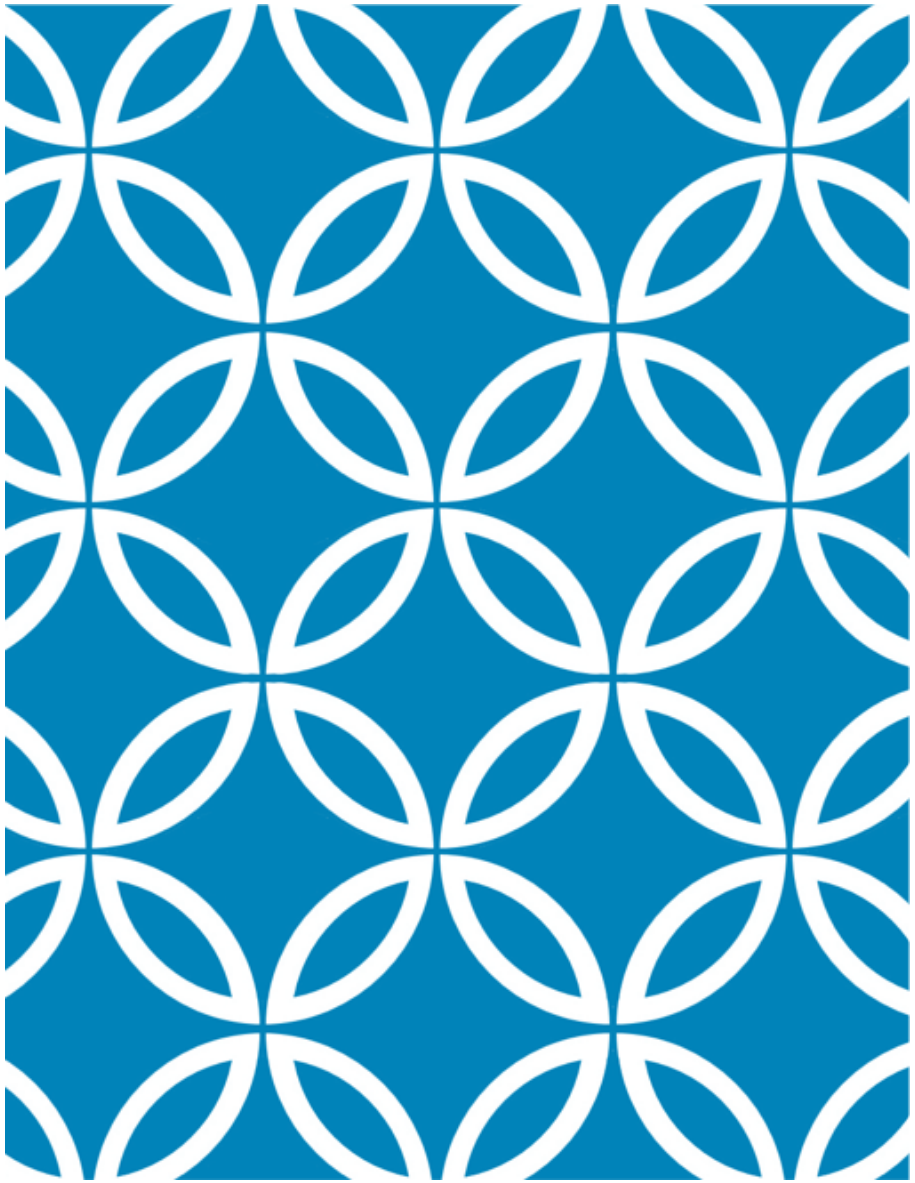


I KNOW YOU'LL BE BACK: INTERPRETABLE NEW USER CLUSTERING AND CHURN PREDICTION ON A MOBILE SOCIAL APPLICATION.

Research Review by William Steimel

SOURCE

I know you'll be back: Interpretable New User Clustering and Churn Prediction on a mobile social application.

Applied Data Science KDD 2018, August 19-23, 2018, London, United Kingdom

Carl Yang, Xiaolin Shi, Jie Luo, Jiawei Han

Collaboration between Snap Inc. and University of Illinois, Urbana Champaign

Keywords: Interpretable Model, User Clustering, Churn Prediction.

TABLE OF CONTENTS

I. INTRODUCTION

Accurate, Stable, and interpretable churn prediction is essential which this study is aimed at.

- Acquiring new users is often more costly than retaining old users.

Churn - User Quits Service

Usually Churn is handled with data analysis and data driven models.

- Data analysis (surveys – hard to scale)
- Data driven models (less interpretable)

This study uses anonymous data from snapchat to create an interpretable, scalable churn prediction system.

I. INTRODUCTION

https://github.com/yangji9181/ClusChurn

Cluschurn – clustering of multidimensional temporal behaviors

Challenges related to discovery of interpretable clusters

Addressing noise/outliers

Leveraging correlation among features

Methods:

Part 1 - Clustering

- Feature Engineering
- K-means Silhouette Analysis
- 3 step clustering mechanism

