = 1) \\ &+ \sum_{k=1}^{M} E_{v_k\sim P_n(v)} \log(1 - p(e_{ik} = 1)) \Big). \end{aligned} \tag{3}$$

$$\begin{aligned} O_{AB} = &- \sum_{(v_i,v_j)\in\mathcal{E}_{AB}} w_{ij} \Big(\log p(e_{ij} = 1) \\ &+ \sum_{k=1}^{M} E_{v_k\sim P_n(v)} \log(1 - p(e_{ik} = 1)) \\ &+ \sum_{k=1}^{M} E_{v_k\sim P_n(v)} \log(1 - p(e_{kj} = 1)) \Big). \end{aligned} \tag{4}$$

Bidirectional Negative Sampling Strategy. We introduce a bidirectional sampling strategy to construct negative edges from both sides by fixing nodes v_i and v_j alternatively, following [28]. Specifically, given a positive edge e_{ij} on the *User-Event* graph, we first fix user node v_i, and sample noise event nodes to form negative edges e_{ik}. Then, we fix the event node v_j and sample noise user nodes v_k to form negative edges e_{kj}. Now, how to sample noise node v_k is critical. The noise distribution $P_n(v) \propto d_v^{0.75}$ has been widely adopted [10], [21] to sample noise nodes, where d_v is the degree of node v. Let M be the number of noise nodes sampled from one side, and our objective function can be redefined as in Eqn. (4). $E_{v_k\sim P_n(v)} \log(1 - p(e_{ik} = 1))$ is the expected value of $\log(1-p(e_{ik} = 1))$. Thus, the task is equivalent to distinguish positive edge e_{ij} from the corresponding negative edges, using the logistic regression method.

A straightforward way to optimize Eqn. (4) is using the standard stochastic gradient descent (SGD). However, it will lead to a problem: when sampling a positive edge e_{ij} for model updating, its weight w_{ij} will be multiplied into the gradients. When the values of these edge weights are significantly diverse (i.e., within a wide range), the norms of the gradients also vary a lot, in which case it is extremely difficult to choose an appropriate uniform learning rate α. We adopt the approach of edge sampling proposed in [21]. We randomly sample positive

edges with the probability proportional to their weights and then treat the sampled edges as binary edges. Using this sampling strategy, the objective function remains the same and the learning process will not be affected by the variance of the edge weights. For each sampled positive edge e_{ij} with 2M negative edges, the embedding vectors $\vec{v}_i$, $\vec{v}_j$ and $\vec{v}_k$ will be updated as in Eqn. (5). Besides, we introduce the rectifier activation function to project the updated node vectors to non-negative values inspired by its great success in deep learning models. To further speed up the training process, we adopt the asynchronous stochastic gradient descent algorithm for model optimization, which guarantees the scalability and efficiency of our model in practice, following [15], [21].

$$\begin{aligned}\vec{v}_i &= \vec{v}_i + \alpha\Big((1 - f(\vec{v}_i^{\mathrm{T}}\vec{v}_j))\vec{v}_j - \sum_{k=1}^{M} E_{v_k\sim P_n(v)} f(\vec{v}_i^{\mathrm{T}}\vec{v}_k)\vec{v}_k\Big),\\ \vec{v}_j &= \vec{v}_j + \alpha\Big((1 - f(\vec{v}_i^{\mathrm{T}}\vec{v}_j))\vec{v}_i - \sum_{k=1}^{M} E_{v_k\sim P_n(v)} f(\vec{v}_k^{\mathrm{T}}\vec{v}_j)\vec{v}_k\Big),\end{aligned} \tag{5}$$

$$\vec{v}_k = \begin{cases}\vec{v}_k - \alpha f(\vec{v}_i^{\mathrm{T}}\vec{v}_k)\vec{v}_i, & v_k \in \mathcal{V}_B\\ \vec{v}_k - \alpha f(\vec{v}_k^{\mathrm{T}}\vec{v}_j)\vec{v}_j, & v_k \in \mathcal{V}_A.\end{cases}$$

Time Complexity Analysis. The time complexity for each stochastic gradient step is $O(K \cdot M) = O(K)$, where M is often very small (e.g., 1 or 2) in large-scale datasets [21], [10] and thus can be ignorable; K is the vector dimension and typically smaller than 100. We assume that our model needs N samples (i.e., N stochastic gradient steps) to reach convergence, thus the overall time complexity is $O(K \cdot N)$. In practice, the required number of stochastic gradient steps N is typically proportional to the number of edges [21].

B. Adaptive Sampler for Adversarial Negative Edges

The motivation of developing an adaptive noise sampler is to select a noise node v_k for a given edge e_{ij} to form a negative edge e_{ik} (or e_{kj}) such that the generated negative edge is sufficiently informative for the current values of the model parameters $\hat{\Theta}$ to effectively update their gradients. Θ is a set of model parameters (i.e., nodes' embeddings). We call the node v_i (or v_j) as the context node and this kind of negative edges as adversarial negative edges [4]. However, the widely adopted degree-based noise sampler [10], [21] and the recently proposed stratified sampling [1] cannot generate adversarial negative edges, because (1) they are static and thus do not consider that the estimated similarity between a pair of nodes changes during the learning process. For example, the similarity between v_i and a sampled noise node v_k is high at the beginning, but after several gradient steps it becomes low; and (2) these samplers are global and do not reflect how informative a noise node is w.r.t. a specific context node.

In light of this, we develop an adaptive noise sampler that considers both the context node and the current values of the model parameters $\hat{\Theta}$, inspired by [17]. Specifically, we propose a ranking-based noise distribution. Our intuition is that when sampling a noise node v_k for a context node v_i, the higher v_k is ranked, the more adversarial and informative the generated negative edge e_{ik} is. Below, we formally define our adaptive noise sampling distribution based on the predicted rank (i.e., $\hat{r}(v_k|v_c)$) of the noise node w.r.t. context node v_c:

$$P_n(v_k|v_c) \propto exp(-\hat{r}(v_k|v_c)/\lambda), \lambda \in \mathbb{R}^+, \tag{6}$$

where v_c can be v_i or v_j in the given positive edge. $\hat{r}(v_k|v_c)$ is the rank of node v_k w.r.t. the given context node v_c based on the currently estimated similarity score between them $f(\vec{v}_c^{\mathrm{T}}\vec{v}_k)$. Here we adopt the Geometric distribution to characterize our noise distribution, following [17]. λ is a hyper-parameter which tunes the probability density. Thus, when sampling a noise node v_k for a given context node v_c based on the above noise distribution, the higher v_k is ranked, the more likely it is to be sampled.

