A Convolutional Neural Tensor Network for Exploring User Preferences in Event-based Social Networks

ABSTRACT

In event-based social networks (EBSN), people of similar interest form groups on different themes. Understanding why people join a group and how groups evolve is critical for revealing the formation of different communities and the evolution of the entire network. In this paper, we study the problem of exploring user preferences in event-based social networks (EBSN) by considering content information, i.e., geographic information and social tags associated with each group. We propose a novel Convolutional Neural Tensor Network (CNTN) model to simultaneously model the content information and the social network structure. CNTN is able to extract and combine the features from various sources. We also develop a two-step pre-training algorithm for parameter initialization, which seems to be very effective in our experiments on a real-world EBSN social network. Our experiments also show clearly better performance of the proposed model over several comparison methods.

KEYWORDS

neural tensor network,event-based social network,embedding

ACM Reference format:
. 2016. A Convolutional Neural Tensor Network for Exploring User Preferences in Event-based Social Networks. In *Proceedings of ACM Conference, Tokyo,Japan, August 2017 (SIGIR'17),* 10 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In social networks, group is a fundamental concept and plays an important role in users social behavior involvement. While Charles Horton Cooley[1] proposed that social groups in offline social networks can be categorized into primary group, secondary group, and reference group, the situation for online social networks is much more complicated because of the rapid proliferation of users and social events. In online social networks, people may join several different groups for very different reasons. For example, a user may tentatively stay in a product discussion group to seek for opinions about a product, and participate in a reading group because of reading interest. Studies on *group* such as the formation of groups [5, 28] and predicting users' preferences on social groups [41] have been important issues in social networks.

In this paper we study the problem of predicting user preferences on social groups in a specific type of social networks, i.e., event-based social networks (EBSN). In ESBN, three types of entities users, groups and events are involved. Users can join different groups and attend events held within this group. Thus, in EBSN, the activities and associations of users are established within each group based on their attending events. Predicting user preferences on social groups is one of the most important tasks for EBSN to understand users and to make proper recommendations.

One important aspect of EBSN is the need of taking relational information into account. Many approaches have been proposed during the past years, e.g., the general purpose approaches like [23] combining various contents in the same factorization machine way, [18] considering various contents as equivalent nodes in a graph-based solution, and specific purpose approaches like [2] for location recommendation, [10, 20] for event recommendation. While works studying the effect of tags [41] and geographical event information [38] have been proposed, few systematically study the combination of the embedding of contents with the original distribution and gridding of contents.

Deep learning has dramatically improved the extractions of features and predictions of user preferences in social networks. Previous works [34, 39] use convolutional neural networks (CNN) to embed 2-dimension contents like images into features, and word embedding techniques [3] are proposed to embed contents to help prediction. [13, 29] proposes an auto-encoder based network to extract user features, while hierarchy models [21, 30] are proposed to deal with different contents specifically. However for locations in ESBN, traditional embedding often overlooks the natural distribution and clustering attributes of locations and the geographical relations between them.

Neural tensor network (NTN), which contains a bilinear tensor layer that connects two vectors, relates the two vectors across multiple dimensions with a tensor [26]. The tensor generally measures the relations of the two vectors and outputs a bilinear tensor product to represent the relationship. [40] firstly adopts NTN to measure the inner-relations of one concatenated vector, which is composed of multiple vectors from various aspects with concatenation. In this work, we extract user features and group features from multiple contexts, and adopts NTN to learn the inner-relations of concatenated feature vectors. The tensor in NTN plays a role in combining features from various contents with relations considering. Plus, it is an indispensable part of our pre-training strategy. Moreover, pre-training for deep neural networks have been researched for a long time. It has been proven that well-designed pre-training strategy could largely improve the initialization of parameters and improve the result of the training procedure.

In this paper, we propose a Convolutional Neural Tensor Network (CNTN) model that simultaneously considers the two types of relational information, the tags and geographical event information of users/events, to help predicting user preferences on social groups in EBSN. We propose a novel embedding algorithm to embed tags and locations into 2-dimension matrices, which considers the distribution

[1] https://en.wikipedia.org/wiki/Charles_Cooley

SIGIR'17, Tokyo,Japan

DOI: 10.1145/nnnnnnn.nnnnnnn

and neighbourhood effects of contents. A deep nueral tensor network is designed to further refine the features of users/groups from different contents and predict the users' preferences on groups. Furthermore, considering the importance of parameter initialization, a two-step pre-training algorithm is proposed for the proposed model.

Experiments on a real-world large-scale EBSN data set *Meetup*[2] show that our approach achieves significant better performance on both normal users and cold start users compared to state-of-the-art techniques.

2 RELATED WORK

Predicting user preferences on social groups has been a hot issue in recent years, especially with the rise of EBSNs. Existing research can be classified into general-purpose approaches and specific-purpose approaches. General-purpose approaches propose general and adaptable solutions for this issue. Graph-based solutions [18, 32] to take groups and users as vertices in social network graphs, together with other information. Tensor Decomposition solutions [41] translate relational contents into high-dimension tensor models. However these general approaches do not effectively catch the characteristics of EBSN and correlations of contents, thus we propose our model with a specific-purpose approach, which utilizes textual and geographical information as well as their correlations in EBSN, to solve our specific task.

For specific-purpose approaches in EBSNs, [31] extends Maximum Margin Matrix Factorization methods with social relations of users, and [36] extends Bayesian Poisson Factorization methods with textual information. [38] combines linear model with matrix factorization by extracting features from existing information. While textual and social information have been widely discussed, geographical event information is still in research in the task of predicting individuals' preferences for groups. [27, 33] propose local recommending systems for POI and news article recommendation, which provide different results according to users' current locations. However, user preferences for groups are relatively long-term interests and often do not change with users' current locations, these approaches are not adaptable to our task.

Deep neural network is an artificial neural network(ANN) which contains multiple hidden layers, which is proposed for feature extraction and representation. Specifically, convolutional neural network(CNN) is designed for 2-dimension inputs such as image features for prediction problems, which proves to be successful on various applications [11, 35, 40]. For instance, [14] uses embedding of events as inputs of CNN for click prediction problems, while [25] takes semantic embedding of word sequence as inputs for information retrieval. Neural Tensor Network, as a new kind of NN which is researched and developed recently, also contributes to the embedding and combination of features from various types of contents. Tensor neural network was originally proposed by [26] and adopted by [40] to measure the inner-relations of one concatenated vector, which is composed of multiple vectors from various aspects with concatenation. Our work proves again a NTN model captures more unrevealed characteristics about the relations, and better. Moreover, by unifying the representations/embeddings of different context, and combining with CNN process it can further be improved.

[2]http://www.meetup.com

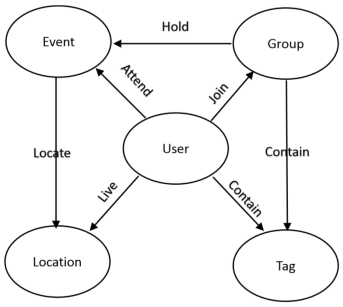

Figure 1: The entity relationship of an EBSN.