Result - 6 user types

Part 2 - Churn Prediction

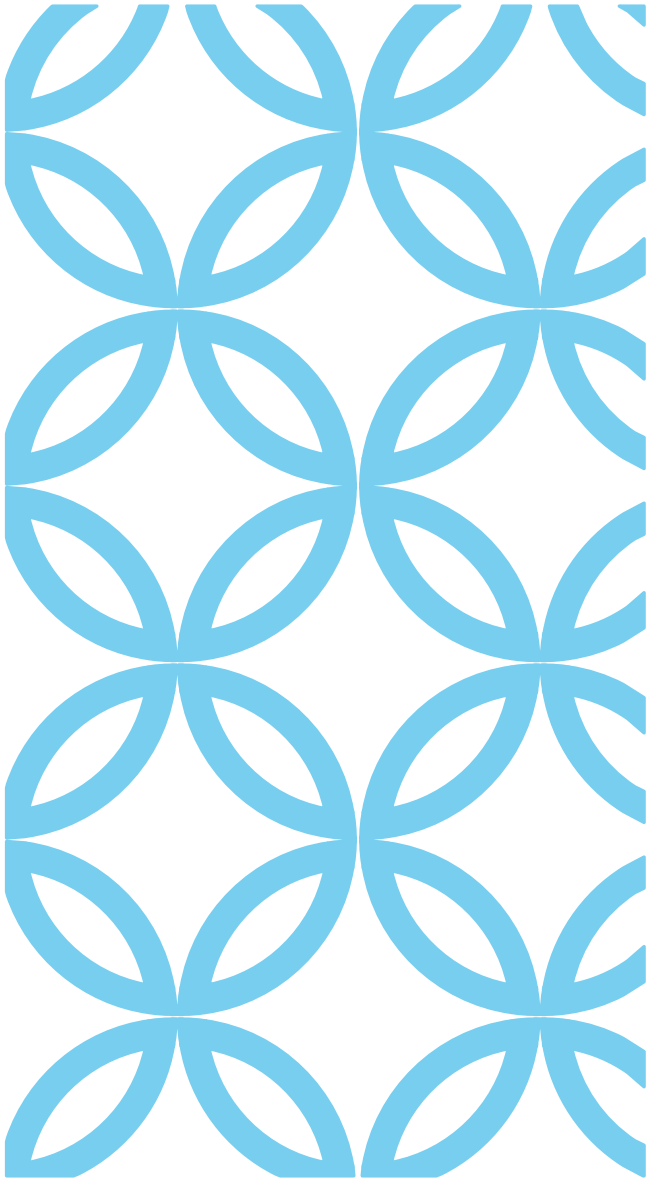
- Parallel LSTM / Attention (Learn Temporal Activities)

Provides valuable insight into why users churn

Cluschurn is applicable to any any data with pytorch implementation available

The project has 4 major contributions

1. Modeling of interpretable user churn through large scale data analysis.
2. New Clustering pipeline provides insight into user types.
3. Prediction pipeline allows for prediction of User types & Interpretable user churn
4. Cluschurn by SNAP INC. can be used on any online systems that create activity data.



II. LARGE SCALE DATA ANALYSIS

2.1 DATASET

Time period of study

- New Users – Aug 1, 2017 - Aug 14, 2017

Data

- Daily Use Activities (Table 1) (10)
- Ego-network structures (Formed by direct friends) (2)
  - Size – Number of notes/total friends
  - Density- Number of links/All Possible Links

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.

$given\ a\ set\ of\ N\ users\ \mathcal{U}, for\ each\ user\ u_i\ \in\ \mathcal{U}$

- 10 dimensional daily activities
- 2 dimensional network properties

This forms a 12 dimensional time-series denoted A_i

The length of A_i is 14 because 2 weeks (14 days) – activity per day

A_i is a matrix 12 x 14

2.2 DAILY ACTIVITY ANALYSIS

Figure 1(a) shows daily measures on users chat received activities

Each curve represents chat received by one user, every day in first two weeks using app.

- This data is however very noisy/bursty – Not easy to see patterns

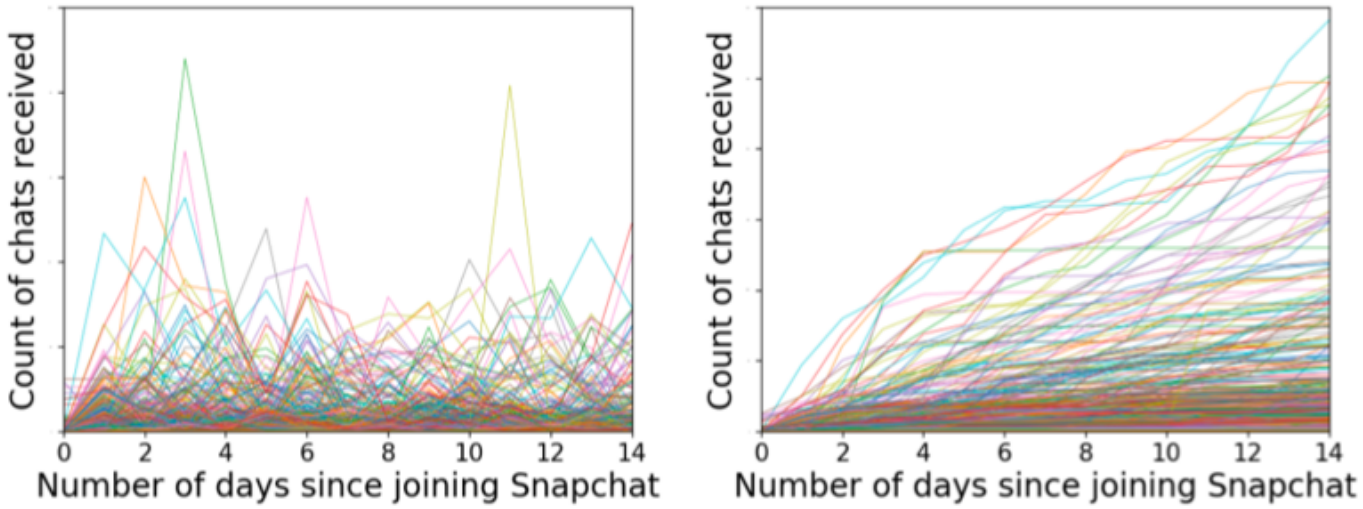
This is not so good for time series models like HMM (Hidden Markov Models)

Two parameters were computed to address this:

- μ – mean of daily measures to capture activity volume
- $l - lag(1)$ of daily measures to capture burstiness

Both are common metrics for time series analysis

Figure 1(b) shows the above aggregated measures on users chat_received activities



(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.

