I Know You'll Be Back: Interpretable New User Clustering and Churn Prediction on a Mobile Social Application

Carl Yang*†, Xiaolin Shi†, Jie Luo†, Jiawei Han*
*University of Illinois, Urbana Champaign, 201 N Goodwin Ave, Urbana, IL 61801, USA
†Snap Inc., 64 Market St, Venice, CA 90291, USA
*{jiyang3, hanj}@illinois.edu, †{xiaolin.shi, roger.luo}@snap.com

ABSTRACT

As online platforms are striving to get more users, a critical challenge is user churn, which is especially concerning for new users. In this paper, by taking the anonymous large-scale real-world data from Snapchat as an example, we develop *ClusChurn*, a systematic two-step framework for interpretable new user clustering and churn prediction, based on the intuition that proper user clustering can help understand and predict user churn. Therefore, *ClusChurn* firstly groups new users into interpretable typical clusters, based on their activities on the platform and ego-network structures. Then we design a novel deep learning pipeline based on LSTM and attention to accurately predict user churn with very limited initial behavior data, by leveraging the correlations among users' multi-dimensional activities and the underlying user types. *ClusChurn* is also able to predict user types, which enables rapid reactions to different types of user churn. Extensive data analysis and experiments show that *ClusChurn* provides valuable insight into user behaviors, and achieves state-of-the-art churn prediction performance. The whole framework is deployed as a data analysis pipeline, delivering real-time data analysis and prediction results to multiple relevant teams for business intelligence uses. It is also general enough to be readily adopted by any online systems with user behavior data.

KEYWORDS

interpretable model; user clustering; churn prediction

ACM Reference Format:
Carl Yang, Xiaolin Shi, Jie Luo, Jiawei Han. 2018. I Know You'll Be Back: Interpretable New User Clustering and Churn Prediction on a Mobile Social Application. In *KDD 2018: 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3219819.3219821

1 INTRODUCTION

Promoted by the widespread usage of internet and mobile devices, hundreds of online systems are being developed every year, ranging from general platforms like social media and e-commerce websites to vertical services including news, movie and place recommenders. As the market is overgrowing, the competition is severe too, with every platform striving to attract and keep more users.

While many of the world's best researchers and engineers are working on smarter advertisements to expand businesses by acquisition [20, 34], retention has received less attention, especially from the research community. The fact is, however, acquiring new users is often much more costly than retaining existing ones[1]. With the rapid evolution of mobile industry, the business need for better user retention is larger than ever before[2], for which, *accurate*, *scalable* and *interpretable* churn prediction plays a pivotal role[3].

Churn is defined as a user quitting the usage of a service. Existing studies around user churn generally take one of the two ways: data analysis and data-driven models. The former is usually done through user surveys, which can provide valuable insights into users' behaviors and mindsets. But the approaches require significant human efforts and are hard to scale, thus are not suitable for nowadays ubiquitous mobile apps. The development of large-scale data-driven models has largely improved the situation, but no existing work has looked into user behavior patterns to find the reasons behind user churn. As a consequence, the prediction results are less interpretable, and thus cannot fundamentally solve the problem of user churn.

In this work, we take the anonymous data from Snapchat as an example to systematically study the problem of interpretable churn prediction. We notice that online platform users can be highly heterogeneous. For example, they may use (and leave) a social app for different reasons[4]. Through extensive data analysis on users' multi-dimensional temporal behaviors, we find it intuitive to capture this heterogeneity and assign users into different clusters, which can also indicate the various reasons behind their churn. Motivated by such observations, we develop *ClusChurn*, a framework that jointly models the types and churn of new users (Section 2).

To understand user types, we encounter the challenges of automatically discovering interpretable user clusters, addressing noises and outliers, and leveraging correlations among features. As a series of treatments, we apply careful feature engineering and adopt k-means with Silhouette analysis [38] into a three-step clustering mechanism. The results we get include six intuitive user types, each having typical patterns on both daily activities and ego-network structures. In addition, we also find these clustering results highly indicative of user churn and can be directly leveraged to generate type labels for users in an unsupervised manner (Section 3).

KDD 2018, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3219821

[1] https://www.invespcro.com/blog/customer-acquisition-retention
[2] http://info.localytics.com/blog/mobile-apps-whats-a-good-retention-rate
[3] https://wsdm-cup-2018.kkbox.events
[4] http://www.businessofapps.com/data/snapchat-statistics

To enable interpretable churn prediction, we propose to jointly learn user types and user churn. Specifically, we design a novel deep learning framework based on LSTM [21] and attention [13]. Each LSTM is used to model users' temporal activities, and we parallelize multiple LSTMs through attention to focus on particular user types. Extensive experiments show that our joint learning framework delivers supreme performances compared with baselines on churn prediction with limited user activity data, while it also provides valuable insights into the reasons behind user churn, which can be leveraged to fundamentally improve retention (Section 4).

Note that, although we focus on the example of Snapchat data, our *ClusChurn* framework is general and able to be easily applied to any online platform with user behavior data. A prototype implementation of *ClusChurn* based on PyTorch is released on Github[5].

The main contributions of this work are summarized as follows:

(1) Through real-world large-scale data analysis, we draw attention to the problem of interpretable churn prediction and propose to jointly model user types and churn.
(2) We develop a general automatic new user clustering pipeline, which provides valuable insights into different user types.
(3) Enabled by our clustering pipeline, we further develop a prediction pipeline to jointly predict user types and user churn and demonstrate its interpretability and supreme performance through extensive experiments.
(4) We deploy *ClusChurn* as an analytical pipeline to deliver real-time data analysis and prediction to multiple relevant teams within Snap Inc. It is also general enough to be easily adopted by any online systems with user behavior data.

2 LARGE-SCALE DATA ANALYSIS

To motivate our study on user clustering and churn prediction, and gain insight into proper model design choices, we conduct an in-depth data analysis on a large real-world dataset from Snapchat. Sensitive numbers are masked for all data analysis within this paper.

2.1 Dataset

We collect the anonymous behavior data of all new users who registered their accounts during the two weeks from August 1, 2017, to August 14, 2017, in a particular country. We choose this dataset because it is a relatively small and complete network, which facilitates our in-depth study on users' daily activities and interactions with the whole network. There are a total of 0.5M new users registered in the specific period, and we also collect the remaining about 40M existing users with a total of approximately 700M links in this country to form the whole network.

