Social-Aware Sequential Modeling of User Interests: A Deep Learning Approach

Chi Harold Liu, *Senior Member, IEEE*, Jie Xu, Jian Tang, *Senior Member, IEEE*, and Jon Crowcroft, *Fellow, IEEE*

Abstract—In this paper, we propose to leverage the emerging deep learning techniques for sequential modeling of user interests based on big social data, which takes into account influence of their social circles. First, we present a preliminary analysis for two popular big datasets from Yelp and Epinions. We show statistically sequential actions of all users and their friends, and discover both temporal autocorrelation and social influence on decision making, which motivates our design. Then, we present a novel hybrid deep learning model, Social-Aware Long Short-Term Memory (SA-LSTM), for predicting the types of item/PoIs that a user will likely buy/visit next, which features stacked LSTMs for sequential modeling and an autoencoder-based deep model for social influence modeling. Moreover, we show that SA-LSTM supports end-to-end training. We conducted extensive experiments for performance evaluation using the two real datasets from Yelp and Epinions. The experimental results show that (1) the proposed deep model significantly improves prediction accuracy compared to widely used baseline methods; (2) the proposed social influence model works effectively; and (3) going deep does help improve prediction accuracy but a not-so-deep deep structure leads to the best performance.

Index Terms—Social networking, user interest modeling, deep learning, recurrent neural network, autoencoder

1 Introduction

Recent years have witnessed the emergence and popularity of Internet and smart mobile devices, which has stimulated the excessive use of various online services (such as online urban guide, online booking and shopping) via websites or mobile apps such as Yelp, TripAdvisor, Epinions, Facebook, Foursquare, etc. Moreover, the rise of social networking enables online users to build connections with each other and share their interests and experiences. We notice that some popular online services, such as Yelp and Epinions, have also integrated social information (such as friendship) into their systems for better services. In order to provide high-quality services and build a cost-effective and profitable business, it is critical for those online service providers to understand interests, needs, habits, lifestyles and/or preferences of their users, as well as their social circles and the corresponding influences.

Towards this end, recommender systems and algorithms [20], [29], [30], [32], [42] have been investigated and developed to understand user interests and help them find the most interesting products or content (e.g., a book from Amazon, a news article from CNN, or a game from PS Plus) without spending too much time on searching for the next buy. Another line of related works is to analyze/predict Points of Interest (PoIs) or types of PoIs of online users based on contextual (such as location) information collected by a mobile app. For example, a recent work [33] presented a system called LIPS, which can learn lifestyles of mobile users via mobile phone sensing using both a clustering algorithm for PoI identification and Support Vector Machines (SVM) for classification. In [31], the authors presented NextPlace, an approach to location prediction based on nonlinear time series analysis of the arrival and residence times of users in relevant places. However, most of these methods failed to capture the inherited temporal autocorrelations and dynamics hidden in the historical data. Specifically, they simply treat all user data (likely collected at different time) in the same and static way, which essentially implies that user interests and needs are not time-sensitive or temporally correlated. This may not be true, for example, a user may likely go to a restaurant after shopping at a grocery store on Saturday (rather than on a weekday). In addition, scant research efforts on recommendation has paid enough attention to how social circles influence user's decision making. However, a recent trend is the introduction of social features [27] into online services. For example, Yelp records and maintains a list of friends for each user; Epinions shows a user's recent reviews of her friends; and Amazon allows users to share their purchases on Facebook and/or Twitter. These social information are apparently beneficial: for a user, friends are more trustful and tend to share certain level of common interests; and therefore, he/she is more likely to

- *C.H. Liu is with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China, and the Department of Computer and Information Security, 209 Neungdong-ro, Gwangjin-gu, Seoul, Sejong University. E-mail: liuchi02@gmail.com.*
- *J. Xu is with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China. E-mail: 752377627@qq.com.*
- *J. Tang is with the Department of Electrical Engineering and Computer Science, Syracuse University, NY 13244. E-mail: jtang02@syr.edu.*
- *J. Crowcroft is with the Computer Laboratory, University of Cambridge, Cambridge CB2 1TN, United Kingdom. E-mail: jon.crowcroft@cl.cam.ac.uk.*

Manuscript received 11 Apr. 2018; revised 24 Sept. 2018; accepted 6 Oct. 2018. Date of publication 9 Oct. 2018; date of current version 4 Oct. 2019. (Corresponding author: Chi Harold Liu).
Recommended for acceptance by J. Gama.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2018.2875006

adopt from his/her friends' personal recommendations than those recommended by the system.

In this paper, we aim to model and predict user interests with full consideration for temporal autocorrelation and social influence, using the emerging Deep Learning techniques. Specifically, we consider the following sequential modeling problem: given a sequence of items/PoIs an online user bought/visited in the past as well as such sequences of his/her friends, predict the types of items/PoIs (e.g., restaurant, coffee shop, grocery store, etc) he/she will purchase/visit next. The solution to this problem can find its application in many scenarios. For example, if an online service system knows a user will likely go to a restaurant next and there happens to be several new restaurants in his/her neighborhood, then it can quickly recommend these to the user, which will likely lead to a hit.

Another contribution of our work lies in leveraging emerging deep learning models and algorithms for modeling and predicting user interests, which have barely been used in this context. Deep learning is a multi-layer representation learning method [12], which leverages Deep Neural Networks (DNNs) to automatically discover a simple but proper representation for raw data. Deep learning has recently made tremendous successes by substantially improving the state-of-art on many application domains, including image/video processing, natural language processing, etc [12]. It is particularly suitable to infer information from large datasets and requires very little domain knowledge and engineering by hand. A Recurrent Neural Network (RNN) [12] is a special type of neural network, where connections between units form a directed cycle. This allows it to model temporal sequence. Multiple RNNs can be stacked together to form a Deep RNN (DRNN), which has been shown to offer better performance than a regular (shallow) RNN [25]. Similarly, DRNNs, especially deep gated RNNs such as Long Short-Term Memory (LSTM) [16] and Gated Recurrent Unit (GRU) [4], have emerged as the state-of-the-art solutions for sequence learning. We leverage the recent results and advances of deep learning to tackle the above sequential modeling problem. Our main contributions are summarized as follows:

- We perform a statistical preliminary analysis for two real datasets from Yelp and Epinions to show temporal autocorrelation in the sequence of PoIs for all users visited and social influence from their friends, which motivates our design.
- We present a novel hybrid deep learning model, Social-Aware LSTM (SA-LSTM), for predicting the types of items/PoIs that a user will likely buy/visit next, which features stacked LSTMs for sequential modeling and an autoencoder-based deep model for social influence modeling. Moreover, we show that the proposed model is end-to-end trainable.
- We conduct extensive experiments for performance evaluation using two real datasets from Yelp and Epinions. The experimental results well justify effectiveness and superiority of the proposed deep model.
- We also conduct a comprehensive empirical study for the structure (the numbers of layers and hidden units) of SA-LSTM, and made an interesting finding: going deep does help improve prediction accuracy but a not-so-deep deep structure leads to the best performance.

To the best of our knowledge, we are one of the first to leverage the emerging deep learning techniques for sequential modeling of user interests with consideration for social influence. The rest of the paper is organized as follows. We discuss related work in Section 2. We then present the preliminary data analysis and the proposed model in Sections 3 and 4, respectively. Experimental results are presented and analyzed in Section 5. We conclude the paper in Section 6.

2 Related Work

In this section, we review the related research activities and point out the differences from them.

Recently, recommender systems/algorithms and user behaviors/interests studies have attracted extensive research attention. Matrix Factorization (MF) [20], which aims to solve the rating prediction task, has become one of the state-of-the-art approaches of Collaborative Filtering (CF). The basic idea of MF is to factorize a user-item rating matrix into two low rank matrices, each of which represents the latent factors of users and items separately. The original matrix can be approximated via multiplication. Similar to MF, AutoRec in [32] is a new CF method, which, however, leverages the non-linear (instead of linear method in MF) autoencoder model. The authors of [29] enhanced MF models using browsing logs and search query logs, which represent a rich profile of individual interests and partly solve the sparsity problem. More related works on this topic include [26], [30]. In [7], Eagle et al. built a model to identify the structure inherent in daily behaviors by finding out the principal components (eigen behaviors) in the data set and representing an individual's behavior over a specific day by a weighted sum of his/her primary eigen behaviors. In [19], the authors proposed new techniques based on clustering and regression for analyzing cellular network data to identify generally important locations, and to discern semantically-meaningful locations such as home and work. More related works along this line can be found in [5], [23], [31], [33]. Unlike these related works which addressed rather static cases/scenarios, we focus on *sequential* modeling of user interests and explore *temporal autocorrelations* hidden in user sequences.