2.2 DAILY ACTIVITY ANALYSIS

Motivated by a previous study on social network user behavior modeling snap fit a sigmoid function to each curve.

$$y(t) = \frac{1}{1+e^{-q(t-\phi)}}$$

Two parameters capture shapes of the curves $(q \; and \; \phi)$

After feature engineering, each of 12 features is described by a vector of 4 parameters.

- f $= \{\mu, l, q, \phi\}$

F_i is used to denote the feature matrix of u_i

- size 12 x 4

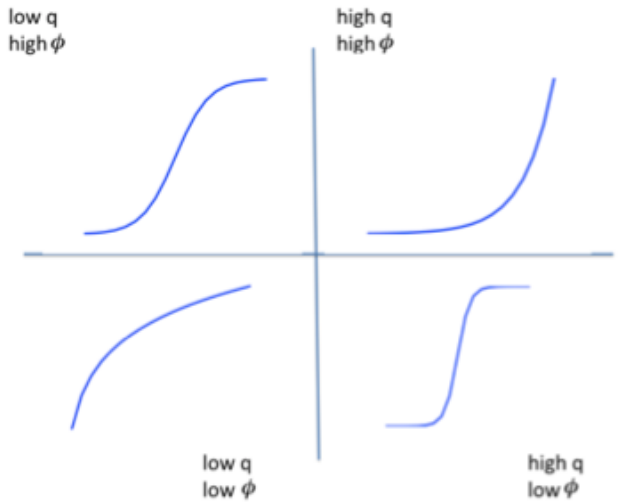


Figure 2: Main curve shapes captured by sigmoid functions with different parameter configurations.

2.3 NETWORK STRUCTURE ANALYSIS

In addition to daily activities, Snap investigates how users connect with other users.

0.5 M new users – make friends with subset of few million users in their first two weeks.

Absolute number of users is masked and represented as K

The K users are very interesting

- 114 M links formed among them / 478 M links to them
- However, there are fewer than 700 m links created in a whole network of 40 M users.

Core of network – Small group of well connected popular users

- Core overlaps with a lot of K direct friends of new users

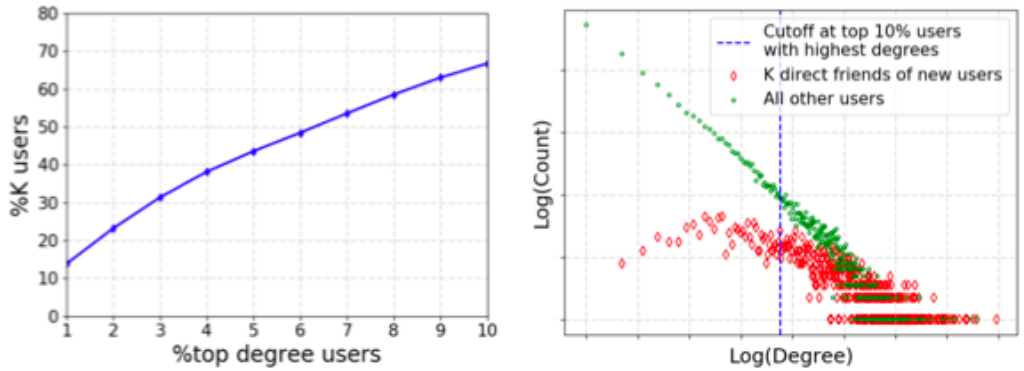
Through previous research of social network analysis (validation) the Core is defined as set of users with most friends (nodes with highest degrees)

Result:

- 44 % of K users are among top 5 % nodes with highest degrees
- 67 % of them have 10 % highest degrees