Pre-training in Deep neural network can improve the network [6] and thus researches are made on the strategy and method of pre-training. [17] adopts Restricted Boltzmann Machine (RBM) for the pre-training of network layers, while [19] introduces Sparse Encoding Symmetric Machines to find better initialization of parameters. [1, 7] proposes an auto-encoder based pre-training strategy for deep neural networks. However, in CNTN which contains various parts, a single kind of pre-training method can not involve the whole pre-training of the model. Thus we propose a two-step pre-training method for CNTN, which proves to be effective and improves the results of the prediction.

3 THE APPROACH

3.1 Preliminaries

In event-based social networks (EBSN) with tags and locations contextual information, the data can be described by the set $\{U, G, E, T, L\}$, where $U = \{u_i\}_{i=1}^{|U|}$, $G = \{g_i\}_{i=1}^{|G|}$, $E = \{e_i\}_{i=1}^{|E|}$,$T = \{t_i\}_{i=1}^{|T|}$, $L = \{l_i\}_{i=1}^{|L|}$, are respectively the sets of groups, users, events, tags and locations. The events E are held by the groups G and users U can join different groups and events based on personal interests. For user u, the set of groups it participates in is denoted as $GP_u \subset G$, while the set of missing data as $GN_u \subset G \setminus GP_u$, where $GP_u \cap GN_u = \emptyset$. The set of tags associated with user u and group g are respectively denoted as T_u and T_g. The own address of user u is denoted as L_u. E_g denotes the events held by group g, and E_u denotes the events user u participated in. The location of event e is denoted as L_e. Users, groups, events are typically described by their user ID, group ID and events ID. Tags can be arbitrary strings like phrase description for users/events. The locations are recorded by their coordinate (longitude, latitude). An example of EBSN is given in Fig. 1.

The entire CNTN model is depicted in Fig. 3. It is composed of three parts: the location&tag embedding, convolution neural network, and neural tensor network. The details of the model design is given in the following sections.

3.2 Location&Tag Embedding

In this section we introduce the process of gridding and embedding locations and tags as the the representaion of the users and groups. Firstly, as depicted in Fig. 2(1), the locations are positioned by their real longitude and latitude coordinates in the area. Then we divide the area into grids with equal size, where each grid contains a subset

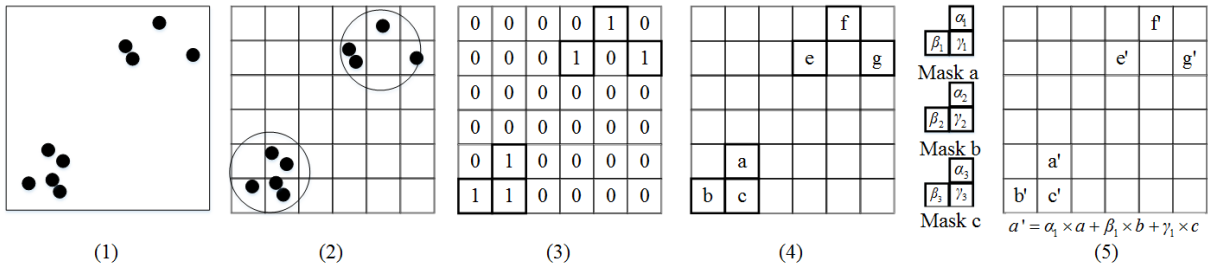

Figure 2: A typical example for the event region matrix of a certain user.

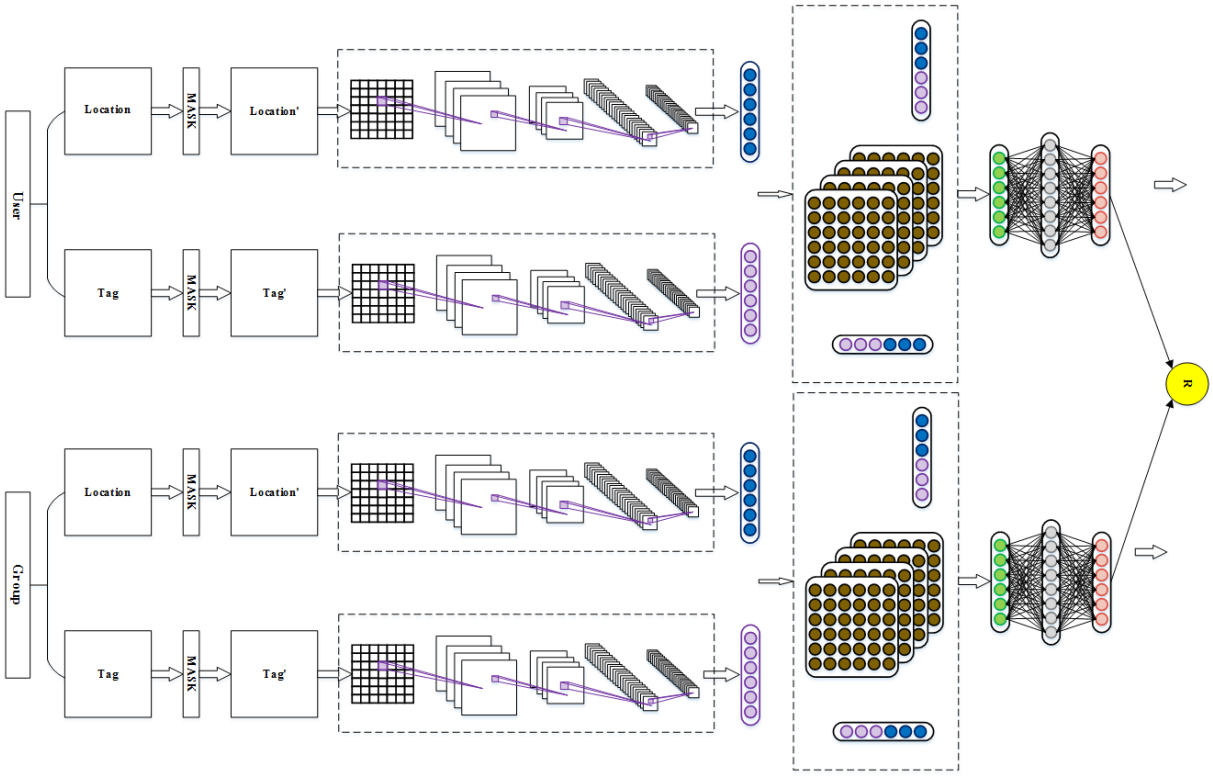

Figure 3: Convolutional Neural Tensor Network (CNTN)

region of the whole region, as depicted in Fig. 2(2). Each grid is regarded as a *event region*.

Similarly, other contexts in EBSN such as tags that contain semantic information can be treated in the same way as locations, with a certain process. That is, we put the tags into the corresponding grids according to their semantic vector and the gridding of the *semantic region* are obtained accordingly. The semantic vector is calculated by word embedding techniques to compress each tag into a 2-dimension vector like longitude and latitude for locations. Suppose a tag with its corresponding vector (a, b), then the tag is mapped into a point with coordinate (a, b) to generate the semantic region.