We leverage two types of features associated with users, *i.e.*, their daily activities and ego-network structures. Both types of data are collected during the two-week time window since each user's account registration. Table 1 provides the details of the daily activities data we collect, which are from users' interactions with some of the core functions of Snapchat: *chat, snap, story, lens*. We also collect each user's complete ego-network, which are formed by her and her direct friends. The links in the networks are bi-directional friendships on the social app. For each user, we compute

[5]https://github.com/yangji9181/ClusChurn

ID	Feat. Name	Feat. Description
0	chat_received	# textual messages received by the user
1	chat_sent	# textual messages sent by the user
2	snap_received	# snap messages received by the user
3	snap_sent	# snap messages sent by the user
4	story_viewed	# stories viewed by the user
5	discover_viewed	# discovers viewed by the user
6	lens_posted	# lenses posted to stories by the user
7	lens_sent	# lenses sent to others by the user
8	lens_saved	# lenses saved to devices by the user
9	lens_swiped	# lenses swiped in the app by the user

Table 1: Daily activities we collect for users on Snapchat.

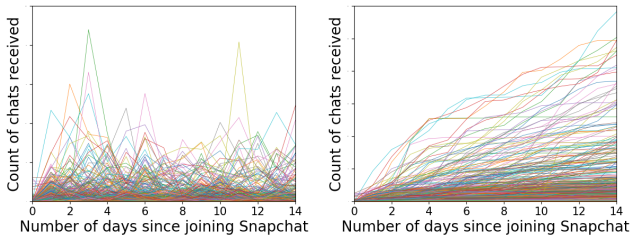
the following two network properties and use them as a description of her ego-network structures.

- Size: the number of nodes, which describes how many friends a user has.
- Density: the number of actual links divided by the number of all possible links in the network. It describes how densely a user's friends are connected.

As a summary, given a set of N users $\mathcal{U}$, for each user $u_i \in \mathcal{U}$, we collect her 10-dimensional daily activities plus 2-dimensional network properties, to form a total of 12-dimensional time series $\mathbf{A}_i$. The length of $\mathbf{A}_i$ is 14 since we collect each new user's behavioral data during the first two weeks after her account registration. Therefore, $\mathbf{A}_i$ is a matrix of 12×14.

2.2 Daily Activity Analysis

Figure 1 (a) shows an example of daily measures on users' *chat_received* activities. Each curve corresponds to the number of chats received by one user every day during the first two weeks after her account registration. The curves are very noisy and bursty, which poses challenges to most time series models like HMM (Hidden Markov Models), as the critical information is hard to be automatically captured. Therefore we compute two parameters, *i.e.*, μ, the mean of daily measures to capture the activity volume, and l, the $lag(1)$ of daily measures to capture the activity burstiness. Both metrics are commonly used in time series analysis [6].



(a) Daily Measures (b) Aggregated Measures

Figure 1: Activities on *chat_received* in the first two weeks. Y-axis is masked in order not to show the absolute values.

Figure 1 (b) shows the aggregated measures on users' *chat_received* activities. Every curve corresponds to the total number of chats received by one user until each day after her account registration. The curves have different steepness and inflection points. Motivated by a previous study on social network user behavior modeling [8], we fit a sigmoid function $y(t) = \frac{1}{1+e^{-q(t-\phi)}}$ to each curve, and use the two parameters q and ϕ to capture the shapes of the curves.

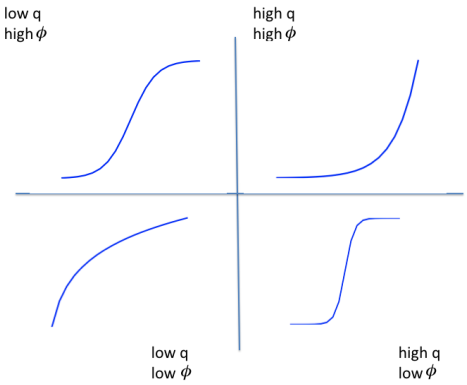


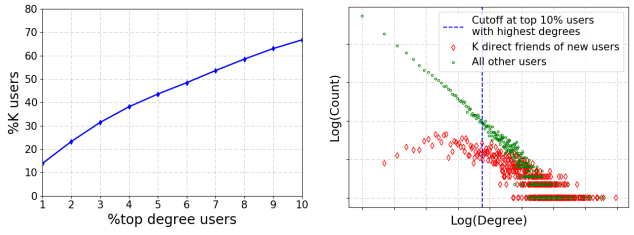
Figure 2: Main curve shapes captured by sigmoid functions with different parameter configurations.

Figure 2 shows 4 example shapes of curves captured by the sigmoid function with different q and ϕ values.After such feature engineering on the time series data, each of the 12 features is described by a vector of 4 parameters $\mathbf{f} = \{\mu, l, q, \phi\}$. We use $\mathbf{F}_i$ to denote the feature matrix of u_i and $\mathbf{F}_i$ is of size 12×4.

2.3 Network Structure Analysis

In addition to daily activities, we also study how new users connect with other users. The 0.5M new users in our dataset directly make friends with a subset of a few million users in the whole network during the first two weeks since their account registration. We mask the absolute number of this group of users and use κ to denote it.

We find these κ users very interesting since there are about 114M links formed among them and 478M links to them. However, there are fewer than 700M links created in the whole network of the total about 40M users in the country. It leads us to believe that there must be a small group of well-connected popular users in the network, which we call the *core* of a network, and in fact, this core overlaps with a lot of the κ direct friends of new users.



(a) Overlapping of core and the κ users (b) Degree distribution of the κ users

Figure 3: Most of the κ users are within the core.

To validate this assumption, we define the core of social networks as the set of users with the most friends, *i.e.*, nodes with highest degrees, motivated by earlier works on social network analysis [40]. Figure 3 (a) shows the percentage of the κ users within the core as we increase the size of the core from the top 1% nodes with highest degrees to the top 10%. Figure 3 (b) shows the particular degrees of the κ users drawn together with all other nodes, ordered by degrees on the x-axis. As we can see, 44% of the κ users are among the top 5% nodes with highest degrees, and 67% of them have 10% highest degrees. This result confirms our hypothesis that most links created by new users at the beginning of their journeys are around the network core. Since the κ direct friends do not entirely overlap with the core, it also motivates us to study how differently new users connect to the core, and what implications such differences can have on user clustering and churn prediction.