Sequential recommendation has also been studied very recently. Many improved collaborative filtering approaches have been proposed along this direction. For example, the authors in [22] provided a few incremental matrix factorization approaches with five new forgetting techniques which aim to forget the outdated information and keep models up-to-date with the current state of the environment and the current preferences of users. A probabilistic framework was proposed in [28], which uses a probabilistic generative framework to suggest a sequence of tourist spots for travelers, incorporating users' categorical preferences, influence from their social circles, dynamic travel patterns and popularity of venues. In [38], the authors presented Recurrent Recommender Networks (RRN), which also aims at estimating missing ratings. Unlike traditional recommender systems assuming that user profiles and item attributes are static, they also considered temporal dynamics. RRN is a nonlinear recommender system based on LSTMs, which can accurately model user and item dynamics. A sequential personalized recommender system, SPORE, was proposed in [37] for

spatial item recommendation. They introduced a novel latent variable topic-region to model and fuse sequential influence with personal interests in the latent and exponential space. Liu et al. proposed Spatial Temporal Recurrent Neural Networks (ST-RNN) in [21] for predicting the next location, which models local temporal and spatial contexts using an RNN. A dynamic recurrent model for next basket recommendation was proposed in [40], which leverages an RNN to not only learn a dynamic representation for users but also to capture global sequential features among baskets. SPMC (Socially aware Personalized Markov Chain) is another model proposed in [2]. It is the most similar work to ours considering impact from feedback sequence and social circle. The model is based on the assumption that a user can be affected both by their own feedback sequence as well as that of their friends' and proposed to exploit both sequential and social information for recommendation.

We summarize the differences from these related works:

- Most of them [21], [22], [37], [38], [40] have not considered social influence, which, however, is the main focus of our work.
- The sequential modeling problem considered here (Section 4) is mathematically different from those in all these related works.
- We showed the superiority of the proposed SA-LSTM model (compared to LSTM). Methods proposed in [21], [38], [40] applied RNNs for sequential modeling, whose ideas are similar [38] or likely less effective [21], [40] than LSTM (no gates were used for modeling long-term dependencies).
- To the best of our knowledge, we are the first to leverage deep learning for sequential modeling of user interests and perform a comprehensive empirical study for the structure of the proposed deep model.

Social influence and social circles have also attracted research attention recently. Qin et al. in [27] proposed a general algorithm for mining users' real friends in social media and dividing them into different social circles automatically according to the closeness of their relationships. A group recommendation method was proposed in [9], which utilizes both social and content interests of group members. In [3], Cai et al. made use of mobile crowdsourced data obtained from location-based social networks to study influence maximization. A trust-based matrix factorization technique is proposed in [11]. In [10], Guo et al. proposed a privacy-preserving social-assisted mobile content dissemination scheme in delay tolerant networks. A novel incentive scheme was proposed in [39] to stimulate selfish nodes to participate in bundle delivery in mobile social networks. It is known that sparse data may cause the "cold start" problem. From a certain point of view, to use the social information will ease the "cold start" problem by providing a user's friends data as a supplement. Unlike these works, we consider a different problem, *sequential modeling*, here. Specifically, we model social influence with a deep model and integrate it into a sequential modeling framework.

3 Preliminary Data Analysis

In this section, we first describe two datasets used for our analysis and evaluation, and then we perform a preliminary analysis for the data, which motivates our design.

3.1 Datasets and Data Pre-Processing

3.1.1 Yelp

The first dataset consists of data collected from Yelp,[1] which is a crowdsourced business review and social networking website and mobile app. Its user community is primarily active in major metropolitan areas. The website has pages devoted to individual PoIs, such as restaurants or coffee shops. Yelp users can submit a review for products or services of a PoI using a one-to-five star rating system. The dataset was collected from users in 7 cities of USA, including Pittsburgh, Charlotte, Urbana-Champaign, Phoenix, etc. We took a major city, Phoenix, as an example to analyze the corresponding user and PoI data. This dataset includes 366,321 ratings from 122,034 users for 11,853 PoIs during the period from 01/01/2010 to 08/30/2016. Each PoI is marked with more than 700 different labels, such as food, sandwiches, ice cream, message, coffee, pets, etc. The dataset is fairly sparse since some users and PoIs are not very active. To obtain meaningful information, we focused on 785 fairly active users, 7,692 PoIs and the corresponding 63,978 ratings. Furthermore, in order to facilitate effective learning, we pre-processed the dataset by using clustering to divide the PoIs into 30 categories (such as food, arts & entertainment, health & medical, etc). Specifically, each PoI is represented with a 731-dimensional vector, each of which corresponds to a label. If the PoI has a label, the corresponding value in the vector is set to 1; otherwise 0. Then, we used the K-means clustering [15] algorithm to partition 7,692 PoIs into different clusters, each of which corresponds to a category.

3.1.2 Epinions

The second dataset collects data from Epinions,[2] which is a non-profitable web community established to serve as an easy way to share knowledge and opinions between people with similar interests. The website provides reviews on digital cameras, cars, books, movies, music and more, and similarly it also provides a one-to-five star rating system. Besides, the web of trust includes a listing of all the members that this individual has deemed trustworthy. It also includes a list of all the members who trust. This dataset includes 922,267 ratings from 22,166 users for 296,277 PoIs ranging from 07/05/1999 to 05/09/2011. Each PoI has a unique category (such as games, books, music, etc). Similar to Yelp, in order to facilitate effective learning, we pre-processed the dataset by using clustering to divide the PoIs into 27 categories.

According to the datasets described above, each user made a sequence of visits at different times, which can naturally be denoted by a sequence $\mathbf{S}_i = [c_i^1, c_i^2, \cdots, c_i^t, \cdots, c_i^T]$, where t is the timestep (which is not the actual time but an index indicating the ordering of visiting events over time). c_i^t gives the corresponding category of user i visits at timestep t. Moreover, both datasets also include information about the social circle $\mathbf{F}_i = \{f_i^k\}$ (i.e., a list of friends) of each user i, where f_i^k is the k-th friend of user i. We illustrate the scenario we consider in this paper as shown in Fig. 1.

1. https://www.yelp.com/dataset_challenge
2. https://www.cse.msu.edu/ tangjili/trust.html

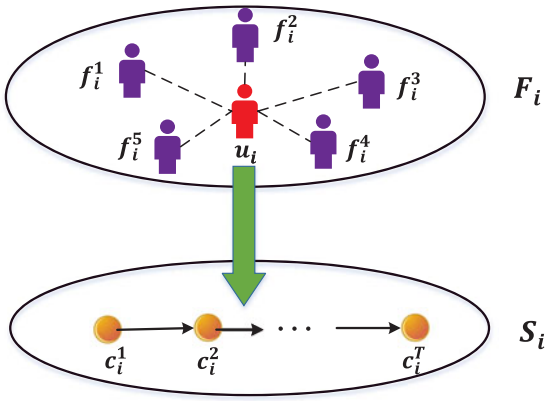


Fig. 1. Sequential modeling of user interests with social circle information.

3.2 Data Analysis

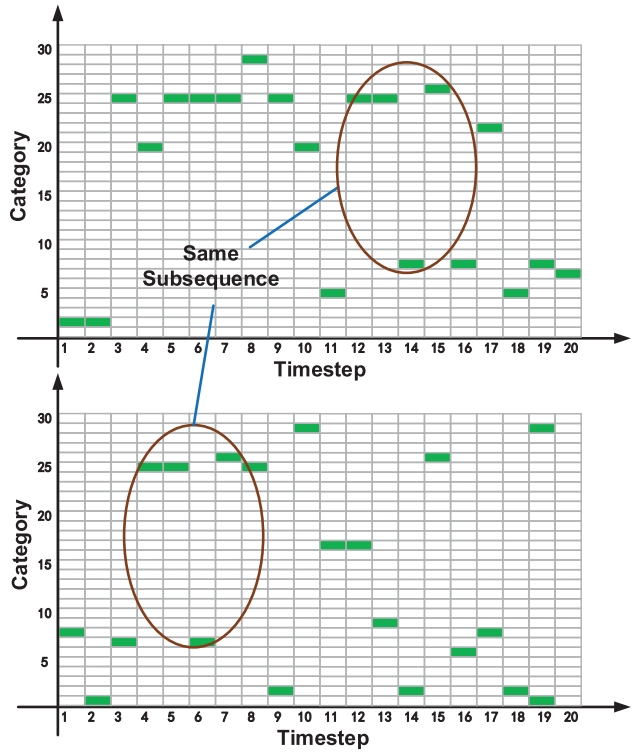
In our preliminary analysis, we try to explore the factors that may influence users' decision-making process. We summarize our main findings as follows.

Observation 1: User Sequences Reveal Strong Social Influence on Decision Making. We aim to show the impact of a user's friend sequential behavior on his/her visiting pattern, where each sequence is associated with multiple activities and each activity is only associated with 1 item. We first randomly selected a user from Yelp dataset, whose sequence is shown in Fig. 2a. We can see that this user visited a PoI belonging to category 25 at timestep 12, followed by PoIs of categories 25, 8, 26. We also show the sequence of one of his friends. It is interesting to see that this friend had the same subsequence (i.e., visited the same types of places) in timesteps 4-7.

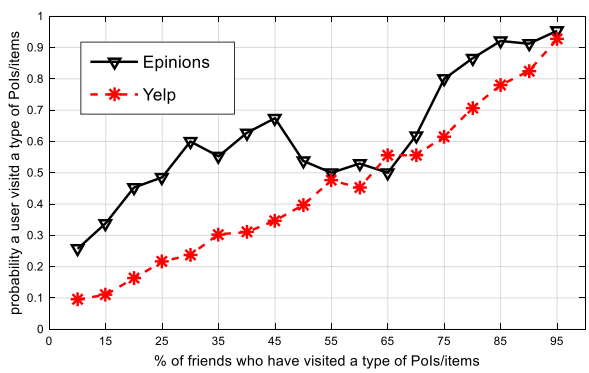
To better show social influence in a statistical way, we calculated the (conditional) probability a user visited a PoI of certain category under the condition that certain percentage of his/her friends have visited it based on two datasets (both Yelp and Epinions). The outcome of this analysis is depicted in Fig. 2b. From the figure, we can see that the curve of Yelp increases almost monotonically, i.e., the more friends visited a PoI, the more likely a user will visit the same or the same type of PoI, and thus the users of Yelp exhibit strong social influence. While the curve of Epinions fluctuates around the value 55 percent of friends who have have visited a type of item, the overall trend is still increasing and the values of most points are higher than that of Yelp. That is, compared with Yelp, users of Epinions reveal stronger social influence on their decision-making process.