Then we classify the grids into clusters to show the inner relations of grids based on their positions. We apply DB-scan to perform the clustering, which is a density-based clustering algorithm, to divide locations into clusters without predefined clusters number. For a certain unvisited point, DB-scan finds density reachable points and generates a cluster, as depicted in Fig. 2(2). Then the girds with some locations in it is marked by 1, otherwise marked by 0. And these obtained clusters are mapped to Fig. 2(3), where each cluster is composed of several neighbored grids marked by 1, e.g., there are two clusters located in the up-right corner and left-down corner, respectively.

In the following, let $g_{i,j}$ be the grid for location (i, j), and c_g be the corresponding cluster for grid g. Let ER_u, ER_g be the *event region* for user u and group g, respectively. For each user u, we mark the grids that includes locations that u has visited, and Fig. 2(4) shows a typical example of user u's event region ER_u. For tags, we similarly obtain *semantic region* for each user and group, denoted respectively as ET_u, ET_g. Then for each grid $(i, j) \in ER_u$, we set a unique mask with shape of cluster $c_{g_{i,j}}$. Fig. 2(4) shows the shape of the mask$_{a,b,c}$ for a certain cluster, which is used to involve neighbourhood effects of the region. With the mask, the value of

Table 1: The structure of CNN.

Input Size	*Layer Info*
50×50	5×5, 32
46×46	2×2, max pooling
23×23	4×4, 64
20×20	5×5, average pooling
$5 \times 5 \times 64$	1024, flat concatenation

the grid (i, j) K^{ij} is set as parameters to evaluate the neighbourhood effect from other grids within the cluster and the final embedding of grid ER^K_{uij} is obtained by

$$ER^K_{uij} = \sum_{(m,n) \in c_{g_{i,j}}} ER_{umn} * K^{ij}_{m,n} \tag{1}$$

For instance, in Fig. 2(5), $a' = \alpha_1 a + \beta_1 b + \gamma_1 c$. In this way, for a specific point the cluster mask combines the nearby contexts in the same cluster.

3.3 CNN

Convolutional Neural Networks (CNN) are designed to process data with grid-like topology as well as reflect correlations of neighbourhood location effects. We adopt CNN to obtain the feature of event regions and semantic regions for each user u and group g, respectively denoted as UE_u, GE_g and UT_u, GT_g. Each convolutional layer can be defined as:

$$L^k_{x,y} = pool(\sigma(\sum_{i=0}^{K_x-1} \sum_{j=0}^{K_y-1} L^{k-1}_{x+i,y+j} K^k_{i,j})) \tag{2}$$

where $L^k_{x,y}$ denotes the value of position (x, y) for the k^{th} layer, and K^k is the kernel for the k^{th} layer. $\sigma(x)$ is the activation function, which is often set as the sigmoid function or tanh function. $pool(x)$ is a sub-sampling function which replaces the output of a certain position with a summary statistic of nearby outputs. The structure of the CNN part is summarized in Table 1.

On the other hand, David *et al.* [4] indicated that users in the same group contain similar tags/event regions, which means that user's preferences can be reflected by users within the same group. Thus we construct two more features for users/groups to gather additional information from their neighbourhoods as:

$$\begin{aligned} UEN_u &= \frac{1}{|N_u|} \sum_{v \in N_u} UE_u, GEN_g = \frac{1}{|N_g|} \sum_{v \in N_g} GE_g \\ UTN_u &= \frac{1}{|N_u|} \sum_{v \in N_u} UT_u, GTN_g = \frac{1}{|N_g|} \sum_{v \in N_g} GT_g \end{aligned} \tag{3}$$

where N_u is the set of users that attend same groups with u, and N_g is the set of groups that contain same users with g. Similarly we obtain the feature of neighbourhood event regions and semantic regions for each user u and group g, respectively denoted as UEN_u, GEN_g and UTN_u, GTN_g.

3.4 Neural Tensor Network

For each user u, we obtain UE_u, UT_u, UEN_u, UTN_u as measurements of u's preferences from different aspects. Similarly for each group g, GE_g, GT_g, GEN_g, GTN_g are obtained. To calculate the user preference vector P_u from the four relational vectors, traditional solutions [3] uses simple concatenation to obtain user/group vectors, i.e., we denote the concatenation of the four vectors of user u and group g as U_u and G_g:

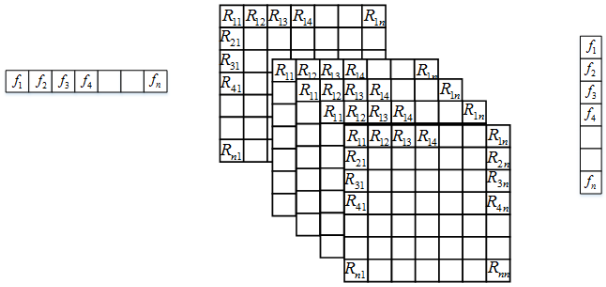

Figure 4: An Example of Neural Tensor Network.

$$\begin{aligned} U_u &= [UE_u; UT_u; UEN_u; UTN_u] \\ G_g &= [GE_g; GT_g; GEN_g; GTN_g] \end{aligned} \tag{4}$$

However, concatenation of vectors of users/groups fail to reveal the inner-relations of vectors. To this purpose, neural tensor network was originally designed and used for learning presentations for two vectors by [26]. And [40] adopted neural tensor network to learn presentations of one concatenated vector. In this paper we adopt the neural tensor network to obtain the user preference vector. For a feature $f \in \mathbb{R}^n$ which consists of various sub-features from different parts, i.e., $f = [f_1, f_2, ..., f_n]$, we try to find a relationship $R \in \mathbb{R}^{n \times n \times k}$ which reflects the inner relations of features. As shown in Fig. 4, $R_{i,j}$ measures the relations of f_i and f_j, and the result $f^T R f \in \mathbb{R}^k$ can be viewed as the feature that combines the sub-features with relation R, which better represents the overall feature instead of simply concatenating the sub-features. The relationship R is viewed as a parameter and can be obtained during training procedure.

Thus we define the neural tensor network as follows:

$$H_u = U_u^T M_u^{[1:ku]} U_u, H_g = G_g^T M_g^{[1:kg]} G_g \tag{5}$$

where $M_u \in \mathbb{R}^{|U_u| \times ku \times |U_u|}$, $M_g \in \mathbb{R}^{|G_g| \times kg \times |G_g|}$ is a 3-dimension relation tensor. ku and kg are the number of slices for the relation tensor, which also equals the length of H_u and H_g. Each slice can be interpreted as a way of measuring the relationship for the input concatenated vector. The output H_u and H_g are overall user/group feature vectors, which is composed through the relation tensor R. Note that the length of H_u and H_g may differ. To unify the length of user preference vector and group preference vector, we add a fully connected layer for H_u and H_g, and the final prediction for user u on group g is calculated as follows:

$$\hat{R}_{u,g} = (H_u W_u)^T H_g W_g \tag{6}$$

where $W_u \in \mathbb{R}^{k_u \times k}$ and $W_g \in \mathbb{R}^{k_g \times k}$ are parameters for the fully connected layer. W_u and W_g can be viewed as a weight parameter of a neural network layer for the output, where $H_u W_u, H_g W_g \in \mathbb{R}^k$ are user/group latent feature factors which reflect user/group's preference for each latent topic. The final prediction $\hat{R}_{u,g}$ for user u of group g is calculated as their interaction result.

