# Deep Learning Driven Venue Recommender for Event-Based Social Networks

Soumajit Pramanik, Rajarshi Haldar, Anand Kumar, Sayan Pathak, and Bivas Mitra

**Abstract**—Event-based online social platforms, such as Meetup and Plancast, have experienced increased popularity and rapid growth in recent years. In EBSN setup, selecting suitable venues for hosting events, which can attract a great turnout, is a key challenge. In this paper, we present a deep learning based venue recommendation system $DeepVenue$ which provides context driven venue recommendations for the *Meetup event-hosts* to host their events. The crux of the proposed model relies on the notion of similarity between multiple Meetup entities such as events, venues, groups, etc. We develop deep learning techniques to compute a compact descriptor for each entity, such that two entities (say, venues) can be compared numerically. Notably, to mitigate the scarcity of venue related information in Meetup, we leverage on the cross domain knowledge transfer from popular LBSN service Yelp to extract rich venue related content. For hosting an event, the proposed $DeepVenue$ model computes a success score for each candidate venue and ranks those venues according to the scores and finally recommend the top k venues. Our rigorous evaluation on the Meetup data collected for the city of Chicago shows that $DeepVenue$ significantly outperforms the baselines algorithms. Precisely, for 84 percent of events, the correct hosting venue appears in the top 5 of the $DeepVenue$ recommended list.

**Index Terms**—Deep learning, venue recommender system, transfer learning, EBSN

## 1 Introduction

Meetup is an event based social networking (EBSN) portal for hosting events in various localities around the world[1] [1]. Event recommendation is an important problem in Meetup [1], [2], [3]. Prior studies recommend suitable events to a single or a group of users based on users' past preferences and current context [1], [2], [3] considering factors such as users' profile, location and social features [4]. However, only few attempts have been made to provide guidance to the Meetup hosts for organizing popular events. For instance, portals like 'Peerspace'[2] display the availability of venues to host a few specific types of events (say, Corporate Dinner, Photo Shoot, Retreat, Wedding etc); however, they do not provide any intelligent recommendation.

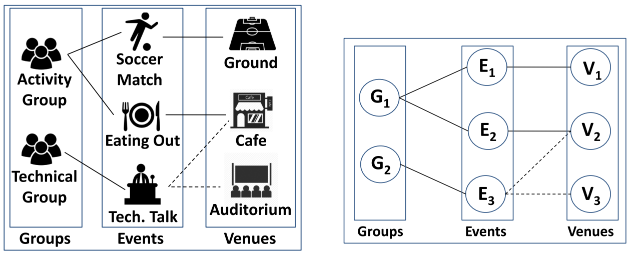
Identifying suitable venue from multiple options to organize a successful Meetup event (attracting large population) is essential for the event hosts. For instance, a technical talk can be hosted in an auditorium as well as in a cafeteria depending on the size and type of the event (see Fig. 1a). In [5] Liu et al. demonstrate a sharp decrease in event participation with increasing distance between attendees and the event venue. While recommending suitable events [1], [2] and groups [6] to Meetup users, researchers have taken into account the users' preference of visiting venues extensively. Motivated by these endeavors, in this paper, we propose a venue recommendation methodology for Meetup hosts to organize popular events.

Recommending suitable venues involves multiple challenges. 'Data sparsity' is a major challenge. In general, most user-item recommender systems [7], [8] learn user preferences based on past user-item interaction histories. Unique to the venue recommendation problem is the lack of history for a particular event. Second, a widely adopted approach to mitigate data sparsity in user-item recommendation is to use additional metadata regarding users & items. For instance, in the context of Point-of-Interest (POI) recommendations in Location Based Social Network (LBSN), recent studies [9], [10] have leveraged on content information of prior visited locations to infer user interests. Unfortunately, Meetup does not provide any additional semantic information associated with venues. Third, the popularity of an event also depends on the preference of the group members. For instance, two Meetup groups 'Chicago Bar Trivia Group' and 'The Fun Chicago Single' have availed 'Bull and Bear' restaurant in Chicago to host their events. In case of the former one, all their events at 'Bull and Bear' have attracted more than 30 percent members whereas for the later one attendance is below 5 percent for all the events hosted at the same restaurant. Finally, the choice of the venues for hosting successful Meetup events depend on the event specific factors. For instance, a thoughtful conversation requires location where the noise level is low, whereas

1. http://www.meetup.com/about/
2. https://www.peerspace.com

• S. Pramanik, R. Haldar, A. Kumar, and B. Mitra are with the Department of Computer Science & Engineering, IIT Kharagpur, West Bengal 721302, India. E-mail: soumajit.pramanik@cse.iitkgp.ernet.in, {rajarshi.haldar, anandsit043}@iitkgp.ac.in, bivas@cse.iitkgp.ac.in.
• S. Pathak is with Microsoft AI & Research, Redmond, WA 98052. E-mail: sayanpa@microsoft.com.

*Manuscript received 12 Mar. 2018; revised 29 Aug. 2018; accepted 22 Apr. 2019. Date of publication 8 May 2019; date of current version 6 Oct. 2020. (Corresponding author: Soumajit Pramanik.) Recommended for acceptance by S. Yan. Digital Object Identifier no. 10.1109/TKDE.2019.2915523*



(a) Venue recommendation in EBSN (b) Graphical representation

Fig. 1. Pictorial and graphical representations of venue recommendation problem in EBSN.

a corporate presentation needs a space with high quality audio-visual facility. In a nutshell, the suitability of a specific venue for hosting an event depends on the complex interactions across multiple entities such as groups, events etc.

The aforementioned challenges reflect that venue recommendation for hosting popular Meetup events involves multiple entities interacting and influencing each other (see Fig. 1a). Illustrated in Fig 1b, this multi-entity recommendation problem can be easily translated into a multipartite graph problem, where the graph based approaches [11], [12] appear to provide straightforward solutions.

Unfortunately, there exist quite a few limitations for the existing graph based approaches. (a) First, most of the typical graph based approaches tend to aggregate past interaction history, which results in severe information loss of temporal interactions. (b) Second, performance of the graph based approaches drops sharply with increasing data sparsity. As a result, this type of methodologies fail to perform in situations where there is a lack of historical information required for building the graph. On the contrary, a recommendation system is particularly needed when historical data is lacking.

In this paper, we propose a *Deep* learning based *Venue* recommendation engine (*DeepVenue*) which recommends a ranked list of venues for hosting a target event (Section 4). To the best of our knowledge, this is the first attempt involving deep learning for handling multi-entity interactions. The functionality of *DeepVenue* is based on a simple idea. A Meetup event can be successfully hosted by a group at a particular venue if (a) *similar* type of events have been recently hosted by *similar* type of groups at the same venue successfully (b) the venue is *similar* to the type of venues where *similar* type of events have been recently hosted successfully. Evidently, the proposed model relies on the notion of similarity between multiple entities (events, venues, groups etc.). However, each of these entities carry multiple complex perspectives, which makes its representation challenging. For instance, representation of a venue includes reviews and tips posted by the clients for that venue, services and facilities available at that venue (say WiFi, garage, parking etc.), type & category of the venue (say 'Restaurants', 'Nightlife', 'Shopping', 'Hotels', 'Mass Media' etc.) etc. Hence, our first step is to develop a compact descriptor for each entity, such that two entities (say, venues) can be compared numerically. We develop a deep learning based framework to compute compact descriptors for Meetup entities through embedding (Section 5). Moreover, Meetup does not provide the API necessary for collecting the requisite venue details (except locational information). Close inspection reveals that in Location Based Social Networks (LBSN) such as Yelp, FourSquare etc., people provide substantial volume of information about the venues [13], [14]. We leverage on the cross domain knowledge transfer [15] from popular LBSN services to extract rich venue related content for supplementing the Meetup venue details.

We evaluate the performance of *DeepVenue* based on the Meetup data collected for the city Chicago (Section 6). We introduce state-of-the-art competing algorithms as well as evaluation metrics for comparative study and show that *DeepVenue* significantly outperforms the baselines such as 'Successive Event Recommendation Based on Graph Entropy' (*SERGE*) [16], 'Deep Cooperative Neural Networks' (*DeepCoNN*) [17], 'Collaborative Topic Regression' (*CTR*) [18]. Details of the evaluation results are provided in Section 7. Compared to the baseline algorithms, *DeepVenue* works especially well for the cases where the hosting venue does not have any prior history of organizing any event or the target event does not have enough history of similar events. Finally, we dissect *DeepVenue* by examining the model deficiencies as well as the robustness, and implement multiple model variations (Section 8).

## 2 Related Work

In this section, first we present a brief description of the state of the art literature on recommendations in event and location based social network. Heterogeneous network is a novel framework to represent the multi-entity interactions; we present the work done in the gamut of recommendations in heterogeneous networks. Finally we present the potential of the deep learning literature to address the recommendation problems.

### 2.1 Recommendations in Event & Location Based Social Networks

Depending on the target users, the recommendation systems on EBSN can broadly be categorized into two classes (a) recommendation for general Meetup members and (b) recommendation for Meetup group organizers and event hosts. Major portion of the literature concentrated on the general recommendation for Meetup members, where most of the endeavors focused on recommending suitable events to a single or collection of Meetup users. Multiple approaches have been chosen for event recommendation, such as collaborative filtering [19], machine learning techniques [1], [2], graph-based models [12]. generative models [3] etc. Apart from event recommendation, few attempts have been made in recommending potential Meetup groups to the members for joining. For instance, in [4] the authors proposed a matrix-factorization based approach to recommend groups, which considers location, social features, as well as implicit user-preference patterns together in a unified model. Attempts have been made in bits and pieces to develop recommendation system for Meetup group organizers and event hosts. For instance, in [20] authors proposed recommendation system for group organizers which can help to identify the most influential and preferable members as invitees. However, in the EBSN literature, researchers mostly overlooked the problem of venue recommendation for hosting popular events, which is major concern for Meetup group organizers.