Observation 2: User Sequences Exhibit Strong Temporal Autocorrelations. We visualize the visiting sequences of users by using figures with the x axis as the timestep and the y axis as the corresponding category of the PoI this user visited. The visiting sequence of a random user for Yelp dataset is shown in Fig. 3a. From this figure, we can see that this user visited a PoI belonging to category 9 (restaurant) at timestep 3, followed by PoIs of categories 2 (hotel) and 9 (restaurant) respectively. It is interesting to see that this subsequence was repeated at both timesteps 8-10 and timesteps 18-20, which are highlighted by red circles.

This observation appears statistically in both Yelp and Epinions datasets, that it is easy to find some visiting subsequences of users repeated from time to time. For example, a



(a) Visiting sequences of a random user (top) and one of his/her friends (bottom) in Yelp dataset.

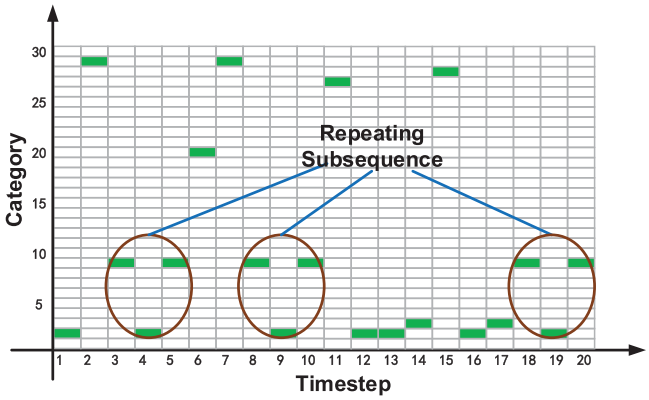


(b) Statistical results of social influence on decision making for both Yelp and Epinions datasets.

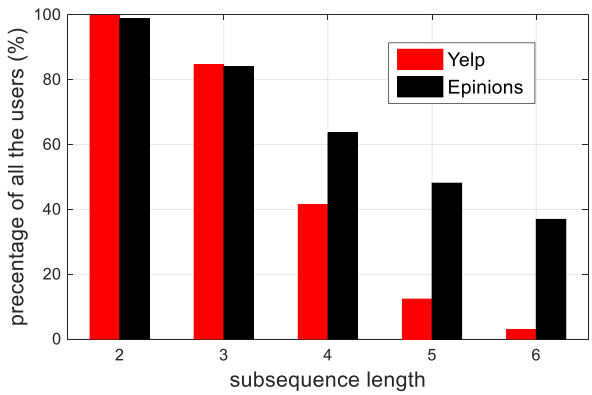
Fig. 2. User sequences reveal strong social influence on decision making.

user might bought a iPhone in an e-commerce site first, most likely the user will buy a mobile phone protecting shell then. The subsequence "phone-phone protecting shell" may form a common pattern. To see the temporal autocorrelations of all users statistically in two datasets (both Yelp and Epinions), we calculated the percentage of all users having different visiting/buying behavior lengths repeated. For example, "phone-phone protecting shell" is a length-2 subsequence and "restaurant-hotel-restaurant" is a length-3 subsequence that may repeat in different user behaviors. We count the proportion of all users who have the same length subsequence repetition from time to time, irrespective of what specific subsequence is, as shown in Fig. 3b.

From this figure, we can see that users with length- 2 or 3 repeated subsequence occupies for a large proportion (above 80 percent, for both Yelp and Epinions), which



(a) Visiting sequence of a random user in Yelp dataset.



(b) Statistical results of the percentage of all users having sequential behaviors of different lengths.

Fig. 3. User sequences exhibit strong temporal autocorrelations.

means that most users repeated their past events. When the subsequence length reaches 4 and higher, the proportion of users having subsequence repetition in Epinions is clearly bigger than that in Yelp, i.e., Epinions shows a stronger temporal autocorrelation than Yelp, which will be well justified by our experiments (see Section 5.2). Therefore, visiting behaviors are not completely random, but follow certain patterns since many users have certain habits or lifestyles, which actually determines the PoIs they visited over time. For example, when a user is traveling, a common pattern is to have lunch at a restaurant, go back to his/her hotel, and then have dinner at a restaurant, as illustrated above.

In short, user sequences exhibit strong temporal autocorrelations.

4 Social-Aware Sequential Model

In this section, we describe the problem we aim to tackle and then we present the proposed sequential model. First, we summarize the major notations in the Table 1 for quick reference.

Here, we consider a sequential modeling problem, in which we are given a set of M users and a set of items/PoIs with C categories. As mentioned above, for each user i, we have a sequence $\mathbf{S}_i$ of items/PoIs he/she bought/visited in the past, the corresponding time and the ratings (usually an integer in $[1, 5]$) he/she gave. Moreover, the social circle $\mathbf{F}_i$ of each user i (i.e., the list of his/her friends) are also known. The problem is to predict the category of item/PoI a user will buy/visit next. Note that even though our preliminary data analysis (see Section 3) was based on Yelp and Epinions involving PoIs, we aim to consider sequential modeling of user interests in general, which also includes the item purchasing case on e-commerce websites/apps (such as Amazon).

TABLE 1
Major Notations Used in This Paper

Notation	Description
i, j and k	The indices of user, category of PoIs and friend
M and C	The numbers of users and categories
t and S_i	Timestep and the sequence of user i
f_i^k and $\mathbf{F}_i$	The k-th friend and the set of friends of user i
$\mathbf{x}_t^i$ and $\mathbf{y}_t$	The input and the final output of stacked LSTMs at timestep t
$\mathbf{Z}_t^i$	The input of SDAEs at timestep t

Motivated by our observations from the real world datasets, we propose a deep learning model, which can well capture the temporal autocorrelation using a gated RNN, stacked LSTMs; as well as influence from social circles using a stacked autoencoder. Next, we describe sequential modeling with LSTMs first and then discuss how to combine it with the social influence model to form an end-to-end trainable and social-aware deep model.

4.1 Sequential Modeling with LSTMs

There are quite a few models and methods for sequential modeling and prediction. We chose a gated RNN, LSTM, as the starting point for our design due to the following reasons:

- Gated RNNs, such as LSTM and GRU, use gates to control how to update hidden states and specify how much past information should be let through, which have been shown to be effective on modeling long-term dependencies (as observed in our preliminary data analysis).
- Similar as other neural networks, multiple LSTMs/GRUs can be stacked together to form a deep model to handle complicated non-linear cases, which are quite common in real applications. We actually observed that going deep can actually help improve performance, which is shown and discussed in Section 5.
- Gated RNNs, especially gated DRNNs, have been shown to deliver the state-of-the-art performance for quite a few complicated sequential modeling problems such as video processing [6], speech recognition [13] and text generation [34].

An RNN is a generalization (in the temporal domain) of the regular feedforward neural network for modeling sequence data [12], which allows for important information to be carried over in the internal memory cell. However, it is well-known that standard RNNs (direct extension in the temporal domain) suffer from vanishing gradient problem [17] and fail to model long-term dependencies [18] effectively. LSTM unit [16] is one of the first gated RNNs, which has been shown to be surprisingly good at modeling long-term dependencies. As mentioned above, it leverages multiple gates (see Fig. 4) to control information flow, which learn (via training data) how to forget previous hidden states and

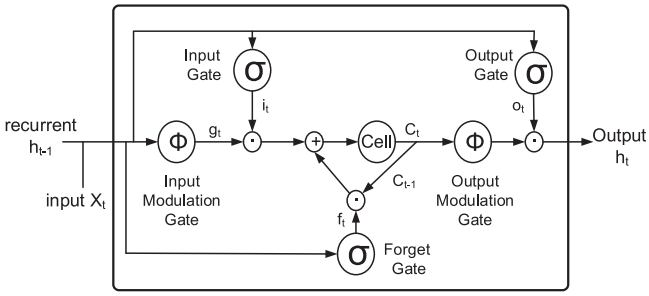


Fig. 4. LSTM unit [41].

how to update the current states. Note that we focus on the LSTM because it is quite representative and general. We believe other gated RNNs (which usually have a similar structure), such as GRU [4], may also be applied here.

For completeness of presentation, we show an LSTM unit [41] in Fig. 4, which is a slight simplification of [14]. A LSTM unit is composed of a single memory cell $\mathbf{c}_t$, three gates (input $\mathbf{i}_t$, output $\mathbf{o}_t$ and forget $\mathbf{f}_t$) and an input and output modulation gate ($\mathbf{g}_t$ and $\mathbf{h}_t$). $\odot$ and $\oplus$ denote the dot product and sum of two vectors respectively. $\sigma(\cdot)$ and $\phi(x) = 2\sigma(2x) - 1$ are the sigmoid and hyperbolic tangent functions respectively. The $\mathbf{W}$ terms denote the weight matrices. For example, $\mathbf{W}_{hi}$ is the hidden-input weight matrix; while the $\mathbf{b}$ terms are the biases. The memory cell $\mathbf{c}_t$ combines the previous cell states, current input and previous output, to update hidden states. The input gate $\mathbf{i}_t$ decides how much impact the current input has on the cell states; the output gate $\mathbf{o}_t$ learns how much the memory cell should affect the hidden states; and the forget gate $\mathbf{f}_t$ determines how much the current information should be forgotten or remembered.