M_u and M_g are tensors that links the concatenated vector U_u and G_g across multiple dimensions, which better reveals the relationship of the concatenated vector to generate better representations for the vectors. Furthermore, the pre-training of neural tensor network can be combined with the fully connected layer by variational Bayesian inference, which can greatly improve the prediction of the network. Based on the above two reasons, we adopt neural tensor network to generate user/group preference vectors. Another solution to give a final prediction on user preference for groups is to introduce another tensor to link the relations between user feature and group feature. However, due to the higher time complexity of variational Bayesian inference pre-training for a neural tensor network, we adopt a fully connected layer instead in our model structure design.

3.5 Optimization

The loss function of the model is to minimize the error of the prediction of $\hat{R}_{u,g}$ and the actual preference of $R_{u,g}$ for each (u, g) that user u joins/dislikes the group g:

$$L = \sum_{(u,g)} (\hat{R}_{u,g} - R_{u,g})^2 + \lambda ||\theta||^2 \tag{7}$$

where θ is the parameters involved in the model. It can be trained by back-propagation algorithm, which uses the gradient of the loss function to adjust the weights of the model as follows:

$$\theta := \theta - \alpha \frac{\partial}{\partial \theta} L(\theta)$$

where θ denotes the parameters of the model, like the weights and convolutional kernels in the neural network, and α is a factor that adjusts the learning rate. The overall algorithm are summarized in Algorithm 1.

In deep neural networks, proper initialization of parameters can obtain better results of the model, thus we design a two-step pre-training procedure for CNTN, which will be introduced in detail in the following.

3.6 Pre-training Strategies

For deep neural networks, parameter initialization influences the result of the network. Pre-training for the network can help find a better initialization of parameters can improve the result of the prediction. The pre-training for CNTN contains two parts: pre-training for user/group embedding part and for neural tensor network part. In the following we will introduce them in details.

3.6.1 Location and Tag Embedding Part Pre-training. For location and tag embedding part, we adopt the convolutional auto encoder (CAE) [16], which convolute the input in the same way as CNN and reconstruct the input from feature map. The encoder and decoder operations for CAE is as follows:

$$\begin{aligned} Encoding: h^k &= \sigma(x * W^k + b^k) \\ Decoding: y &= \sigma(\sum_{k \in H} h^k \cdot w'^k + b'^k) \end{aligned} \tag{8}$$

where x, h^k respectively denotes the input and output for the k^{th} channel, for the encoding operation. W^k, b^k denote the convolutional kernel and bias parameters for the convolutional layer, and $*$ denotes the 2D-Convolution operation. Similarly, h^k, y respectively denotes the input and output for the k^{th} channel, and $\cdot$ denotes the linear multiplication operation, for the decoding operation. The cost function is to minimize the mean square error (MSE) between the input x and output y of CAE:

$$L = \frac{1}{2n} \sum_{i=1}^{n} (x_i - y_i)^2 + \lambda ||\theta||^2 \tag{9}$$

Input: {U,G,E,T,L} and their corresponding relations (See Section.3.1, Fig. 1);
Output: Group prediction matrix $\hat{R}_{u,g}$;

Using DB-Scan to generate the shape of kernels for each grid; (See Section.3.2, Fig. 2)
foreach $u \in U$ **do**
  Generate ER_u, ET_u (See Section.3.2, Fig. 2)
end
foreach $g \in G$ **do**
  Generate ER_g, ET_g (See Section.3.2, Fig.2)
end
All the parameters are initialized randomly within [0,0.01];

Pre-train the embedding part;
repeat
  foreach $u \in ER_u$ **do**
    Calculate output of CAE; Eq.8
    Calculate the loss of CAE and update W^k and b^k; Eq.9
  end
until *Convergence*;
Set W^k and b^k as the corresponding convolutional layer parameter in CNN part of CNTN;
Similarly pre-training CAE for ET_u, ER_g and ET_g;

Pre-train the neural tensor part;
repeat
  foreach *(u, g),* $u \in U$, $g \in G$ **do**
    Obtain H_u and H_g as output for neural tensor network; Eq.10
    Calculate U_u and G_g from CNN using the previous pre-training parameters; Eq.4
    Use back propagation to update M_u and M_g of neural tensor network part;
  end
until *Convergence*;
Set M_u and M_g as the tensor in neural tensor part of CNTN;

repeat
  foreach *(u, g),* $u \in U$, $g \in G$ **do**
    Calculate ER_u^K, ET_u^K (Eq.1)
    Calculate ER_g^K, ET_g^K (Eq.1)
    Use ER_u^K, ET_u^K as inputs to obtain UE_u, UT_u, UEN_u, UTN_u (Eq.3)
    Use ER_g^K, ET_g^K as inputs to obtain UE_g, UT_g, UEN_g, UTN_g (Eq.3)
    Calculate U_u and G_g (Eq.4)
    Obtain H_u and H_g (Eq.5)
    Calculate $R_{u,g}$ (Eq.6)
    Use back-propagation algorithm to update the weights;
  end
until *Convergence*;
Calculate the prediction matrix $\hat{R}_{u,g}$; (Eq.6)

Algorithm 1: CNTN Algorithm

In our model, we set $x = \{ER_u, ET_u, ER_g, ET_g\}$ to respectively pre-train their parameters in the CNN part. After pre-training, the value each convolution kernel parameter W^k and bias b^k for each layer are introduced as the initialized value of CNTN in the corresponding convolutional layer.

3.6.2 Neural Tensor Network Part Pre-training. For neural tensor network part, we combine back propagation method with variational Bayesian inference to pre-train this part. we assume the preference of users for groups as a Poisson distribution:

$$Poisson(R_{u,g}|U_u^T V_g) = (U_u^T V_g)^{R_{u,g}} exp(-U_u^T V_g)/R_{u,g}! \quad (10)$$

where $U_u, V_g \in \mathbb{R}^k$ are latent preferences for user u and group g. Furthermore, the two parameters are designed as a Gamma distribution:

$$Gamma(U_u|\lambda_{ua}, \lambda_{ub}) = \frac{\lambda_{ub}^{\lambda_{ua}}}{\Gamma(\lambda_{ua})} U_u^{\lambda_{ua}-1} exp(-\lambda_{ub} U_u)$$

$$Gamma(V_g|\lambda_{ga}, \lambda_{gb}) = \frac{\lambda_{gb}^{\lambda_{ga}}}{\Gamma(\lambda_{ga})} U_u^{\lambda_{ga}-1} exp(-\lambda_{gb} V_g)$$

where λ_{ua}, λ_{ga} are shape parameters of the Gamma distribution, and λ_{ub}, λ_{gb} is rate parameters of the Gamma distribution.

