Identifying Key Observers to Find Popular Information in Advance

Takuya Konishi, †‡ Tomoharu Iwata, § Kohei Hayashi, †‡ Ken-ichi Kawarabayashi †‡
†National Institute of Informatics
‡JST, ERATO, Kawarabayashi Large Graph Project
§NTT Communication Science Laboratories
takuya-ko@nii.ac.jp iwata.tomoharu@lab.ntt.co.jp kohei-h@nii.ac.jp k_keniti@nii.ac.jp

Abstract

Identifying soon-to-be-popular items in web services offers important benefits. We attempt to identify users who can find prospective popular items. Such visionary users are called observers. By adding observers to a favorite user list, they act to find popular items in advance. To identify efficient observers, we propose a feature selection based framework. This uses a classifier to predict item popularity, where the input features are a set of users who adopted an item before others. By training the classifier with sparse and non-negative constraints, observers are extracted as users whose parameters take a non-zero value. In experiments, we test our approach using real social bookmark datasets. The results demonstrate that our approach can find popular items in advance more effectively than baseline methods.

1 Introduction

Users in web services often share their favored or adopted items, such as messages (e.g., Twitter), web pages (Reddit), images/movies (Instagram), products (Amazon), and local services (Yelp). When a user finds a favorite item in the social networking services (SNSs), he/she adopts it, for example, by bookmarking a web page in a social bookmarking site or by retweeting on Twitter. Also, when a user purchases a product via an e-commerce website, he/she sometimes submits the review of this product, which is accessible online.

The early detection of popular items is beneficial for anyone. For example, it helps a company to make the plan of marketing strategies effective [Yu and Kak, 2012]. Or it helps a researcher to identify prospective research topics before other teams. Or it helps a gourmet to make a reservation at a hidden fine restaurant before it gets crowded.

To obtain popular items in advance, we investigate the task of identifying observers that are early adopters of popular items. Although the majority of users knows popular information after becoming a trend, some users may find it before it starts trending. Such special users would be experts who know the items more than others, heavy users who constantly monitor information, prescient users who have good insight for the future, or influential people whose choices influence those of other users. If such users are recruited to a favorite user list, they can help identify likely future trends.

Identifying observers provides more benefits than the approaches that merely predict popular items such as previous works [Kupavskii et al., 2012; Li et al., 2014; Kong et al., 2014]. It can easily personalize the obtained items by changing observers: if an observer has different preferences with us, we should remove him/her in our favorite user list. Also, when the followed observers are special users as described above, we will acquire expert knowledge from their activities. For example, their reviews in e-commerce websites give insight for finding out good products by ourselves.

Selecting appropriate observers from the massive number of users is a hard task. If inefficient observers are selected, an excessive number of items will be collected, and identified popular items will be swamped by many unpopular items. Also, if the selected observers are late majority, the most of obtained items will have become popular, and the first-mover advantage will be lost. One solution is to apply the methods for estimating influential users on SNSs [Trusov et al., 2010; Tang and Yang, 2010]. However, these methods suppose that the network among users is given. Such user network does not often exist in web services such as e-commerce.

To identify efficient observers, we take an approach based on feature selection in machine learning. We use a classifier to predict whether a given item would become popular or not in the future. The input features are users who adopted the item in advance. By training the classifier with sparse and non-negative constraints, only parameters of users who frequently adopted popular items in advance take a non-zero value. We then select the users as observers. Moreover, for augmenting the effective number of training samples, we design the loss function as an expectation over the time evolving of adoptions, which is solved by stochastic optimization. Our approach only uses event data: an event is an adoption by a user for an item with a time stamp. Such event log can be obtained from many web services, and the approach will be more widely applicable than user network based methods.