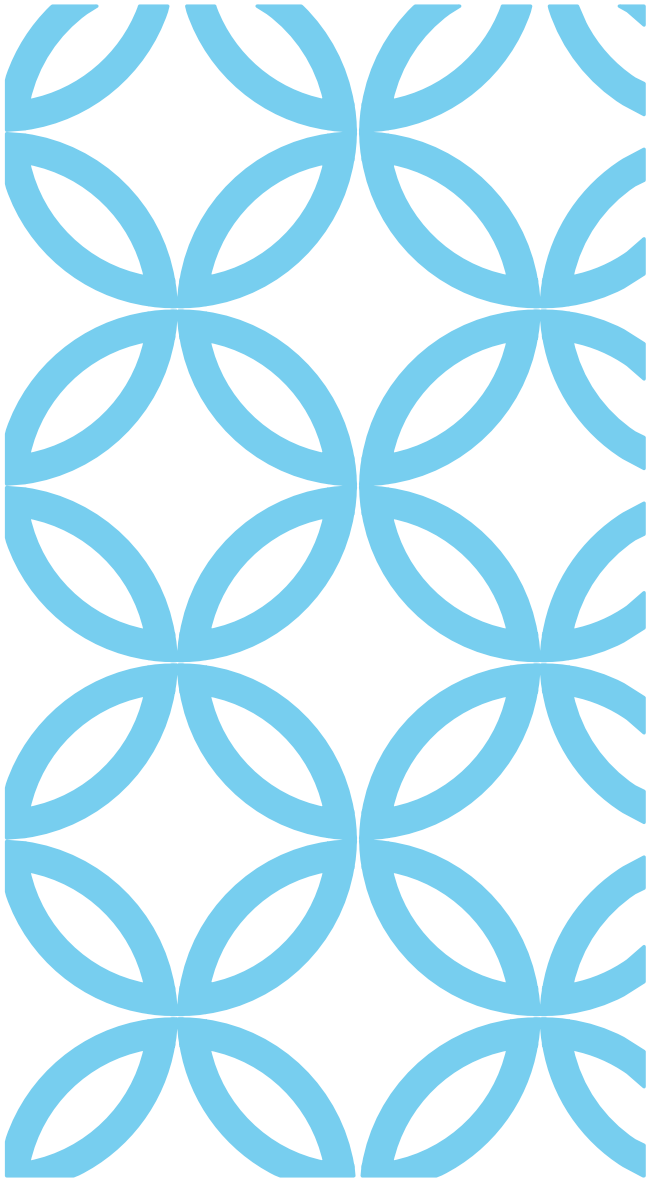
Confirms Assumption – Most links by new users at the beginning are around network core.

- This motivates further research on how new users connect with the core.



(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.



III. INTERPRETABLE USER CLUSTERING

INTRODUCTION

This section studies the typical characteristics of new users on snap chat.

Goal

- Find Interpretable clustering based on initial behaviors/evolution patterns
- Study correlations between user types and churn

These methods are also useful for user engagement promotion as they aim to use interpretable user clustering to influence product based designs and user engagement.

3.1 CHALLENGES

Multidimensional time series data poses some challenges making k-means and principal component analysis impractical.

Challenge 1- Zero-shot discovery of typical user types – users are very different. Some share content while some passively consume. Is there any good system for discovering user types without any knowledge or even total clusters.

Challenge 2- Handling correlated multidimensional behavior data. There are many correlated features like chat sent/received but these activities cannot be regarded as the same. What is a good way to identify and leverage correlations among multiple dimensions of behavior data?

Challenge 3- Addressing noise/outliers – User behavior is usually very noisy with outliers. A good clustering framework needs to be robust to outliers.

Challenge 4- Producing interpretable clustering results – The user clusters need to be interpretable of user types so they can be used in downstream applications like fast-response/targeted user retention.

3.2 METHODS

A robust three-step clustering framework was adopted.

This toy example considers two features and four users.

- chat_received, chat_sent
- u_1, u_2, u_3, u_4

Step 1: Single-Feature Clustering K-Means with Silhouette analysis to automatically decide the proper number of Clusters K

For chat_received feature we have the feature of four users $\{f_1^1, f_2^1, f_3^1, f_4^1\}$ each of which is a 4-dimensional vector $\mathrm{f} = \{\mu, l, q, \phi\}$

- Assume K chosen by algorithm is 3 then record cluster belongingness.
  - $\{l_1^1 = 1, l_2^1 = 1, l_3^1 = 2, l_4^1 = 3\}$ and cluster centers $\{c_1^1, c_2^1, c_3^1\}$
- Assume K -chosen by chat-sent is 2
  - $\{l_1^2 = 1, l_2^2 = 1, l_3^2 = 1, l_4^2 = 2\}$ and cluster centers $\{c_1^2, c_2^2\}$
- This process helps find meaningful types of users with each individual feature.

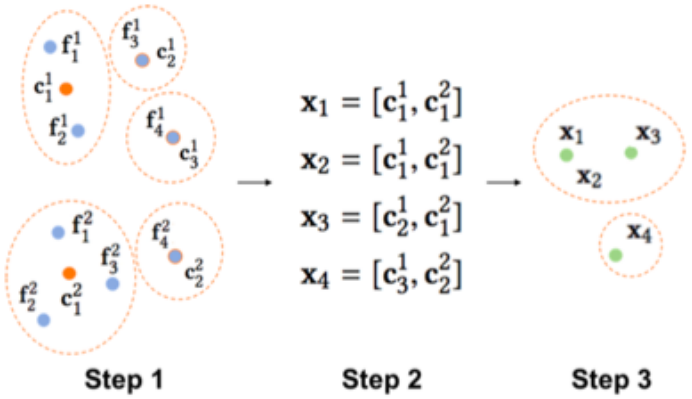


Figure 4: A toy example of our 3-step clustering framework.

3.2 METHODS

Step 2: Feature Combination – Convert the features of each user into a combination of the features of the nearest cluster center.

Since user u_1, belongs to first cluster in chat_sent and first cluster in chat_received, it is replaced by x_1, which is a concatenation of c_1^1 and c_1^2

- This process also happens respectively for u_2, u_3, u_4
- This process helps eliminate noise from outliers as every single feature is replaced with a cluster center.

Step 3: Multi- Feature Clustering – We apply K-means with silhouette analysis again on the feature combinations to get the final cluster.