To obtain the latent factors U_u and V_g in our model, we use variational Bayesian inference [8, 37]. The general idea of variational Bayesian inference is to derive the lower bound of the normalization term $p(R_{u,g}|\Theta)$, and then optimizes the lower bound through standard learning algorithms. The lower bound is usually obtained by applying Jensen's equality through a new designed variational distribution $q(\Theta)$,

$$\begin{aligned} log p(R) &= log \int p(R, \Theta) d\Theta \\ &= log \int p(R, \Theta) \frac{q(\Theta)}{q(\Theta)} d\Theta \\ &\geq q(\Theta) log \int \frac{p(R, \Theta)}{q(\Theta)} d\Theta \\ &= L(q) \end{aligned} \quad (11)$$

To implement variational Bayesian inference, we first introduce several types of auxiliary latent variables. Specifically, for each user-group pair (u, g), we add latent variables $r_{u,g} \sim Poisson(U_u^T V_g)$. After adding those new latent variables defined as Z, the variational distribution becomes $q(\Theta, Z)$.

Then we derive the complete conditional distribution for each latent variable. We take U_u as an instance and fix other latent variables, then we obtain the complete conditional distribution of U_u as follows:

$$\begin{aligned} & p(U_u|z, V_g, \lambda_{ua}, \lambda_{ub}) \\ & \propto U_u^{\lambda_{ua}-1} exp(-\lambda_{ub} U_u) \prod_g U_u V_g^{r_{u,g}} exp(-U_u V_g) \\ & \propto U_u^{\lambda_{ua}+\sum_g r_{u,g}-1} exp(-(\lambda_{ub} + \sum_g V_g) U_u) \\ & = Gamma(\lambda_{ua} + \sum_g r_{u,g}, \lambda_{ub} + \sum_g V_g) \end{aligned} \quad (12)$$

Then we update the parameters of Gamma distribution for each latent factor. We take U_u as an example and λ_{ua} and λ_{ub} are calculated as follows:

$$\lambda_{ua} = \lambda_{ua} + \sum_g R_{u,g} e^{\psi(\alpha)+\psi(\lambda_{ga})-log(\lambda_{gb})}$$

$$\lambda_{ub} = \lambda_{ub} + \sum_g \frac{\lambda_{ga}}{\lambda_{gb}}$$

where $\psi(x)$ is the digamma function. After we use variational Bayesian inference to obtain the values of U_u and V_g, which is also the output for the neural tensor network, we further apply back propagation for pre-training parameters within the neural tensor network.

4 EXPERIMENTS

Datasets We conduct experiments on the real-world EBSN Meetup data [15], which is an online social event service helping people publish and participate in face-to-face events. As observed, only 1% events are held across different cities. With such a weak connection the entire Meetup data actually consists of some sub-datasets around several big cities. Therefore, we select three metropolises, *LA (Los Angeles)*, *SF (San Francisco)* and *NYC (New York City)* from Meetup to form three data sets. Table 2 lists statistics of the datasets.

Baseline Algorithms Six baselines from recommendation fields are incorporated as baselines.

- **SVD++** takes implicit feedback of users into consideration [12], which provides an additional indication of user preferences to matrix factorization. In EBSN, implicit feedbacks are important for group recommendation.
- **BPR-MF** [22] is a latent factor collaborative filtering algorithm with the optimization of Bayesian Personalized Ranking criterion for personalized ranking problems, which aims at ranking positive groups higher than negative ones.
- **PTA** PTARMIGAN [38] is a pair-wise tag enhanced and feature-based matrix factorization algorithm, which extracts and utilizes features from contexts such as tags and geographical information with a combination of linear model and latent factor model, and is designed specifically for EBSN group recommendation.
- **NMF** Non-negative Matrix Factorization [9] puts a Gamma prior on user/group vectors and uses Bayesian inference to update the parameters, which restricts user/group feature as non-negative vectors. NMF does not require negative examples, i.e., user dislike groups, which is hard to achieve in EBSN, thus it fits our task.
- **ConvMF** Convolutional Matrix Factorization [11] is a type of deep neural network which extracts user/group features from 2D inputs like images or event regions for recommendation systems, which is originally designed for cloth recommendation but also suits for our task.
- **NNED** [40] is a deep network with neural tensor that is used to calculate relations of semantic phrases and sentences with word embedding techniques through a specifically-designed tensor that captures the documents' underlying features.

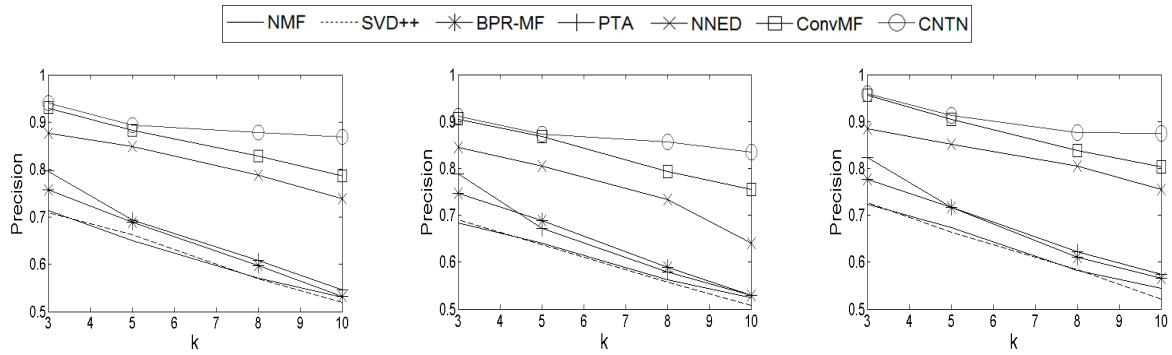

Figure 5: The *Precision* results on data sets *SF*, *LA*, *NYC*, with $k = 3, 5, 8, 10$.

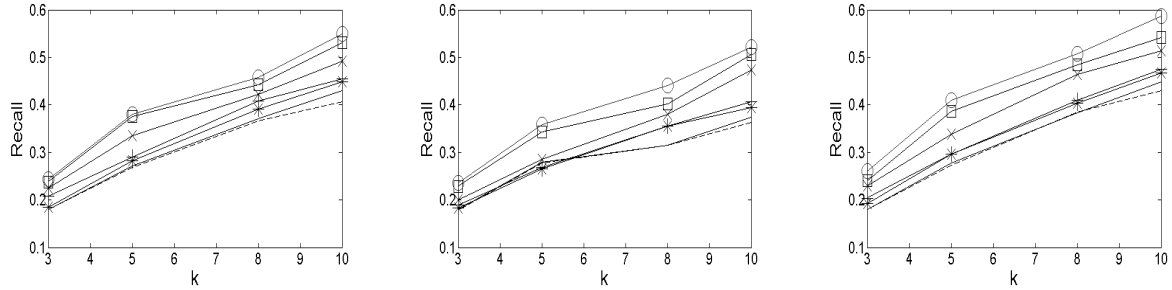

Figure 6: The *Recall* results on data sets *SF*, *LA*, *NYC*, with $k = 3, 5, 8, 10$.