In the context of Location Based Social Networks (LBSN), stand-alone point-of-interest (POI) recommendation recommends customers with individual venues (say restaurant or shopping mall) for future visits, depending on their past visits and preferences. For instance, Liu et al. [9] proposed a geographical probabilistic factor model (Geo-PFM) which strategically incorporates customers' mobility patterns as well as the geographical influences on their subsequent check-in behaviors. Gupta et al. [14] demonstrated the combined effect of reviews and tips posted by customers on location recommendation in Yelp by proposing a graph-based framework. In our paper, we have extended this framework for Meetup venue recommendation by implementing the baseline algorithm *Graph*. Additionally, apart from single POI recommendations, researchers have worked on trajectory recommendation where a sequence of locations are recommended (e.g., a popular travel route) to a visiting user [21]. Albeit the presence of rich venue recommendation literature in LBSN, they cannot be directly adopted for EBSN due to the following reasons - (a) first, unlike LBSN where target customer's business preferences can be easily mined from her past visit patterns, in case of EBSN, no history is available for the target event, (b) second, the LBSN literature is mostly customized for typical two entity (customer-business) recommendation and hence, cannot be simply extended for recommendation amidst multi-entity (group-event-venue) interactions present in EBSN, (c) third, in LBSN there exists ample metadata for each business (reviews, tips, description, category, services) and customer (check-in history, reviews, tips, social connections) which is largely missing for venues and events in EBSN.

## 2.2 Recommendations in Heterogeneous Networks

Venue recommendation for hosting popular Meetup events is a perfect example of a recommendation problem where multiple entities interact and influence each other. The suitability of a specific venue depends on the complex interactions across multiple entities such as groups, events etc. Since, multi-entity interactions can be efficiently represented using heterogeneous networks, venue recommendation problem can be suitably mapped to recommendation in a heterogeneous network. Recent works on recommendation in heterogeneous networks can be classified into three major categories - graph based, latent factor based and topic model based.

(a) *Graph based models* represent entities (e.g., users, items etc.) as nodes and the relations among them as edges in a heterogeneous network structure and apply various graph mining techniques for recommendation. Most graph based algorithms adopt a Random Walk with Restart (RWR) [22] approach for generating recommendation scores [16], [22]. However, as RWR is unable to discriminate among different types of entities present in the graph, this model fails to realize the influence strength across various types of entities. In contrast, Pham et al. proposed a multivariate Markov chain (MMC) [12] model which is able to explicitly quantify the influence between different types of entities.

(b) *Factorization models* work well on feature rich datasets accompanying plenty of metadata (e.g., tags, categories etc.) as they provide a natural way to incorporate extra features and higher order interactions in the model. State of the art factorization techniques include collective matrix factorization [23], regression-based latent factor model [4], and Factorization Machine [24]. However, these models primarily focus on discovering the interactions among entities, not the types of entities themselves, making it difficult to realize the role of each entity type and customize the recommendation results accordingly.

(c) *Topic Models* extract user and item preferences as latent topics with the aid of extra information. Those models may be applied in location and tweet recommendation [25], news article recommendation [26], and POI recommendation [10]. Topic models are usually less flexible and domain dependent as they require assumptions on dependencies among domain-specific variables.

## 2.3 Deep Learning Based Recommendation

In recent years, deep learning has made radical advancements in recommendation architecture. Deep learning effectively captures latent user-item relationships and represents complex abstractions in higher layers. Deep learning based recommender systems can be classified into following two types.

(a) *Single deep learning techniques:* The advantage of adopting single deep model approach is that we can leverage on the strength of that particular deep learning technique depending on the application scenario. For instance, MLPs [27] can be used to model non-linear relationships between users and items, CNNs (Convolutional Neural Networks) are capable of extracting local and global representations from both textual and visual information [28] and RNNs (Recurrent Neural Networks) can be used to model the temporal dynamics of rating data and sequential influences of content information [29].

(b) *Composition of multiple deep, non-deep techniques:* Collaborative Deep Learning (CDL) [30] model performs article recommendation by tightly coupling deep learning for content information and collaborative filtering [31] for ratings. This shows a significant improvement over the widely used Collaborative Topic Regression (CTR) [18]. However, CDL is a feed-forward model that uses bag-of-words as input and hence, could not model the order aware generation of sequences (such as movie plot). In order to address this limitation, Collaborative Recurrent Autoencoder (CRAE) [32] was proposed, which combines collaborative filtering (CF) along with Recurrent Neural Networks (RNNs), Autoencoders (AE) and denoisers to perform accurate recommendations on real world dataset from various domains such as CiteULike and Netflix. Recently, Zheng et al.[17] proposed Deep

Cooperative Neural Networks (DeepCoNN) model that learns hidden latent features for users and items jointly using two coupled neural networks such that the rating prediction accuracy is maximized. Each of these two coupled networks consists of a convolution layer, a max-pooling layer and a densely connected layer. In our paper, we have implemented $DeepCoNN$ model as a baseline algorithm for Meetup venue recommendation.

## 3 Terminologies and Problem Definition

We first provide the terminologies and notations required to explain different entities present in Meetup. Next, we state the Meetup venue recommendation problem.

### 3.1 Meetup Preliminaries and Terminologies

Meetup is a popular Event Based Social Network, providing opportunities for people to expand their online interactions to physical interactions and participate in real-world events. Leveraging on this platform, users can organize and participate in a variety of events such as watching movie together, hosting a dinner, organizing a group travel etc for socialization. The core goal of such a portal is to bring together like minded people (primarily living in same city) to get engaged in different activities.

#### *3.1.1 Meetup Members and Groups*

The profile of one member in Meetup gets specified by a set of *Tags*, which reflects their respective preferences. Whenever one member joins Meetup, she is asked to choose tags for describing her interests. For instance, 'Dining Out', 'Fitness', 'Live Music', 'Travel', 'Art' are few of the most popular tags used by the members. Like minded members in Meetup join together online and form a *Meetup group* (let, $\mathcal{U}$ & $\mathcal{G}$ denote all members and groups in Meetup). Similar to members, profile of a group also gets specified by the Tags. Whenever a Meetup group gets formed by the group organizer, she is asked to provide a short textual description and select a set of tags which describe the group best. In addition, *Category* field characterizes the activity type performed by the Meetup group. For instance groups from 'Tech' category mostly indulge in hosting technical conversations, whereas 'Fine Arts/Culture' category groups prefer hosting classes and workshops related to a specific art. During formation, each group is assigned to one of the 33 'official' categories defined in Meetup. For example, few popular Meetup categories are 'Career/Business', 'Tech', 'Health/Wellbeing', 'Socializing' etc.

#### *3.1.2 Meetup Events*

In Meetup, groups organize events related to their interests at different venues (let, $\mathcal{E}$ & $\mathcal{V}$ denote all events and venues in Meetup). Meetup members (from inside & outside of the organizing group) physically attend those events for interactions. The *success* or *popularity* of a Meetup event is measured by the population attending that event. Meetup provides a field called "Headcount" which shows the total attendance information of the event. However, this count does not provide the details of the individual attendees, which can be obtained from the RSVP message {"Yes", "No", "Maybe"}. Following a conservative approach, we only consider attendees of an event as the people who send "Yes" responses to RSVPs corresponding to that event. Additionally, for each event, we obtain a short textual description stating the detail of the event.

TABLE 1
Notations

| Notation | Description |
|---|---|
| $\mathcal{E}, \mathcal{G}, \mathcal{V}, \mathcal{U}$ | Set of all events, groups, venues and members. |
| $e^*$ | Target event for recommendation. |
| $g^*$ | Group hosting the target event $e^*$. |
| $v^*$ | A candidate venue for hosting the target event $e^*$. |
| $v_1, v_2, \dots v_N$ | Chronological sequence of venues hosting $N$ most recent popular events similar to $e^*$. |
| $e_1, e_2, \dots e_N$ | Chronological sequence of $N$ most recent popular events hosted at $v^*$. |
| $g_1, g_2, \dots g_N$ | Chronological sequence of corresponding organizing groups for $N$ most recent popular events hosted at $v^*$. |

### 3.2 Problem Statement

The objective of our paper is to recommend top $k$ venues for successfully hosting a particular Meetup event $e^* \in \mathcal{E}$, organized by the group $g^* \in \mathcal{G}$. Essentially, the venue recommender needs to rank all the venues in $\mathcal{V}$ according to their abilities of making the event $e^*$ successful. This rank should be based on a score which indicates the propensity of event $e^*$ being successful at venue $v^* \in \mathcal{V}$. Formally, given a target event $e^*$ hosted by a group $g^*$, we propose a methodology to compute a success score $S(v^*|e^*, g^*)$ for each candidate venue $v^* \in \mathcal{V}$ and recommend the top $k$ most suitable venues for hosting $e^*$. In Table 1, we summarize the relevant notations related to our problem.

#### *3.2.1 Notion of Event Success*

We consider event popularity as a measure of event success. Popularity of an event is defined as the fraction of group members attending the event. For example, for an event $e$ hosted by group $g$, the popularity of event $P_e = \frac{A_g^e}{M_g^e}$ where $M_g^e$ is the size of the group $g$ at the time of hosting $e$ and $A_g^e$ is the number of members of $g$ attending event $e$. Standard Kolmogorov-Smirnov normality test [33] shows that the $P_e$ of all the events in our dataset follow normal distribution (with mean $\mu$ and standard deviation $\sigma$). Following the principle of outlier detection, we designate an event $e$ as 'popular'[3] if the corresponding $P_e$ is above $\mu + 2\sigma$; otherwise, we label $e$ as 'unpopular' [34].

## 4 *DeepVenue*: Venue Recommendation Model

We propose a deep neural network based framework *DeepVenue* for recommending venues to host Meetup events. In this framework, we use 'Long Short Term Memory' (LSTM) [35] cells to efficiently memorize and learn

3. We use the terms 'popular event' and 'successful event' interchangeably throughout the paper.

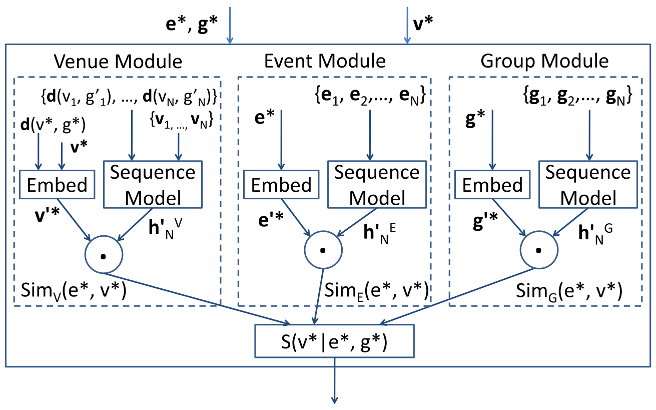


Fig. 2. Schematic diagram of the *DeepVenue* framework for estimating popularity of a given event $e^*$ (organized by group $g^*$) if hosted at venue $v^*$. It consists of three modules, namely, venue module, event module, and group module. Each module aims to asses the suitability of hosting the target event $e^*$ at a candidate venue $v^*$ from a different perspective.

representations for long temporal sequences. We first describe the key idea behind the proposed framework. Next we illustrate the architecture[4] of *DeepVenue* along with the input/output representation (see Fig. 2. Finally, we describe the model training and the procedure to generate a ranked list of recommended venues for any event.