The noise distribution in Eqn. (6) depends on $\hat{r}(v_k|v_c)$ which is the rank of node v_k among all nodes in $\mathcal{V}_B$ (when v_i is the context node) based on their similarity score $f(\vec{v}_c^{\mathrm{T}}\vec{v}_k)$. Thus, the proposed sampler is ***adaptive***: 1) the sampling probability depends on the given context node v_c rather than being globally the same, and 2) the sampler (i.e., the noise distribution) is dynamically changing as the model parameters $\hat{\Theta}$ are updated, because the changes of $\hat{\Theta}$ would result in changes in the ranking $\hat{r}$.

Exact Implementation. Given a positive edge $e_{ij} \in \mathcal{E}_{AB}$, the noise sampler for context node v_i can be implemented as follows. First, sample a rank set $\mathcal{S}$ with size M from the Geometric distribution(i.e., $p(s) \propto exp(-s/\lambda), \lambda \in \mathbb{R}^+$) in $O(M)$. Second, for each rank $s \in \mathcal{S}$ return the node which is currently ranked at the s-th position, i.e., find noise node v_k that meets $\hat{r}(v_k|v_i) = s$ or $v_k = \hat{r}^{-1}(s|v_i)$. An exact implementation of the second step has to compute $f(\vec{v}_i^{\mathrm{T}}\vec{v}_k)$ for all $v_k \in \mathcal{V}_B$, then sort these nodes by their estimated similarity scores and finally return the M ones at positions $\mathcal{S}$. This algorithm has a complexity of $O(|\mathcal{V}_B| \cdot K + |\mathcal{V}_B| \cdot \log|\mathcal{V}_B|)$, which would inevitably increase the time complexity of our model optimization by a factor $O(|\mathcal{V}_B| \cdot K + |\mathcal{V}_B| \cdot \log|\mathcal{V}_B|)$. Therefore, the above noise sampler is not feasible in practice.

Approximate Implementation. To accelerate the negative sampling process, we propose an approximate and fast sampling algorithm for noise nodes: (1) Draw a rank set $\mathcal{S}$ from the Geometric distribution. (2) Draw a dimension f from $p(f|v_i) \propto v_{i,f} \cdot \sigma_f$ where $v_{i,f}$ denotes the value of $\vec{v}_i$ on dimension f, and $\sigma_f = Var(v_{\cdot,f})$. (3) Sort nodes $v_k \in \mathcal{V}_B$ according to their values on dimension f in a descending order, and the ranking is denoted as $\hat{r}^{-1}(\cdot|f)$. (4) Return M nodes in the positions $\mathcal{S}$ in the sorted list. Steps 1 and 4 can be performed efficiently in $O(1)$. But the time complexity of sampling a dimension f in step 2 is $O(|\mathcal{V}_B|)$, as the computation of σ_f needs to access all nodes in $\mathcal{V}_B$. Step 3 is also computationally intensive and its time cost is $O(|\mathcal{V}_B| \cdot \log|\mathcal{V}_B|)$. For context node v_j, repeat the above process and replace v_i and $\mathcal{V}_B$ by v_j and $\mathcal{V}_A$ respectively.

In order to further accelerate the adaptive sampling process, we propose to precompute the ranking $\hat{r}^{-1}(\cdot|f)$ of nodes in $\mathcal{V}_B$ for each of the K dimensions and recompute it every couple of gradient steps instead of each gradient step. After a single gradient step, both the variance value σ_f and the ranking

Algorithm 1: Fast Adaptive Negative Sampling Algorithm

Input: $\mathcal{G}_{AB}$, number of required gradient steps N, number of negative samples M;
Output: the embeddings $\vec{v}_i$ and $\vec{v}_j$ of each node $v_i \in \mathcal{V}_A$ and $v_j \in \mathcal{V}_B$;

Randomly initialize each $\vec{v}_i$ and $\vec{v}_j$ with Gaussian distribution;
Initialize $iter \leftarrow 0$;
while $iter \leq N$ **do**
  if $iter \% |\mathcal{V}_B| \log |\mathcal{V}_B| = 0$ **then**
    for $f \in \{1, \cdots, K\}$ **do**
      Compute ranking $\hat{r}_B^{-1}(\cdot|f)$ of nodes in $\mathcal{V}_B$;
      Compute $\sigma_{f,B}^2$;
    end
  end
  if $iter \% |\mathcal{V}_A| \log |\mathcal{V}_A| = 0$ **then**
    for $f \in \{1, \cdots, K\}$ **do**
      Compute ranking $\hat{r}_A^{-1}(\cdot|f)$ of nodes in $\mathcal{V}_A$;
      Compute $\sigma_{f,A}^2$;
    end
  end
  Draw $e_{ij} \in \mathcal{E}_{AB}$ from $p(e_{ij}) \propto w_{ij}$;
  Draw a rank set $\mathcal{S}_B$ with size M from $p(s) \propto exp(-s/\lambda)$;
  Draw dimension f from $p(f|v_i) \propto v_{i,f} \cdot \sigma_{f,B}$;
  for $s \in \mathcal{S}_B$ **do**
    Collect the noise node $r_B^{-1}(s|f)$;
  end
  Draw a rank set $\mathcal{S}_A$ with size M from $p(s) \propto exp(-s/\lambda)$;
  Draw dimension f from $p(f|v_j) \propto v_{j,f} \cdot \sigma_{f,A}$;
  for $s \in \mathcal{S}_A$ **do**
    Collect the noise node $r_A^{-1}(s|f)$;
  end
  Generate $2M$ adversarial negative edges based on the collected noise nodes;
  Update $\vec{v}_i$, $\vec{v}_j$ and each sampled noise node's vector $\vec{v}_k$ with Eqn. (5);
  iter=iter+1;
end

$\hat{r}^{-1}(\cdot|f)$ change only little and many gradient steps are needed to change them significantly. We propose to recompute the K ranking lists of nodes every $|\mathcal{V}_B| \cdot \log |\mathcal{V}_B|$ gradient updates, which showed good performance in the evaluation. The described precomputation strategy has an amortized runtime of $O(K)$ because every $|\mathcal{V}_B| \cdot \log |\mathcal{V}_B|$ iterations or gradient steps there is an effort to update the K ranking lists with the time complexity $O(K \cdot |\mathcal{V}_B| \cdot (1 + \log |\mathcal{V}_B|)) = O(K \cdot |\mathcal{V}_B| \cdot \log |\mathcal{V}_B|)$. On average, the noise sampling algorithm has an amortized runtime of $O(K)$ for drawing M noise nodes, and that is the same as the cost for a single gradient step. Thus, the time complexity of our model optimization does not increase, and it is still $O(N \cdot K)$ where N is the number of gradient steps. Algorithm 1 shows the detailed pseudocode.