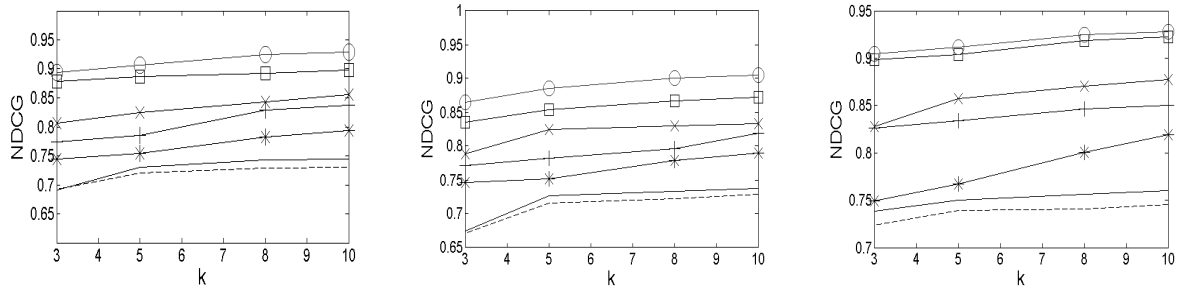

Figure 7: The *NDCG* results on data sets *SF*, *LA*, *NYC*, with $k = 3, 5, 8, 10$.

Evaluation Measures In this work we perform our model and rank the predictions to provide top-k groups for the user, with $k = \{3, 5, 8, 10\}$. Four commonly used metrics are adopted, including three classification measures *precision*, *recall*, F_1 *score* and a ranking measure *NDCG* (Normalized Discounted Cumulative Gain) [24].

Then we calculate the prediction list B by setting the top k groups with highest prediction values as positive labels. $Precision = \frac{|A \cap B|}{|B|}$ measures the fraction of correct groups that a recommendation system give in its prediction list B; $Recall = \frac{|A \cap B|}{|A|}$ measures the fraction of correct groups that occurs in the test list A. F_1 score considers both precision and recall computed as $F_1 = \frac{2 \times precision \times recall}{precision + recall}$

NDCG takes into account the ranking performance which requires relevant groups to be ranked higher. The *NDCG* value for the top-k prediction list is computed as $NDCG_k = \frac{DCG_k}{IDCG_k}$,where $DCG_k = \sum_{i=1}^{k} \frac{2^{rel_i}-1}{\log_2(i+1)}$, with $rel_i = 1$ when the group at rank i is relevant (0 otherwise), and $IDCG_k$ is the maximum possible (ideal) *DCG* for a given set of recommended groups. Note that the larger the *precision*, *recall*, F_1 and *NDCG* values are, the better the performance is.

For each user in the data sets, 90% of his/her involved groups are randomly selected to serve as the training set, and the rest serves as test set. Negative examples are randomly selected from the missing data with an equal size of the positive examples. The algorithms are performed on the datasets with 10-fold cross-validations. For the presented results, the differences between the algorithms are statistically significant ($p < 0.01$ with t-test). All the parameters are initialized randomly within [0, 0.01]. The batch size for the pre-training on embedding part and tensor part is set to 5, while the

Table 2: The statistics of the three data sets *NYC*, *LA*, *SF*.

	NYC	*LA*	*SF*
$\#users$	607, 177	386, 321	427, 112
$\#groups$	6, 099	3, 921	5, 177
$\#events$	110, 929	65, 533	99, 012
$\#tags$	37, 326	31, 694	34, 057
$\#(user, group)\ pairs$	1, 573, 432	1, 029, 248	1, 017, 274
$\#(user, event)\ pairs$	1, 154, 706	752, 245	774, 191
$\#(group, event)\ pairs$	136, 880	82, 613	136, 984

batch size for the overall training is set to 10, and we conduct the experiment on a machine with one GPU (NVIDIA GTX-1080) and one CPU (INTEL Xeon E5-2620).

4.1 Results

Results on three data sets *NYC*, *LA*, *SF* in terms of *NDCG*, *precision*, *recall* and F_1 score are respectively shown in Fig. 7, Fig. 5, Fig. 6 and Table 3. For *NDCG* and F_1 score, results for top-k group recommendation with k varying from 3 to 10 are reported.

Baseline Comparison

First we compare the performance of CNTN with the baseline algorithms. Fig. 7 depicts the results on NDCG. Among them two pure matrix factorization algorithms SVD++ and NMF performs worse than other more complicated strategies. NMF performs slightly better by benefitting from the non-negative restriction. BPR-MF and PTA enforces the Bayesion pairwise ranking mechnism on the matrix factorizagion algorithm, which brings an obvious increase in NDCG. PTA performs better than BPR-MF since it integrates more contexts including tags and geo information. Then, with a deep neural network and the tensor embedding design NNED shows an advantage over the algorithms with a relatively shallow structure. ConvMF gives a superior performance over the rest baseline algorithms since it successfully combines the CNN that also has a deep neural network structure and the tensor mechnism.

From Table 3 and Fig. 7, we can see that our approach performs significantly better than baselines on the three data sets on *NDCG* and F_1 score. Specifically, on *LA* dataset, in terms of F_1 score CNTN improves 5.95% over ConvMF, 39.57% over PTA and 47.16% over NMF. In terms of NDCG, CNTN improves 8.59% over NNED, 14.58% over BPR-MF and 24.09% over SVD++.

The *precision* and *recall* results in Fig. 5 and 6 also validate the big performance improvements of CNTN over baselines. Exploiting the tag and location embedding and combining features with tensors, CNTN is able to rank user preferred groups higher and thus achieves better performance. Note that there is a big gap between the algorithms that exploits the deep neural network structure, ie., NNED, ConvMF and CNTN, and the ones that have not used it.

Model Parts Study

We further study the effects of key designs of our model, i.e., the tag&location embedding design and the neural tensor network design, to see how they impact the performance. To evaluate the effect of the tag&location embedding part, we replace our designed embedding with a direct embedding mechanism with one-hot vector as inputs, i.e., each position represents a specific tag/location and the user tag vector can be obtained as the combinations of all tags he/she

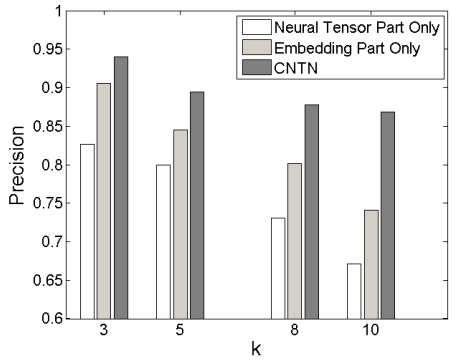

Figure 8: The *Precision* results for each part of the model on data set *SF*, with $k = 3, 5, 8, 10$.