## 4.1 Key Idea

The functionality of *DeepVenue* is based on the following hypothesis. An event $e^*$ can be successfully hosted by the group $g^*$ at a venue $v^*$ if (a) $v^*$ is similar to the venues where events similar to $e^*$ have been recently hosted successfully, (b) $e^*$ is similar to the events which have been hosted at venue $v^*$ successfully and (c) $g^*$ is similar to the groups recently hosted events at $v^*$ successfully. In order to compute the three aforementioned similarities, we introduce three separate modules in *DeepVenue* as shown in Fig. 2. In each of the modules, we have the following components:

(a) *Input:* The key challenge in developing *DeepVenue* model, which involves multiple sources of similarities, is dealing with the heterogeneity present in the input entities. In each module input, we provide the *representations* of the target entity (event, venue, group) and a corresponding chronological sequence of other entities of same type, among which we wish to compute the similarity. For instance, in the 'Event Module', we provide the representation of the target event $e^*$ (denoted as $\mathbf{e}^*$), and the representations of the sequence of past popular events hosted at $v^*$, as input. The detailed procedure to create those representations are illustrated in Section 5.

(b) *Embedding & sequence model:* Next, we pass the representation of the target entity through an embedding layer to get its $d$ dimensional embedding. On the other hand, the corresponding sequence of entities are passed through a LSTM based sequence model to obtain its $d$ dimensional embedding, so that this sequence can be compared with the embedding of the target entity.

(c) *Output:* Finally, in each module, we compute the cosine similarity (dot-product) between the target entity embedding and the corresponding sequence embedding. Once, we obtain the similarity values from each of the three modules, we combine them to obtain a score $S(v^*|e^*, g^*)$ which indicates the estimated popularity of the event $e^*$ if hosted at venue $v^*$.

In the following, we illustrate the individual module in detail.

## 4.2 Architecture

The expected popularity score of the candidate venue $v^* \in \mathcal{V}$ to host target event $e^*$ organized by group $g^*$ is computed in two steps.

### 4.2.1 *Suitability of Venue $v^*$ from Different Perspectives*

First we compute suitability of $v^*$ to host $e^*$ from the perspectives of events, venues and groups with the help of the three corresponding modules.

*(a) Venue module:* Representing venues in Meetup is not trivial due to the scarcity of venue related information. In order to deal with it, we use a transfer learning based approach to leverage Yelp (a popular 'LBSN') corpus for getting Meetup venue details. The detailed procedure of collecting Yelp data and learning venue representations is discussed in Section 5.1. In the venue module, we estimate the similarity of the candidate venue $v^*$ with the $N$ venues where events similar to the target event $e^*$ have been hosted successfully in recent past. The venue module consists of the following components (see Fig. 3),

*Input:* In this module, we provide the representations of the candidate venue $v^*$ and the sequence of venues $v_1, v_2, \ldots v_N$ hosting popular events similar to $e^*$ in recent past. To obtain events similar to the target event $e^*$, we estimate the cosine similarity of its representation $\mathbf{e}^*$ with the representations of all past popular events and choose the most recent $N$ popular events with cosine similarity higher than a threshold $\alpha_E$. We empirically choose the event similarity threshold $\alpha_E = 0.75$ for all our subsequent experiments (see Section 8.2.2 for detailed discussion).

*Embedding:* First, the module learns a $d_V$ dimensional latent embedding ($\mathbf{v'}^*$) of the candidate venue $v^*$ by passing it through an embedding layer with necessary dropout required to prevent over-fitting. Additionally, we concatenate the distance $\mathbf{d}(v^*, g^*)$ between the venue $v^*$ and members of group $g^*$, with the venue representation $\mathbf{v}^*$ before passing it to the embedding layer[5] $\mathbf{d}(v^*, g^*)$ is estimated as a 4-length vector consisting of four distance specific feature fractions of group $g^*$ members staying within 5 miles, 10 miles, 20 miles and 50 miles from the venue $v^*$.

*Sequence model:* Simultaneously, we chronologically pass the venue sequence $v_1, v_2, \ldots v_N$ to the LSTM units. Here also, with each venue representation, we concatenate the distance $\mathbf{d}(v_i, g_i)$ between venue $v_i$ and the members of corresponding group $g_i$. Finally, after passing the $N$ embeddings one-by-one to the LSTM units, we obtain the output

4. Used Microsoft's open-source Cognitive Toolkit (CNTK) for implementation: https://github.com/Microsoft/CNTK/wiki

5. Prior study shows that popularity of an event hosted at a venue depends on the distance of the group members from the venue [1], [2].

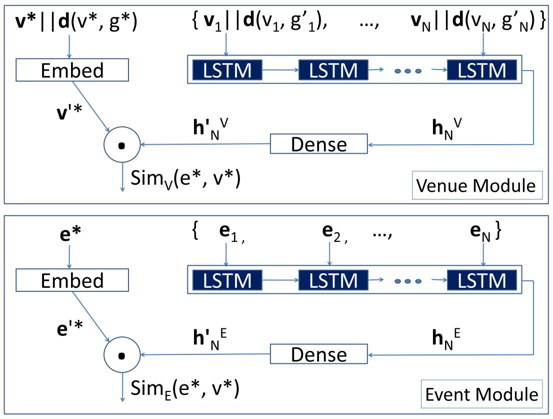


Fig. 3. Detailed ‘Venue Module’ and ‘Event Module’ of the *DeepVenue* framework. ‘II’ symbolizes concatenation.

vector $\mathbf{h}_N^V$ (also known as the ‘thought’ vector [36]) from the final LSTM block. Subsequently, we pass this $\mathbf{h}_N^V$ through a dense layer (with required dropout) and obtain a $d_V$ dimensional embedding $\mathbf{h'}_N^V$ representing the venue sequence.

*Output:* Once we obtain the $d_V$ dimensional representations of the candidate venue $v^*$ and the venue sequence $v_1, v_2, \ldots v_N$, we compute their cosine similarity as

$$Sim_V(e^*, v^*) = \frac{\mathbf{h}_N^{\prime V} \cdot \mathbf{v}'^*}{||\mathbf{h}_N^{\prime V}||\ ||\mathbf{v}'^*||} = \frac{\sum_{i=1}^n h_{N\,i}^{\prime V} v_i'^*}{\sqrt{\sum_{i=1}^n h_{N\,i}^{\prime V^2}}\ \sqrt{\sum_{i=1}^n v_i'^{*2}}}. \quad (1)$$

The output of venue module $Sim_V(e^*, v^*)$ depicts the similarity between the candidate venue $v^*$ and the sequence of venues hosting events similar to the target event $e^*$ successfully.

*(b) Event module:* The aim of this module is to compute the similarity between the target event $e^*$ and the events hosted at the candidate venue $v^*$ successfully.

*Input:* As shown in Fig. 3, in this module we provide the latent representations corresponding to each of the past $N$ events $e_1, e_2, \ldots e_N$ (hosted at the candidate venue $v^*$ successfully) as input along with the representation of the target event $e^*$.

The detailed procedure of learning this representations is discussed in Section 5.2.

*Embedding:* We pass the target event representation $\mathbf{e}^*$ through an embedding layer (with necessary dropout) to obtain its $d_E$ dimensional latent embedding $\mathbf{e}'^*$.

*Sequence model:* Next, we pass the representations corresponding to events $e_1, e_2, \ldots e_N$ as input to the LSTM units. The hidden state $\mathbf{h}_N^E$ from the last LSTM block is again passed through a dense layer to obtain a $d_E$ dimensional embedding $(\mathbf{h}_N^{\prime E})$ of this sequence of events.

*Output:* Finally, we calculate $Sim_E(e^*, v^*)$ as the cosine similarity of these two embeddings $\mathbf{h}_N^{\prime E}$ and $\mathbf{e}'^*$ as

$$Sim_E(e^*, v^*) = \frac{\mathbf{h}_N^{\prime E} \cdot \mathbf{e}'^*}{||\mathbf{h}_N^{\prime E}||\ ||\mathbf{e}'^*||} = \frac{\sum_{i=1}^n h_{N\,i}^{\prime E} e_i'^*}{\sqrt{\sum_{i=1}^n h_{N\,i}^{\prime E^2}}\ \sqrt{\sum_{i=1}^n e_i'^{*2}}}. \quad (2)$$

Essentially $Sim_E(e^*, v^*)$ estimates the similarity of $e^*$ with the recent events successfully hosted at $v^*$.

*(c) Group module:* In the group module, we estimate the similarity of the organizing group $g^*$ with the $N$ groups $g_1, g_2, \ldots g_N$ which recently hosted events $e_1, e_2, \ldots e_N$ successfully at the candidate venue $v^*$. This module looks exactly identical to the ‘Event Module’ shown in Fig. 3 except here we provide $g^*$ and $g_1, g_2, \ldots g_N$ as input. The output of this module is denoted as $Sim_G(e^*, v^*)$ which estimates the similarity of $g^*$ with the groups hosting events successfully at $v^*$.

### 4.2.2 *Computing Success Score for Venue $v^*$*

The output similarity values of the venue, event and group modules ($Sim_V(e^*, v^*)$, $Sim_E(e^*, v^*)$ & $Sim_G(e^*, v^*)$ respectively) indicate the suitability of venue $v^*$ to successfully host the event $e^*$ from three different perspectives. Finally, we concatenate all these similarity values and pass it to a dense layer with sigmoid activation function (equivalent to logistic regression) to calculate the final prediction score $S(v^*|e^*, g^*)$. The score $S(v^*|e^*, g^*)$[6] indicates the suitability of venue $v^*$ to host event $e^*$ organized by the group $g^*$.