C. Joint Multi-Graph Embedding Learning

There are five bipartite graphs in our problem. An intuitive approach is to collectively embed the five bipartite graphs by minimizing the sum of all objective functions as following:

$$O = O_{UX} + O_{XT} + O_{XC} + O_{XL} + O_{UU}. \tag{7}$$

The above objective function can be optimized by training all bipartite graphs simultaneously as follows. We first merge all edges in the five edge sets $\mathcal{E}_{UX}$, $\mathcal{E}_{XT}$, $\mathcal{E}_{XC}$, $\mathcal{E}_{XL}$ and $\mathcal{E}_{UU}$ into a big set, and then randomly sample an edge from the merged big set with the sampling probability proportional to its weight in each step. However, the edge weights of different bipartite graphs are not comparable to each other as they have different scales and semantics. To overcome the challenge of multiple bipartite graphs, we propose a novel joint training algorithm for multiple bipartite graphs, as shown in Algorithm 2. Instead of drawing edges directly for training, we will first draw a bipartite graph from the five bipartite graphs with the sampling probability proportional to its number of edges (Line 3), and then draw a positive edge and $2M$ negative edges from the sampled bipartite graph (Line 4-5). The probability of sampling a specific positive edge e_{ij} in Line 4 is proportional to it edge weight, and $2M$ negative edges are sampled in Line 5 using our adaptive noise sampler developed in Section III-B. Note that our proposed joint training algorithm is different from the one developed in [25], [20] where each bipartite graph is treated equally and has the same opportunity to be sampled at each step. It would result in that bipartite graphs with a small number of edges are over-exploited while those with a large number edges are under-exploited.

Algorithm 2: Joint Training of Multiple Bipartite Graphs

Input: $\mathcal{G}_{UX}$, $\mathcal{G}_{XT}$, $\mathcal{G}_{XC}$, $\mathcal{G}_{XL}$, $\mathcal{G}_{UU}$, number of samples N, number of negative samples M;
Output: The parameter set $\Theta = \{\vec{x}, \vec{l}, \vec{t}, \vec{c}, \vec{u}\}$, i.e., the event, location, time, content word and user vectors;

1 $iter \leftarrow 0$;
2 **while** $iter \leq N$ **do**
3 Draw a bipartite graph $\mathcal{G}_{AB}$ from the five input graphs with the probability proportional to its number of edges;
4 Draw a positive edge $e_{ij} \in \mathcal{E}_{AB}$;
5 Draw $2M$ negative edges for e_{ij} using fast adaptive negative sampling algorithm;
6 Update the associated nodes' vectors according to Eqn.(5);
7 iter=iter+1;
8 **end**

IV. Online Event-Partner Recommendation

A successful event-partner recommendation depends on not only whether the target user will adopt the recommendation, but also whether the recommended partner will accept the invitation from the target user. Thus, the decision process is much more complex than the conventional event recommendation. To efficiently model the complex decision process, we decompose the high order interrelationships among users, events and partners into three pair-wise interaction relationships, i.e., the triple (user, partner, event) is decomposed to (user, event), (partner, event) and (user, partner) pairs. Specifically, given a target user u and a recommended event-partner pair (u', x), the success probability is computed as follows:

$$\begin{aligned} f(u, u', x) &= \frac{1}{1 + \exp -(\vec{u}^{\mathrm{T}}\vec{x} + \vec{u'}^{\mathrm{T}}\vec{x} + \vec{u}^{\mathrm{T}}\vec{u'} + \beta)} \\ &\propto \vec{u}^{\mathrm{T}}\vec{x} + \vec{u'}^{\mathrm{T}}\vec{x} + \vec{u}^{\mathrm{T}}\vec{u'} \end{aligned} \tag{8}$$

where β is a constant bias. As only the ranking matters in the top-n recommendation problem, we only need to compute $\vec{u}^{\mathrm{T}}\vec{x} + \vec{u'}^{\mathrm{T}}\vec{x} + \vec{u}^{\mathrm{T}}\vec{u'}$. The above equation considers not only the target user's preference for the recommended event, but also the partner's preference for the event and the social proximity between the target user and the recommended partner.

TABLE I
BASIC STATISTICS OF DOUBAN EVENT DATASETS

	Beijing	Shanghai
# of users	64,113	36,440
# of events	12,955	6,753
# of venues	3,212	1,990
# of historical attendances	1,114,097	482,138
# of friendship links	865,298	298,105

Naive Method. Given a target user u, to generate top-n recommendations, a naive idea is to first compute a score for each event-partner pair $(u',\ x)$ as in Eqn. (8), and then select top-n pairs with largest scores as recommendations. However, this method is computationally inefficient, as it needs to compute the score for all event-partner pairs, and the number of possible event-partner combinations is extremely large, i.e., $|\mathcal{V}_U| \cdot |\mathcal{V}_X|$ where $|\mathcal{V}_U|$ and $|\mathcal{V}_X|$ are the numbers of users and events, respectively. Thus, the time complexity of this naive method is $O(|\mathcal{V}_U| \cdot |\mathcal{V}_X| \cdot K)$. Real world online recommender systems must adhere to strict response-time constraints, so scoring all event-partner pairs is infeasible.

To speed up the online top-n recommendations, some efficient online recommendation techniques [29], [24], [32], [31], [33], [30] have been developed to index and prune object search space. However, they cannot be straightforwardly applied to our event-partner recommendation problem, as these online recommendation techniques focus on finding the maximum (or top-n) dot-product for a given user vector over a set of item vectors, while our scoring function in Eqn. (8) cannot be expressed as a dot product between $\vec{u}$ and $(\vec{x}, \vec{u'})$.

Fast Online Recommendation. To take advantage of these off-the-shelf online recommendation techniques, we propose a novel space transformation method to map each event-partner pair to one point in a new space. Specifically, we create a new space with $2K + 1$ dimensions in which each event-partner pair (x, u') is mapped to one point $\vec{p}_{xu'} = (\vec{x}, \vec{u'}, \vec{u'}^{\mathrm{T}} \vec{x})$. Similarly, a user vector $\vec{u}$ can be also mapped to a point $\vec{q}_u = (\vec{u}, \vec{u}, 1)$ in the new space. Thus, the scoring function in Eqn. (8) can be reformulated as an inner product between an extended user vector $\vec{q}_u$ and an event-partner combination vector $\vec{p}_{xu'}$ in the new space. After the space transformation, we adopt the TA-based query processing technique developed in [32] to speed up event-partner recommendations in the new space. This technology has the nice property of finding top-n results correctly by examining the minimum number of event-partner pairs without scanning all ones.