2 Problem Formulation

Suppose we seek to extract observers who can find prospective popular items but avoid selecting unpopular ones. More rigorously, we specify two requirements for the adoption behavior of observers as follows: (a) they adopt a s-popular
3 Proposed Methods

3.1 Item classification by popularity

For our task, we use a binary classifier that divides items into popular and unpopular ones. Let $y_i \in \{0, 1\}$ be a binary target variable that takes one when item $i$ is popular, and zero otherwise. The input features of the classifier indicate users who adopt the item. Let $x_i = (x_{i,0}, x_{i,1}, \ldots, x_{i,|U|})$ be a $(|U|+1)$-dimensional input feature vector. The first element is defined as $x_{i,0} = 1$, which is introduced in the feature vector to represent a bias term conveniently. The value $x_{i,u}$ for $1, \ldots, |U|$ is defined as follows:

\[
x_{i,u} = \begin{cases} 
1 & \text{if } m_{i,u} < m \\
0 & \text{otherwise,}
\end{cases}
\]

where $m_{i,u} \in \{0, 1, \ldots, |U|-1\}$ is the ranking of user $u$ adopting item $i$, and $m$ is a threshold. $m_{i,u}$ takes 0 when user $u$ is ranked first for item $i$. The set of feature-target pairs $\{(x_i, y_i)\}_{i=1}^{|I|}$ represents requirements (a) and (b) from Section 2 and is derived from the event log $E$.

As the classifier, we employ logistic regression with $L_1$ regularization and non-negative constraints for weight parameters. The loss function of logistic regression is given by:

\[
\ell(w, x_i, y_i) = (y_i - 1) \ln (1 - \sigma(w^T x_i)) - y_i \ln \sigma(w^T x_i),
\]

where

\[
\sigma(\cdot) \text{ is the sigmoid function, } w = (w_0, w_1, \ldots, w_{|U|}) \text{ is the } (|U|+1) \text{-dimensional weight parameter vector, and } w_0 \text{ is the bias parameter. Under the regularization and constraints, the optimum solution } w^* \text{ is obtained via the following minimization problem:}
\]

\[
w^* = \arg \min_{\bar{w} \in \mathbb{R}_{\geq 0}^{(|U|+1)}, w_0 \in \mathbb{R}} \left\{ \sum_{i=1}^{|I|} \ell(w, x_i, y_i) + R(\bar{w}) \right\},
\]

where $\bar{w} = (w_1, \ldots, w_{|U|})$ is the weight parameter vector except for bias $w_0$, $R(\bar{w}) = \lambda ||\bar{w}||_1$ is the $L_1$ regularized term, and $\lambda > 0$ is the regularized parameter. $\mathbb{R}_{\geq 0}^{(|U|+1)}$ denotes $|U|$-dimensional space on non-negative real numbers, constraining the possible values of $\bar{w}$ on non-negative space.

The non-negative constraints and $L_1$ regularization for $\bar{w}$ help for selecting observers. Without the non-negativity, the classifier would give negative weights for users who frequently adopt unpopular items. Although this will improve prediction performance, this does not serve the purpose of feature selection; we are not interested in such “novice” observers. $L_1$ regularization makes $w^*$ sparse, which means it eliminates weak or redundant observers.

3.2 Augmenting temporal information

In (2), we have $|I|$ training samples where each feature vector has $m$ non-zero elements. However, the setting of $m$ is somewhat arbitrary. For instance, by changing $m$ to $1, \ldots, |U|$, we obtain $|U|$ different training datasets. These “padded” samples contain richer information than the original samples, and will improve the performance. From this intuition, we introduce feature vector $x_i(z) = (x_{i,0}(z), x_{i,1}(z), \ldots, x_{i,|U|}(z))$ with varying thresholds, where each element $x_{i,u}(z)$ for $1, \ldots, |U|$ is defined as follows:

\[
x_{i,u}(z) = \begin{cases} 
1 & \text{if } m_{i,u} < z \\
0 & \text{otherwise.}
\end{cases}
\]