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o) \\
\mathbf{g}_t &= \phi(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \phi(\mathbf{c}_t).
\end{aligned} \tag{1}$$

Multiple LSTMs can be stacked together to form a DRNN [25]. Specifically, all but the first layer input at timestep t is passed from the hidden state of the previous layer $\mathbf{h}_t^{(l-1)}$ (where l denotes the index of layer). Thus, for layer $l > 1$, the activation is:

$$\mathbf{h}_t^l = \sigma^l(\mathbf{h}_t^{l-1}, \mathbf{h}_{t-1}^l). \tag{2}$$

We use LSTMs for sequential modeling. For simplicity, we use $LSTM(\cdot)$ to denote the operations a single LSTM unit or stacked LSTMs perform as mentioned above. Thus we have $\mathbf{h}_t = LSTM(\mathbf{h}_{t-1}, \mathbf{x}_t^i)$, where $\mathbf{x}_t^i$ is the input at timestep t:

$$\mathbf{x}_t^i = [r_1^t, r_2^t ... r_C^t, \tau_t]. \tag{3}$$

Here $r_j, j \in \{1, \cdots, C\}$ is the rating the user i gives for the PoI/item at timestep t (note that $r_j = 0$ if it does not belong to category j) and τ_t gives the weekday of the date the corresponding event occurs. C is the number of categories a user may purchase/visit, which is known *a priori* given by the mobile app/website. The rating vector encodes information about whether the user visits/buys this type of PoI/item and how much he/she likes it, which both may affect his/her decision in the future. τ_t encodes the time information in the format of the weekday (instead of the actual date and time that are not really useful). Note the proposed model is general enough so it is not restricted to such an input format, i.e., the input information can be given in other formats and the other information can also be added into input. However we found that they do not really help improve performance in our experiments.

To obtain the final output at t, we have to add a fully connected layer:

$$\mathbf{y}_t = \text{softmax}(\mathbf{W}_{fc} \cdot \mathbf{h}_t + \mathbf{b}_{fc}), \tag{4}$$

where $\mathbf{y}_t$ is a C-dimensional vector, and $\mathbf{W}_{fc}$ and $\mathbf{b}_{fc}$ are the corresponding weights and biases respectively. The softmax [12] function maps each output value to a probability (i.e., the probability the user will visit/buy an PoI/item of category j). To train this RNN, we use the cross-entropy function as the loss function []:

$$L_{lstm}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{j=1}^{C}(y_j \log(\hat{y_j}) + (1 - y_j)\log(1 - \hat{y_j})), \tag{5}$$

where $\hat{\mathbf{y}}$ gives the target values in the training data.

4.2 Social-Aware Sequential Modeling

The next task is to integrate influence of social circle (i.e., friends) into the above LSTM-based sequential model. A user may have multiple friends, each of who has a visiting/purchasing sequence described above. We need to merge the information hidden in these sequences together to form a representation that represents social influence.

Among all the deep models, we believe autoencoder [12] is the best candidate for this task. Because an autoencoder is a special neural network, which can be used for unsupervised learning of a good and compact representation of high-dimensional complex data. Training an autoencoder does not need any labeled data since its output is the same of its input so it can simply be trained by input data. The most important and useful part is its middle layer, which usually has a smaller or much smaller size. We choose the denoising autoencoder [36] in our implementation it is a stochastic version of the regular autoencoder, which enhances robustness of the representation learning. Autoencoders can also be stacked to form a deep network to effectively deal with highly non-linear cases [1], whose last hidden layer is normally used as the presentation of complex input data.

The proposed model, which we call *Social-Aware LSTM*, consists of two components: Stacked Denoiseing AutoEncoders (SDAEs) for social influence learning which is shown in Fig. 5, and stacked LSTMs for sequential learning, which is illustrated in Fig. 6. SA-LSTM works as follows:

- All the data from recent sequences of K friends of the target user (marked in light blue in Fig. 6) at timestep t are passed to the SDAEs as input to learn a compact social influence representation (marked in dark blue

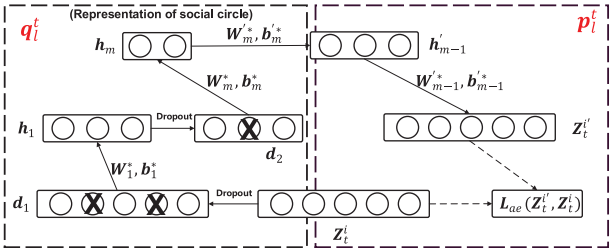


Fig. 5. The SDAEs used to represent social influence.

in Fig. 6). In our experiment, we take a user's K friends' recent activities as the input, since they may potentially have larger impact on his/her decision-making process if compared with older ones.

- The target user's representation (marked in red in Fig. 6) is concatenated ($\oplus$) with the corresponding social influence representation to form the input of LSTMs at timestep t for sequential learning as described in the last section.

Next, we discuss the training of SA-LSTM in detail. We use the data (sequences) of all the users to jointly train the SDAEs and stacked LSTMs. For the sake of illustration, we take data related to user i as an example. For the k-th friend $f_i^k \in \mathbf{F}_i$ of user i, we have a C-dimensional vector (described above), $\mathbf{z}_t^{i,k} = \{r_1, r_2, ..., r_C\}$, to denote if he/she has visited a PoI of category j and the corresponding rate (if yes). The C-dimensional vectors (at timestep t) of all the K friends of user i are concatenated together to form the input of SDAEs at timestep t, which is denoted as $\mathbf{Z}_t^i$. Note that we need to fix the value of K since the input size needs to be fixed. If a user has more than K friends, we select K most active ones; however, if he/she has less than K friends, we do zero-padding to fulfill the input vector. Then we can formally define the output $\mathbf{q}_l^t$ of each layer l of SDAEs' encoder part at timestep t.

$$\mathbf{q}_l^t = \begin{cases} \delta(\mathbf{W}_l \cdot d(\mathbf{Z}_t^i) + \mathbf{b}_l), & l = 1; \\ \delta(\mathbf{W}_l \cdot d(\mathbf{q}_{l-1}^t) + \mathbf{b}_l), & l \in \{2, \cdots, m\} \end{cases}, \tag{6}$$

where $\delta(\cdot)$ is the activation function (we used the sigmoid function in our implementation), $d(\cdot)$ is the dropout function (this is used because it is a denoising autoencoder), and $< \mathbf{W}_l >$ and $< \mathbf{b}_l >$ are the weights and bias of layer l respectively. Then the output $\mathbf{p}_l^t$ of each layer l of SDAEs' decoder part at timestep t can be defined as

$$\mathbf{p}_l^t = \begin{cases} \delta(\mathbf{W}_l^{'} \cdot \mathbf{q}_m^t + \mathbf{b}_l^{'}), & l = m; \\ \delta(\mathbf{W}_l^{'} \cdot \mathbf{p}_{l+1}^t + \mathbf{b}_l^{'}), & l \in \{1, \cdots, m-1\} \end{cases}, \tag{7}$$

where $< \mathbf{W}_l^{'} >$ and $< \mathbf{b}_l^{'} >$ are the weights and bias of layer l respectively. Note that we only take the final output of encoder part $\mathbf{q}_m^t$ as the representation of social circle, and the output of decoder part $\mathbf{p}_1^t$ is supposed to be equal to the input of SDAEs $\mathbf{Z}_t^i$. $L_{ae}(\cdot)$ denotes the loss function of SDAE.

$$L_{ae}(\mathbf{Z}, \mathbf{Z}^{'}) = \sum_{k=1}^{|\mathbf{Z}|} (z_k - z_k^{'})^2. \tag{8}$$

For sake of consistency, we use $\mathbf{Z}$ and $\mathbf{Z}^{'}$ as the input of SDAEs and final output of SDAEs' decoder, respectively, by omitting subscript t and k as the index of vector $\mathbf{Z}$ and $\mathbf{Z}^{'}$, respectively.

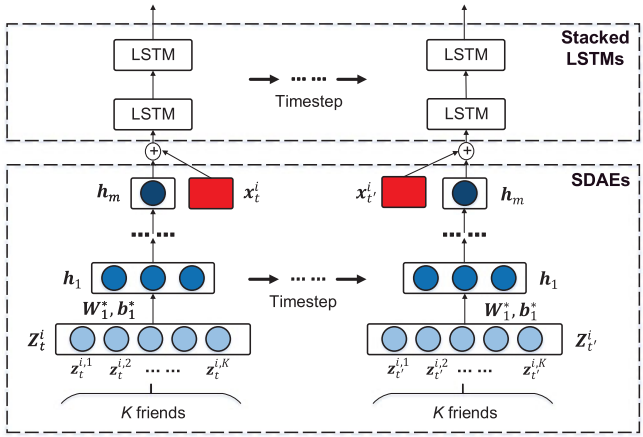


Fig. 6. The proposed SA-LSTM model.