The space transformation is performed offline. Thus, we need $O(|\mathcal{V}_U| \cdot |\mathcal{V}_X| \cdot K)$ space cost to store all event-partner vectors $\vec{p}_{xu'}$. To reduce the required storage space and further improve the efficiency of online recommendation, we propose to narrow down the search space by filtering out unpromising event-partner pairs (x, u'). In practice, we only need to store top-k events for each user u' instead of all events in $\mathcal{V}_X$ ($k \ll |\mathcal{V}_X|$), as the user u' will tend to refuse an invitation to attend her uninterested event x. Thus, the number of candidate event-partner pairs is reduced from $O(|\mathcal{V}_U| \cdot |\mathcal{V}_X|)$ to $O(|\mathcal{V}_U| \cdot k)$.

V. EXPERIMENTS

In this section, we first describe the experimental settings and then report the performance of our GEM models.

A. Dataset

As no benchmark datasets are available for evaluating the event-partner recommendation, we collected a real dataset by crawling Douban Event from Sep 2005 to Dec 2012. Douban Event is the largest online event-based social network in China that helps people publish and participate in social events. On Douban Event, an event is created by an individual or a business by specifying when, where and what the event is. Then, other users express their intent to attend the event by online registration. For each event, its content introduction, geographical location, start time information, and a list of registered users for attending were collected. For each user, we acquired her event attendance list and social friend list.

To simulate real scenarios, we first divide all events according to their located cities, and then choose Beijing and Shanghai, the two largest cities in China, to create two local event datasets. Besides, we filter out users who attended less than 5 events to remove noisy data. The basic statistics of our used datasets are shown in Table I.

Ground-Truth of Event Recommendation. To simulate a more real event recommendation scenario, we divide the events in Shanghai and Beijing into training and test sets based on their chronological order (i.e., their start time) with a widely adopted ratio of $7:3$. For each event in the test set, its associated attendance records of users are removed or marked off when training models, and thus the events in the test set can be regarded as new or cold-start ones. We further partition the test set into a validation set and a test set with a ratio of $1:2$. Thus, the event set $\mathcal{X}$ is divided into three disjoint sets $\mathcal{X}^{training}$, $\mathcal{X}^{validation}$ and $\mathcal{X}^{test}$. Correspondingly, given a user u, her attended events $\mathcal{X}_u$ can be divided into $\mathcal{X}_u^{training}$, $\mathcal{X}_u^{validation}$ and $\mathcal{X}_u^{test}$. Based on this dividing strategy, we can divide the edge set $\mathcal{E}_{UX}$ in the user-event graph $\mathcal{G}_{UX}$ into $\mathcal{E}_{UX}^{training}$, $\mathcal{E}_{UX}^{validation}$ and $\mathcal{E}_{UX}^{test}$.

Ground-Truth of Event-Partner Recommendation. For each event x in the test set $\mathcal{X}^{test}$, if two users u and u' are friends and attend x, we assume that they are suitable partners to each other w.r.t. event x, denoted as (u, u', x). In this way, we create an event-partner test set $\mathcal{Y} = \{(u, u', x) : x \in \mathcal{X}^{test}, u \in \mathcal{U}_x, u' \in \mathcal{U}_x, (u, u') \in \mathcal{E}_{UU}\}$. As our GEM does not limit partners to social friends, we simulate another scenario in which u and u' are potential friends and attend event x together, i.e., they will become friends in the future after attending event x. To mimic this scenario, for each user-partner pair (u, u') in $\mathcal{Y}$, we remove their social links from the graph $\mathcal{G}_{UU}$ when training models.

Hyper-parameter Optimization. There are some hyperparameters in our model, and we perform the cross-validation. Specifically, we use the conventional grid search algorithm to obtain the optimal hyper-parameter setup on the validation dataset, such as learning rate $\alpha = 0.05$, random Gaussian initialization $\mathcal{N}(0, 0.01)$, $M = 2$ and $N = 2,000,000$. We

also present the process and results of tuning some important hyper-parameters (e.g., K) in Section V-E.

B. Evaluation Method

In the following experiments, we aim to evaluate the performance of our GEM model in two tasks: cold-start event recommendation and event-partner recommendation. To measure the performance of our GEM model in these tasks, we adopt the evaluation methodology and metric that were proposed in [2], [32] to evaluate the top-n recommendations.

Evaluation of cold-start event recommendation. For each user-event pair $(u,\ x)$ in the test set $\mathcal{E}_{UX}^{test}$, we randomly choose 1000 events from $\mathcal{X}^{test} - \mathcal{X}_u$ as negative events, and then we compute the scores for event x and all negative events to perform ranking. Next, we form a top-n recommendation list by picking the n ones with highest scores. If event x (i.e., the ground truth) appears in top-n recommendations, we have a hit; otherwise we have a miss. Accuracy@n is computed as the hit ratio in the test set $\mathcal{E}_{UX}^{test}$, as follows:

$$Accuracy@n = \frac{\#Hit@n}{|\mathcal{E}_{UX}^{test}|} \tag{9}$$

where $\#Hit@n$ denotes the total number of hits in the test set, and $|\mathcal{E}_{UX}^{test}|$ is the number of all test cases.

Evaluation of event-partner recommendation. For each triple (u, u', x) in the test set $\mathcal{Y}$, we fix (u, u') and randomly select 500 events from $\mathcal{X}^{test} - (\mathcal{X}_u \bigcap \mathcal{X}_{u'})$ to replace x and form a set of negative triples. Similarly, we fix (u, x) and randomly choose 500 users from $\mathcal{U} - \mathcal{U}_x$ to replace u' and form another set of negative triples. Then, a score is calculated for (u, u', x) and the 1000 negative triples according to Eqn. (8). By ranking these triples in the descending order, we get the rank of the positive triple (u, u', x). If its rank is not larger than n, we have a hit; otherwise we have a miss. Accuracy@n is computed as the hit ratio in the test set $\mathcal{Y}$, as follows:

$$Accuracy@n = \frac{\#Hit@n}{|\mathcal{Y}|} \tag{10}$$

where $\#Hit@n$ denotes the total number of hits.

Evaluation of Model Scalability. As the asynchronous stochastic gradient descent is employed to train our model, we test its speedup ratio by varying the number of threads.

Evaluation of Recommendation Efficiency. The efficiency of the online recommendation mainly depends on the number of available event-partners in the new space and the number of event-partners in the recommendation list (i.e., n). Therefore, we test the efficiency of our fast online recommendation methods over these two factors.

C. Comparison Methods

We compare our GEM model with the following state-of-the-art recommender models. All the selected comparison methods are latent factor models that integrate both content and contextual information to learn latent vector representations of cold-start events, and moreover they can be straightforwardly extended to support event-partner recommendation using the same recommendation framework proposed in Section IV. The hyper-parameters of all the comparison methods are well-tuned on the validation dataset.

PCMF. PCMF [13] is a probabilistic collective matrix factorization model. It extends the BPR matrix factorization [18] to multiple matrices by assigning each entity a K-dimension vector that is shared across all the relations. Compared with our GEM, PCMF can only model the binary relations and employed uniform distribution to generate negative samples.