Here, $z > 0$ is the real-valued varying threshold. As in $x_{i,0}$, the first element is defined as $x_{i,0}(z) = 1$. If the number of events (i.e. adoptions) in item $i$ was $E_i$, $x_i(z)$ can represent $E_i$ different feature vectors. The event log $E$ can have at most $|E|$ different training samples, which is greater than $|I|$.
The training samples with high threshold values \( z \) would not be important for our task because these samples include information of late adoptions. In contrast, samples with low threshold values would be important because they only include information of early adoptions. To take into account these importance, we introduce \( p(z) \) as the probability density function over threshold \( z \). By weighting all possible training samples with \( p(z) \), the augmented loss is derived as the expectation of \( \ell \) with respect to \( z \):

\[
\sum_{i=1}^{\lfloor I \rfloor} \mathbb{E}_{p(z)} [\ell(w, x_i(z), y_i)] = \sum_{i=1}^{\lfloor I \rfloor} \int p(z) \ell(w, x_i(z), y_i) dz.
\]

As a result, the problem is written as follows:

\[
w^* = \arg\min_{w \in \mathbb{R}_{\geq 0}^{\lfloor I \rfloor}, w_0 \in \mathbb{R}} \left\{ \sum_{i=1}^{\lfloor I \rfloor} \mathbb{E}_{p(z)} [\ell(w, x_i(z), y_i)] + R(\bar{w}) \right\}.
\]

Note that this data augmented method (7) includes the above non-augmented method (4) as the special case of setting \( p(z) = \delta(z - m) \) where \( \delta(\cdot) \) is the delta function.

Before solving (7), we need to choose \( p(z) \). Because the optimal \( p(z) \) may differ for each dataset, we prepare some flexible distribution and determine its parameters by cross validation. In the experiments, we used the Weibull distribution of which the density is given by:

\[
p(z|k, \theta) = \left( \frac{k}{\theta} \right) \left( \frac{z}{\theta} \right)^{k-1} \exp \left( -\left( \frac{z}{\theta} \right)^k \right),
\]

where \( k \) is a shape parameter, and \( \theta \) is a scale parameter.\(^2\) The Weibull distribution includes the exponential distribution \((k = 1)\) and the Rayleigh distribution \((k = 2)\) as a special case. It can also represent heavy/right tailed distribution.

### 3.3 Learning algorithm

Unfortunately, (7) may not be solved by standard optimization methods such as gradient descent because the expectation is not analytically written. The expectation is, however, approximated by many samples \( \{z_s\}_{s=1}^{S} \) as follows:

\[
\sum_{i=1}^{\lfloor I \rfloor} \mathbb{E}_{p(z)} [\ell(w, x_i(z), y_i)] \approx \sum_{i=1}^{\lfloor I \rfloor} \sum_{s=1}^{S} \ell(w, x_i(z_s), y_i).
\]

On the basis of the approximation, we consider solving (7) with stochastic optimization. Note that because both \( \ell \) and \( R \) are convex, general stochastic optimization algorithms such as stochastic gradient descent are guaranteed to converge to \( w^* \) [Bottou, 1998].

We use the regularized dual averaging method with the adaptive gradient method (Ada-RDA) [Xiao, 2010; Duchi et al., 2011]. Ada-RDA repeatedly selects one sample randomly and updates the parameters for many times. A feature-target pair \( (x_i(z), y_i) \) is sampled, where \( i \) is randomly drawn from \( \{1, 2, \ldots, \lfloor I \rfloor\} \), and \( z \) is drawn from the probability distribution having the density \( p(z) \). At the \( n \)th step, the Ada-RDA updates the parameter \( w \) by using the current weight vector \( w_n = (w_{n,0}, \ldots, w_{n,\lfloor I \rfloor}) \), as follows:

\[
w_{n+1,u} = \begin{cases} 
\left[ \text{sign}(\bar{g}_{n,u}) \cdot \frac{m}{h_{n,u}} \cdot [\bar{g}_{n,u} - \lambda]_+ \right] + u 
& u \neq 0 \\
-\frac{m}{h_{n,u}} \bar{g}_{n,u} 
& u = 0,
\end{cases}
\]

where

\[
\bar{g}_{n,u} = \frac{1}{n} \sum_{n'=1}^{n} g_{n',u}, \quad h_{n,u} = \epsilon + \sum_{n'=1}^{n} g_{n',u},
\]