- (Interpretable)
- In toy example, clustering is performed on x_1, x_2, x_3, and x_4

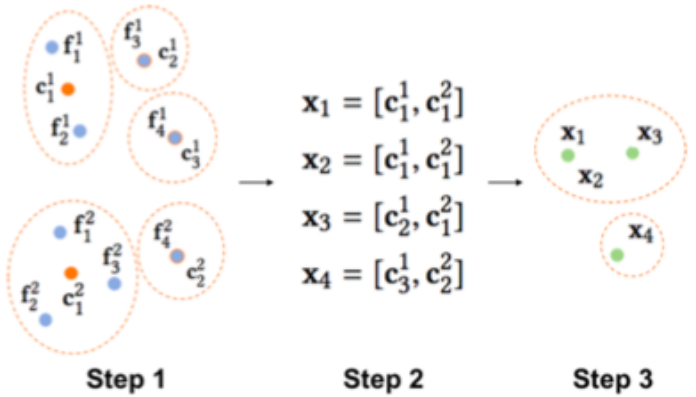


Figure 4: A toy example of our 3-step clustering framework.

3.3 RESULTS – CLUSTERING ON SINGLE FEATURES

This section presents results of clustering on 12 features with the lens_sent feature

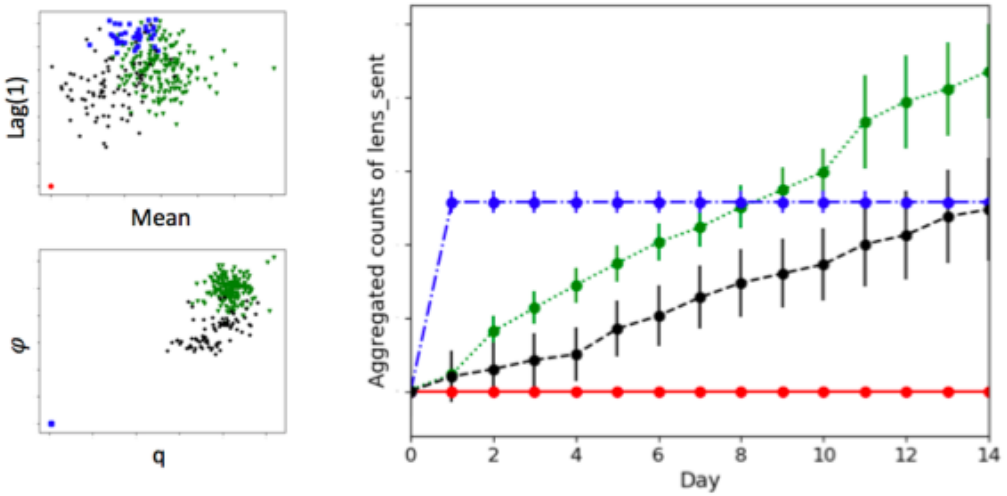
Figure 5(a) shows four parameters computed over 14-day period on users lens sent activities in which 4 clusters were detected by k-means.

- Number of clusters is automatically detected when k is iterated between 2 to 6

Figure 5(b) shows corresponding 4 types of users based on lens_sent.

- (red)- no activity at all
- (green)- stable activities
- (blue) – only active in the beginning
- (black) – occasionally active

These clusters are highly interpretable for understanding activity of users.



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

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

3.3 RESULTS- CLUSTERING ON NETWORK PROPERTIES

Single feature clustering on network properties finds that we get four clusters on ego-network size and 3 on density

- 3x4 = 12 combinations

When applying this multi-feature clustering framework it is found that new users only form three clusters.

- Type 1: Large sizes/high density
- Type 2: Small ego-network and low density
- Type 3: Minimal values on both measure.

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

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

3.3 RESULTS- CLUSTERING ON NETWORK PROPERTIES (CONT.)

If we define the network core as top 5 % users that have the most friends we can pinpoint each of three user types into tendrils, outsiders, and disconnected party.

- Type 1 users: mostly tendrils with about 58 % of friends in the core.
- Type 2: mainly outsiders 20 % direct friends in core.
- Type 3: mostly disconnected almost no friends in core.

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

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

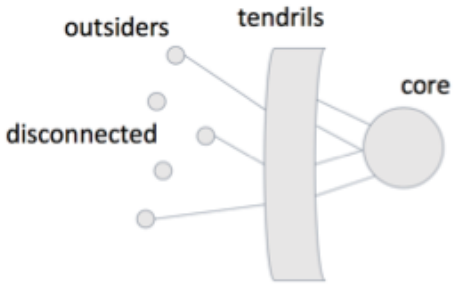


Figure 7: The whole network depicted into a jellyfish shape.

3.3 RESULTS-CLUSTERING ON ALL BEHAVIORS

Combining user network properties with daily activities results in 6 user types which were automatically discovered by the algorithm without prior knowledge.

Different combinations of features are useful for characterizing these types and appropriate names are given to them. (Table 2)

Churn is defined as no activity at all in second week after registration.

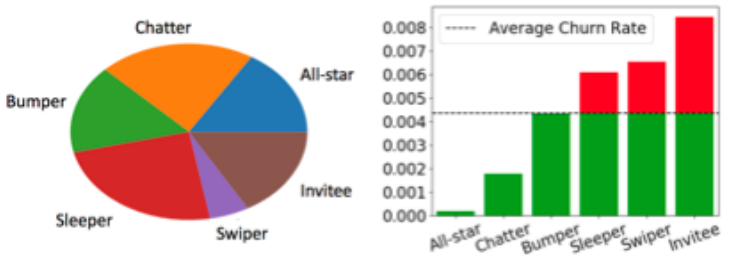
- This is measured by user type in Figure 8

New user cluster modeling is highly intuitive and provides good insight for user modeling, growth, and retention.