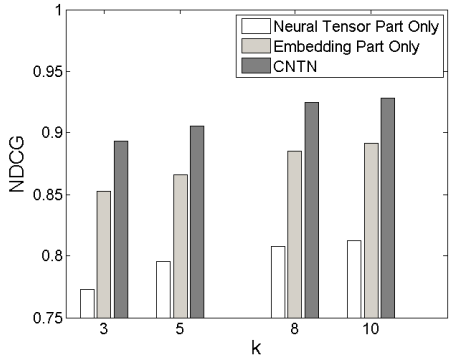

Figure 9: The *NDCG* results for each part of the model on data set *SF*, with $k = 3, 5, 8, 10$.

contains. And to evaluate the effect of the neural tensor network part, we replace the neural tensor network with a fully connected layer of the neural network. Fig. 8, 9 and 4 show the top k *precision*, F_1 and *NDCG* results on *SF* data set with k varying from 3 to 10.

For the sake of page limitation here we only present the results of one dataset, the rest evaluation measures resembles the presented results. It indicates that each part in the models plays an important role in improving the final performance. Specifically, the embedding part design works better than traditional embedding mechanisms which embed locations into one-dimension inputs, and the neural tensor network part has better performance than directly concatenating features by considering inner-relations within features from various contents. From the difference between the entire model and the $model_E$ we can see that the tensor part contributes 17.21% improvement on the performance, on the other hand, the design of our embedding method with CNN contributes 29.37% improvement.

Model Pre-training Study We also study the performance of the pre-training effect for our proposed model. For this study, we compare our designed model without pre-training and the model with pre-training. The top k results comparison results of the model with/without pre-training on the *SF* dataset are shown in Fig. 10, 11 and 5, with k varying from 3 to 10.

Fig. 10 and 11 depict that proposed pre-training achieves great advantages over the model without pre-training procedure, e.g., 5.75%

Table 3: F_1 score of the top-k ($k = 3, 5, 8, 10$) group recommendation on data sets *SF*, *LA*, *NYC*.

City	SF				LA				NYC			
Method/Rec #	3	5	8	10	3	5	8	10	3	5	8	10
NMF	0.286	0.382	0.449	0.474	0.283	0.389	0.403	0.435	0.287	0.394	0.462	0.491
SVD++	0.286	0.381	0.445	0.455	0.283	0.387	0.402	0.422	0.288	0.386	0.464	0.470
BPR-MF	0.297	0.400	0.472	0.486	0.293	0.382	0.443	0.451	0.308	0.418	0.486	0.512
PTA	0.329	0.408	0.488	0.495	0.305	0.383	0.440	0.459	0.326	0.418	0.494	0.519
NNED	0.357	0.479	0.549	0.590	0.323	0.421	0.500	0.544	0.363	0.484	0.589	0.611
ConvMF	0.379	0.527	0.576	0.633	0.365	0.491	0.533	0.605	0.384	0.541	0.614	0.647
CNTN	**0.386**	**0.533**	**0.601**	**0.672**	**0.374**	**0.507**	**0.582**	**0.641**	**0.409**	**0.565**	**0.643**	**0.702**

Table 4: F_1 score for the embedding method and tensor network of CNTN, respectively, on *SF* data set.

Method/Rec# k	3	5	8	10
Neural Tensor Part Only	0.322	0.458	0.508	0.558
Embedding Part Only	0.353	0.492	0.534	0.617
CNTN	**0.386**	**0.533**	**0.601**	**0.672**

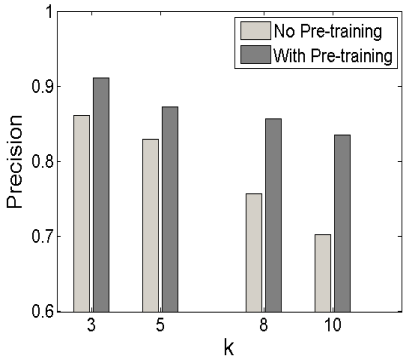

Figure 10: The *Precision* results for pre-training of the model on data set *SF*, with $k = 3, 5, 8, 10$.

improvement in terms of precision and 7.38% improvement in terms of NDCG. This indicates our designed pre-training method for the model plays an important role and improves the effects of the predictions. Moreover, Fig. 12 depicts the convergence curves of the model with and without the pre-training. By observing the value of loss function defined in Eq.7 in each iteration, we can see that with the proper pre-training strategy the algorithm converges in a much faster speed and obtains a much better solution in the end, since the parameters are initialized in a better position. Specifically, the model with the pre-training converges around the first 35 iteration whereas without pre-training it goes down very slowly and takes more than 55 iterations, which indicates that the pre-training process reduces the time cost by 42.86%.

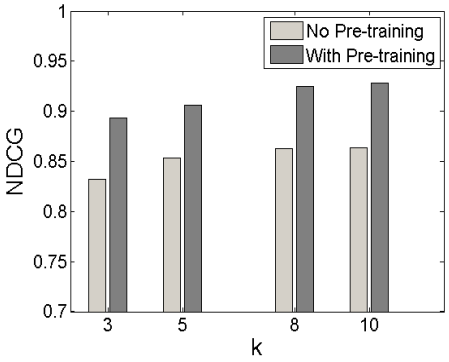

Figure 11: The *NDCG* results for pre-training of the model on data set *SF*, with $k = 3, 5, 8, 10$.

Table 5: F_1 score for the pre-training strategy on *SF* data set.

Method/Rec# k	3	5	8	10
Without Pre-training	0.330	0.474	0.518	0.567
With Pre-training	**0.380**	**0.533**	**0.601**	**0.672**

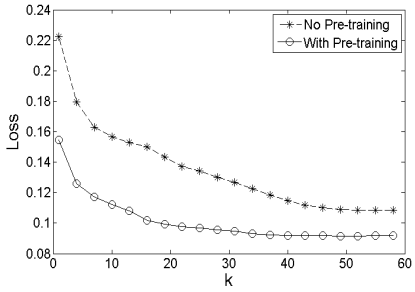

Figure 12: The *NDCG* results for pre-training of the model on data set *SF*.

5 CONCLUSION

In this paper, we study the problem of predicting user preferences for groups in social networks by exploiting event location and semantic tag content information in EBSN. We propose CNTN to model user behaviors taking into account the content information. We propose a novel embedding method for locations considering the geographical position and clustering properties of locations. A neural tensor network is adopted for combining features from various contents to reveal the inner-relations between features. A two-step pre-training procedure is designed to pre-train the parameters of the model.

REFERENCES

[1] Xin Qi Bao and Yun Fang Wu. 2016. A Tensor Neural Network with Layerwise Pretraining:Towards Effective Answer Retrieval. *Journal of Computer Science & Technology* 31, 6 (2016), 1151–1160.

[2] C. Cheng, H. Yang, M. R. Lyu, and I. King. 2013. Where You Like to Go Next: Successive Point-of-interest Recommendation. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*. Beijing, China, 2605–2611.

[3] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *ACM Conference on Recommender Systems*. 191–198.

[4] D.J. Crandall, L. Backstrom, D. Cosley, S. Suri, D. Huttenlocher, and J. Kleinberg. 2010. Inferring social ties from geographic coincidences. *Proceedings of the National Academy of Sciences of the United States of America* 107, 52 (2010), 22436–22441.

[5] Y. Dong, J. Zhang, J. Tang, N. V. Chawla, and B. Wang. 2015. CoupledLP: Link Prediction in Coupled Networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Sydney,Australia, 199–208.