The training samples with high threshold values \( z \) would be important because they only include information of late adoptions. In contrast, samples with low threshold values would not be important for our task because these samples include information of early adoptions. To take into account these importance, we introduce \( p(z) \) as the probability density function over threshold \( z \). By weighting all possible training samples with \( p(z) \), the augmented loss is derived as the expectation of \( \ell \) with respect to \( z \):

\[
\sum_{i=1}^{\lfloor I \rfloor} \mathbb{E}_{p(z)} [\ell(w, x_i(z), y_i)] = \sum_{i=1}^{\lfloor I \rfloor} \int p(z) \ell(w, x_i(z), y_i) dz.
\]

As a result, the problem is written as follows:

\[
w^* = \arg\min_{w \in \mathbb{R}_{\geq 0}^{\lfloor I \rfloor}, w_0 \in \mathbb{R}} \left\{ \sum_{i=1}^{\lfloor I \rfloor} \mathbb{E}_{p(z)} [\ell(w, x_i(z), y_i)] + R(\bar{w}) \right\}.
\]

Note that this data augmented method (7) includes the above non-augmented method (4) as the special case of setting \( p(z) = \delta(z - m) \) where \( \delta(\cdot) \) is the delta function.

Before solving (7), we need to choose \( p(z) \). Because the optimal \( p(z) \) may differ for each dataset, we prepare some flexible distribution and determine its parameters by cross validation. In the experiments, we used the Weibull distribution of which the density is given by:

\[
p(z|k, \theta) = \left( \frac{k}{\theta} \right) \left( \frac{z}{\theta} \right)^{k-1} \exp \left( -\left( \frac{z}{\theta} \right)^k \right),
\]

where \( k \) is a shape parameter, and \( \theta \) is a scale parameter.\(^2\) The Weibull distribution includes the exponential distribution \((k = 1)\) and the Rayleigh distribution \((k = 2)\) as a special case. It can also represent heavy/right tailed distribution.

### 3.3 Learning algorithm

Unfortunately, (7) may not be solved by standard optimization methods such as gradient descent because the expectation is not analytically written. The expectation is, however, approximated by many samples \( \{z_s\}_{s=1}^{S} \) as follows:

\[
\sum_{i=1}^{\lfloor I \rfloor} \mathbb{E}_{p(z)} [\ell(w, x_i(z), y_i)] \approx \sum_{i=1}^{\lfloor I \rfloor} \sum_{s=1}^{S} \ell(w, x_i(z_s), y_i).
\]

On the basis of the approximation, we consider solving (7) with stochastic optimization. Note that because both \( \ell \) and \( R \) are convex, general stochastic optimization algorithms such as stochastic gradient descent are guaranteed to converge to \( w^* \) [Bottou, 1998].

We use the regularized dual averaging method with the adaptive gradient method (Ada-RDA) [Xiao, 2010; Duchi et al., 2011]. Ada-RDA repeatedly selects one sample randomly and updates the parameters for many times. A feature-target pair \( (x_i(z), y_i) \) is sampled, where \( i \) is randomly drawn from \( \{1, 2, \ldots, \lfloor I \rfloor\} \), and \( z \) is drawn from the probability distribution having the density \( p(z) \). At the \( n \)th step, the Ada-RDA updates the parameter \( w \) by using the current weight vector \( w_n = (w_{n,0}, \ldots, w_{n,\lfloor I \rfloor}) \), as follows:

\[
w_{n+1,u} = \begin{cases} 
\left[ \text{sign}(\bar{g}_{n,u}) \cdot \frac{m}{h_{n,u}} \cdot [\bar{g}_{n,u} - \lambda]_+ \right] + u 
& u \neq 0 \\
-\frac{m}{h_{n,u}} \bar{g}_{n,u} 
& u = 0,
\end{cases}
\]

where

\[
\bar{g}_{n,u} = \frac{1}{n} \sum_{n'=1}^{n} g_{n',u}, \quad h_{n,u} = \epsilon + \sum_{n'=1}^{n} g_{n',u},
\]