3 INTERPRETABLE USER CLUSTERING

In this section, we study what the typical new users are like on Snapchat and how they connect to the social network. We aim to find an interpretable clustering of new users based on their initial behaviors and evolution patterns when they interact with the various functions of a social app and other users. Moreover, we want to study the correlations between user types and user churn, so as to enable better churn prediction and personalized retention.

We also note that, besides churn prediction, interpretable user clustering is crucial for the understanding of user behaviors so as to enable various product designs, which can ultimately lead to different actions towards the essentially different types of users. Therefore, while we focus on the end task of churn prediction, the framework proposed in this work is generally useful for any downstream applications that can potentially benefit from the understanding of user types, such as user engagement promotion.

3.1 Challenges

Automatically finding interpretable clustering of users *w.r.t.* multi-dimensional time series data poses quite a few challenges, which makes the canonical algorithms for clustering or feature selection such as k-means and principal component analysis impractical [19].

Challenge 1: Zero-shot discovery of typical user types. As we discuss in Section 1, users are often heterogeneous. For example, some users might actively share contents, whereas others only passively consume [22]; some users are social hubs that connect to many friends, while others tend to keep their networks neat and small [25]. However, for any arbitrary social app, is there a general and systematic framework, through which we can automatically discover the user types, without any prior knowledge about possible user types or even the proper number of clusters?

Challenge 2: Handling correlated multi-dimensional behavior data. Users interact with a social app in multiple ways, usually by accessing different functions of the app as well as interacting with other users. Some activities are intuitively highly correlated, such as *chat_sent* and *chat_received*, whereas some correlations are less obvious, such as *story_viewed* and *lens_sent*. Moreover, even highly correlated activities cannot be simply regarded as the same. For example, users with more chats sent than received are quite different from users in the opposite. Therefore, what is a good way to identify and leverage the correlations among multiple dimensions of behavior data, including both functional and social activities?

Challenge 3: Addressing noises and outliers. User behavior data are always noisy with random activities. An active user might pause accessing the app for various hidden reasons, and a random event might cause a dormant user to be active for a period of time as well. Moreover, there are always outliers, with extremely high activities or random behavior patterns. A good clustering framework needs to be robust to various kinds of noises and outliers.

Challenge 4: Producing interpretable clustering results. A good clustering result is useless unless the clusters are easily interpretable. In our scenario, we want the clustering framework to provide insight into user types, which can be readily turned into actionable items to facilitate downstream applications such as fast-response and targeted user retention.

3.2 Methods

To deal with those challenges, we design a robust three-step clustering framework. Consider a total of two features, namely, $\mathbf{f}^1$ (*chat_received*) and $\mathbf{f}^2$ (*chat_sent*), for four users, u_1, u_2, u_3 and u_4. Figure 4 illustrates a toy example of our clustering process with the details described in the following.

Step 1: Single-feature clustering. For each feature, we apply k-means with Silhouette analysis [38] to automatically decide the proper number of clusters K and assign data into different clusters. For example, as illustrated in Figure 4, for *chat_received*, we have the feature of four users $\{\mathbf{f}_1^1, \mathbf{f}_2^1, \mathbf{f}_3^1, \mathbf{f}_4^1\}$, each of which is a 4-dimensional vector (*i.e.*, $\mathbf{f} = \{u, l, q, \phi\}$). Assume K chosen by the algorithm is 3. Then we record the cluster belongingness, *e.g.*, $\{l_1^1 = 1, l_2^1 = 1, l_3^1 = 2, l_4^1 = 3\}$, and cluster centers $\{\mathbf{c}_1^1, \mathbf{c}_2^1, \mathbf{c}_3^1\}$. Let's also assume that for *chat_sent*, we have $K = 2$, $(l_1^2 = 1, l_2^2 = 1, l_3^2 = 1, l_4^2 = 2)$ and $\{\mathbf{c}_1^2, \mathbf{c}_2^2\}$. This process helps us to find meaningful types of users *w.r.t.* every single feature, such as users having high volumes of *chat_received* all the time versus users having growing volumes of this same activity day by day.

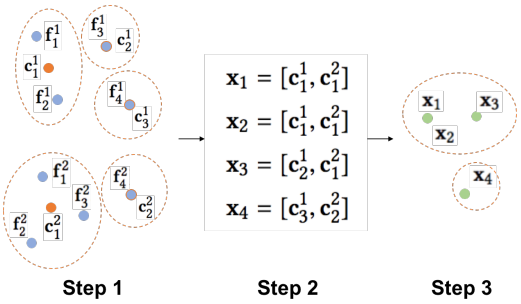


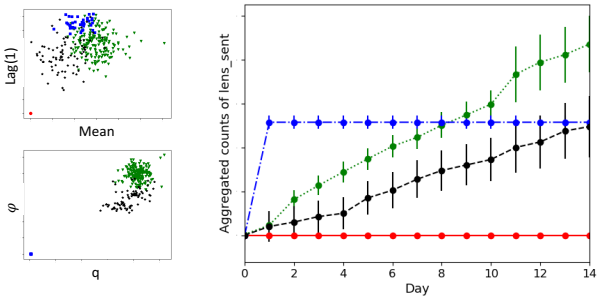
Figure 4: A toy example of our 3-step clustering framework.

Step 2: Feature combination. We convert the features of each user into a combination of the features of her nearest cluster center in each feature. Continue our toy example in Figure 4. Since user u_1 belongs to the first cluster in feature *chat_received* and first cluster in feature *chat_sent*, it is replaced by $\mathbf{x}_1$, which is a concatenation of $\mathbf{c}_1^1$ and $\mathbf{c}_1^2$. u_2, u_3 and u_4 are treated in the same way. This process helps us to largely reduce the influence of noises and outliers because every single feature is replaced by that of a cluster center.

Step 3: Multi-feature clustering. We apply k-means with Silhouette analysis again on the feature combinations. As for the example, the clustering is done on $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$. The algorithm explores all existing combinations of single-dimensional cluster centers, which record the typical values of combined features. Therefore, the multi-feature clustering results are the typical combinations of single-dimensional clusters, which are inherently interpretable.

3.3 Results

Clustering on single features. We first present our single-feature clustering results on each of users' 12-dimensional behaviors. Figure 5 provides the detailed results on *lens_sent* as an example. The results on other features are slightly different regarding the numbers of clusters, shapes of the curves, and numbers of users in each cluster. However, the method used is the same and they are omitted to keep the presentation concise.



(a) Parameter dist. (b) Activity patterns.