CBPF. CBPF [36] is a collective Bayesian Poisson factorization model for cold-start event recommendation. In CBPF, each user, location, time and textual document are represented by a K-dimension vector, and an event is modeled as a weighted average of the vector representations of its content, location and time. CBPF takes Bayesian Poisson factorization as its basic unit to model user response to events.

PER. Personalized Entity recommendation [34] first models the user-event interaction and other auxiliary information as a heterogenous information network, and then extract meta-path based latent features to represent the similarity between users and events along different types of meta paths.

PTE. PTE was proposed for large-scale heterogeneous text information network embedding in [20] and recently extended for POI recommendation by [25]. It adopts the degree-based noise sampler to generate negative edges from one side, as shown in Eqn. (3). Compared with PTE, our GEM uses a new bidirectional negative sampling method and a novel adaptive noise sampler to generate adversarial negative edges. Besides, PTE treats each bipartite graph equally and ignores their differences when performing joint training.

GEM-P. GEM-P means that our GEM adopts the degree-based noise sampler to generate negative edges.

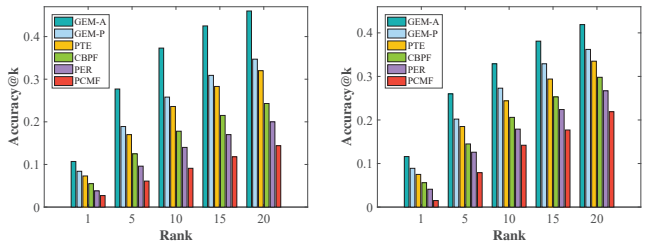
GEM-A. GEM-A means that our GEM adopts the adaptive noise sampler to generate adversarial negative edges.

As there is no existing method to solve our proposed new problem of joint event-partner recommendation, we try to extend the above comparison methods for event-partner recommendation using our proposed pairwise interaction-based recommendation framework in Section IV. Specifically, given a target user u and an event-partner pair (x, u'), we first use the above comparison methods to compute the preferences of u and u' for event x respectively, then compute the social affinity between u and u' based on their vector representations. Finally, we combine them together to compute the final matching score between user u and event-partner pair (x, u').

CFAPR-E. CFAPR was reported to be the best method for the activity-based partner recommendation in [22], and it aims to find suitable partners for a given user to attend a given event. Inspired by [23], we develop CFAPR-E for joint event-partner recommendation by extending CFAPR.

D. Results on Recommendation Effectiveness

In this section, we assess the effectiveness of our proposed GEM model for both cold-start event recommendation and event-partner recommendation. In particular, we aim to answer the following research questions. Q1. How effective is GEM for cold-start event recommendation? Q2. How effective is GEM for event-partner recommendation? Q3. Which negative sampling strategy or method is more effective, *unidirectional*



(a) Beijing (b) Shanghai

Fig. 3. Cold-start Event Recommendation.

sampling vs. *bidirectional sampling*, *uniform sampling* vs. *degree-based sampling* vs. *adaptive sampling*?

Figures 3, 4 and 5 report the comparison results between our GEM and other competitor models with well-tuned parameters. All differences between GEM and others are statistically significant ($p < 0.01$). We only show the performance where n is set to 1, 5, 10, 15, 20.

Cold-start Event Recommendation. Figure 3 presents the recommendation accuracy in the task of cold-start event recommendation, where the recommendation accuracy of GEM-A in Beijing is about 0.373 when $k = 10$ (i.e., the model has a probability of 37.3% of placing an appealing new event in the top-10). Clearly, our GEM-A outperforms other competitor models significantly, and the relative improvements, in terms of $Accuracy@10$, are 58%, 109.55%, 166.42% and 309.01% compared with PTE, CBPF, PER and PCMF, respectively.

Several observations are made from the results. 1) GEM-A, GEM-P and PTE outperform other methods significantly, showing the advantage of graph-based embedding models in the cold-start recommendation scenario. 2) Among the three graph-based embedding models GEM-A, GEM-P and PTE, GEM-A achieves the highest recommendation accuracy and GEM-P performs better than PTE, demonstrating the advantages of our proposed bidirectional negative sampling strategy and adaptive noise sampler, respectively. 3) Although PCMF looks similar to the graph-based embedding models GEM-A, GEM-P and PTE, it performs the worst. This is because PCMF can only model the binary relation between two entities, and fail to capture the strength of relations. Moreover, PCMF adopts the uniform distribution to generate negative cases while GEM-A, GEM-P and PTE use more advanced sampling strategies, i.e., our proposed adaptive sampling and degree-based sampling strategies. 4) CBPF performs worse than all the graph-based embedding models. To solve the cold-start issue, CBPF represents an event as the average latent feature vectors learnt from its auxiliary information. This scheme refrains CBPF from learning a more robust representation from the auxiliary information, since users choosing of events can be influenced by many unknown factors.

Joint Event-Partner Recommendation. Figures 4 and 5 report the performance of all the recommendation models in the task of joint event-partner recommendation. We report the comparison results in two different scenarios: partners are friends in Figure 4, and partners are potential friends in Figure 5. Obviously, our GEM models significantly outperforms the competitor models in these two scenarios. Although CFAPR-E adopts the vectors of users and events learned from GEM-A to compute $p(x|u)$, CFAPR limits the recommended partners to those who have been partners with u in the past, because it exploits the historical user-partner together data to produce partners. Besides, CFAPR cannot work for users who do not have the historical data of attending events with partners together. Another observation is that the recommendation accuracies of all models are lower in Figure 5 than in Figure 4. This is because the second scenario is more difficult. In the second scenario, we need to not only predict the events the users are interested in, but also find the partners with whom the users are most likely to become friends in the future.

E. Parameter Tuning and Sensitivity Analysis

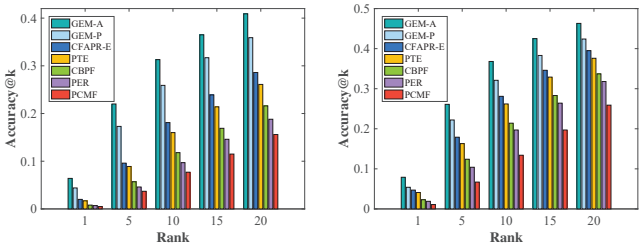
In the embedding models GEM-A, GEM-P and PTE, there is an important parameter: the number of samples N as shown in Algorithm 1. In this experiment, we study the converging performance of different models with increasing N. Due to space constraints, we only show the results of GEM-A, GEM-P and PTE w.r.t. the number of samples on the Beijing dataset in Tables II and III, and similar results are also achieved on the Shanghai dataset. From the results, we can see that GEM-A, GEM-P and PTE need 2 million, 4 million and 10 million samples to reach their convergence in both tasks. Therefore, we conclude that our GEM-A consistently outperforms GEM-P an PTE, and both GEM models converge much faster than PTE, which demonstrates the advantage of our proposed bidirectional sampling strategy and adaptive noise sampler.