\(^2\)In contrast to (5), \( z \) in the Weibull distribution can take 0. However, this will not be a serious problem as explained in Section 3.3.\(^3\) As noted above, \( z \) in the Weibull distribution can take 0. However, this means all users take just 0, and thus the effect would be almost ignored. When \( z \) takes 0, the sampled feature is “empty”; this means all users take 0. While our experiments did not exclude this empty feature, we would be also able to ignore the feature.
been also proposed the sources of information diffusion in social networks has personalized information methods, these suggest users who have similar preference to obtain who can adopt popular items in advance. our proposed method uses classifiers to identify observers Kong been proposed before Methods for predicting the number of adoptions in SNSs have without the missing sets. 

\[ O_d, \text{ after removing } d' - d \text{ users taking smaller values would give a good approximation of } O_d. \]

Based on this insight, we perform a forward backward algorithm to extract \( D \) sets of observers. The algorithm is given as Algorithm 2. We prepare \( \lambda = \{ \lambda_j \}_{j=1}^{J} \), which is sorted in descending order.\(^4\) This indicates that the larger index \( j \) (i.e. smaller \( \lambda_j \)) are used, the more non-zero elements the learned \( w^* \) tends to have. In lines 5 to 20, the algorithm repeats extracting and storing observers by forward grid search of \( \lambda \). If \( D \) or more observers were obtained, the algorithm finishes the grid search. Next, the algorithm performs backward interpolation of missing sets of observers (lines 21 to 27). “remove(\( w, r \))” in line 24 returns a vector in which the smaller non-zero \( r \) elements in \( w \) are replaced by zero. If \( O_d \) is empty, the algorithm interpolates it with the larger and non-empty set \( O_{d+1} \). Finally, the algorithm outputs \( \{O_d\}_{d=1}^{D} \) without the missing sets.

4 Related Work

Methods for predicting the number of adoptions in SNSs have been proposed before [Kupavskii et al., 2012; Li et al., 2014; Kong et al., 2014], and attempts have been made to improve predictive performance. In contrast with earlier approaches, our proposed method uses classifiers to identify observers who can adopt popular items in advance.

In most of existing user or followee recommendation methods, these suggest users who have similar preference to obtain personalized information [Hannon et al., 2011; Armentano et al., 2013; Ying et al., 2012]. A similar method for locating the sources of information diffusion in social networks has been also proposed [Pinto et al., 2012]. Our aim of obtaining popular information in advance is different.

Algorithms for social network inference have been proposed [Du et al., 2013; Iwata et al., 2013; Gomez Rodriguez et al., 2013; Zaman et al., 2010]. By simulating information diffusion on inferred networks, item popularity could be predicted. However, inferring networks is a challenging task as the number of unknown parameters to be estimated is the square of the number of users. In contrast, the number of unknown parameters in our proposed method is the number of users, which is more tractable. In addition, our proposed method directly learns classifiers for predicting popularity using log data, which leads to better predictive performance.

Our proposed method can be seen as a method for finding influential users. Identifying such users leads to detection of popular items as soon as possible: once those influential users adopt an item, many other users are likely to adopt the same item. There are some existing methods for detecting influential users [Trusov et al., 2010; Tang and Yang, 2010], Garcia-Herranz et al. showed that even randomly selected users are helpful for detecting outbreaks on Twitter [Garcia-Herranz et al., 2014]. However, these methods need to user network, which is unavailable in many web services.