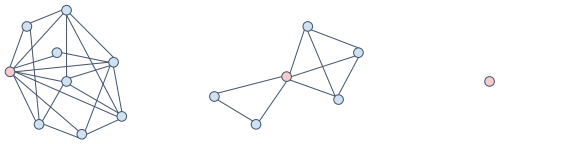
Figure 5: 4 types of users shown with different colors.

Figure 5 (a) shows the four parameters we compute over the 14-day period on users' *lens_sent* activities, as they distribute into the four clusters detected by the k-means algorithm. The number of clusters is automatically selected with the largest average Silhouette score when k is iterated from 2 to 6, which corresponds to clusters that are relatively far away from each other while having similar sizes. Figure 5 (b) shows the corresponding four types of users with different activity patterns on *lens_sent*. The first type of users (red) have no activity at all, while the second type (green) have stable activities during the two weeks. Type 3 users (blue) are only active in the beginning, and type 4 users (black) are occasionally active. These activity patterns are indeed well captured by the volume and burstiness of their daily measures, as well as the shape of the curves of their aggregated measures. Therefore, the clusters are highly interpretable. By looking at the clustered curves, we can easily understand the activity patterns of each type of users.

Clustering on network properties. For single-feature clustering on network properties, as we get four clusters on ego-network size and three clusters on density, there is a total of $4 \times 3 = 12$ possible combinations of different patterns. However, when putting these two features of network properties together with the ten features of daily activities through our multi-feature clustering framework, we find that our new users only form three typical types of ego-networks. This result proves the efficacy of our algorithm since it automatically finds that only three out of the twelve combinations are essentially typical.

Figure 6 illustrates three example structures. The ego-networks of type 1 users have relatively large sizes and high densities; type 2 users have relatively small ego-network sizes and low densities; users of type 3 have minimal values on both measures.

Through further analysis, we find that these three types of new users clustered by our algorithm based on the features of their ego-networks have strong correlations with their positions in the whole social network. Precisely, if we define network core as the top 5% users that have the most friends in the entire network, and depict



(a) Type 1 users (b) Type 2 users (c) Type 3 users

Figure 6: Examples of 3 types of ego-network structures.

the whole network into a jellyfish structure as shown in Figure 7, we can exactly pinpoint each of the three user types into the tendrils, outsiders, and disconnected parts. Specifically, type 1 users are mostly tendrils with about 58% of direct friends in the core; type 2 users are primarily outsiders with about 20% of direct friends in the core; type 3 users are mostly disconnected with almost no friends in the core. Such result again proves that our clustering framework can efficiently find important user types.

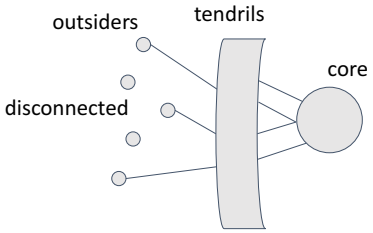


Figure 7: The whole network depicted into a jellyfish shape.

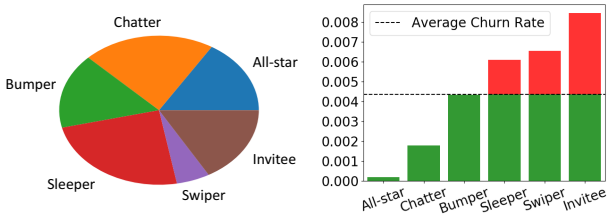
Clustering on all behaviors. Combining new users' network properties with their daily activities, we finally come up with six cohorts of user types, which is also automatically discovered by our algorithm without any prior knowledge. Looking into the user clusters, we find their different combinations of features quite meaningful, regarding both users' daily activities and ego-network structures. Subsequently, we are able to give the user types intuitive names, which are shown in Table 2. Figure 8 (a) shows the portions of the six types of new users.

We define a user churns if there is no activity at all in the second week after account registration. To get more insight from the user clustering results and motivate an efficient churn prediction model, we also analyze the churn rate of each type of users and present the results in Figure 8 (b). The results are also very intuitive. For example, All-star users are very unlikely to churn, while Swipers and Invitees are the most likely to churn.

ID	Type Name	Daily Activities	Ego-network Type
0	All-star	Stable active chat, snap, story & lens	Tendril
1	Chatter	Stable active chat & snap, few other acts	Tendril
2	Bumper	Unstable chat & snap, few other acts	Tendril
3	Sleeper	Inactive	Disconnected
4	Swiper	Active lens swipe, few other acts	Disconnected
5	Invitee	Inactive	Outsider

Table 2: 6 types of new users and their characteristics.

Note that, our new user clustering results are highly intuitive, and in the meantime provide a lot of valuable insights. For example, the main differences between All-star users and Chatters are their activities on *story* and *lens*, which are the additional functions of Snapchat. Being active in using these functions indicates a much lower churn rate. The small group of Swipers is impressive too since they seem to only try out the lenses a lot without utilizing



(a) Portions (b) Churn rates

Figure 8: Portions and churn rates of the six new user types. The y-axis is rescaled to not show the absolute values.

any other functions of the app, which is related to an entirely high churn rate. Quite a lot of new users seem to be invited to the app by their friends, but they are highly likely to quit if not interacting with their friends, exploring the app functions or connecting to core users. Insights like these are highly valuable for user modeling, growth, retention and so on.

Although we focus our study on Snapchat data in this paper, the clustering pipeline we develop is general and can be applied to any online platforms with multi-dimensional user behavior data. The code of this pipeline has also been made publicly available.

4 FAST-RESPONSE CHURN PREDICTION

Motivated by our user type analysis and the correlations between user types and churn, we aim to develop an efficient algorithm for interpretable new user churn prediction. Our analysis of real data shows that new users are most likely to churn in the very beginning of their journey, which urges us to develop an algorithm for fast-response churn prediction. The goal is to accurately predict the likelihood of churn by looking at users' very initial behaviors, while also providing insight into possible reasons behind their churn.

4.1 Challenges

New user churn prediction with high accuracy and limited data is challenging mainly for the following three reasons.

Challenge 1: Modeling sequential behavior data. As we discuss in Section 2.1, we model each new user by their initial interactions with different functions of the social app as well as their friends, and we collect a 12-dimensional time series $\mathbf{A}_i$ for each new user $u_i \in \mathcal{U}$. However, unlike for user clustering where we leverage the full two-week behavior data of each user, for fast-response churn prediction, we only focus on users' very limited behavior data, *i.e.*, from the initial few days. The data are naturally sequential with temporal dependencies and variable lengths. Moreover, the data are very noisy and bursty. These characteristics pose great challenges to traditional time series models like HMM.