We also investigate the impact of the dimension K on the top-10 recommendations in Beijing. Table IV reports the results, and we observe that the performances of all the three models first improve quickly with the increase of K and then the increments become ignorable. The dimension represents the model complexity. Thus, when K is too small, these models have limited ability to describe the data. However, when K exceeds a threshold (e.g., $K = 60$), the models are complex enough to handle the event-based social network data. At this point, it is less helpful to improve the model performance by increasing K, but will increase the time cost of model training. As a result, we choose the dimension $K = 60$ as the best trade-off between effectiveness and efficiency.

Another important parameter in our GEM-A model is λ for the Geometric distribution in Eq. (6). Table V reports the performance of GEM-A by varying λ value on the Beijing dataset. Similar observations are also made on the Shanghai dataset. From the results, we observe that the recommendation accuracy of GEM-A first increases with the increasing value λ and then does not vary much when λ is larger than 200.

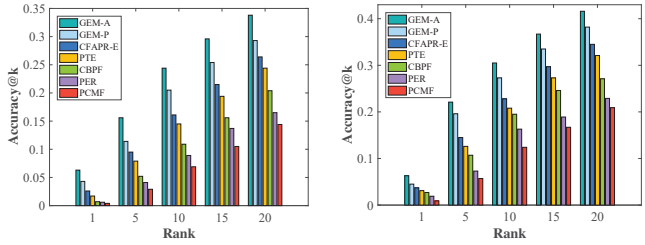
F. Scalability

In this experiment, we investigate the scalability of the GEM-A model optimized by the asynchronous stochastic gradient descent, which deploys multiple threads for optimization. Figure 6(a) shows the speedup ratio w.r.t. the number of threads on the Beijing dataset. The speedup ratio is quite close to linear. Figure 6(b) shows that the recommendation



(a) Beijing (b) Shanghai

Fig. 4. Joint Event-Partner Recommendation Scenario 1.



(a) Beijing (b) Shanghai

Fig. 5. Joint Event-Partner Recommendation Scenario 2.

TABLE II
COLD-START EVENT RECOMMENDATION IN BEIJING.

N	GEM-A		GEM-P		PTE	
	Ac@5	Ac@10	Ac@5	Ac@10	Ac@5	Ac@10
1 million	0.217	0.310	0.069	0.107	0.011	0.022
2 millions	**0.277**	**0.373**	0.134	0.194	0.013	0.023
3 millions	0.277	0.373	0.172	0.243	0.015	0.028
4 millions	0.277	0.373	**0.179**	**0.254**	0.033	0.052
5 millions	0.277	0.373	0.179	0.254	0.061	0.091
6 millions	0.277	0.373	0.179	0.254	0.096	0.141
7 millions	0.277	0.373	0.179	0.254	0.125	0.178
8 millions	0.277	0.373	0.179	0.254	0.147	0.208
9 millions	0.277	0.373	0.179	0.254	0.163	0.224
10 millions	0.277	0.373	0.179	0.254	**0.171**	**0.236**
15 millions	0.277	0.373	0.179	0.254	0.171	0.236

TABLE III
JOINT EVENT-PARTNER RECOMMENDATION IN BEIJING.

N	GEM-A		GEM-P		PTE	
	Ac@5	Ac@10	Ac@5	Ac@10	Ac@5	Ac@10
1 million	0.122	0.194	0.069	0.129	0.005	0.012
2 millions	**0.156**	**0.244**	0.089	0.163	0.006	0.014
3 millions	0.156	0.244	0.105	0.193	0.007	0.018
4 millions	0.156	0.244	**0.114**	**0.205**	0.013	0.031
5 millions	0.156	0.244	0.114	0.205	0.022	0.047
6 millions	0.156	0.244	0.114	0.205	0.031	0.061
7 millions	0.156	0.244	0.114	0.205	0.037	0.077
8 millions	0.156	0.244	0.114	0.205	0.046	0.097
9 millions	0.156	0.244	0.114	0.205	0.057	0.118
10 millions	0.156	0.244	0.114	0.205	**0.079**	**0.145**
15 millions	0.156	0.244	0.114	0.205	0.079	0.145

performance remains stable when using multiple threads for model updating. The two figures together show that the model optimization algorithm of GEM is quite scalable.

G. Recommendation Efficiency

This experiment is to evaluate the online event-partner recommendation efficiency where the partners are not limited to friends. For the online recommendation of GEM, we first use our proposed space transformation method to map each event-partner pair to one point in a new space. Then, we adopt two methods to produce top-n event-partner recommendations based on the new space. The first one is called GEM-TA which adopts the TA-based retrieval algorithm [32] to produce online recommendation. The second is called GEM-BF which uses a naive brute-force algorithm. Both methods were implemented in Java and run on a Windows Server with 200G RAM.

Table VI presents the average online efficiency on the Beijing dataset. We show the performance where n is set to 5, 10, 15 and 20. For example, on average GEM-TA finds the top-10 event-partner recommendations from about $2,590 \times 64,113$ event-partner pairs in 5.45s, where $2,590$ is the number of new events in the test set and $64,113$ is the number of users in

TABLE IV
IMPACT OF THE PARAMETER DIMENSION K.

K	Event Recommendation			Event-Partner Recommendation		
	GEM-A	GEM-P	PTE	GEM-A	GEM-P	PTE
20	0.339	0.235	0.218	0.223	0.193	0.126
40	0.365	0.248	0.231	0.240	0.201	0.139
60	**0.373**	**0.254**	**0.236**	**0.244**	**0.205**	**0.145**
80	0.373	0.254	0.236	0.244	0.205	0.145
100	0.373	0.254	0.237	0.244	0.205	0.146

TABLE V
IMPACT OF THE PARAMETER λ.

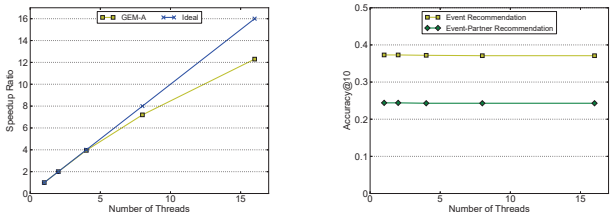
λ	Event Recommendation			Event-Partner Recommendation		
	Ac@5	Ac@10	Ac@20	Ac@5	Ac@10	Ac@20
50	0.228	0.312	0.396	0.105	0.165	0.245
100	0.261	0.354	0.446	0.128	0.194	0.282
150	0.276	0.363	0.448	0.154	0.239	0.331
200	**0.277**	**0.373**	**0.461**	**0.156**	**0.244**	**0.338**
500	0.277	0.372	0.460	0.156	0.244	0.338

the Beijing dataset. From the results, we observe that GEM-TA outperforms GEM-BF significantly, which demonstrates that the TA-based query processing technique can significantly speed up the process of online event-partner recommendation. By tracking the computation process of GEM-TA, we find it only needs to access around 8% of all the event-partner pairs on average for top-10 recommendations.