We can train SA-LSTM in an end-to-end manner (rather than training SDAEs and stacked LSTMs separately), which usually yields better performance [14]. We use $\pi(\cdot)$ and $\boldsymbol{\theta}$ to denote stacked SDAEs and their parameters respectively; use $\rho(\cdot)$ and $\boldsymbol{\beta}$ to denote LSTMs and their parameters respectively; and use $L(\cdot)$ to denote the loss function, which is the sum of cross-entropy function shown in (5) and least-square function shown in (8).

$$\mathbf{y} = \text{softmax}(\rho(\pi(\mathbf{Z}) + \mathbf{x})) \tag{9}$$

$$L = -\sum_{j=1}^{C} (y_j \log(\hat{y}_j) + (1 - y_j) \log(1 - \hat{y}_j)) + \sum_{k=1}^{|\mathbf{Z}|} (z_k - z_k^{'})^2 \tag{10}$$

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \gamma \nabla_{\boldsymbol{\theta}} L \tag{11}$$

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \gamma \frac{1}{C} \sum_{j=1}^{C} (\hat{y}_j - y_j) \nabla_{\boldsymbol{\theta}} \rho \nabla_{\boldsymbol{\theta}} \pi + 2 \sum_{k=1}^{|\mathbf{Z}|} (z_k - z_k^{'}) \nabla_{\boldsymbol{\theta}} \pi \tag{12}$$

$$\boldsymbol{\beta} = \boldsymbol{\beta} - \gamma \nabla_{\boldsymbol{\beta}} L \tag{13}$$

$$\boldsymbol{\beta} = \boldsymbol{\beta} - \gamma \frac{1}{C} \sum_{j=1}^{C} (\hat{y}_j - y_j) \nabla_{\boldsymbol{\beta}} \rho, \tag{14}$$

where γ is the learning rate, $< \hat{y}_j >$, $\mathbf{x}$ and $\mathbf{Z}$ give target output values, all the data (C-dimensional or $(C+1)$-dimensional vectors) in a user sequence and the corresponding data in the sequences of his/her friends respectively in the training set. Note that we use slightly different notations here by omitting the subscript t for $\mathbf{x}$ and $\mathbf{Z}$ because we need to use data in all the timesteps of user/friend sequences for training to predict the result of the next step.

We can see that while the hybrid model SA-LSTM is being trained, the parameters of LSTMs and SDAEs can be jointly updated. In our implementation, we applied the commonly-used Stochastic Gradient Descent (SGD) [35] algorithm to train the proposed model. Other methods, such as RMSProp and AdaGrad [35], can also be applied here for training.

5 Performance Evaluation

To evaluate effectiveness of the proposed deep models, we conducted extensive experiments using the two real dataset, Yelp and Epinions.

5.1 Experimental Settings

For performance evaluation, we used the datasets described in Section 3, which include real data from both Yelp and Epinions. The dataset Yelp records user activities from 01/01/2010 to 08/30/2016. We used the data before 01/01/2015 for training and the rest for testing. As mentioned before, we pre-processed the data by dividing all the PoIs into a number of categories. The dataset Epinions consists of user activities from 07/05/1999 to 05/09/2011. We used the data before 05/11/2001 for training and the rest for testing. For both datasets, we predicted which types of PoIs a user would visit next. To avoid the potential concept drift issues [8], i.e., users can hardly keep their purchase interests stable for a long period, we use a "sliding window" approach. That is, a "window" of two months of user purchase sequence slides to generate the training data and avoid keeping remembering user interests long time ago. It is worth noting that, if the sequence is too short, there may not have enough data for training, and too long window size will bring more outdated data that impact on the accuracy of the model. Therefore, in this paper we empirically chose two months as the window size.

5.1.1 Metrics

We used three widely-used performance metrics for evaluation, including Recall@N, F1-Score@N and Mean Average Precision (MAP), where N specifies the degree of precision, i.e., we consider we have an accurate prediction if one of the top-N predicted categories matches the actual one during testing, and N here means the number of categories recommended finally. We report Recall@N and F1-score@N in our experiments.

They are standard metrics for evaluating the quality of the whole ranked lists recommended. All these metrics somehow measure the prediction accuracy, but in different ways. Moreover, for all these metrics, the larger the values are, the better the performance is.

5.1.2 Baselines

We compared our SA-LSTM (see Section 4.2) with several representative baseline methods, which have all been widely used for recommendation/prediction. They are generally classified into two groups, as non-sequential models, and sequential models.

For non-sequential models, we compare with:

- TOP [21]: For each user, the most popular categories of PoIs in the training dataset are selected as the predicted result.
- Matrix Factorization [20]: One of the state-of-the-art Collaborative Filtering methods for estimating user ratings for items, which has been widely used in recommender systems. A natural application of MF to our task is to select the category of the item (i.e., PoI) with the highest estimated rating from the reconstructed matrix as the predicted result.
- AutoRec [32]: It is proposed in a recent work [32], whose basic idea is similar to MF but it leverages an autoencoder (a non-linear model) instead of linear matrix multiplication (used in MF) for estimating user ratings.

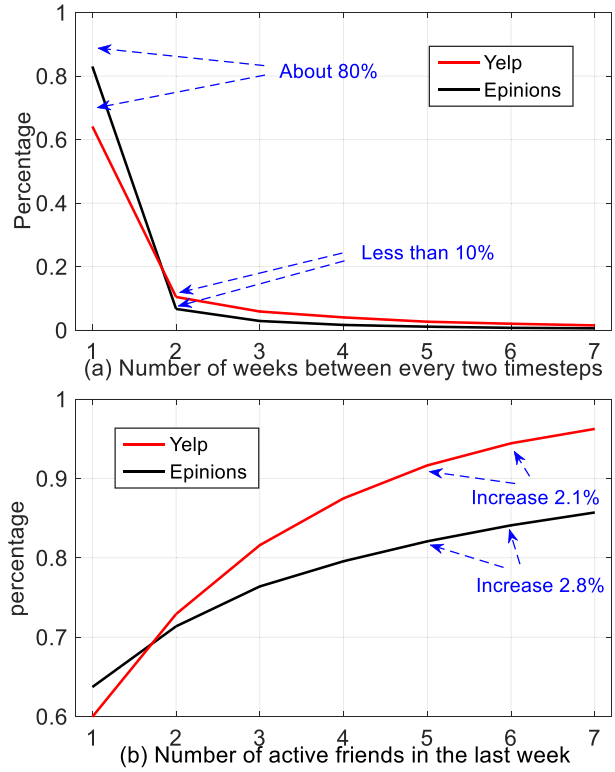


Fig. 7. Statistical analysis on the period of friends' data and number of active friends of a user.

- TrustSVD [11]: It is a trust-based MF model which incorporated both rating and trust information into consideration.

For sequential models, we compare with:

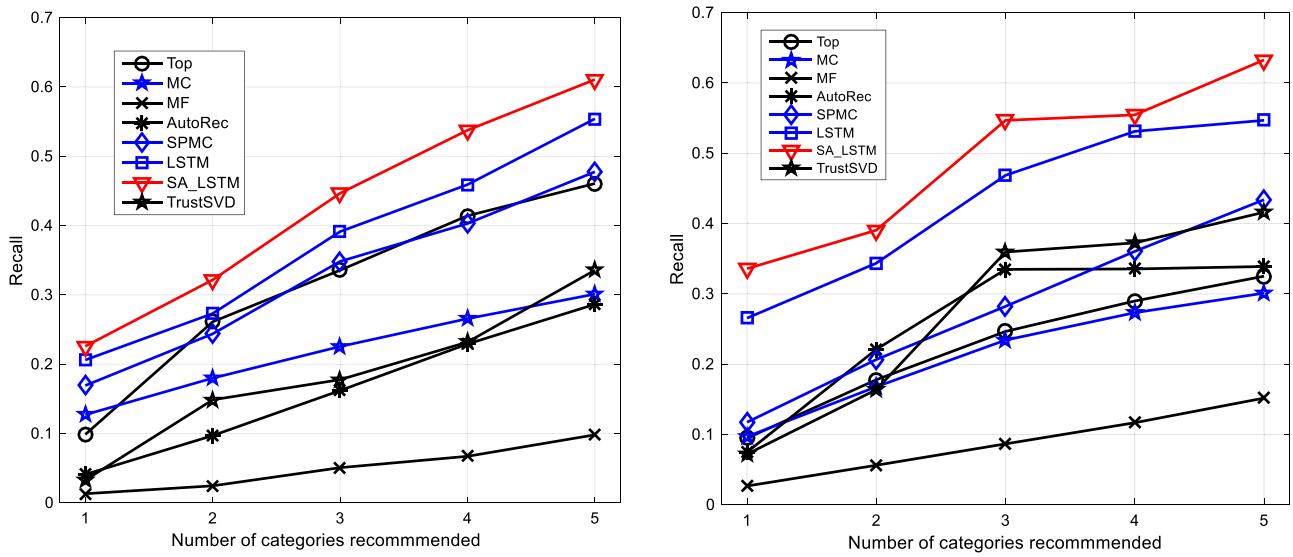
- Markov Chain (MC) [24]: The Markov chain is often used for sequential modeling, which can be used to calculate the transition probability to each category based on the dataset.
- LSTM: It makes model user sequences and make predictions as described in Section 4.1 without explicitly considering the social information.
- SPMC [2]: Socially-Aware Personalized Markov Chains, which leverages feedback from sequences, as well as social interactions in the same model. The model is based on the assumption that a user can be affected both by their own feedback sequence as well as that of their friends'. It is a improved version of Markov Chain.

Note that both SPMC and TrustSVD are state-of-art social based models.