- Quite many people are invited by their friends but they are highly likely to quit if not interacting with friends.

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.



(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.

IV. FAST RESPONSE CHURN PREDICTION

4.1 CHALLENGES

There are many challenges associated with predicting user churn with high accuracy and limited data.

Challenge 1: Modeling sequential behavior data

- In comparison to the previous user clustering problem where two weeks of data was used, for fast-response churn prediction only the first few days of data is focused on. (limited data) The data are naturally sequential with temporal dependencies and variable lengths. The data is also very noisy and bursty. These factors are extremely challenging for traditional time series modeling.

Challenge 2: Handling Sparse, skewed and correlated activity

- The time series data generated by each user is multi-dimensional and very sparse. The distributions of activity counts are highly skewed instead of uniform and many activities are correlated.

Challenge 3: Leveraging underlying user types

- As shown by the previous clustering model, clustering is highly indicative of which users churn and should be leveraged for better churn prediction. However, since access is limited to the first few days of data it is challenging to design a model that can simultaneously learn user type and churn.

4.2 METHODS AND RESULTS

Data - Anonymous internal dataset of Snapchat

- 37 M users
- 697 M bi-directional links

Metrics - Accuracy, Recall, and Precision are all computed

Baseline Comparison – Logistic Regression and Random Forest

Training/Test Split – ratio 8:2 (10 x) and average performance for evaluation

Machine Specs- 12-core 2.2GHz CPU and no GPU

Proposed Solutions to previous challenges:

Solution 1: Sequence-to-sequence learning with LSTM

Solution 2: LSTM with activity embedding

Solution 3: parallel LSTMs with joint training

SOLUTION 1: SEQUENCE-TO-SEQUENCE LEARNING WITH LSTM

Sequence Modeling is the hardest challenge with understanding user behavior.

It is important to convert sequences of arbitrary lengths with temporal dependencies into a fixed-length vector for further use.

To solve this problem, a multi-layer LSTM is applied to multi-dimensional input A

Each layer of the LSTM computes the follow functions:

1. $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
2. $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$
3. $c_t = f_t * c_{t-1} + i_t * \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$
4. $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$
5. $h_t = o_t * \tanh(c_t)$

Dropout is also applied to avoid overfitting.

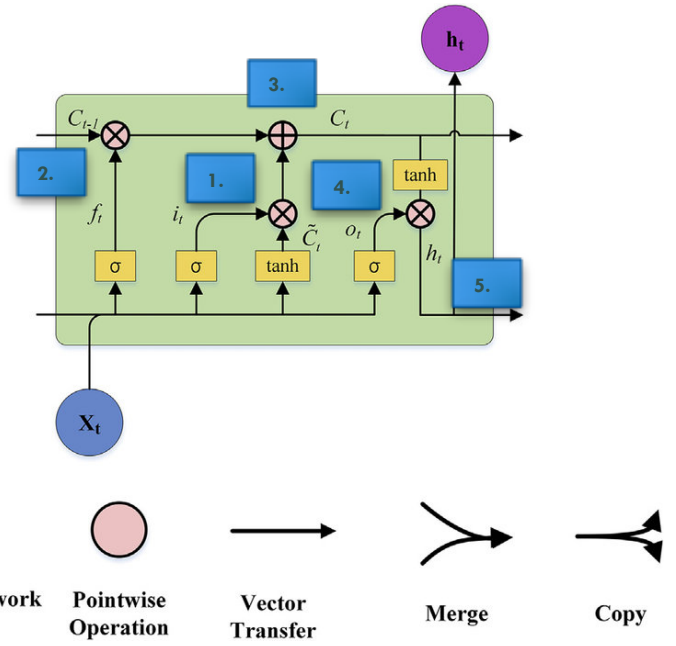
Θ_l - Is used to denote the set of parameters in all LSTM layers.

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

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

Θ_c - Is used to denote parameters in the last layer. i.e (W_c and b_c)

- t – time step in terms of days (14 days)
- h_t – hidden state at time
- c_t - cell state at time t
- x_t -hidden state of previous layer at time t
  $x_t = a_{.t}$ for first layer
- i_t – input gate
- f_t - forget gate
- o_t - out gate
- σ is the Sigmoid Function $\sigma(x) = \frac{1}{(1+e^{-x})}$



SOLUTION 1: SEQUENCE-TO-SEQUENCE LEARNING WITH LSTM

Model Comparison

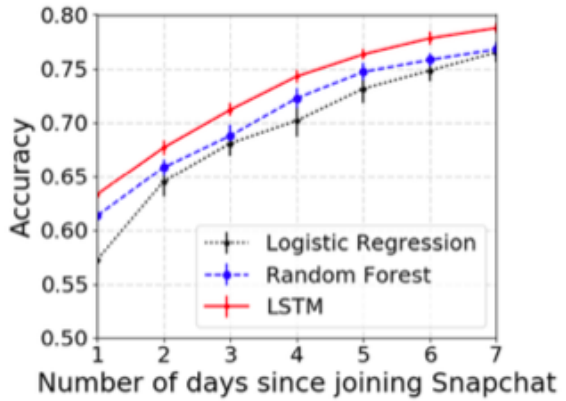
Compared to logistic regression/random forest

- LSTM Is able to model user behavior data as time series and capture involvement of user activities through recognizing temporal dependencies.

Compared to standard time series models like HMM

- LSTM is good at capturing both long term and short term dependencies with variable length sequences.

A simple LSTM model outperforms all baselines on CPU within ten times the run times of the other models.