TABLE VI
ONLINE RECOMMENDATION EFFICIENCY.

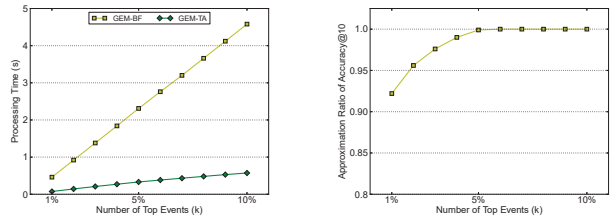
Methods	Online Recommendation Time Cost (s)			
	n=5	n=10	n=15	n=20
GEM-TA	2.21	4.45	7.65	9.28
GEM-BF	45.34	45.75	45.89	45.94

To further improve the efficiency of online recommendation, we propose to narrow down the search space by filtering out unpromising event-partner pairs. In practice, we only need to store top-k events for each user u' instead of all events. Next, we study the performance of GEM-TA and GEM-BF in top-10 recommendations by varying k from 1% to 10% of the total events, and present the results in Figure 7. From Figure 7(a), we observe that the time cost of GEM-BF is linear to k, as expected. The online recommendation time cost of GEM-TA is approximately linear to k, but much lower than GEM-BF. Figure 7(b) shows that the approximation ratio of Accuracy@10 achieved by our GEM model is extremely close and even equal to 1 when k is larger than 5% of the total number of events. The approximation ratio is defined as the recommendation accuracy achieved in the pruned search space that stores only top-k events for each user divided by the recommendation accuracy obtained in the original search



(a) Speedup vs. #threads (b) Accuracy@10 vs. #threads

Fig. 6. Performance w.r.t. the number of threads.



(a) Time cost vs. k (b) Approximation Rate vs. k

Fig. 7. Performance w.r.t. the number of top-events (k).

space with all event-partner combinations. The two figures together show that our proposed fast online event-partner recommendation techniques are quite scalable and efficient.

VI. Related Work

In general, our work is closely related with event recommendation, event-partner recommendation and graph embedding.

A. Event Recommendation

With the increasing popularity of event-based social networks [8], event recommendation is becoming an important means to help people discover their interested events. [14] developed a Bayesian matrix factorization approach using the Bayesian Personalized Ranking optimization criterion for event recommendation from the implicit feedback. Recently, Qiao et al. [13] extended their previous recommender model [14] into a collective matrix factorization model [19] to jointly model user-event, event-location and social relations. But, they ignored the content and time information of events. Zhao et al. [37] used a supervised learning model to predict the event attendance based on the extracted semantic, temporal and spatial features. Pham et al. [12] developed a general-purpose Heterogeneous graph-based Recommendation System model (HeteRS) on EBSN data. HeteRS used a multivariate Markov chain (MMC) to compute the proximity between a pair of nodes. However, HeteRS cannot separate the model training process from the online recommendation, compared with the latent factor models. The computation of MMC on the graph is very time-consuming, resulting in an unbearably long response time (hundreds of and even thousands of seconds).

Cold-start Event Recommendation. Most of the above works ignore that event recommendation is intrinsically a cold start problem. To solve the cold-start issue in event recommendation, Zhang et al. [36] proposed a collective Bayesian Poisson factorization model (CBPF). CBPF represents an event as the average latent vectors learned from its auxiliary information. This scheme refrains CBPF from learning a more robust representation, since users choosing of events can be influenced by many unknown factors which are not captured by the auxiliary information. Macedo [9] combined content, social, spatial and temporal signals and used the list-wise learning to rank approach for event recommendation. Their method is not a generic recommendation method and needs specialized recommender model tailored to each signal. Thus, it is hard to be extended for event-partner recommendation without much effort. That is why we do not compare with this approach in the experiment.

B. Event-Partner Recommendation

Although our proposed event-partner recommendation looks similar to the group recommendation [35], activity-based partner recommendation [7], [22], partner-aware activity recommendation [23] and participant recommendation [6], there are significant differences between them. First, partners are known (i.e., inputs) in the group recommendation, while partners are part of recommendation results (i.e., outputs) in our event-partner recommendation. Second, both a target user and an activity/event are given (i.e., inputs) in the activity-based partner recommendation, while events are part of recommendation results (i.e., outputs) in our proposed new recommendation problem. Third, an event is given (i.e., inputs) in the participant recommendation, and the aim is to find suitable participants for the given event; while a target user is given (i.e., inputs) in our proposed event-partner recommendation, and both events and partners are recommendation results (i.e., outputs). Fourth, although the availability of partners is exploited and integrated in the partner-aware activity recommendation [23], the purpose is to improve the traditional activity/event recommendation rather than providing joint event-partner recommendation.

In summary, there are only one type of entities as the recommendation results in these existing recommendation paradigms on EBSNs, but the recommendation results in our proposed event-partner recommendation are combinations of two types of interdependent entities. Obviously, our proposed new recommendation paradigm is more difficult than these existing ones, and our developed GEM model can be applied to all existing recommendation problems on EBSNs.

C. Scalable Information Network Embedding

Recently, there are some works attempting to embed large-scale information networks into low dimensional spaces. Perozzi et al. [11] proposed a random walk-based embedding model called DeepWalk, but it is only applicable for large-scale networks with binary edges. Grover et al. [5] extended DeepWalk by designing a biased random walk procedure and proposed a network embedding model called node2vec. Tang et al. [21] developed a network embedding model, called LINE, based on the word2vector model Skip-gram, and it is suitable for networks with both binary and weighted graphs. All Deepwalk, LINE and node2vec can only handle single homogeneous networks. Tang et al. [20] further extended LINE and proposed PTE to deal with heterogeneous bipartite graphs. However, all these existing network embedding models

adopt the degree-based sampling strategy to generate negative edges for the model optimization. Although PTE is designed for handling heterogenous bipartite graphs, it still samples noise nodes from only one side to construct negative edges. Besides, when performing joint training of multiple bipartite graphs, PTE treats each bipartite equally and ignores their differences (e.g., edge distributions).

Compared with these existing graph embedding models, we develop a new bidirectional negative sampling method and a novel adaptive noise sampler to generate adversarial negative edges for model optimization. Besides, our proposed joint training approach achieves a good sampling balance among multiple bipartite graphs due to its consideration of the skewness of edge distributions over different graphs.