## 4.3 Training of *DeepVenue* Model

The *DeepVenue* model is trained by minimizing the *Mean Square Loss* between the estimated success score $S(v^*|e^*, g^*)$ for hosting an event $e^*$ at venue $v^*$ and the corresponding ground truth success score $P_{e^*}$ (obtained from the empirical data, described in Section 3.2.1). Hence, the corresponding loss becomes

$$Loss(e^*, v^*) = (P_{e^*} - S(v^*|e^*, g^*))^2. \quad (3)$$

We apply ‘Adam-SGD (Stochastic Gradient Descent)’ [37] learner with a fixed learning rate of 0.005, fixed batch size ($B$) of 128 and momentum of $B/(-log(0.9)) \approx 1215$.

## 4.4 Venue Recommendation

Once we train the *DeepVenue* model from empirical dataset, the model is ready to be used as a venue recommendation engine. Given a target event $e^*$ to be organized by group $g^*$, we compute a success score $S(v^*|e^*, g^*)$ for every venue $v^* \in \mathcal{V}$ using our model. Finally, we rank those venues based on the obtained scores and recommend the top $k$ venues $L_k = \{v_1, v_2, \ldots v_k\}$.

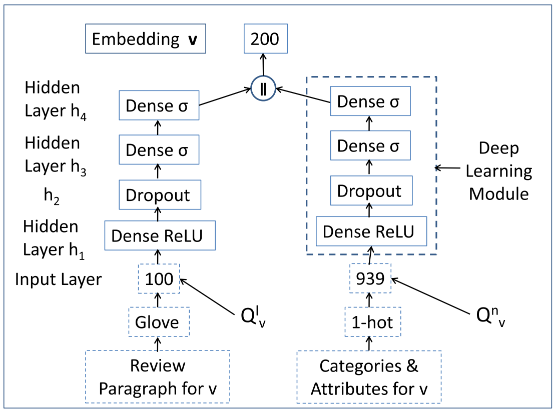
## 5 Representation of Meetup Entities

Implementation of the venue recommendation model *DeepVenue* requires suitable representation of the Meetup entities and feeding them as the input of *DeepVenue* model. We borrow the concept from deep embedding [38] and propose a principled method to represent Meetup entities (venues, events and groups) from the empirical data.

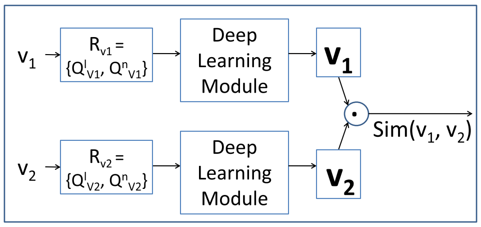
## 5.1 Meetup Venue Representation

The embedding of Meetup venues is challenging. (a) In Meetup, the available venue information is limited to only physical location such as latitude, longitude, address, zip-code etc. This information is insufficient to properly

6. $S(v^*|e^*, g^*)$ values always range between [0,1] due to the final sigmoid activation function.



(a) Deep learning framework for representing venues; each dense layer contains 100 fully connected nodes. 'II' symbolizes concatenation.



(b) Framework for learning venue representations via venue similarity prediction. Here, $R_{v_1} = \{Q^l_{v_1}, Q^n_{v_1}\}$ and $R_{v_2} = \{Q^l_{v_2}, Q^n_{v_2}\}$ denote the combinations of review vectors and attribute vectors for venues $v_1$ and $v_2$ respectively. Similarly, $\mathbf{v_1}$ and $\mathbf{v_2}$ denote the learnt representations of venues $v_1$ and $v_2$ respectively.

Fig. 4. Deep learning framework for learning venue representations via pair-wise venue similarity prediction.

represent the diverse aspect of a venue, significantly limiting the quality of venue embedding. (b) Once available, properly combining heterogeneous venue information and constructing suitable representation is challenging. In the following, we outline an approach to address the aforesaid challenges.

#### 5.1.1 Knowledge Transfer from Yelp

In order to enrich the Meetup venue information, we leverage on cross-domain knowledge transfer [15] from the popular LBSN services (Yelp, FourSquare etc) to extract substantial location related content. For instance, in Yelp[7] [13], [39] , customers frequently provide tips, reviews and various other information about the locations they visit. Yelp provides APIs to easily and efficiently crawl such information, which makes them a reliable source of location related detailed information. We collect (a) unstructured text such as venue reviews & tips posted by the visitors as well as (b) structured information such as venue categories and the facilities available (WiFi, parking etc.). We develop a simple heuristic (based on name matching, phone number matching and distance) to map each Meetup venue $v$ to the corresponding Yelp venue $u$. Subsequently, we assign the collected location details of Yelp venue $u$ to the corresponding Meetup venue $v$. Details of this mapping heuristic and corresponding evaluation can be found in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/TKDE.2019.2915523. In a nutshell, two types of information gets associated with each Meetup venue (a) qualitative information, which includes reviews and tips posted for that venue (b) quantitative information, which indicates the venue category and services available at that venue.

7. https://www.yelp.com

#### 5.1.2 Deep Embedding

We present a deep learning based venue embedding model (see Fig. 4a) to represent the aforesaid diverse venue information, which is motivated by the architecture similar to the DSSM [38]. The necessity of incorporating both qualitative & quantitative information for venue representation stems from the fact that the suitability of a venue to host an event depends on both the factors. For instance, positive & negative reviews may affect the reputation of a venue, thereby affecting the event attendance. Similarly services like 'Wifi' might be essential for hosting a 'technical' event whereas a large 'parking' space might be needed for a 'sports' event.

(a) *Input layer:* We primarily represent a venue $v$ in Meetup as $R_v = \{Q^l_v, Q^n_v\}$ which is a combination of following two vector components.
   (1) *Qualitative information: Review Vector $Q^l_v$:* The raw unstructured textual data corresponding to reviews and tips of a venue $v$ can be represented as a single word embedding vector with the help of a standard NLP tool Glove.[8] A Glove model is trained[9] incorporating all the reviews and tips provided for all venues and subsequently, it generates the embedded vector (real) $Q^l_v$ corresponding to the reviews and tips submitted for each individual venue $v$. The dimension of this review vector $Q^l_v$ is flexible. In our representation, we use 100-length (fixed empirically[10]) review vectors.
   (2) *Quantitative information: Attribute Vector $Q^n_v$:* The quantitative information attributing a venue $v$ can be primarily of two types - category and service. According to the collected data, venues can be hierarchically distributed into 22 primary, 498 secondary and 178 tertiary categories[11]. We construct a 698-length sparse 'one-hot' vector to represent this category information, where each element of the vector indicates the category $i$. Additionally, the collected venue data contains the information of 88 different facilities and services (such as WiFi, garage, parking etc) provided at each venue, which we represent using another 241 (total number of service options; for instance the service 'parking' can take values like 'street', 'private lot' etc.) length sparse vector. Finally, we

8. http://nlp.stanford.edu/projects/glove/
9. http://github.com/maciejkula/glove-python
10. Varying this length from 50 to 200 yields similar results.
11. Detailed list is available here: http://www.localvisibilitysystem.com /2013/07/19/yelp-business-categories-list/

concatenate the category and service indicator vectors to obtain a consolidated attribute vector $Q_v^n$ of length 939.

(b) *Dense layers:* Fig. 4a shows that $Q_v^l$ & $Q_v^n$ vectors of the input layer are successively passed through a *'Deep Learning Module'* which embeds them into a latent embedding space. This module contains three dense layers, containing 100 fully connected nodes with one 'ReLU', two 'Sigmoid' activation functions respectively (with necessary dropouts). After getting passed through the dense layers, both $Q_v^l$ and $Q_v^n$ are represented in a latent space of dimension 100.

(c) *Output layer:* Finally, at the output layer, we concatenate the 100 dimension latent representations of $Q_v^l$ and $Q_v^n$ to obtain a unified 200 dimension representation $\mathbf{v}$ for the venue $v$.

#### 5.1.3 Model Training

In order to train and evaluate the model, we leverage on the mutual similarity between the venue pairs, represented through our model. First, for each venue pair, we assign a ground truth label (similar & dissimilar) deliberately relying on the Meetup data only, so that this labeling becomes independent of the model we built through knowledge transfer from Yelp. We explore various venue similarity indicators (such as, types of events they hosted, the profile of members visiting them etc) and finally following a principled methodology [40], label each pair of venues as similar or dissimilar. This ground truth venue similarity labeling procedure is described in detail in Appendix B, available in the online supplemental material.

Next, during training, for each of the training venue pair $(v_1, v_2)$, we obtain their latent representations from our model and calculate the cosine similarity $Sim(v_1, v_2)$ (as shown in Fig. 4b). Finally, we compute the 'Mean Square Loss' with respect to ground truth venue similarity and train the model accordingly.

#### 5.1.4 Evaluation

For each venue in the test set, we first obtain their 200 dimensional latent embeddings from our trained model and subsequently compute the venue pair similarity as the cosine distance. We evaluate the model performance with respect to the ground truth venue pair similarity labels and observe the accuracy of 88 percent (at a similarity threshold 0.51) and area under the corresponding Precision-Recall curve (AUC) as 0.93 (with 5-fold cross validation). These results clearly demonstrate the high reliability of the venue embedding, obtained from the model.

Finally, we compare our deep embedding model with vanilla machine learning models such as Support Vector Machine (SVM), Decision Tree (DT) and Logistic Regression (LR). Here we represent each venue $v$ as naive combination of qualitative and quantitative information $R_v = \{Q_v^l, Q_v^n\}$ (as defined in Section 5.1.2). In order to estimate the similarity between venue pair $v_1$ and $v_2$ using vanilla ML models, we compute their respective qualitative similarity ($cosine(Q_{v_1}^l, Q_{v_2}^l)$) and quantitative similarity $cosine(Q_{v_1}^n, Q_{v_2}^n)$) as features. The accuracy and AUC results presented in Table 2 clearly shows that the proposed deep learning based venue embedding outperforms the baseline models.