(a) Single LSTM

SOLUTION 2: LSTM WITH ACTIVITY EMBEDDING

To deal with the challenge of sparse, skewed, and correlated activity data.

- An activity embedding layer in front of the standard LSTM layer is proposed.

A fully connected feedforward neural network is connected to the original daily activity vectors.

- Converts users' sparse feature activity of each day into distributional activity embeddings

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

where

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

- H – number of hidden layers in activity embedding network
- Θ_e – set of parameters in H layers
- With activity Layers, we simply replace A by E for the input of the first LSTM layer.
  - Architecture remains unchanged

SOLUTION 2: LSTM WITH ACTIVITY EMBEDDING

Figure 9. shows the performance of LSTM with activity embedding with a varying number of layers and embedding sizes.

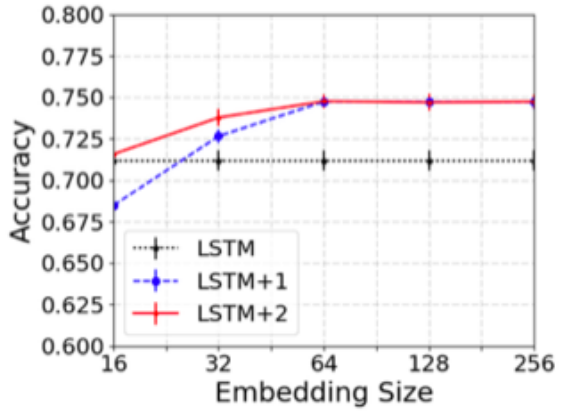
- The length of the output sequence is set as 64

Performances are significantly improved with one single layer of full connected non-linear embedding (LSTM +1)

More Layers (LSTM +2) yield similar performance.

The results are intuitive because a single embedding layer usually can deal with sparsity, skewness, and correlations of daily activity data.

Model overfitting was also not observed due to the dropout technique and the large size of data compared with number of parameters.



(b) Activity embedding

SOLUTION 3: PARALLEL LSTMS WITH JOINT TRAINING

To improve churn prediction this experiment also leverages the new user types from the clustering model.

Training Set - For users in the training set two-week behavior data is used for not only the churn labels but for computing user types through the clustering framework.

Test set - For users in the test set we can compare the initial behaviors with the training set to get the user types

- Correlation between user types and churn can be used to further improve the model.

Parallel LSTM's with joint training are proposed to implement this.

- K user types are assumed
- K is either chosen automatically by clustering framework or set to specific values

K sub-LSTM's are trained on the training set.

- Each sub LSTM is good at modeling one type of users
- K sub-LSTM's are parallelized and merged through Attention mechanism to jointly infer user types and user churn. (Figure 10)

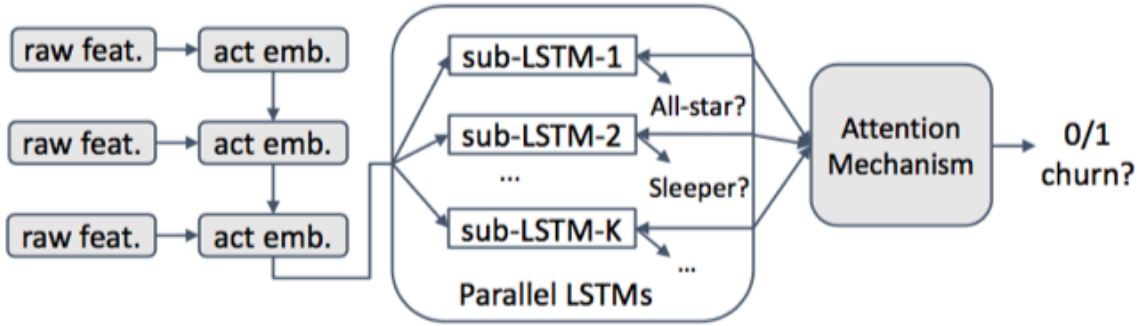


Figure 10: Parallel LSTMs with user type attention.

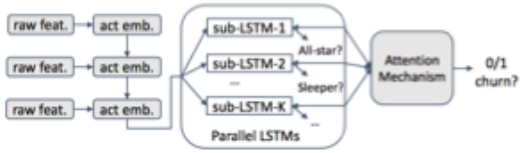


Figure 10: Parallel LSTMs with user type attention.

SOLUTION 3: PARALLEL LSTMS WITH JOINT TRAINING

- For each user, the input sequence of activity embedding vectors E is put into K sub LSTM's in parallel to generate K typed sequences

$$\mathbf{s}_k = \mathrm{LSTM}_k(\mathbf{E}).$$

- To use user types to improve churn prediction, an attention mechanism is introduced to generate user behavior embeddings by focusing on their latent types.
  - A positive attention w_k is placed to indicate probability of user being a specific type
  - w_k is computed as similarity of the corresponding sequence s_k and a global unique typing vector v_t (Which is jointly learned during the training process.)
  - Softmax function is used to normalize weights

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

- User behavior embedding u is then computed as sum of typed sequences weighted by their importance weights.

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

- Linear projection with sigmoid function is connected to u then used to predict user churn as binary classification

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

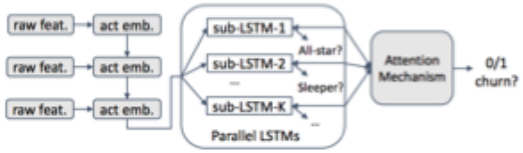


Figure 10: Parallel LSTMs with user type attention.