Challenge 2: Handling sparse, skewed and correlated activities. The time series activity data generated by each new user are multi-dimensional. As we show in Section 3, such activity data are very sparse. For example, *Chatters* are usually only active in the first four dimensions as described in Table 1, while *Sleepers* and *Invitees* are inactive in most dimensions. Even *All-star* users have a lot of 0's in certain dimensions. Besides the many 0's, the distributions of activity counts are highly skewed instead of uniform and many activities are correlated, like we discuss in Section 3.1.

Challenge 3: Leveraging underlying user types. As shown in our new user clustering analysis and highlighted in Figure 8 (b), our clustering of new users is highly indicative of user churn and should be leveraged for better churn prediction. However, as we only get access to initial several days instead of the whole two-week behaviors, user types are also unknown and should be jointly inferred with user churn. Therefore, how to design the proper model that can simultaneously learn the patterns for predicting user types and user churn poses a unique technical challenge that cannot be solved by existing approaches.

4.2 Methods and Results

We propose a series of solutions to treat the challenges listed above. Together they form our efficient churn prediction framework. We also present comprehensive experimental evaluations for each proposed model component. Our experiments are done on an anonymous internal dataset of Snapchat, which includes 37M users and 697M bi-directional links. The metrics we compute include accuracy, precision, and recall, which are commonly used for churn prediction and multi-class classification [47]. The baselines we compare with are logistic regression and random forest, which are the standard and most widely practiced models for churn prediction and classification. We randomly split the new user data into training and testing sets with the ratio 8:2 for 10 times, and run all compared algorithms on the same splits to take the average performance for evaluation. All experiments are run on a single machine with a 12-core 2.2GHz CPU and no GPU, although the runtimes of our neural network models can be largely improved on GPUs.

Solution 1: Sequence-to-sequence learning with LSTM. The intrinsic problem of user behavior understanding is sequence modeling. The goal is to convert sequences of arbitrary lengths with temporal dependences into a fixed-length vector for further usage. To this end, we propose to leverage the state-of-the-art sequence-to-sequence model, that is, LSTM (Long-Short Term Memory) from the family of RNN (Recurrent Neural Networks) [21, 33]. Specifically, we apply a standard multi-layer LSTM to the multi-dimensional input $\mathbf{A}$. Each layer of the LSTM computes the following functions

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ c_t &= f_t * c_{t-1} + i_t * \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \\ o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t * \tanh(c_t) \end{aligned} \tag{1}$$

where t is the time step in terms of days, h_t is the hidden state at time t, c_t is the cell state at time t, x_t is the hidden state of the previous layer at time t, with $x_t = a_{\cdot t}$ for the first layer, and i_t, f_t, o_t are the input, forget and out gates, respectively. σ is the sigmoid function $\sigma(x) = 1/(1 + e^{-x})$. Dropout is also applied to avoid overfitting. We use Θ_l to denote the set of parameters in all LSTM layers.

A linear projection with a sigmoid function is connected to the output of the last LSTM layer to produce user churn prediction as

$$\hat{y} = \sigma(\mathbf{W}_c o_T + \mathbf{b}_c). \tag{2}$$

We use Θ_c to denote the parameters in this layer, *i.e.*, $\mathbf{W}_c$ and $\mathbf{b}_c$.

Unlike standard methods for churn prediction such as logistic regression or random forest, LSTM is able to model user behavior data as time series and capture the evolvement of user activities through recognizing the intrinsic temporal dependencies. Furthermore, compared with standard time series models like HMM, LSTM is good at capturing both long term and short term dependences within sequences of variable lengths. When the lengths are short, LSTM acts similarly as basic RNN [33], but when more user behaviors become available, LSTM is expected to excel.

Figure 9 (a) shows the performances of compared models. The length of the output sequence of LSTM is empirically set to 64. In the experiments, we vary the amounts of user behavior data the models get access to and find that more days of behavior data generally lead to better prediction accuracy. We can also see that new users' initial activities in the first few days are more significant in improving the overall accuracy. A simple LSTM model can outperform all compared baselines with a substantial margin. The runtime of LSTM on CPU is within ten times of the runtimes of other baselines, and it can be significantly improved on GPUs.

Solution 2: LSTM with activity embedding. To deal with sparse, skewed and correlated activity data, we propose to add an activity embedding layer in front of the standard LSTM layer. Specifically, we connect a fully connected feedforward neural network to the original daily activity vectors, which converts users' sparse activity features of each day into distributional activity embeddings, while deeply exploring the skewness and correlations of multiple features through the linear combinations and non-linear transformations. Specifically, we have

$$e_{\cdot t} = \psi^H(\ldots \psi^2(\psi^1(a_{\cdot t}))\ldots), \tag{3}$$

where

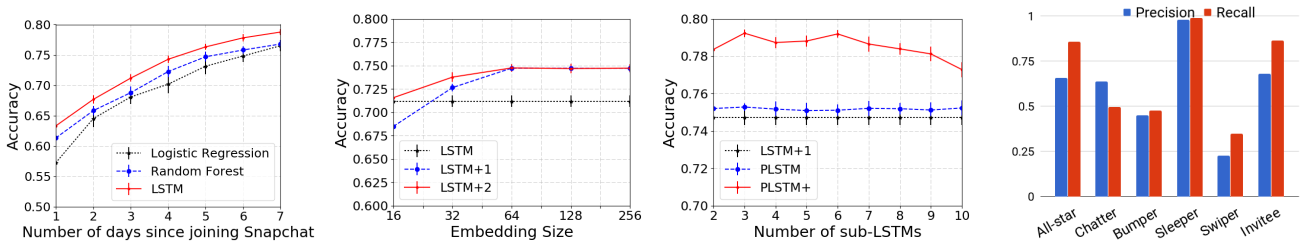
$$\psi^h(e) = \text{ReLU}(W_e^h \text{Dropout}(e) + b_e^h). \tag{4}$$

H is the number of hidden layers in the activity embedding network. Θ_e is the set of parameters in these H layers. With the activity embedding layers, we simply replace $\mathbf{A}$ by $\mathbf{E}$ for the input of the first LSTM layer, with the rest of the architectures unchanged.