5.1.3 Training and Parameter Selection

We used TensorFlow 1.3 [35] and Python 3.5 in a Ubuntu 16.04.3 server with two NVIDIA TITAN Xp graphics cards to implement the LSTM and SA-LSTM models, train and run the experiments. For SA-LSTM, we trained 9,000 times (but training different networks structures different times, the most complex one 12,000 times, and the most simple one 7,000 times); for LSTM, we trained 6,000 times.

For social influence modeling, it is important to choose the right historical period of friends' data and number of friend K. Fig. 7a shows the interval between every two



(a) Yelp (b) Epinions

Fig. 8. Recall@N given by all the methods.

timesteps in terms of number of weeks (1-7) in a statistical way. We see that most of the data is within one week, e.g., 82 percent for Yelp, and 65 percent for Epinions, and it drastically drops to very low percentage (both below 10 percent for longer period of time). Therefore, we use a friend's historical activities within one week. Fig. 7b shows the statistical data counting the percentage of the number of friends who visited/purchased a category/item (i.e., referred to as "active" friend) in the past week, if a user has made an activity as well in that week. We see that when $K > 5$, the number of users having K active friends increases very slightly, e.g., 2.1 and 2.8 percent for Yelp and Epinions, respectively. Therefore, including more friends will not generate clear benefits, but much higher input dimension. We chose top-5 active friends for each user, i.e., $K = 5$.

In the first experiment, for both datasets, the SDAEs had two hidden layers: the first layer had 75 hidden units and the second layer had 30 hidden units. The stacked LSTMs also had 2 layers, each of which had 64 hidden units. In the second experiment, we investigated how the structure of the proposed model will affect the performance and performed a comprehensive empirical study for different structures (the numbers of layers and hidden units) of SDAEs and stacked LSTMs. Note that we used those settings in the first experiment because we discovered that they led to best performance in the second experiment, which will be discussed in the next section.

5.2 Experimental Results and Analysis

We conduct two sets of experiments on both Yelp and Epinions datasets, i.e., results of recall, F1-Score and MAP (a) given fixed network structures to compare with other state-of-art solutions, and (b) different network structures to show the impact on its own performance.

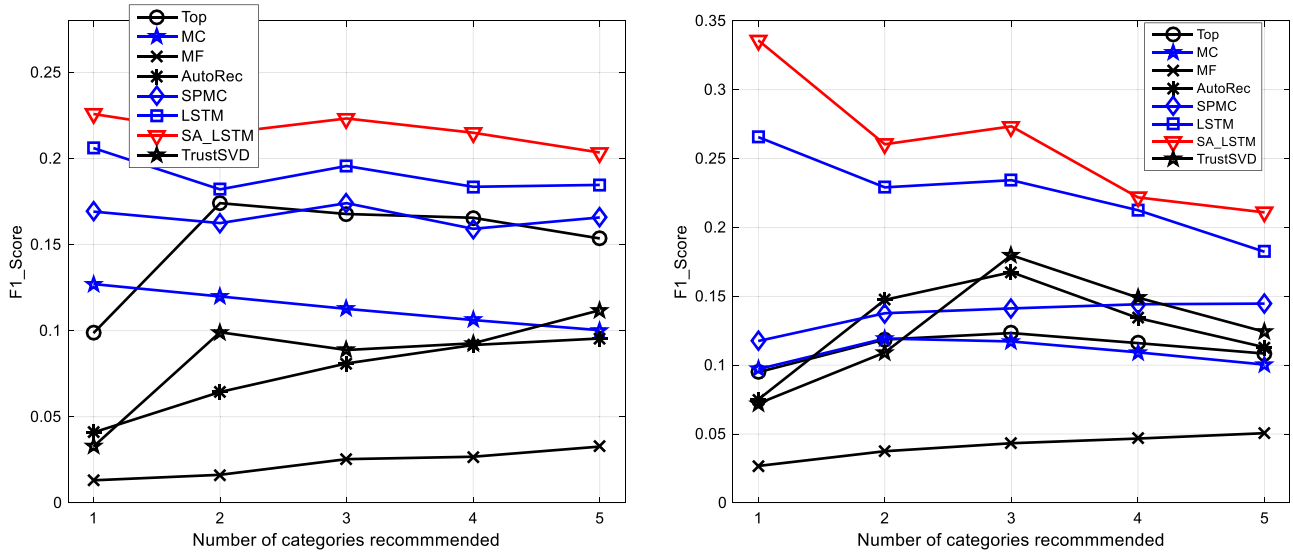
5.2.1 Comparison with State-of-Art Solutions

We report the experimental results on Yelp related to Recall@N and F1-Score@N with $N = \{1, 2, ..., 5\}$, as shown in Figs. 8a and 9a, respectively, and the results in terms of MAP in Fig. 10a. For Epinions, the experimental results related to Recall@N and F1-Score@N with $N = \{1, 2, ..., 5\}$ are shown in Figs. 8b and 9b, respectively, and the results in terms of MAP are depicted in Fig. 10b. We can make the following observations from these figures.

1) From Fig. 8a, we can see first that both the proposed LSTM and SA-LSTM models consistently outperform all the baseline methods in terms of Recall@N on the Yelp datasets. When $N = 1$, all the algorithms give relatively low recall scores because this setting requires the prediction results to match the actual ones exactly. In this case, SA-LSTM gives a recall score of 0.2259, compared to 0.1692 given by the best baseline, SPMC, which represents an improvement of 33.5 percent. As expected, no matter which algorithm is used, the recall score increases with N because the larger the value of N, the more likely we can have a match. When $N = 5$, compared to the six baselines, SA-LSTM significantly improves the recall score by $0.1780, 0.3096, 0.5126, 0.3242, 0.1331, 0.2750$, respectively.

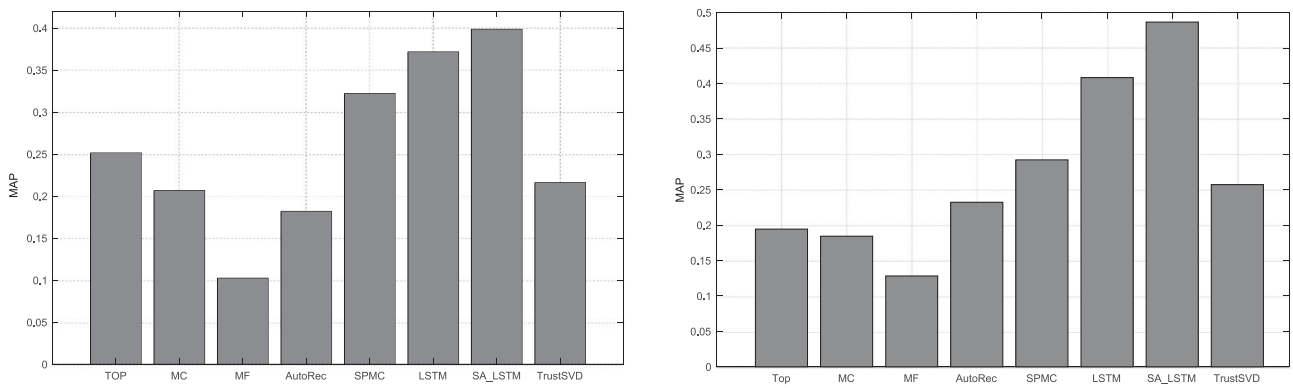
Another interesting observation is that even though both LSTM-based models perform better than the baselines, SA-LSTM offers further improvement over the LSTM-based model that does not take account of social influence. For example, when $N = 5$, SA-LSTM achieves a noticeable improvement of 10.21% over the LSTM-based model. Besides, although our proposed SA-LSTM model and TrustSVD both consider the impact from social circle, SA-LSTM gets an improvement of 0.275 over TrustSVD, when $N = 5$. This is because SA-LSTM considers the impact of autocorrelations within a user sequence, as Yelp and Epinions reveal in Fig. 3b.

Similar observations on Epinions dataset can be made from Fig. 8b. In terms of Recall@1, all methods achieve relatively low recall scores and in this case, SA-LSTM gives a recall score of 0.3359, while the best baseline SPMC gives a recall score of 0.1175, which represents an improvement of 0.2184. Compared with the six baseline methods, SA-LSTM still have the best performance, and improves the Recall@5



(a) Yelp (b) Epinions

Fig. 9. F1-Score@N given by all the methods.



(a) Yelp (b) Epinions

Fig. 10. MAP given by all the methods.

score by $0.3076, 0.3317, 0.4810, 0.2938, 0.1988, 0.2166$, respectively. The noticeable improvement of 15.71% in terms of Recall@5 achieved by SA-LSTM over the LSTM-based model proves the SA-LSTM considering social influence dose help increase the prediction accuracy effectively.

2) From Figs. 9a and 9b, we can see that both LSTM and SA-LSTM consistently beat all the baselines in terms of F1-Score@N.

For example, the F1-Scores@5 given by SA-LSTM and the best baseline are 0.2035 and 0.1658 respectively on Yelp dataset, and on Epinions dataset, the score given by SA-LSTM and the baseline are 0.2109 and 0.1447 respectively. Similarly, we can see that SA-LSTM makes a consistent and noticeable improvement over the LSTM-based model in terms of F1-Score@N on both Yelp and Epinions datasets. The only difference is that no matter which method (except MF) is used, F1-Score decreases with N when $N \geq 3$. This is because F1-Score jointly considers both the precision and recall. When the number of recommended categories increases, the possibility for having false positive value increases, especially under the condition of the true positive value is high enough where recommending wrong categories will make a critical impact on the overall accuracy. However, compared approaches like MF and AutoRec show the pretty low absolute value of true positive, and their trend is increasing with respect to the number of recommended categories (see Figs. 9a and 9b). This is because more accurate recommended categories (i.e., which results in higher true positive value) dominates the possible wrongly recommended ones (i.e., false positive), and thus its trend is slightly increasing.