TABLE 2
Evaluation Results of Deep Learning Based Venue Embedding with Respect to SVM, DT, and LR

| Metric | Deep Learning | SVM | Decision Tree | Logistic Regression |
|---|---|---|---|---|
| Accuracy | **0.88** | 0.50 | 0.51 | 0.50 |
| AUC | **0.93** | 0.51 | 0.54 | 0.51 |

### 5.2 Meetup Event Representation

In order to obtain embeddings for events in Meetup, we adhere a similar approach. We represent an event $e$ as $R_e = \{Q_e^l, Q_e^n\}$ which is a combination of following two vector components.

*(a) Qualitative information: Description Vector $Q_e^l$:* The raw unstructured event description text of an event $e$ can be represented as a single word embedding vector with the help of Glove. A Glove model is trained incorporating all the descriptions for all the events in Meetup and subsequently, it generates an embedded vector $Q_e^l$ corresponding to the description of each event $e$. We keep the dimension of $Q_e^l$ as 100.

*(b) Quantitative information: Tag Vector $Q_e^n$:* The type of an event can be best inferred from the preferences (tags) of its attendees. Hence, we combine the tags specified in the profiles of all the members sending 'Yes' RSVP for the event and create a weighted tag vector $Q_e^n$. In order to build such a tag vector, we first choose the top 1000 most popular member-tags, each of which designates a dimension of $Q_e^n$. The coefficient of each tag in $Q_e^n$ is computed as the fraction of attendees using it in their profiles.

In order to learn a combined representation $\mathbf{e}$ of the event $e$, we pass the $Q_e^l$ & $Q_e^n$ vectors through *'Deep Learning Modules'* as shown in Fig. 4a and obtain a latent embedding for the event $e$. For training the model, we estimate similarities between pair of events and compute the loss as depicted in Fig. 4b. We designate two events as similar if their corresponding hosting venues are also labeled as similar (following Appendix B, available in the online supplemental material). Finally, we obtain a 200 length embedding for each Meetup event.

### 5.3 Meetup Group Representation

In order to generate embedding $\mathbf{g}$ of a group $g$, we use the vector combination $R_g = \{Q_g^l, Q_g^n\}$. Here $Q_g^l$ is the 100-length qualitative vector obtained from the Glove representation of textual description available for each group $g$ and $Q_g^n$ is the 1000-length one-hot tag vector representing the profile of $g$. In order to combining both $\{Q_g^l, Q_g^n\}$, we pass them through *'Deep Learning Modules'* as shown in Fig. 4a. For training the model, we label two groups as 'similar' if the fraction of venues used by both of them to host events is above a threshold (66.67th percentile); otherwise they are labeled as 'dissimilar' groups. Finally, with the help of the framework shown in Fig. 4b, we minimize the loss and learn a 200 length embedding for each Meetup group.

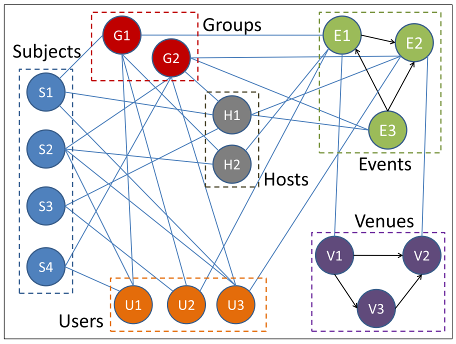


Fig. 5. Graph constructed by the $SERGE$ algorithm to recommend venues for hosting event $E3$.

## 6 Dataset & Experimental Setup

In this section, first we describe the crawled Meetup dataset and perform the necessary data filtering. Next we illustrate the procedure to evaluate the proposed recommendation model *DeepVenue*. Moreover, we explain multiple baseline algorithms to demonstrate the superiority of the proposed model.

### 6.1 Data Preparation

We develop a crawler to collect the Meetup data for the city Chicago during a period of 20 months (from August 2015 to March 2017). In order to evaluate the proposed venue recommendation model *DeepVenue*, we first choose 3178 Meetup venues (a) which have hosted at least one Meetup event and (b) for which we can extract venue information from Yelp (see Section 5.1.1). Finally, we extract the 10840 Meetup events (hosted by 433 groups) which were hosted at those 3178 venues for our experiments.

### 6.2 Evaluation Procedure and Metrics

We randomly divide the aforementioned set of 10840 Meetup events into 5 equal size partitions and perform standard 5-fold cross-validation. We train the *DeepVenue* model using 'Adam-SGD (Stochastic Gradient Descent)' [37] learner for 2000 iterations. It takes approximately 2 hours to train on a machine with 6 Intel(R) Xeon(R) E5-2620 v3 2.40GHz CPU cores, 32GB RAM and 16GB GPU. We reduce the chance of overfitting by keeping the dataset sufficiently larger compared to the number of model parameters and adjusting the dropouts accordingly.[12] For all our subsequent experiments, we fix the history length ($N$) as 100 (except for Section 8.2.1 where we show the impact of varying $N$). During evaluation, we concentrate only on the 'popular' events (as defined in Section 3.2.1) from each of these 5 folds as the test events. Consider the set of test events as $T_e$ and assume a popular event $e^* \in T_e$ is originally hosted at venue $v^* \in \mathcal{V}$. Intuitively, a successful recommender is expected to include venue $v^*$ within the top recommended venues for target event $e^*$.[13] Following this intuition, we define two metrics to quantify and evaluate the performance of recommendation model *DeepVenue*.

#### 6.2.1 Metrics

(a) *Recall@k:Recall@k* measures the fraction of times the original venue, corresponding to the target event appears in the top $k$ recommended venues. Mathematically, $Recall@k = \frac{1}{|T_e|}\sum_{i=1}^{|T_e|}\{(v^i \in {L_k}^i)?1:0\}$ where $v^i$ is the hosting venue of the target event $e^i$ and ${L_k}^i$ is the list of $k$ top venues recommended by the model for hosting $e^i$.[14]

(b) *Mean Inverse Rank (MIR): MIR* measures the rank of the hosting venues in the recommended venue list. For instance, rank of the hosting venue $v^i$ in the recommendation list of a target event $e^i \in T_e$ is $r^i$. Then the mean inverse rank ($MIR$) of the recommender can be measured as $MIR = \frac{1}{|T_e|}\sum_{i=1}^{|T_e|}\frac{1}{r^i}$ Higher the $MIR$, better is the performance of the recommender.

### 6.3 Baseline Algorithms

From the rich literature in the gamut of user-item recommendation, we implement the following baseline algorithms to compare their performance with our proposed model *DeepVenue*.

*Successive Event Recommendation Based on Graph Entropy (SERGE).* Liu et al. recently proposed a model 'SERGE' [16] for recommending events to members of Douban, a popular EBSN.[15] We adapt this model and customize to make it suitable for Meetup venue recommendation. This model first constructs the primary graph (see Fig. 5) involving six types of entities - Meetup groups, events, event hosts, members, subjects[16] and candidate venues as nodes and suitably link them following [16]. The primary graph contains eleven types of links (see Fig. 5), for instance, link between group & a member indicates group-membership, whereas link between a pair of events is placed based on their description similarity. Notably, in our adaptation for venue recommendation, we introduce the venue nodes in the primary graph and connect them with the past hosted events. Additionally, the venue nodes are connected to each other based on their similarity in qualitative and quantitative information obtained from Yelp (see Section 5.1.2). Finally, we apply random walk with restart (RWR) [42] on the primary graph, initiated on the target event to compute the event-venue similarity scores. These scores essentially provide the ranked list of recommended venues for the target event. Since the venue recommendation is an one time task (unlike event recommendation, where recommendation varies successively depending on the received feedback from the members), we only rely on the primary graph to compute the event-venue similarity scores and skip the construction of feedback graph, which is required in [16].

12. In https://github.com/Soumajit-Pramanik/Venue_Recommendation, we provide the implementations of *DeepVenue* and the major baseline algorithms along with the parameter values used for running them.
13. A similar argument does not hold for the unpopular events; hence, we could not keep them in test set.

14. Since there exists only *one* hosting venue for each target event, other standard metrics such as 'Precision' or 'Accuracy' would be identical to Recall.
15. https://www.douban.com
16. Meetup tags are clustered into 'subjects' by using the unweighted pair-group method with arithmetic mean (UPGMA) [16], [41]. The similarity between a pair of tags is computed based on the overlap of users and groups using them.

TABLE 3
Overall Model's Evaluation Results

| Algorithm | Recall@1 | Recall@5 | Recall@10 | MIR |
|---|---|---|---|---|
| *DeepVenue* | **0.785** | **0.840** | **0.858** | **0.808** |
| *SERGE* | 0.701 | 0.731 | 0.735 | 0.726 |
| *DeepCoNN* | 0.435 | 0.634 | 0.648 | 0.522 |
| *CTR* | 0.483 | 0.568 | 0.596 | 0.526 |
| *MFR* | 0.460 | 0.506 | 0.539 | 0.488 |
| *SR* | 0.268 | 0.413 | 0.478 | 0.333 |

TABLE 4
Overall Model's Evaluation Results for 'New' Venues

| Algorithm | Recall@5 | Recall@10 | Recall@15 | MIR |
|---|---|---|---|---|
| *DeepVenue* | **0.071** | **0.125** | **0.200** | **0.143** |
| *SERGE* | 0.025 | 0.025 | 0.025 | 0.041 |
| *DeepCoNN* | 0.018 | 0.056 | 0.117 | 0.088 |
| *CTR* | 0.000 | 0.000 | 0.000 | 0.001 |
| *MFR* | 0.000 | 0.000 | 0.000 | 0.001 |
| *SR* | 0.000 | 0.000 | 0.005 | 0.002 |

*Deep Cooperative Neural Networks (DeepCoNN).* The *DeepCoNN* model[17] [17] consists of two parallel neural networks coupled through a shared layer where one network focuses on learning user behaviors exploiting reviews written by them, and the other one learns item properties from the reviews written for the items. We adapt this model for venue recommendation by suitably mapping events to users and venues to items.[18]

*Collaborative Topic Regression based Ranking (CTR).* Collaborative Topic Regression [18] is a model performing both topic modeling and collaborative filtering simultaneously. It uses additional context information (say, reviews) for the venues and represents them as latent topic vectors with the help of Latent Dirichlet Allocation [43].

*Matrix Factorization based Ranking (MFR).* We create an event-venue popularity matrix where each entry $(i, j)$ represents the popularity score of the event $e_i$ hosted at venue $V_j$. Taking this sparse matrix as input, Matrix Factorization [44] derives two low-rank matrices for events and venues respectively. We estimate the popularity score of the venue $V_i$ to host an event $e_i$ as the product of their corresponding rows from the generated low-rank matrices.