SOLUTION 3: PARALLEL LSTMS WITH JOINT TRAINING

Training

- To use user types for churn prediction loss l_t and churn loss l_c are jointly trained:

- l_t - first compute users soft clustering labels as Q

$$q_{ik} = \frac{(1 + ||\mathbf{f}_i - \mathbf{c}_k||^2)^{-1}}{\sum_j (1 + ||\mathbf{f}_i - \mathbf{c}_j||^2)^{-1}}. \qquad l_t = -\sum_i \sum_k q_{ik} \log(w_{ik}).$$

  - q_{ik} is a kernel function that measures similarity between feature f_i of user u_i and cluster center c_k
    - It is computed as the probability of assigning u_i to the kth type
  - w_{ik} is used to denote attention weight for user u_i on type t_k
  - For each user u_i typing loss is computed as cross-entrophy on q_i and w_i

- For l_c we compute the log loss for binary predictions

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

- The objective function of the parallel LSTM with joint training is represented below.
  - λ is a hyperparameter controlling trade-off between churn prediction and type prediction

$$l = l_c + \lambda l_t,$$

SOLUTION 3: PARALLEL LSTMS WITH JOINT TRAINING

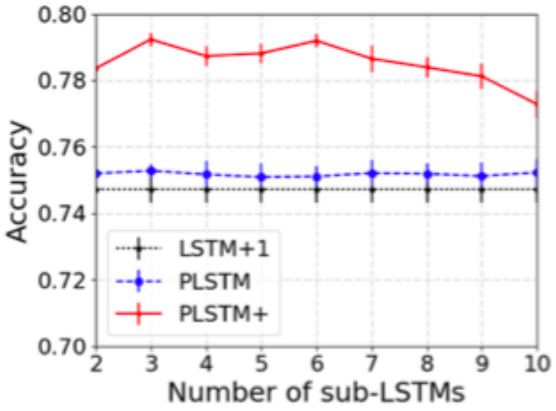
Figure 9c shows the performances of parallel LSTM's with and without joint training

- (PLSTM + vs. PLSTM)
- PLSTM is not trained with the correct user types produced by the clustering framework

In experiments, the number of clusters sub-LSTM's is varied and joint training is significantly helpful

Joint training peaks at 3 and 6 number of sub-LSTM's

- Optimal performance of 3 may be a coincidence
- Optimal performance of 6 sub-LSTM's align with the previous 6 user types determined from the clustering framework.

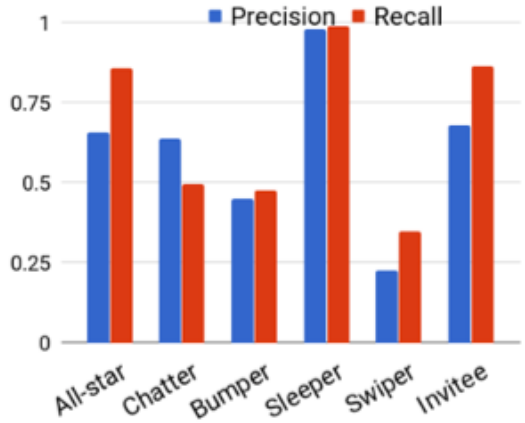


(c) Parallel LSTMs

SOLUTION 3: PARALLEL LSTMS WITH JOINT TRAINING

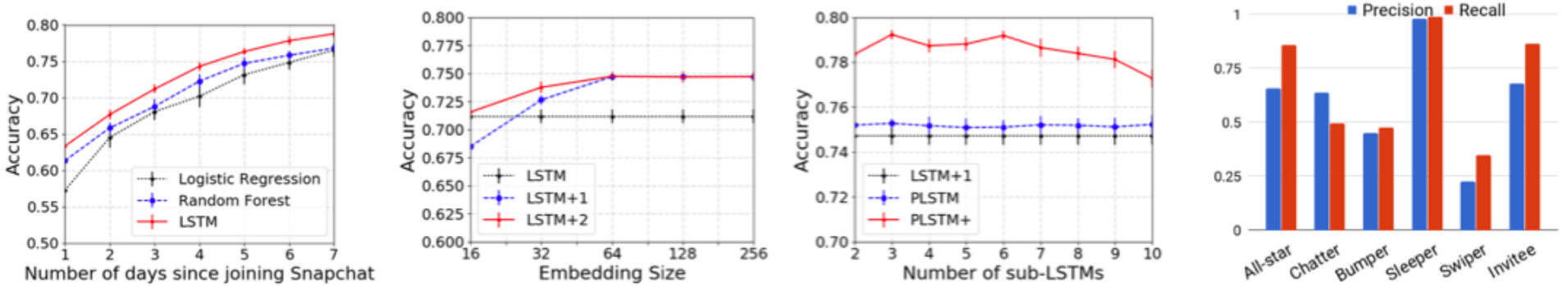
Besides Churn prediction, figure 9(d) shows that predicting a type of new user can be done with initial behaviors rather than 2 week data.

- The algorithm is good at capturing All-Star, Sleeper, and Invitee due to their distinct behavior patterns
- Swiper and Bumper are harder to predict due to their less regular activity patterns.



(d) User type prediction

RESULTS OF SNAPCHURN



(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.

CONCLUSIONS

This paper conducted analysis on snap chat users behavior and developed ClusChurn a framework for predicting churn of new users for fast response churn prediction.

The techniques from this paper can be leveraged for any online platform where users interact with platform functions as well as other users