VII. Conclusions

In this paper, we proposed a novel problem of joint event-partner recommendation in EBSNs, which is extremely challenging due to the intrinsic cold-start property of events, the more complex decision-making process and the huge prediction space. To address the issue of cold start, we proposed a bipartite graph-based embedding model to embed multiple heterogenous relations into a shared low-dimensional space so that the representations of cold-start events can be learned from their associated content and contextual information. To speed up the convergence of our model, we proposed a bidirectional negative sampling method and a novel adaptive noise sampler to generate adversarial negative samples. Besides, we proposed a novel space transformation method to project each event-partner pair to one point in a new space, based on which effective space pruning and efficient online recommendation techniques were developed. Extensive experiments were conducted, and the experimental results showed the superiority of our proposed event-partner recommendation techniques.

Acknowledgement

This work was supported by ARC Discovery Early Career Researcher Award (Grant No. DE160100308), ARC Discovery Project (Grant No. DP170103954) and New Staff Research Grant of The University of Queensland (Grant No.613134). It was also partially supported by National Natural Science Foundation of China (Grant No.61572335).

References

[1] T. Chen, Y. Sun, Y. Shi, and L. Hong. On sampling strategies for neural network-based collaborative filtering. In *KDD*, pages 767–776, 2017.

[2] P. Cremonesi and Y. Koren. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys*, pages 39–46, 2010.

[3] R. Du, Z. Yu, T. Mei, Z. Wang, Z. Wang, and B. Guo. Predicting activity attendance in event-based social networks: Content, context and social influence. In *UbiComp*, pages 425–434, 2014.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014.

[5] A. Grover and J. Leskovec. Node2vec: Scalable feature learning for networks. In *KDD*, pages 855–864, 2016.

[6] J. Jiang and C. Li. Analyzing social event participants for a single organizer. In *ICWSM*, pages 599–602, 2016.

[7] Y. Liao, W. Lam, S. Jameel, S. Schockaert, and X. Xie. Who wants to join me?: Companion recommendation in location based social networks. In *ICTIR*, pages 271–280, 2016.

[8] X. Liu, Q. He, Y. Tian, W.-C. Lee, J. McPherson, and J. Han. Event-based social networks: linking the online and offline social worlds. In *KDD*, pages 1032–1040, 2012.

[9] A. Q. Macedo, L. B. Marinho, and R. L. Santos. Context-aware event recommendation in event-based social networks. In *RecSys*, 2015.

[10] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[11] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *KDD*, pages 701–710, 2014.

[12] T. A. N. Pham, X. Li, G. Cong, and Z. Zhang. A general graph-based model for recommendation in event-based social networks. In *ICDE*, pages 567–578, 2015.

[13] Z. Qiao, P. Zhang, Y. Cao, C. Zhou, L. Guo, and B. Fang. Combining heterogenous social and geographical information for event recommendation. In *AAAI*, pages 145–151, 2014.

[14] Z. Qiao, P. Zhang, C. Zhou, Y. Cao, L. Guo, and Y. Zhang. Event recommendation in event-based social networks. In *AAAI*, 2014.

[15] B. Recht, C. Re, and S. Wright. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, pages 693–701, 2011.

[16] S. Rendle. Pairwise interaction tensor factorization for personalized tag recommendation. In *WSDM*, pages 81–90, 2010.

[17] S. Rendle. Improving pairwise learning for item recommendation from implicit feedback. In *WSDM*, pages 273–282, 2014.

[18] S. Rendle, C. Freudenthaler, and Z. Gantner. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.

[19] A. P. Singh and G. J. Gordon. Relational learning via collective matrix factorization. In *KDD*, pages 650–658, 2008.

[20] J. Tang, M. Qu, and Q. Mei. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *KDD*, 2015.

[21] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *WWW*, 2015.

[22] W. Tu, D. W. Cheung, N. Mamoulis, M. Yang, and Z. Lu. Activity-partner recommendation. In *PAKDD*, pages 591–604, 2015.

[23] W. Tu, D. W. Cheung, N. Mamoulis, M. Yang, and Z. Lu. Activity recommendation with partners. *ACM Trans. Web*, 12(1):4:1–4:29, Sept. 2017.

[24] W. Wang, H. Yin, S. Sadiq, L. Chen, M. Xie, and X. Zhou. Spore: A sequential personalized spatial item recommender system. In *ICDE*, pages 954–965, 2016.

[25] M. Xie, H. Yin, H. Wang, and F. Xu. Learning graph-based poi embedding for location-based recommendation. In *CIKM*, 2016.

[26] Z. Yao, Y. Fu, B. Liu, Y. Liu, and H. Xiong. Poi recommendation: A temporal matching between poi popularity and user regularity. In *ICDM*, pages 549–558, 2016.

[27] M. Ye, D. Shou, W.-C. Lee, P. Yin, and K. Janowicz. On the semantic annotation of places in location-based social networks. In *KDD*, pages 520–528, 2011.

[28] H. Yin, H. Chen, X. Sun, H. Wang, Y. Wang, and Q. V. H. Nguyen. Sptf: A scalable probabilistic tensor factorization model for semantic-aware behavior prediction. In *ICDM*, pages 585–594, 2017.

[29] H. Yin and B. Cui. *Spatio-temporal recommendation in social media*. Springer, 2016.

[30] H. Yin, B. Cui, Y. Sun, Z. Hu, and L. Chen. Lcars: A spatial item recommender system. *ACM Trans. Inf. Syst.*, 32(3):11:1–11:37, July 2014.

[31] H. Yin, B. Cui, X. Zhou, W. Wang, Z. Huang, and S. Sadiq. Joint modeling of user check-in behaviors for real-time point-of-interest recommendation. *ACM Trans. Inf. Syst.*, 35(2):11:1–11:44, Oct. 2016.

[32] H. Yin, Y. Sun, B. Cui, Z. Hu, and L. Chen. Lcars: A location-content-aware recommender system. In *KDD*, pages 221–229, 2013.

[33] H. Yin, X. Zhou, B. Cui, H. Wang, K. Zheng, and Q. V. H. Nguyen. Adapting to user interest drift for poi recommendation. *IEEE Trans. on Knowl. and Data Eng.*, 28(10):2566–2581, Oct. 2016.

[34] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han. Personalized entity recommendation: A heterogeneous information network approach. In *WSDM*, pages 283–292, 2014.

[35] Q. Yuan, G. Cong, and C.-Y. Lin. Com: A generative model for group recommendation. In *KDD*, pages 163–172, 2014.

[36] W. Zhang and J. Wang. A collective bayesian poisson factorization model for cold-start local event recommendation. In *KDD*, 2015.

[37] X. Zhang, J. Zhao, and G. Cao. Who will attend? – predicting event attendance in event-based social network. In *MDM*, pages 74–83, 2015.