Figure 9 (b) shows the performances of LSTM with activity embedding of varying number of embedding layers and embedding sizes. The length of the output sequence of LSTM is kept as 64. The overall performances are significantly improved with one single layer of fully connected non-linear embedding (*LSTM+1*), while more layers (*e.g.*, *LSTM+2*) and larger embedding sizes tend to yield similar performances. The results are intuitive because a single embedding layer is usually sufficient to deal with the sparsity, skewness, and correlations of daily activity data. We do not observe significant model overfitting due to the dropout technique and the large size of our data compared with the number of model parameters.

Solution 3: Parallel LSTMs with joint training. To further improve our churn prediction, we pay attention to the underlying new user types. The idea is that, for users in the training set, we get their two-week behavior data, so besides computing their churn labels y based on their second-week activities, we can also compute their user types $\mathbf{t}$ with our clustering framework. For users in the testing set, we can then compare the initial behaviors with those in the training set to guess their user types, and leverage the correlation between user types and churn for better churn prediction.

To implement this idea, we propose parallel LSTMs with joint training. Specifically, we assume there are K user types. K can be



(a) Single LSTM (b) Activity embedding (c) Parallel LSTMs (d) User type prediction

Figure 9: Comprehensive experimental results on our churn prediction framework compared with various baseline methods.

either chosen automatically by our clustering framework or set to specific values. Then we jointly train K sub-LSTMs on the training set. Each sub-LSTM is good at modeling one type of users. We parallelize the K sub-LSTMs and merge them through attention [13] to jointly infer hidden user types and user churn.

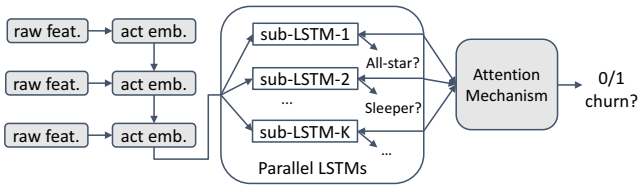


Figure 10: Parallel LSTMs with user type attention.

As shown in Figure 10, for each user, the input of a sequence of activity embedding vectors $\mathbf{E}$ is put into K sub-LSTMs in parallel to generate K *typed sequences*:

$$\mathbf{s}_k = \text{LSTM}_k(\mathbf{E}). \tag{5}$$

To differentiate hidden user types and leverage this knowledge to improve churn prediction, we introduce an attention mechanism to generate user behavior embeddings by focusing on their latent types. A positive attention weight w_k is placed on each user type to indicate the probability of the user to be of a particular type. We compute w_k as a similarity of the corresponding typed sequence $\mathbf{s}_k$ and a global unique *typing vector* $\mathbf{v}_t$, which is jointly learned during the training process.

$$w_k = \text{softmax}(\mathbf{v}_t^T \mathbf{s}_k). \tag{6}$$

Here softmax is taken to normalize the weights and is defined as $\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$. The user behavior embedding $\mathbf{u}$ is then computed as a sum of the typed sequences weighted by their importance weights:

$$\mathbf{u} = \sum_{k=1}^{K} w_k \mathbf{s}_k. \tag{7}$$

The same linear projection with sigmoid function as in Eq. 2 is connected to $\mathbf{u}$ to predict user churn as binary classification.

$$\hat{y} = \sigma(\mathbf{W}_c \mathbf{u} + \mathbf{b}_c). \tag{8}$$

To leverage user types for churn prediction, we jointly train a typing loss l_t and a churn loss l_c. For l_t, we firstly compute users' soft clustering labels $\mathbf{Q}$ as

$$q_{ik} = \frac{(1 + ||\mathbf{f}_i - \mathbf{c}_k||^2)^{-1}}{\sum_j (1 + ||\mathbf{f}_i - \mathbf{c}_j||^2)^{-1}}. \tag{9}$$

q_{ik} is a kernel function that measures the similarity between the feature $\mathbf{f}_i$ of user u_i and the cluster center $\mathbf{c}_k$. It is computed as the probability of assigning u_i to the kth type, under the assumption of Student's t-distribution with degree of freedom set to 1 [30].

We use w_{ik} to denote the attention weight for user u_i on type t_k. Thus, for each user u_i, we compute her typing loss as the cross entropy on q_i. and w_i.. So we have

$$l_t = -\sum_i \sum_k q_{ik} \log(w_{ik}). \tag{10}$$

For l_c, we simply compute the log loss for binary predictions as

$$l_c = \sum -y_i \log \hat{y}_i - (1 - y_i) \log(1 - \hat{y}_i), \tag{11}$$

where y_i is the binary ground-truth churn label and $\hat{y}_i$ is the predicted churn label for user u_i, respectively.

Subsequently, the overall objective function of our parallel LSTM with joint training is

$$l = l_c + \lambda l_t, \tag{12}$$

where λ is a hyper-parameter controlling the trade-off between churn prediction and type prediction. We empirically set it to a small value like 0.1 in our experiments.

Figure 9 (c) shows the performances of parallel LSTMs with and without joint training (*PLSTM+* vs. *PLSTM*). The only difference between the two frameworks is that *PLSTM* is not trained with the correct user types produced by our clustering framework. In the experiments, we vary the number of clusters and sub-LSTMs and find that joint training is always significantly helpful. The performance of parallel LSTMs with joint training peaks with 3 or 6 sub-LSTMs. While the number 3 may accidentally align with some trivial clusters, the number 6 actually aligns with the six interpretable cohorts automatically chosen by our clustering framework, which illustrates the coherence of our two frameworks and further supports the sanity of splitting the new users into six types.

Besides churn prediction, Figure 9 (d) shows that we can also predict what type a new user is by looking at her initial behaviors rather than two-week data, with different precisions and recalls. Our algorithm is good at capturing *All-star*, *Sleeper* and *Invitee*, due to their distinct behavior patterns. *Swiper* and *Bumper* are harder to predict because their activity patterns are less regular. Nonetheless, such fast-response churn predictions with insights into user types can directly enable many actionable production decisions such as fast retention and targeted promotion.

5 RELATED WORK

5.1 User Modeling

The problem of user modeling on the internet has been studied since the early 80's [37], with the intuitive consideration of stereotypes that can group and characterize individual users. However, during that time, with the scarce data, a lot of knowledge has to be manually collected or blindly inferred. Such labor and uncertainty make the models hard to capture stereotypes accurately. While the lack of large datasets and labeled data have hindered deep user modeling for decades [48, 54], the recent rapid growth of online systems ranging from search engine and social media to vertical recommenders have been collecting vast amounts of data and generating tons of new problems, which has enabled the resurgence of machine learning in user modeling.