*Similarity based Ranking (SR).* This algorithm [31] computes the recommendation score of a venue as the average popularity of the events, highly similar to the target event, hosted at this venue. Notably, this is basically a standard neighborhood based approach which does not use any additional contextual information about the venues.

## 7 Evaluation of *DeepVenue*

In this section, we illustrate the performance of *DeepVenue* with respect to the baseline algorithms. We evaluate the model from two different perspectives; (a) overall evaluation and (b) category specific evaluation.

### 7.1 Overall Model Evaluation

In Table 3, we demonstrate that the proposed *DeepVenue* model outperforms all the baseline algorithms[19] both in terms of *Recall@k* and *MIR*. Specially, the high *Recall@1* and *Recall@5* indicates that for most of the popular events (see Section 3.2.1), the corresponding actual hosting venue comes at the top of the *DeepVenue* recommended venue list. Close inspection reveals that the deficiency of simple venue similarity information results in poor performance of *SR*, *MFR* & *CTR* baseline algorithms. On the other hand, poor performance of *SERGE* and *DeepCoNN* ascertains the importance and necessity of sequence encoding that incorporates time based state information. However, even after employing deep convolutional neural network structures, *DeepCoNN* performs relatively worse than *SERGE* because it treats each event-venue interaction as a distinct data-point and refrains from utilizing the past history of event-venue interactions while estimating the corresponding score. On the other hand, *SERGE* incorporates the 'memory' component to some extent (however naively) by aggregating the entire history of event-venue interactions in its network structure.

### 7.2 Recommending *new* Venues

In our Meetup venue recommendation, one critical problem is to recommend *new venues* (hosting an event for the first time) due to the unavailability of their past history. In order to examine the performance of *DeepVenue* for those venues, we handpick all the 'popular' events hosted at such new venues (19 percent of total 10840 events). We designate them as target events in the test set and recommend venues following *DeepVenue* as well as baseline algorithms. In Table 4, we present the performance of the proposed model *DeepVenue* against baselines. Due to their sheer dependence on the past event hosting history information, most of the baseline algorithms fail to recommend the correct venues, hosting an event for the first time. Albeit graceful degradation, *DeepVenue* exhibits close to 20 percent for *Recall@15* which is almost twice better than the best baseline result (*DeepCoNN*). This hike in performance is solely contributed by the 'venue' module of *DeepVenue* (the other two modules are fully dependent on the past event hosting history, hence ineffective for *new venues*) which judiciously exploits the *new venue's* similarity with past venues successfully hosting similar events. On the contrary, the performance of *SERGE* mainly depends on memorizing the detail of past event-venue interactions which is inadequate for recommending *new venues*. Observing the poor performance of *SR*, *MFR* & *CTR* in both Tables 3 and 4, we drop these naive baselines in our subsequent evaluation.

### 7.3 Category Specific Model Evaluation

According to [40], we cluster the official Meetup group categories into the following five broad categories - a) Activity b) Hobby c) Social d) Entertainment and e) Technical. In this evaluation, we train the models based on only the

17. It outperforms most of the state-of-the-art baseline recommender systems such as Collaborative Deep Learning [30].

18. For venues, we use client reviews, venue category, and services. For events, we use event descriptions as a replacement of reviews.

19. We optimize the parameters of each of them suitably for our dataset.

TABLE 5
Category Wise Models' Evaluation Results

| Category | Algorithm | Recall @5 | Recall @10 | Recall @15 | MIR |
|---|---|---|---|---|---|
| Act. | *DeepVenue* | **0.892** | **0.902** | **0.911** | **0.874** |
| | *SERGE* | 0.843 | 0.843 | 0.861 | 0.801 |
| | *DeepCoNN* | 0.760 | 0.760 | 0.773 | 0.765 |
| Hobby | *DeepVenue* | **0.727** | **0.812** | **0.818** | **0.633** |
| | *SERGE* | 0.631 | 0.684 | 0.747 | 0.635 |
| | *DeepCoNN* | 0.571 | 0.571 | 0.576 | 0.311 |
| Social | *DeepVenue* | **0.809** | **0.857** | **0.872** | **0.774** |
| | *SERGE* | 0.788 | 0.803 | 0.812 | 0.750 |
| | *DeepCoNN* | 0.348 | 0.449 | 0.463 | 0.239 |
| Ent. | *DeepVenue* | **0.921** | **0.921** | **0.933** | **0.923** |
| | *SERGE* | 0.904 | 0.904 | 0.904 | 0.916 |
| | *DeepCoNN* | 0.887 | 0.914 | 0.920 | 0.786 |
| Tech. | *DeepVenue* | **0.746** | **0.764** | **0.789** | **0.716** |
| | *SERGE* | 0.734 | 0.734 | 0.747 | 0.709 |
| | *DeepCoNN* | 0.100 | 0.100 | 0.115 | 0.112 |

TABLE 6
Percentage of Total Events and Events Hosted at *New Venues* for Different Categories of Groups

| | Act. | Hobby | Social | Ent. | Tech. |
|---|---|---|---|---|---|
| Event Count | 52% | 7% | 27% | 8% | 6% |
| Events hosted at *New Venues* | 1% | 18% | 5% | 4% | 3% |

events and venues belonging to one specific category.[20] Table 5 depicts that for category specific models, *DeepVenue* outperforms all the baseline algorithms for almost all the categories. The performance of the algorithms across different categories largely depend on the following two factors. (i) *Number of total events belonging to that category:* higher the number, larger the training set ensuring better performance and (ii) *Fraction of events hosted at new venues in that category:* lower the fraction, better the performance as correctly ranking the *new venues* is one of the most challenging task for any recommender. In Table 6, we summarize the aforementioned factors for each category of groups. In this context, two interesting observations can be reported:

(a) In 'Activity' category of events, most of the algorithms perform decently well due to the sheer volume of total events (52 percent) with a mere 1 percent of them hosted at *new venues*.
(b) On the other hand, the 'Hobby' category contains only 7 percent of total events among which 18 percent are hosted at *new venues* which clearly causes poor performance of all the competing algorithms; nevertheless, *DeepVenue* exhibits graceful degradation.

## 7.4 Complexity Analysis

In the following, we illustrate the time complexity of *DeepVenue* along with the major baseline algorithms. Table 7 shows the comparative study of the time complexity of *DeepVenue*, *SERGE* and *DeepCoNN* algorithms; side by side it depicts the mean runtime of the aforesaid algorithms on the platform specified in Section 6.2.

20. Each event's category is assumed to be the same as its organizing group's category.

TABLE 7
Comparison of Runtime Complexity of All Models with *DeepVenue*

| Algorithm | Complexity | Runtime in seconds |
|---|---|---|
| *DeepVenue* | $O(\mathcal{V}NI^2 + \mathcal{V}logk)$ | 3.69 |
| *SERGE* | $O(t\mathcal{U}^2 + \mathcal{V}logk)$ | 5.28 |
| *DeepCoNN* | $O(\mathcal{V}I^3 + \mathcal{V}logk)$ | 201.65 |

### 7.4.1 *DeepVenue*

The proposed *DeepVenue* model consists of three components - venue module, event module and group module (see Fig. 2). Since the structure of all three modules are mostly identical, we concentrate only on one module to compute the runtime complexity of *DeepVenue*. Consider that the dimension of input entity for each module is $I$. In order to learn a $D$ dimensional embedding of the input entity (via the dense embedding layer), the complexity becomes $O(I \times D)$. Additionally, the complexity of the sequence model of length $N$ can be estimated as $O(NI^2)$ [35] (assuming the hidden dimension and memory dimension are of $O(I)$). Hence, the time complexity of *DeepVenue* model for computing the score of each candidate venue $v^*$ hosting a target event $e^*$ can be estimated as $O(ID + NI^2) \approx O(NI^2)$ (assuming $D$ to be of $O(I)$). Finally, complexity of computing this score for all $\mathcal{V}$ candidate venues and generating the top-$k$ recommendation becomes $O(\mathcal{V}NI^2 + \mathcal{V}logk)$. Table 7 shows that mean (run)time taken by *DeepVenue* on the platform specified in Section 6.2 to recommend venues for a single event is 3.69 seconds.

### 7.4.2 *SERGE*

The complexity of *SERGE* algorithm [16] is mostly regulated by (i) the size of the primary graph and (ii) the random walk with restart (RWR) process running on top of it. For each target event $e^*$, we construct the primary graph containing the past events ($N$), their corresponding hosts, venues ($\mathcal{V}$), groups ($\mathcal{G}$), all the members ($\mathcal{U}$) and subjects associated to them. Notably, the size of the graph is mostly dominated by the concerned members, which can be estimated as $O(\mathcal{U})$. Additionally, each iteration of random walk with restart (RWR), initiated at the target event $e^*$ takes approximately $O(\mathcal{U}^2)$ operations [45]. On average, if the random walk converges after $t$ steps, the overall complexity of *SERGE* algorithm recommending top-k venues becomes $O(t\mathcal{U}^2 + \mathcal{V}logk)$. Table 7 depicts that on average *SERGE* takes 5.28 seconds to generate the list of recommended venues for a single event.

### 7.4.3 *DeepCoNN*

The *DeepCoNN* architecture [17] consists of the following neural network layers - a convolution layer, a max-pooling layer, and a dense layer. However, the complexity of

TABLE 8
Results for Different Model Variants of *DeepVenue*

| Model | Recall@1 | Recall@5 | Recall@10 | MIR |
|---|---|---|---|---|
| *Venue* | 0.57 | 0.65 | 0.70 | 0.61 |
| *EVenue* | 0.76 | **0.83** | 0.85 | 0.80 |
| *DeepVenue* | **0.77** | **0.83** | **0.86** | **0.81** |

*DeepCoNN* is mostly dominated by the convolution layer which takes $O(I^3)$ operations [46], where $I$ is the input word vector length[21] (assuming the number of words per event & venue, number of filters and width of filters are of $O(I)$; the height of each filter is negligible). Hence, the overall time complexity of *DeepCoNN* for computing the score of all $\mathcal{V}$ venues and recommending the top k becomes $O(\mathcal{V}I^3 + \mathcal{V}logk)$. Compared to *DeepVenue*, it takes quite a long time (201.65 seconds) to generate recommendations for a single event (see Table 7) due to its heavy dependency on the expensive convolution operation.