[6] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why Does Unsupervised Pre-training Help Deep Learning? *Journal of Machine Learning Research* 11, 3 (2010), 625–660.

[7] Stefan Glüge, Ronald Böck, and Andreas Wendemuth. 2013. Auto-encoder pre-training of segmented-memory recurrent neural networks.. In *ESANN*.

[8] P. Gopalan, L. Charlin, and D. M. Blei. 2014. Content-based recommendations with Poisson factorization. *Advances in Neural Information Processing Systems* 4 (2014), 3176–3184.

[9] Prem Gopalan, Jake M. Hofman, and David M. Blei. 2013. Scalable Recommendation with Poisson Factorization. *Computer Science* (2013).

[10] X. Ji, Z. Qiao, M. Xu, P. Zhang, C. Zhou, and L. Guo. 2015. Online Event Recommendation for Event-based Social Networks. In *Proceedings of the 24th International Conference on World Wide Web Companion*. Florence, Italy, 45–46.

[11] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. 2016. Convolutional Matrix Factorization for Document Context-Aware Recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. 233–240.

[12] Y. Koren, R. Bell, and C. Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer* 42, 8 (2009), 30–37.

[13] Sheng Li, Jaya Kawale, and Yun Fu. 2015. Deep Collaborative Filtering via Marginalized Denoising Auto-encoder. In *The ACM International*. 811–820.

[14] Qiang Liu, Feng Yu, Shu Wu, and Liang Wang. 2015. A Convolutional Click Prediction Model. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM '15)*. 1743–1746.

[15] X. Liu, Q. He, Y. Tian, W.-C. Lee, J. McPherson, and J. Han. 2012. Event-based Social Networks: Linking the Online and Offline Social Worlds. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Beijing, China, 1032–1040.

[16] Jonathan Masci, Ueli Meier, Ciresan Dan, and Jurgen Schmidhuber. 2011. Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction. In *Artificial Neural Networks and Machine Learning - ICANN 2011 - International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings*. 52–59.

[17] Toru Nakashika, Tetsuya Takiguchi, and Yasuo Ariki. 2015. Voice Conversion Using RNN Pre-trained by Recurrent Temporal Restricted Boltzmann Machines. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.* 23, 3 (March 2015), 580–587.

[18] T.A.N. Pham, X. Li, G. Cong, and Z. Zhang. 2015. A general graph-based model for recommendation in event-based social networks. In *Proceeding of the 31st IEEE International Conference on Data Engineering*. Seoul, South Korea, 567–578.

[19] C Plahl, T. N Sainath, B Ramabhadran, and D Nahamoo. 2012. Improved pre-training of Deep Belief Networks using Sparse Encoding Symmetric Machines. In *IEEE International Conference on Acoustics, Speech and Signal Processing*. 4165–4168.

[20] Z. Qiao, P. Zhang, Y. Cao, C. Zhou, L. Guo, and B. Fang. 2014. Combining heterogenous social and geographical information for event recommendation. In *AAAI Conference on Artificial Intelligence*.

[21] Yogesh Singh Rawat and Mohan S. Kankanhalli. 2016. ConTagNet: Exploiting User Context for Image Tag Recommendation. In *ACM*. 1102–1106.

[22] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. 2012. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. Montreal, Canada, 452–461.

[23] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. 2011. Fast Context-aware Recommendations with Factorization Machines. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Beijing, China, 635–644.

[24] M. Richardson, E. Dominowska, and R. Ragno. 2007. Predicting Clicks: Estimating the Click-through Rate for New Ads. In *Proceedings of the 16th International Conference on World Wide Web*. Banff, Canada, 521–530.

[25] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM '14)*. 101–110.

[26] R. Socher, D. Chen, C. D. Manning, and A. Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. (2013).

[27] J.-W. Son, A.Y. Kim, and S.-B. Park. 2013. A Location-based News Article Recommendation with Explicit Localized Semantic Analysis. In *Proceedings of the 36th International ACM SIGIR conference on research and development in Information Retrieval*. Dublin, Ireland, 293–302.

[28] Y. Sun, J. Tang, J. Han, C. Chen, and M. Gupta. 2014. Co-Evolution of Multi-Typed Objects in Dynamic Star Networks. *IEEE Transaction on Knowledge and Data Engineering* 26, 12 (2014), 2942–2955.

[29] Hao Wang, Naiyan Wang, and Dit Yan Yeung. 2014. Collaborative Deep Learning for Recommender Systems. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1235–1244.

[30] Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. 2016. CNN-RNN: A Unified Framework for Multi-label Image Classification. (2016).

[31] Yilun Wang, Liang Chen, and Jian Wu. 2013. Content-boosted Maximum Margin Matrix Factorization for Flickr Group Recommendation. In *Proceedings of the 2013 Workshop on Data-driven User Behavioral Modelling and Mining from Social Media (DUBMOD '13)*. New York, NY, USA, 13–16.

[32] L. Wu, X. Ying, X. Wu, and Z.-H. Zhou. 2011. Line Orthogonality in Adjacency Eigenspace with Application to Community Partition. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*. Barcelona,Spain, 2349–2354.

[33] H. Yin, Y. Sun, B. Cui, Z. Hu, and L. Chen. 2013. LCARS: A Location-content-aware Recommender System. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Chicago, IL, 221–229.

[34] Chao Zhang, Junchi Yan, Changsheng Li, Xiaoguang Rui, Liang Liu, and Rongfang Bie. 2016. On Estimating Air Pollution from Photos Using Convolutional Neural Network. In *ACM on Multimedia Conference*. 297–301.

[35] Chao Zhang, Junchi Yan, Changsheng Li, Xiaoguang Rui, Liang Liu, and Rongfang Bie. 2016. On Estimating Air Pollution from Photos Using Convolutional Neural Network. In *Proceedings of the 2016 ACM on Multimedia Conference (MM '16)*. 297–301.

[36] Wei Zhang and Jianyong Wang. 2015. A Collective Bayesian Poisson Factorization Model for Cold-start Local Event Recommendation. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. 1455–1464.

[37] Wei Zhang and Jianyong Wang. 2015. A Collective Bayesian Poisson Factorization Model for Cold-start Local Event Recommendation. In *Kdd*. 1455–1464.

[38] W. Zhang, J. Wang, and W. Feng. 2013. Combining Latent Factor Model with Location Features for Event-based Group Recommendation. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Chicago, IL, 910–918.

[39] Shichao Zhao, Youjiang Xu, and Yahong Han. 2016. Large-Scale E-Commerce Image Retrieval with Top-Weighted Convolutional Neural Networks. 285–288.

[40] Shichao Zhao, Youjiang Xu, and Yahong Han. 2016. Large-Scale E-Commerce Image Retrieval with Top-Weighted Convolutional Neural Networks. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval (ICMR '16)*. 285–288.

[41] N. Zheng, Q. Li, S. Liao, and L. Zhang. 2010. Flickr Group Recommendation Based on Tensor Decomposition. In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. Geneva, Switzerland,, 737–738.