Nowadays, there has been a trend in fusing personalization into various tasks, including search [18, 41], rating prediction [16, 44], news recommendation [1, 28, 53], place recommendation [27, 45, 49, 56], to name a few. Most of them design machine learning algorithms to capture users' latent interests and mutual influences. However, while boosting the overall performance for particular tasks, such algorithms work more like black boxes, without yielding interpretable insight into the user behaviors.

Instead of building the model based on vague assumptions of user behavior patterns, in this work, we systematically analyze users' activity data and strive to come up with a general framework that can find interpretable user clusters. For any real-world online system as we consider, such interpretation can lead to both better prediction models and more personalized services.

5.2 Churn Prediction

It has been argued for decades that acquiring new users is often more expensive than keeping the old ones [10, 15]. Surprisingly, however, user retention and its core component, churn prediction, have received much less attention from the research community. Only a few papers can be found discussing user churn, by modeling it as an evolutionary process [3] or based on network influence [23]. While significant research efforts have been invested on brilliant ads to acquire new users, when dealing with old users, the most common practice is to simply plug in off-the-shelf logistic regression or random forest model[6][7].

To the best of our knowledge, this is the first effort in the research community to seriously stress the emergence and challenge of churn prediction for online platforms. We are also the first to shift the focus of churn prediction from black box algorithms to deep understanding while maintaining high performance and scalability.

5.3 Network Analysis

Recent algorithms on network analysis are mostly related to the technique of network embedding [7, 12, 17, 24, 35, 36, 46, 50–52]. They are efficient in capturing the high-order similarities of nodes regarding both structural distance and random-walk distance in large-scale networks. However, the latent representations of embedding algorithms are hard to interpret, as they do not explicitly capture the particular essential network components, such as hubs, cliques, and isolated nodes. Moreover, they are usually static and do not capture the evolution or dynamics of the networks.

On the other hand, traditional network analysis mostly focuses on the statistical and evolutionary patterns of networks [14, 26], which provides more insights into the network structures and dynamics. For example, the early work on Bowtie networks [2] offers key insight into the overall structure of the web; more recent works like [11, 29, 32] help people understand the formation and evolution of online communities; the famous Facebook paper analyzes the romantic partnerships via the dispersion of social ties [4] and so on. Such models, while providing exciting analysis of networks, do not help improve general downstream applications.

In this work, we combine traditional graph analysis techniques such as ego-network construction, degree and density analysis, as well as network core modeling, together with advanced neural network models, to coherently achieve high performance and interpretability on user clustering and churn prediction.

5.4 Deep Learning

While much of the current research on deep learning is focused on image processing, we mainly review deep learning models on sequential data, because we model user activities as multi-dimensional time series. Current research on deep learning has agreed that RNN [33] is the best model for sequential data. Its network design with loops allows information to persist and has been widely used in tasks like sentiment classification [43], image captioning [31] and language translation [9], as a great substitute to traditional models like HMM. Among many RNN models, LSTM [21], which specifically deals with long-term dependencies, is often the most popular choice. Variants of LSTM have also been developed, such as GRU (Gated Recurrent Unit) [9], bi-directional LSTM [39], tree LSTM [42], latent LSTM [55] and parallel LSTM [5]. They basically tweak on the design of LSTM neural networks to achieve better task performance, but the results are still less interpretable.

In this work, we leverage the power of LSTM, but also care about its interpretability. Rather than using neural networks as black boxes, we integrate it with an interpretable clustering pipeline and leverage the hidden correlations among user types and user churn with an attention mechanism. Attribute embedding is also added to make the model work with sparse noisy user behavior data.

6 CONCLUSIONS

In this paper, we conduct in-depth analysis of Snapchat's new user behavior data and develop *ClusChurn*, a coherent and robust framework for interpretable new user clustering and churn prediction. Our study provides valuable insights into new user types, based on their daily activities and network structures, and our model can accurately predict user churn jointly with user types by taking limited data of new users' initial behaviors after joining Snapchat on large scales. While this paper focuses on the Snapchat data as a comprehensive example, the techniques developed here can be readily leveraged for other online platforms, where users interact with the platform functions as well as other users.

[6] http://blog.yhat.com/posts/predicting-customer-churn-with-sklearn.html
[7] https://www.dataiku.com/learn/guide/tutorials/churn-prediction.html

We deployed *ClusChurn* in Snap Inc. to deliver real-time data analysis and prediction results to benefit multiple productions including user modeling, growth, retention, and so on. Future works include but are not limited to the study on user ego-network heterogeneity, where we hope to understand how different types of users connect with each other, as well as the modeling of user evolution patterns, where we aim to study how users evolve among different types and how such evolvements influence their activeness and churn rates.

REFERENCES

[1] Fabian Abel, Qi Gao, Geert-Jan Houben, and Ke Tao. 2013. Twitter-Based User Modeling for News Recommendations.. In *IJCAI*, Vol. 13. 2962–2966.

[2] Arvind Arasu, Jasmine Novak, Andrew Tomkins, and John Tomlin. 2002. PageRank computation and the structure of the web: Experiments and algorithms. In *WWW, Poster Track*. 107–117.

[3] Wai-Ho Au, Keith CC Chan, and Xin Yao. 2003. A novel evolutionary data mining algorithm with applications to churn prediction. *IEEE transactions on evolutionary computation* 7, 6 (2003), 532–545.

[4] Lars Backstrom and Jon Kleinberg. 2014. Romantic partnerships and the dispersion of social ties: a network analysis of relationship status on facebook. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*. ACM, 831–841.

[5] Mohamed Bouaziz, Mohamed Morchid, Richard Dufour, Georges Linarès, and Renato De Mori. 2016. Parallel Long Short-Term Memory for multi-stream classification. In *Spoken Language Technology Workshop (SLT)*. IEEE, 218–223.

[6] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time series analysis: forecasting and control*. John Wiley & Sons.

[7] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. GraRep:Learning Graph Representations with Global Structural Information. In *CIKM*. 891–900.

[8] Simla Ceyhan, Xiaolin Shi, and Jure Leskovec. 2011. Dynamics of bidding in a P2P lending service: effects of herding and predicting loan success. In *WWW*. ACM, 547–556.

[9] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*.

[10] John L Daly. 2002. *Pricing for profitability: Activity-based pricing for competitive advantage*. Vol. 11. John Wiley & Sons.

[11] Cristian Danescu-Niculescu-Mizil, Robert West, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. No country for old members: User lifecycle and linguistic change in online communities. In *WWW*. ACM, 307–318.