## 8 Dissecting *DeepVenue*

In this section, we delve deep and inspect *DeepVenue* recommendation model through the lens of the model parameters and variants.

### 8.1 Model Variants

We explore multiple variations of *DeepVenue* model (Fig. 2) to investigate the role of the event, venue and group modules. We derive the following two recommendation models from *DeepVenue* (a) *Venue* model, incorporating only venue module, (b) *EVenue* model, incorporating both venue and event modules. In Table 8, we present a comparative study of the aforesaid two variants with *DeepVenue* model. Clearly, *Venue* exhibits poor performance, whereas *EVenue* results in sharp improvement, manifesting the importance of event module. Moreover, the vanilla *DeepVenue* model (consisting of event, venue and group modules) shows further (albeit marginal) improvement over *EVenue*. Evidently, *DeepVenue* model proves to be a classic instance of multi-entity recommender, where individual entities play role in improving the model performance.

### 8.2 Impact of Model Parameters on *DeepVenue*

In the following, we vary several model parameters such as the history length $N$, event similarity threshold $\alpha_E$, dropout to analyze the dependency of our proposed *DeepVenue* model on these parameters and justify the parameter values chosen for performing our experiments in previous sections.

#### 8.2.1 Varying N: Analyzing History Dependency

In order to recommend venues for a target event $e^*$, *DeepVenue* model requires history information designating (a) venues hosting past $N$ popular events similar to $e^*$, (b) past $N$ popular events hosted at the candidate venue $v^*$ & (c) groups hosting past $N$ popular events at the candidate venue $v^*$. We vary the volume of history information $N$ to

21. Each event and venue is represented as a $n \times I$ word vector matrix where $n$ is the number of most frequent words considered (chosen from descriptions, reviews, etc.).

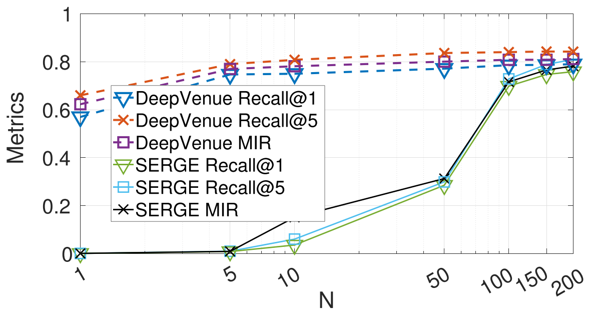


Fig. 6. History length $N$ versus recall & MIR values for *DeepVenue* and *SERGE*.

investigate the impact on model performance. In Fig. 6, we observe that *DeepVenue* attains a decent performance at $N = 5$ and remains consistent with $N > 50$, which implies that the history dependence of our model is almost negligible. It mainly occurs due to the ability of our proposed model (especially, LSTM cells) to learn efficient representations of the historical events, venues and groups even when the history length $N$ is very small. On the contrary, as *SERGE* mainly depends on memorizing past events, the performance of *SERGE* is heavily dependent on the volume of history information. As a result, it performs poorly at $N < 50$ and improves rapidly with increasing $N$ until $N = 100$. Hence in practice, when only few events similar to the target event $e^*$ are available, or the candidate venue $v^*$ is relatively new, *DeepVenue* outperforms *SERGE* in terms of both Recall and MIR. Moreover, we observe that for both *DeepVenue* & *SERGE*, the recall and MIR values do not improve significantly with $N > 100$, which explains our choice of $N = 100$ for all other experiments related to them.

#### 8.2.2 Varying $\alpha_E$: Analyzing Impact of Event Similarity Threshold

In the venue module of *DeepVenue* model (see Section 4.2.1), we identify the sequence of past $N$ events $e_1, e_2, \ldots e_N$ similar to the target event $e^*$. In this process, we compute the cosine similarity of $e^*$ with the past events and choose the $N$ events above the similarity threshold $\alpha_E$ (0.75). Notably, in model development, we only consider the target events $e^*$ with at least $N$ similar past events. Hence, the volume of target events $e^*$ in model training gets regulated by $\alpha_E$. Importantly, the performance of *DeepVenue* depends on the (a) quality of similar event sequence (higher $\alpha_E$ yields better quality) and (b) the volume of target events (yields better model training). Fig. 7 depicts that the performance of *DeepVenue* sharply improves as we increase $\alpha_E$, which improves the event sequence quality. However, if we increase $\alpha_E$ beyond 0.75, performance does not significantly improve as the number of target events $e^*$ with at least $N$ similar past events becomes too low. This experiment indicates that choosing $\alpha_E$ between $[0.7 - 0.9]$ yields a decent performance. In our evaluation experiments, we fix $\alpha_E$ as 0.75. For a different dataset, similar experiments can be conducted to decide on $\alpha_E$.

#### 8.2.3 Varying Dropout: Analyzing Robustness

In order to assess robustness, we vary the percentage of dropout in *DeepVenue* model (introduced at the embedding

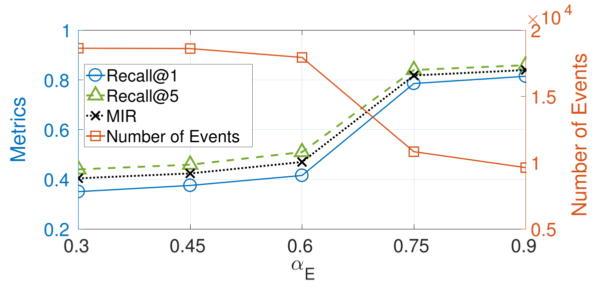


Fig. 7. The left Y axis shows how the recall & MIR values of *DeepVenue* model varies with the event similarity threshold $\alpha_E$; The right Y axis depicts the number of events selected for the experiments depending on $\alpha_E$.

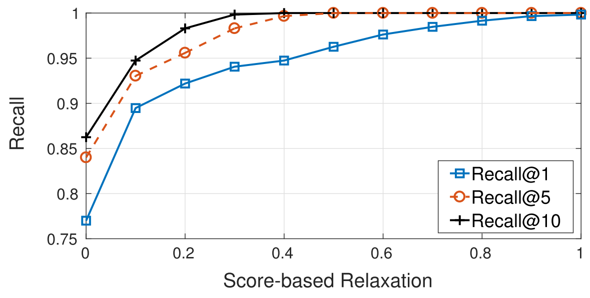


Fig. 8. Improvement of recall with score-based relaxation in venue similarity.

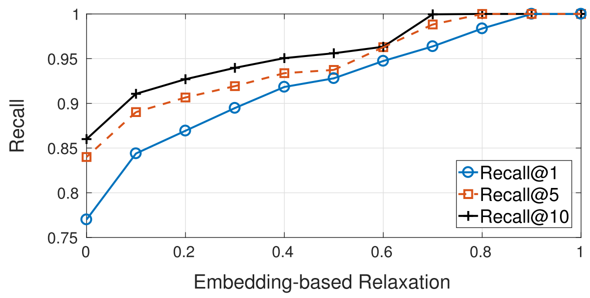


Fig. 9. Improvement of recall with embedding-based relaxation in venue similarity.

and sequence models of each module) and measure the corresponding test errors (median, upper & lower quartiles). Our model has a stable median error for a wide range of dropout $(0.1 - 0.7)$ indicating *DeepVenue* exhibiting a decent performance, even if the test sample significantly deviates from the training data. The variation in the error reduces till dropout of 20 percent and then increases till the end. Hence, a dropout of 20 percent seems to be ideal for alleviating overfitting for our dataset.

## 8.3 Analyzing Model Deficiency: Recommending Suitable Alternate Venues

In Table 3, we observed that for 15-20 percent of cases, *DeepVenue* model fails to recommend the correct venue for hosting a popular event. In this section, we concentrate on those (failure) cases and investigate the properties of the corresponding recommended venues. Through our experiments, we demonstrate the fact that although the recommended list fails to include the exact hosting venue, nevertheless, the recommended venues are indeed similar to the hosting venue. Hence, the recommendation may be considered as suitable *alternate* venues to host that event. The suitability of the alternate venues is characterized by the a *relaxation factor*, which helps to (virtually) include the original hosting venue in the recommended list. We measure this relaxation factor from two different perspectives (a) score-based relaxation and (b) embedding-based relaxation.

### 8.3.1 Score-Based Relaxation

In score-based relaxation, we define the extent of relaxation $x_S$ as an inclusion factor of the hosting venue in the *DeepVenue* recommended list. If the difference (root mean square) in success score (see Section 4.2.2) between the hosting venue and the venues in the recommended list falls below $x_S$, we include the hosting venue in the recommended list. The effectiveness of this relaxation gets directly reflected in the Recall rate. In Fig. 8, we show that even a small amount of relaxation $x_S$ results in a significant improvement in the corresponding recall. Of course, no relaxation ($x_S = 0$) converges the results to Table 3.

### 8.3.2 Embedding-Based Relaxation

In this case, relaxation is characterized by the similarity between the hosting venue and the *DeepVenue* recommended venues; we measure the cosine similarity between the venues following the embedded representation, introduced in Section 5.1. Considering the extent of relaxation as $x_E$, we include the original hosting venue in the *DeepVenue* recommended list, if the average similarity between them exceeds $1 - x_E$. In Fig. 9, we show how the recall improves if we slowly increase the relaxation $x_E$. This precisely confirms the fact that even if the exact venue is not present in the recommended list, *DeepVenue* is able to recommend venues highly similar to it, which can be easily considered as alternate options for hosting the same event.