3) The proposed SA-LSTM model is also superior to all the baselines in terms of MAP on both datasets, which can be seen in Figs. 10a and 10b, respectively. The achieved MAP values of seven baseline methods ToP, MC, MF, AutoRec, SPMC, LSTM, TrustSVD for Yelp are 0.2516, 0.2068, 0.1026, 0.1821, 0.3222, 0.3718, and 0.2161, respectively; and the MAP value of our proposed method SA-LSTM is 0.3987. We observe that SA-LSTM improves the MAP scores by 0.1471, 0.1919, 0.2961, 0.2166, 0.0765, 0.0269, and 0.1826 compared to seven baselines respectively on Yelp dataset. On the other hand, the MAP values of seven baselines for Epinions are 0.1943, 0.1843, 0.1283, 0.2323, 0.2922, 0.4083, and 0.2571, respectively, and the MAP value of our proposed method SA-LSTM is 0.4867. That is, SA-LSTM improves MAP scores by 0.2924, 0.3024, 0.3584, 0.2544, 0.1945, 0.0784, and 0.2296, respectively.

TABLE 2
Impact of Network Structures of SDAE and Stacked LSTMs on Recall, F-1 Score, and MAP

Dataset	Structure	Recall@1	Recall@3	Recall@5	F1-Score@1	F1-Score@3	F1-Score@5	MAP
Yelp	(2, 1*64)	0.2231	0.4393	0.6009	0.2231	0.2197	0.2003	0.3963
	(2, 2*64)	**0.2259**	**0.4466**	**0.6107**	**0.2259**	**0.2233**	**0.2035**	**0.3987**
	(2, 3*64)	0.2249	0.4454	0.6021	0.2249	0.2227	0.2007	0.3943
	(2, 1*128)	0.2132	0.4388	0.5959	0.2132	0.2194	0.1986	0.3904
	(2, 2*128)	0.2161	0.4362	0.5975	0.2161	0.2181	0.1991	0.3901
	(2, 3*128)	0.2195	0.4275	0.5809	0.2195	0.2138	0.1936	0.3863
	(1, 2*64)	0.2213	0.4438	0.5921	0.2213	0.2219	0.1973	0.3935
	(3, 2*64)	0.2234	0.4398	0.5913	0.2234	0.2199	0.1971	0.3976
Epinions	(2, 1*64)	0.2813	0.4922	0.6406	0.2813	0.2461	0.2154	0.4436
	(2, 2*64)	**0.3359**	**0.5469**	**0.6328**	**0.3359**	**0.2734**	**0.2109**	**0.4867**
	(2, 3*64)	0.2734	0.4219	0.5001	0.2734	0.2110	0.1667	0.4004
	(2, 1*128)	0.2422	0.3828	0.5547	0.2422	0.1914	0.1849	0.3918
	(2, 2*128)	0.2812	0.4375	0.5703	0.2812	0.2187	0.1901	0.4191
	(2, 3*128)	0.2891	0.3906	0.5625	0.2891	0.1953	0.1875	0.4148
	(1, 2*64)	0.2969	0.4219	0.5547	0.2969	0.2109	0.1849	0.4309
	(3, 2*64)	0.2188	0.4063	0.5313	0.2188	0.2031	0.1771	0.3718

4) From all the figures, the improvement made by SA-LSTM compared to LSTM model is much more obvious on Epinions than Yelp dataset. Specifically, compared to LSTM model, the SA-LSTM improved by 10.21, 10.17 and 7.23 percent in terms of Recall@5, F1-Score@5 and MAP respectively on Yelp dataset, while the improvement made by SA-LSTM is 15.70, 15.69 and 19.20 percent respectively. This is because Epinions dataset reveals much more social impact than Yelp, which can be observed in Fig. 2b (see Section 3).

5) From all these figures, we can observe that the traditional recommendation methods, such as MF and AutoRec, deliver poor performance. This is mainly because these methods fail to capture the temporal autocorrelations hidden in the user sequences, which are shown and discussed in Section 3.

In summary, these results well justify effectiveness of the proposed LSTM-based models on sequential modeling of user interests, and more importantly, the superiority of our SA-LSTM model, which takes into account social influence for prediction.

5.2.2 Impact of Different Network Structures

In the second experiment, we investigated how the structure of SA-LSTM affects the performance by setting the depth and the number of hidden unites of stacked LSTMs and SDAEs to various values. We present the corresponding results on Yelp and Epinions dataset in Table 2, where each row corresponds to a specific structure, each column is a performance metric, and the numbers of each setting specifies the depth of SDAEs, the depth of LSTMs and the number of hidden units of each LSTM. For example, (2,2*64) means the SDAEs have 2 layers, the stacked LSTMs has 2 layers and each layers has 64 hidden units. Note that the numbers of hidden units in the SDAEs cannot be set arbitrarily since they are related to K (friends number, $K = 5$ in our case).

From this table, on the Yelp dataset, we can see that if we fix the depth of SDAEs to 2 and the number of hidden units of each LSTM to 64, and change the depth of the stacked LSTMs from 1 to 3, the Recall@5 score increases from 0.6009 to 0.6107 then goes back down to 0.6021. Similar trends can be observed for the other metrics. If we fix the structure of stacked LSTMs to $2 * 64$ and we change the depth of SDAEs from 1 to 3, similarly, we can see the Recall@5 score increases from 0.5921 to 0.6107 then goes back down to 0.5913; and we can observe the similar trends in terms of the other metrics. In addition, when we fix both the depths of both SDAEs and the stacked LSTMs to 2 layers, and we change the number of hidden units from 64 to 128, we find that it leads to unexpected performance degradation and longer running time.

Similar observation can be made on Epinions dataset. When fixing the depth of SDAEs to 2 and the number of hidden units of each LSTM to 64 and changing the depth of the stacked LSTMs from 1 to 3, the MAP first increases from 0.4436 to 0.4867 and then goes back down to 0.4004. Most of other metrics provide the same trend. When the structure of stacked LSTMs is fixed to $2 * 64$ and the depth of SDAEs is changed from 1 to 3, similarly, we can observe that MAP increases from 0.4309 to 0.4436 and then goes back to 0.3718. Similar trends in terms of other metrics can be observed. Besides, when both the depths of both SDAEs and the stacked LSTMs are set to 2 layers, and we change the of hidden units from 64 to 128, we find that it also leads to an unexpected performance degradation.

Overall, we observe that the structure corresponding to setting $(2, 2 * 64)$ (highlighted) offers the best performance in terms of all the metrics for both Yelp and Epinions datasets.

In short, we conclude that going deep does help improve prediction accuracy, which well justifies our motivation for using deep models for this prediction task. However, deeper models and more hidden units for Autoencoder and LSTM do not necessarily yield better performance; on the contrary, they may cause overfitting issue and unexpected performance degradation.

6 Conclusion

In this paper, we first performed a preliminary analysis for two big social datasets, Yelp and Epinions. On both datasets, we statistically showed that both temporal autocorrelations hidden in the data and social influence on user decision

making, which form the motivations of our design. Then we presented a novel hybrid deep model, SA-LSTM, for predicting the types of item/PoIs that a user will likely buy/visit next, which leverages stacked LSTMs for sequential modeling and SDAEs for social influence modeling. Moreover, we showed that the proposed model supports end-to-end training. We conducted extensive experiments for performance evaluation using two real datasets from Yelp and Epinions. The experimental results show that 1) the proposed SA-LSTM model significantly improves prediction accuracy compared to widely used baseline methods; 2) SA-LSTM outperforms the LSTM-based model, which justifies effectiveness of the proposed social influence model; 3) going deep does help improve performance but a not-so-deep deep structure leads to the best performance.

Acknowledgments

The research of C.H. Liu and J. Xu was supported in part by National Natural Science Foundation of China (No. 61772072). The research of J. Tang was supported in part by NSF grants 1525920 and 1704662.

References

[1] Y. Bengio, P. Lamblin, D. Popovici and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proc. 19th Int. Conf. Neural Inf. Process. Syst.*, 2006, pp. 153–160.

[2] C. Cai, R. He and J. McAuley, "SPMC: Socially-Aware Personalized Markov chains for sparse sequential recommendation," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, 2017, pp. 1476–1482.

[3] J. Li, Z Cai, M. Yan, and Y. Li, "Using crowdsourced data in location-based social networks to explore influence maximization," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.

[4] K. Cho, et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2014, pp. 1724–1734.

[5] Y. Chon, N. D. Lane, F. Li, H. Cha, and F. Zhao, "Automatically characterizing places with opportunistic crowdsensing using smartphones," in *Proc. ACM Conf. Ubiquitous Comput.*, 2012, pp. 481–490.

[6] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 2625–2634.

[7] N. Eagle and A. Pentland, "Eigenbehaviors: Identifying structure in routine," *Behavioral Ecology Sociobiology*, vol. 63, no. 7, pp. 1057–1066, 2009.

[8] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, 2014, Art. no. 44.

[9] M. Gartrell, X. Xing, Q. Lv, A. Beach, R. Han, S. Mishra and K. Seada, "Enhancing group recommendation by incorporating social relationship interactions," in *Proc. 16th ACM Int. Conf. Supporting Group Work*, 2010, pp. 97–106.