[12] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NIPS*.

[13] Misha Denil, Loris Bazzani, Hugo Larochelle, and Nando de Freitas. 2012. Learning where to attend with deep architectures for image tracking. *Neural computation* 24, 8 (2012), 2151–2184.

[14] Alessandro Epasto, Silvio Lattanzi, Vahab Mirrokni, Ismail Oner Sebe, Ahmed Taei, and Sunita Verma. 2015. Ego-net community mining applied to friend suggestion. *VLDB* 9, 4 (2015), 324–335.

[15] Terry Gillen. 2005. *Winning new business in construction*. Gower Publishing, Ltd.

[16] Jennifer Golbeck, James Hendler, et al. 2006. Filmtrust: Movie recommendations using trust in web-based social networks. In *CCNC*, Vol. 96. 282–286.

[17] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*. ACM, 855–864.

[18] Ramanathan Guha, Vineet Gupta, Vivek Raghunathan, and Ramakrishnan Srikant. 2015. User modeling for a personal assistant. In *WSDM*. ACM, 275–284.

[19] Jiawei Han, Jian Pei, and Micheline Kamber. 2011. *Data mining: concepts and techniques*. Elsevier.

[20] Neil Harris. 2006. Method for customizing multi-media advertisement for targeting specific demographics. US Patent App. 11/441,529.

[21] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[22] Yuheng Hu, Lydia Manikonda, Subbarao Kambhampati, et al. 2014. What We Instagram: A First Analysis of Instagram Photo Content and User Types.. In *ICWSM*.

[23] Jaya Kawale, Aditya Pal, and Jaideep Srivastava. 2009. Churn prediction in MMORPGs: A social influence based approach. In *CSE*, Vol. 4. IEEE, 423–428.

[24] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[25] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. 2010. What is Twitter, a social network or a news media?. In *WWW*. ACM, 591–600.

[26] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2005. Graphs over time: densification laws, shrinking diameters and possible explanations. In *KDD*. ACM, 177–187.

[27] Huayu Li, Yong Ge, and Hengshu Zhu. 2016. Point-of-Interest Recommendations: Learning Potential Check-ins from Friends. In *KDD*.

[28] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. 2010. Personalized news recommendation based on click behavior. In *ICIUI*. ACM, 31–40.

[29] Caroline Lo, Dan Frankowski, and Jure Leskovec. 2016. Understanding Behaviors that Lead to Purchasing: A Case Study of Pinterest.. In *KDD*. 531–540.

[30] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *JMLR* 9, Nov (2008), 2579–2605.

[31] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. 2015. Deep captioning with multimodal recurrent neural networks (m-rnn). In *ICLR*.

[32] Julian John McAuley and Jure Leskovec. 2013. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *WWW*. ACM, 897–908.

[33] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model.. In *Interspeech*, Vol. 2. 3.

[34] Sandra Moriarty, Nancy D Mitchell, William D Wells, Robert Crawford, Linda Brennan, and Ruth Spence-Stone. 2014. *Advertising: Principles and practice*. Pearson Australia.

[35] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In *ICML*. 2014–2023.

[36] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*. ACM, 701–710.

[37] Elaine Rich. 1979. User modeling via stereotypes. *Cognitive science* 3, 4 (1979), 329–354.

[38] Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics* 20 (1987), 53–65.

[39] Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45, 11 (1997), 2673–2681.

[40] Xiaolin Shi, Matthew Bonner, Lada A Adamic, and Anna C Gilbert. 2008. The very small world of the well-connected. In *Proceedings of the nineteenth ACM conference on Hypertext and hypermedia*. ACM, 61–70.

[41] Mirco Speretta and Susan Gauch. 2005. Personalized search based on user search histories. In *ICWI*. IEEE, 622–628.

[42] Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*.

[43] Duyu Tang, Bing Qin, and Ting Liu. 2015. Document Modeling with Gated Recurrent Neural Network for Sentiment Classification.. In *EMNLP*. 1422–1432.

[44] Duyu Tang, Bing Qin, Ting Liu, and Yuekui Yang. 2015. User Modeling with Neural Network for Review Rating Prediction.. In *IJCAI*. 1340–1346.

[45] Jiliang Tang, Xia Hu, Huiji Gao, and Huan Liu. 2013. Exploiting Local and Global Social Context for Recommendation. In *IJCAI*. 264–269.

[46] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW*. ACM, 1067–1077.

[47] Grigorios Tsoumakas and Ioannis Katakis. 2006. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining* 3, 3 (2006).

[48] Geoffrey I Webb, Michael J Pazzani, and Daniel Billsus. 2001. Machine learning for user modeling. *User modeling and user-adapted interaction* 11, 1 (2001), 19–29.

[49] Carl Yang, Lanxiao Bai, Chao Zhang, Quan Yuan, and Jiawei Han. 2017. Bridging Collaborative Filtering and Semi-Supervised Learning: A Neural Approach for POI Recommendation. In *KDD*. ACM, 1245–1254.

[50] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. 2015. Network Representation Learning with Rich Text Information.. In *IJCAI*. 2111–2117.

[51] Carl Yang, Hanqing Lu, and Kevin Chang Chang. 2017. CONE: Community Oriented Network Embedding. *arXiv preprint arXiv:1709.01554*.

[52] Carl Yang, Chao Zhang, Xuewen Chen, Jieping Ye, and Jiawei Han. 2018. Did You Enjoy the Ride: Understanding Passenger Experience via Heterogeneous Network Embedding. In *ICDE*. IEEE.

[53] Carl Yang, Lin Zhong, Li-Jia Li, and Luo Jie. 2017. Bi-directional Joint Inference for User Links and Attributes on Large Social Graphs. In *WWW*. 564–573.

[54] Hongzhi Yin, Bin Cui, Ling Chen, Zhiting Hu, and Xiaofang Zhou. 2015. Dynamic user modeling in social media systems. *ACM Transactions on Information Systems (TOIS)* 33, 3 (2015), 10.

[55] Manzil Zaheer, Amr Ahmed, and Alexander J Smola. 2017. Latent LSTM Allocation: Joint Clustering and Non-Linear Dynamic Modeling of Sequence Data. In *ICML*. 3967–3976.

[56] Jia-Dong Zhang and Chi-Yin Chow. 2013. iGSLR: personalized geo-social location recommendation: a kernel density estimation approach. In *SIGSPATIAL*. ACM, 334–343.