## 9 Conclusion

The major contribution of this paper is to propose a deep neural network based framework *DeepVenue* for recommending venues to host popular Meetup events. The development of *DeepVenue* is based on the following simple intuition: a Meetup event can be successfully hosted by a group at a particular venue if (a) *similar* type of events have been recently hosted by *similar* type of groups at the same venue successfully (b) the venue is *similar* to the type of venues where *similar* type of events have been recently hosted successfully. In order to realize the notion of similarity, we have developed a principled method to obtain latent representation of the Meetup entities (venue, event and groups) via embedding. In *DeepVenue*, we use "Long Short Term Memory" (LSTM) cells to efficiently memorize and learn representations for long temporal sequences. Notably, this is a generic model and can be seamlessly adopted & extended for any multi-entity recommendation problem, even incorporating additional entities. During evaluation, we have observed a high *Recall*@1 (0.78) and *Recall*@5 (0.84) for *DeepVenue* (with decent runtime complexity), indicating that for most of the popular events, our system recommended the correct venue as the top recommended venue. Interestingly, unlike the closest baseline *SERGE*, the history dependence of *DeepVenue* is negligible. As a result, *DeepVenue* works especially well for recommending *new venues* (venues without any prior history of organizing events); this gets manifested in the 'Hobby' category, where *DeepVenue* performs substantially well (*Recall*@10 - 0.812) than the closest baseline *SERGE* (*Recall*@10 - 0.684). Moreover, we have observed that in case of incorrect recommendations made by *DeepVenue* (16 percent of cases), the recommended venues exhibit high similarity with the correct hosting venues, pointing towards alternate venues for hosting that event.

## REFERENCES

[1] A. Q. Macedo, L. B. Marinho, and R. L. Santos, "Context-aware event recommendation in event-based social networks," in *Proc. 9th ACM Conf. Recommender Syst.*, 2015, pp. 123–130.

[2] Z. Qiao, P. Zhang, Y. Cao, C. Zhou, L. Guo, and B. Fang, "Combining heterogenous social and geographical information for event recommendation," in *Proc. AAAI Conf. Artif. Intell.*, 2014, pp. 145–151.

[3] Q. Yuan, G. Cong, and C.-Y. Lin, "COM: A generative model for group recommendation," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 163–172.

[4] W. Zhang, J. Wang, and W. Feng, "Combining latent factor model with location features for event-based group recommendation," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 910–918.

[5] 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 *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 1032–1040.

[6] S. Purushotham and C.-C. J. Kuo, "Personalized group recommender systems for location- and event-based social networks," *ACM Trans. Spatial Algorithms Syst.*, vol. 2, no. 4, pp. 16:1–16:29, Nov. 2016.

[7] P. Lops, M. De Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends," in *Recommender Systems Handbook*. Berlin, Germany: Springer, 2011, pp. 73–105.

[8] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Procedings 20th Int. Conf. Neural Inf. Process. Syst.*, 2008, pp. 1257–1264.

[9] B. Liu, H. Xiong, S. Papadimitriou, Y. Fu, and Z. Yao, "A general geographical probabilistic factor model for point of interest recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 5, pp. 1167–1179, May 2015.

[10] H. Yin, X. Zhou, B. Cui, H. Wang, K. Zheng, and Q. V. Nguyen, "Adapting to user interest drift for poi recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 10, pp. 2566–2581, Oct. 2016.

[11] S. Lee, S.-i. Song, M. Kahng, D. Lee, and S.-g. Lee, "Random walk based entity ranking on graph for multidimensional recommendation," in *Proc. 5th ACM Conf. Recommender Syst.*, 2011, pp. 93–100.

[12] T.-A. N. Pham, X. Li, G. Cong, and Z. Zhang, "A general recommendation model for heterogeneous networks," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 12, pp. 3140–3153, Dec. 2016.

[13] S. Scellato, A. Noulas, R. Lambiotte, and C. Mascolo, "Socio-spatial properties of online location-based social networks," in *Proc. 5th Int. AAAI Conf. Weblogs Social Media*, 2011, vol. 11, pp. 329–336.

[14] S. Gupta, S. Pathak, and B. Mitra, "Complementary usage of tips and reviews for location recommendation in yelp," in *Proc. Pacific-Asia Conf. Knowl. Discovery Data Mining*, 2015, pp. 720–731.

[15] M. Jiang, P. Cui, X. Chen, F. Wang, W. Zhu, and S. Yang, "Social recommendation with cross-domain transferable knowledge," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 11, pp. 3084–3097, Nov. 2015.

[16] S. Liu, B. Wang, and M. Xu, "Serge: Successive event recommendation based on graph entropy for event-based social networks," *IEEE Access*, vol. 6, pp. 3020–3030, 2018.

[17] L. Zheng, V. Noroozi, and P. S. Yu, "Joint deep modeling of users and items using reviews for recommendation," in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, 2017, pp. 425–434.

[18] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 448–456.

[19] C. Luo, W. Pang, Z. Wang, and C. Lin, "Hete-cf: Social-based collaborative filtering recommendation using heterogeneous relations," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2014, pp. 917–922.

[20] Z. Yu, D. Zhang, Z. Yu, and D. Yang, "Participant selection for offline event marketing leveraging location-based social networks," *IEEE T. Syst. Man Cybern.: Syst.*, vol. 45, no. 6, pp. 853–864, Jun. 2015.

[21] Y. Wen, J. Yeo, W. Peng, and S. Hwang, "Efficient keyword-aware representative travel route recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 8, pp. 1639–1652, Aug. 2017.

[22] H. Wang, M. Terrovitis, and N. Mamoulis, "Location recommendation in location-based social networks using user check-in data," in *Proc. 21st ACM SIGSPATIAL Int. Conf. Advances Geographic Inf. Syst.*, 2013, pp. 374–383.

[23] M. Jamali and L. Lakshmanan, "Heteromf: Recommendation in heterogeneous information networks using context dependent factor models," in *Proc. 22nd Int. Conf. World Wide Web*, 2013, pp. 643–654.

[24] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme, "Fast context-aware recommendations with factorization machines," in *Proc. 34th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2011, pp. 635–644.

[25] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M. Thalmann, "Who, where, when and what: Discover spatio-temporal topics for twitter users," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 605–613.

[26] Y. Kim, Y. Park, and K. Shim, "DIGTOBI: A recommendation system for digg articles using probabilistic modeling," in *Proc. 22nd Int. Conf. World Wide Web*, 2013, pp. 691–702.

[27] W. Huang, Z. Wu, L. Chen, P. Mitra, and C. L. Giles, "A neural probabilistic model for context based citation recommendation," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 2404–2410.

[28] R. He and J. McAuley, "Vbpr: Visual Bayesian personalized ranking from implicit feedback," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 144–150.

[29] J. Song, J. Xiao, F. Wu, H. Wu, T. Zhang, Z. M. Zhang, and W. Zhu, "Hierarchical contextual attention recurrent neural network for map query suggestion," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 9, pp. 1888–1901, Sep. 2017.

[30] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2015, pp. 1235–1244.

[31] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: An open architecture for collaborative filtering of netnews," in *Proc. ACM Conf. Comput. Supported Cooperative Work*, 1994, pp. 175–186.

[32] H. Wang, S. Xingjian, and D.-Y. Yeung, "Collaborative recurrent autoencoder: Recommend while learning to fill in the blanks," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 415–423.

[33] F. J. Massey Jr, "The Kolmogorov-Smirnov test for goodness of fit," *J. Amer. statistical Assoc.*, vol. 46, no. 253, pp. 68–78, 1951.

[34] F. E. Grubbs, "Procedures for detecting outlying observations in samples," *Technometrics*, vol. 11, no. 1, pp. 1–21, 1969.

[35] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proc. 15th Annu. Conf. Int. Speech Commun. Assoc.*, 2014, pp. 338–342.

[36] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 3294–3302.

[37] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, 2014.

[38] J. Gao, P. Pantel, M. Gamon, X. He, and L. Deng, "Modeling interestingness with deep neural networks," in *Proc. Conf. Empirical Methods Natural Language Process.*, 2014, pp. 2–13.

[39] A. K. Bhowmick, S. Suman, and B. Mitra, "Effect of information propagation on business popularity: A case study on Yelp," in *Proc. IEEE Int. Conf. Mobile Data Manage.*, 2017, pp. 11–20.

[40] S. Pramanik, M. Gundapuneni, S. Pathak, and B. Mitra, "Can i foresee the success of my meetup group?" in *Proc. IEEE/ACM Int. Conf. Advances Social Netw. Anal. Mining*, Aug. 2016, pp. 366–373.

[41] R. R. Sokal, "A statistical method for evaluating systematic relationship," *University Kansas Sci. Bulletin*, vol. 28, pp. 1409–1438, 1958.

[42] H. Tong, C. Faloutsos, and J.-Y. Pan, "Fast random walk with restart and its applications," in *Proc. 6th Int. Conf. Data Mining*, 2006, pp. 613–622.

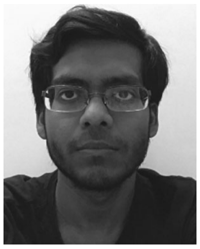
[43] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003.

[44] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Comput.*, vol. 42, pp. 30–37, 2009.

[45] C. Zhang, S. Jiang, Y. Chen, Y. Sun, and J. Han, "Fast inbound top-k query for random walk with restart," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2015, pp. 608–624.

[46] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 5353–5360.

**Soumajit Pramanik** received the MTech degree from the Department of Computer Science of the Indian Statistical Institute, Kolkata, India, in 2013. He is working toward the PhD degree from the Department of Computer Science & Engineering, Indian Institute of Technology Kharagpur, India. His research interests include complex networks, data mining, and machine learning.



**Rajarshi Haldar** received the BTech degree from the Department of Computer Science & Engineering, Indian Institute of Technology Kharagpur, India, in 2018. He is presently working toward the PhD degree in the Department of Computer Science, University of Illinois at Urbana-Champaign. His research interests include data mining and machine learning.

**Anand Kumar** received the MTech degree from the Department of Computer Science & Engineering, Indian Institute of Technology Kharagpur, India, in 2017. He is presently working at Qualcomm. His research interests include data mining and machine learning.

**Sayan Pathak** received the BTech degree from IIT Kharagpur, India, and the MS and PhD degrees from the University of Washington, USA. His research are in machine learning, artificial intelligence applied to Big data problems in vision, speech, and natural language processing. He is a principal machine learning scientist at Microsoft AI & Research, Redmond, Washington, USA.

**Bivas Mitra** received the PhD degree in computer science and engineering from IIT Kharagpur, India, in 2010. He worked as a chief engineer at Samsung Electronics, Noida, India. He held post-doctoral positions at CNRS, Paris, France, and the Universit Catholique de Louvain (UCL), Ottignies-Louvain-la-Neuve, Belgium. He is an assistant professor with the Department of Computer Science and Engineering, IIT Kharagpur. His current research interests include network science, social network, and socio-mobile applications.