A closely related work is the method of [Menjo and Yoshikawa, 2008]. For predicting item popularity, this method also uses event logs. The method scores the importance of users and outputs the ranking of potential item popularity at a certain time by using the scores. Although the method seems to be able to use our problem at first glance, it does not meet our task. The method assumes each item has some growing interval that rapidly increases the number of

\[ \begin{align*}
\text{Algorithm 1 Ada-RDA (} E, \lambda, \eta, \epsilon \})
1: & \text{Input: } E = \{(i_e, u_e, t_e)\}_{e=1}^{\|E\|}, \lambda, \eta, \epsilon \\
2: & \text{Initialize: } w_1 = 0, n = 1 \\
3: & \text{for epoch } c = 1, \ldots, C \text{ do} \\
4: & \{i'_1, \ldots, i'_|I| \} = \text{shuffle}(I) \\
5: & \text{for } i = i'_1, \ldots, i'_|I| \text{ do} \\
6: & z \sim p(z) \\
7: & \text{Set } y_i \text{ and } x_i(z) \text{ according to (5)} \\
8: & \text{Compute } g_n \text{ according to (12)} \\
9: & \text{Compute } g_n \text{ and } h_n \text{ according to (11)} \\
10: & \text{Compute } w_{n+1} \text{ according to (10)} \\
11: & n = n + 1 \\
12: & \text{end for} \\
13: & \text{end for} \\
14: & \text{Output: } w^* = w_n \\
\end{align*} \]

\[ \begin{align*}
\text{Algorithm 2 Forward-backward extraction (} E, \lambda, \eta, \epsilon \})
1: & \text{Input: } E = \{(i_e, u_e, t_e)\}_{e=1}^{\|E\|}, \lambda = \{\lambda_j\}_{j=1}^{J}, \eta, \epsilon \\
2: & \text{Initialize:} \\
3: & O_d = \emptyset, \text{ for } d = 1, \ldots, D + 1 \\
4: & w_d = 0, \text{ for } d = 1, \ldots, D + 1 \\
5: & \text{for } j = 1, \ldots, J \text{ do} \\
6: & w_j = \text{Ada-RDA (} E, \lambda_j, \eta, \epsilon \}) \\
7: & d = |\text{observers}(w_j)| \\
8: & \text{if } 0 < d < D \text{ and } O_d = \emptyset \text{ then} \\
9: & w_d = w_j \\
10: & O_d = \text{observers}(w_d) \\
11: & \text{else if } d = D \text{ then} \\
12: & w_D = w_j \\
13: & O_D = \text{observers}(w_D) \\
14: & \text{break} \\
15: & \text{else if } d > D \text{ then} \\
16: & w_{D+1} = w_j \\
17: & O_{D+1} = \text{observers}(w_{D+1}) \\
18: & \text{break} \\
19: & \text{end if} \\
20: & \text{end for} \\
21: & \text{for } d = D, \ldots, 1 \text{ do} \\
22: & \text{if } O_d = \emptyset \text{ then} \\
23: & r = |\text{observers}(w_{d+1})| - |\text{observers}(w_d)| \\
24: & w_d = \text{remove}(w_{d+1}, r) \\
25: & O_d = \text{observers}(w_d) \\
26: & \text{end if} \\
27: & \text{end for} \\
28: & \text{Output: } \{O_d\}_{d=1}^{D} \\
\end{align*} \]
adoptions. Then, the method gives high scores to users that adopt an item before all the growing intervals even when the item has been already popular. In contrast, we require to obtain items before becoming a trend, and thus the method is not relevant to our task.

5 Experiments

To evaluate our proposed augmented and non-augmented methods, we used Delicious datasets [Wetzker et al., 2008], which comprise records of events where Delicious users bookmarked (adopted) web pages with time stamps. We used 11 datasets separated by tag information in Delicious. Users who appeared less than 30 times in every data set were excluded, and only items that were adopted more than 10 times were used. We assigned popular labels to the top s = 10 percent of items, and unpopular labels to the others. The summary of the datasets is given in Table 1.