[10] L. Guo, C. Zhang, H. Yue, and Y. Fang, "A privacy-preserving social-assisted mobile content dissemination scheme in DTNs," in *Proc. IEEE INFOCOM*, 2013, pp. 2301–2309.

[11] G. Guo, J. Zhang, and N. Yorke-Smith, "TrustSVD: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings," in *Proc. 29th AAAI Conf. Artif.*, 2015, pp. 123–125.

[12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[13] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2013, pp. 6645–6649.

[14] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proc. 31st Int. Conf. Mach. Learn.*, 2014, pp. 1764–1772.

[15] J. Han, M. Kamber and J. Pei, *Data Mining: Concepts and Techniques, 3rd ed.* San Mateo, CA, USA: Morgan Kaufmann, 2011.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, 1997, pp. 1735–1780.

[17] Hochreiter, Sepp, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *Int. J. Uncertainty Fuzziness Knowl.-Based Syst.*, vol. 6, pp. 107–116, 1998.

[18] S. Hochreiter, Y. Bengio, P. Frasconi and J. Schmidhuber, "Gradient flow in recurrent nets: The difficulty of learning long-term dependencies," in *A Field Guide to Dynamical Recurrent Neural Networks*. New York, NY, USA: Wiley-IEEE Press, 2001.

[19] S. Isaacman, R. Becker, R. Caceres, S. Kobourov, M. Martonosi, J. Rowland, and A. Varshavsky, "Identifying important places in peoples lives from cellular network data," in *Proc. Int. Conf. Pervasive Comput.*, 2011, pp. 133–151.

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

[21] Q. Liu, S. Wu, L. Wang, and T. Tan, "Predicting the next location: A recurrent model with spatial and temporal contexts," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 194–200.

[22] P. Matuszyk, J. Vinagre, M. Spiliopoulou, and J. Gama, "Forgetting methods for incremental matrix factorization in recommender systems," in *Proc. 30th Annu. ACM Symp. Appl. Comput.*, 2015, pp. 47–953.

[23] J. McInerney, S. Stein, A. Rogers and N. R. Jennings, "Breaking the habit: measuring and predicting departures from routine in individual human mobility," *Pervasive Mobile Comput.*, vol. 9, pp. 808–822, 2013.

[24] J. R. Norris, *Markov Chains*. Cambridge, U.K.: Cambridge Univ. Press, 1998.

[25] R. Pascanu, C. Gulcehre, K. Cho and Y. Bengio, "How to construct deep recurrent neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2014.

[26] A. Paterek, "Improving regularized singular value decomposition for collaborative filtering," in *Proc. SIGKDD*, 2017, pp. 5–8.

[27] H. Qin, T. Liu, and Y. Ma, "Mining user's real social circle in microblog," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*, 2012, pp. 348–352.

[28] V. Rakesh, N. Jadhav, A. Kotov, and C.K. Reddy, "Probabilistic social sequential model for tour recommendation," in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, 2017, 2017.

[29] R. Ronen, E. Yom-Tov, and G. Lavee, "Recommendations meet web browsing: Enhancing collaborative filtering using internet browsing logs," *IEEE 32nd Int. Conf. Data Eng.*, 2016, pp. 1230–1238.

[30] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Proc. 20th Int. Conf. Neural Inf. Process. Syst.*, 2007, pp. 1257–1264.

[31] S. Scellato, M. Musolesi, C. Mascolo, V. Latora, and A.T. Campbell, "Nextplace: A spatio-temporal prediction framework for pervasive systems," in *Proc. Pervasive Comput.*, 2011, pp. 152–169.

[32] S. Sedhain, A.K. Menon, S. Sanner and L. Xie, "Autorec: Autoencoders meet collaborative filtering," *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 111–112.

[33] X. Sheng, J. Tang, J. Wang, T. Li, G. Xue, and D. Yang, "LIPS: Lifestyle learning via mobile phone sensing," in *Proc. IEEE Global Commun. Conf.*, 2016, pp. 1–7.

[34] I. Sutskever, J. Martens, and G.E. Hinton, "Generating text with recurrent neural networks," in *Proc. 28th Int. Conf. Int. Conf. Mach. Learn.*, pp. 1017–1024, 2011.

[35] TensorFlow. [Online]. Available: https://www.tensorflow.org/

[36] P. Vincent, H. Larochelle, Y. Bengio and P. A. Manzagol, "Extracting and composing robust features with denoising autoencoders," *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 1096–1103.

[37] W. Wang, H. Yin, S. Sadiq, L. Chen, M. Xie, and X. Zhou, "Spore: A sequential personalized spatial item recommender system," in *Proc. IEEE 32nd Int. Conf. Data Eng.*, 2016, pp. 954–965.

[38] C.Y. Wu, A. Ahmed, A. Beutel, A.J. Smola, and H. Jing, "Recurrent recommender networks," in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, 2017, pp. 495–503.

[39] Q. Xu, Z. Su, and S. Guo, "A game theoretical incentive scheme for relay selection services in mobile social networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 8, pp. 6692–6702, Aug. 2016.

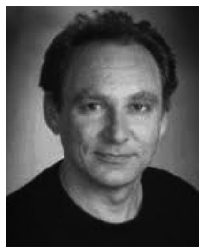
[40] F. Yu, Q. Liu, S. Wu, L. Wang, and T. Tan, "A dynamic recurrent model for next basket recommendation," *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2016, pp. 729–732.

[41] W. Zaremba and I. Sutskever, Learning to execute," *arXiv'14*, pp. 1410–4615.
[42] C. H. Liu, Z. Zhang, M. Chen, "Personalized multimedia recommendations for cloud-integrated cyber-physical systems," *IEEE Syst. J.*, vol. 11, no. 1, pp. 106–117, Mar. 2017.

Chi Harold Liu (SM'15) received the BEng degree in electronic and information engineering from Tsinghua University, China, in 2006 and the PhD degree in electronic engineering from imperial college, United Kingdom, in 2010. He is currently a full professor and vice dean with the School of Computer Science and Technology, Beijing Institute of Technology, China. He also serves as the director of the IBM Mainframe Excellence Center (Beijing) and the IBM Big Data and Analysis Technology Center. Before moving to academia, he worked for IBM Research - China as a staff researcher and project manager from 2010 to 2013, worked as a postdoctoral researcher with Deutsche Telekom Laboratories, Germany, in 2010, and as a research staff member with the IBM T. J. Watson Research Center, USA, in 2009. His current research interests include the big data analytics, mobile computing, and machine learning. He received the IBM First Plateau Invention Achievement Award in 2012, and the IEEE DataCom'16 Best Paper Award. He has published more than 90 prestigious conference and journal papers and owned 14 EU/UK/U.S./China patents. He serves as the area editor for the *KSII Transactions on Internet and Information Systems* and (lead) guest editor for the *IEEE Transactions on Emerging Topics in Computing* and the *IEEE Sensors Journal*. He was the book editor for nine books published by Taylor & Francis Group and China Machine Press, China. He also has served as the general chair of the IEEE SECON'13 workshop on IoT Networking and Control, IEEE WCNC'12 workshop on IoT Enabling Technologies, and the ACM UbiComp'11 Workshop on Networking and Object Memories for IoT. He served as the consultant to the Asian Development Bank, Bain & Company, and KPMG, and the peer reviewer for the Qatar National Research Foundation, and National Science Foundation, China. He is a senior member of the IEEE.

Jie Xu received the BEng and MEng degrees from Beijing Institute of Technology, China, in 2015 and 2018, respectively. He is a software engineer. with Tencent Big Data, China. His research interests include big data and machine learning. He receives the best paper award from IEEE DataCom'16.

Jian Tang received the PhD degree in computer science from Arizona State University in 2006. He is a professor in the Department of Electrical Engineering and Computer Science at Syracuse University. His research interests lie in the areas of wireless networking, machine learning, big data, and cloud computing. Dr. Tang has published over 120 papers in premier journals and conferences. He received an NSF CAREER award in 2009, the 2016 Best Vehicular Electronics Paper Award from the IEEE Vehicular Technology Society, and Best Paper Awards from the 2014 IEEE International Conference on Communications (ICC) and the 2015 IEEE Global Communications Conference (Globecom), respectively. He has served as an editor for a few IEEE journals, including the *IEEE Transactions on Big Data, IEEE Transactions on Mobile Computing, IEEE Transactions on Network Science and Engineering*, *IEEE Transactions on Wireless Communications, IEEE Internet of Things Journal*, and *IEEE Transactions on Vehicular Technology*. In addition, he served as a TPC co-chair for the 2018 International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (Mobiquituous), the 2015 IEEE International Conference on Internet of Things (iThings) and the 2016 International Conference on Computing, Networking and Communications (ICNC); as the TPC vice chair for the 2019 IEEE International Conference on Computer Communications (INFOCOM); and as an area TPC chair for INFOCOM 2017-2018. He is also the vice chair of the Communications Switching and Routing Committee of IEEE Communications Society.



Jon Crowcroft (F'04) received the graduate degree in physics from Trinity College, Cambridge University, United Kingdom, in 1979, and the MSc degree in computing, in 1981 and the PhD degree, in 1993 from University College London (UCL), United Kingdom. He is currently the Marconi professor of cmmunications systems in the Computer Lab with the University of Cambridge, United Kingdom. He was a recipient of the ACM Sigcomm Award in 2009. He is a fellow of the IEEE, ACM, United Kingdom Royal Academy of Engineering, and IET.