We prepared six baseline methods. The baselines give a score to each user and select users having a high score as observers. The baselines assign the following score to user u:

**Random:** A random real number in [0.0, 1.0).

**Nadoptions:** The number of items adopted by u.

**Nfollowers:** The sum of the number of users adopting the same item as u but after u.

**Nearly:** The number of times that u adopted items before m other users adopted them.

**Nearly-pos:** The number of times that u adopted popular (i.e. positive) items before m other users.

**Nearly-pos/neg:** Nearly-pos divided by the number of times that u adopted unpopular (i.e. negative) items before m other users.

Random and Nadoptions fail to consider requirements (a) and (b) from Section 2, because it does not take account of popularity and the order of adoptions. Nfollowers and Nearly consider (b) by giving priority to users adopting items early. However, they do not consider (a). Nearly-pos and Nearly-pos/neg consider (a) and (b) by taking account of both popularity and the order of adoptions.

We did not compare any previous work described in Section 4. The methods using user network was unavailable in the Delicious datasets where the network does not exist as in many other web services. We did not also compare the methods for directly predicting popularity because the purpose is different from ours, and they require more information that is not included in the Delicious datasets. The method of [Menjo and Yoshikawa, 2008] is applicable for our experiments. However, the definition of important users and the purpose are different from ours as discussed in Section 4.

We conducted the experiments with the following settings. Items were split into ten subsets, with 90 percent of the items used as training data and the other 10 percent as test data. In the training dataset, all events were fully observed. From this training data, we extracted 100 sets of observers \((O_d)_{d=1}^{100}\) using the proposed methods and the baselines respectively. In the test data, we assumed that the initial \(m = 10\) users were only observed for each item. If several users adopted items at the same date around the threshold \(m\), we randomly selected ten users to be observed. If at least one observer was in the initial 10 users, each method classified the item as a popular item by supposing that we could obtain the item before more than 10 other users through observers. We evaluated the methods by using the F-measure. To simplify the comparison, we computed the area under the curve (AUC) of the #observers-F-measure curve, after normalizing the axis of #observers from \([1, 100]\) to \([0, 1.0]\). We repeated the above procedure ten times while changing the training and test data (i.e. 10-fold cross validation) and took the average of the AUCs.

We set the parameters as follows. For the non-augmented method, Nearly, Nearly-pos, and Nearly-pos/neg, we set \(m\) to 10 so as to meet the above experimental setting. Although we only tested the case of \(m = 10\), it will not be so critical for the methods; these methods are adapted according to \(m\). The augmented and non-augmented methods are learned using Algorithm 2. We set both \(\eta\) and \(\epsilon\) to 1.0, and \(C\) to 20. We also set \(\lambda\) to the set of 500 points in [0.00001, 0.01]. For the augmented method, we needed to select the pair of parameters \((k, \theta)\). We performed \(v\)-fold cross validation on \(k \in K = \{0.5, 1.0, 2.0, 5.0, 10.0, 50.0\}\), \(\theta \in \Theta = \{1.0, 5.0, 9.5, 15.0\}\), and \(v = 5\) using training data. After computing the AUC for each fold, the pair taking the best average was used in the test. Thus, the augmented method requires to validate the parameters \(v |K| |\Theta|\) times before extracting observers. However, the validation can be executed in parallel. We used 24 (= |\(K|\) |\(\Theta|\) times in order to validate all \((k, \theta)\)s in parallel for each fold. Note that the augmented method is closest to the non-augmented method when \((k, \theta)\) is set to \((50.0, 9.5)\). This setting ensures that the augmented method performs at least as well as the non-augmented method.

All the results are shown in Table 2. Figure 2 illustrates the example of #observers-F-measure curve in the webdesign dataset. The results show that the augmented method had

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#items</th>
<th>#users</th>
<th>#events</th>
</tr>
</thead>
<tbody>
<tr>
<td>ajax</td>
<td>7,924</td>
<td>9,553</td>
<td>458,706</td>
</tr>
<tr>
<td>css</td>
<td>11,647</td>
<td>16,528</td>
<td>955,829</td>
</tr>
<tr>
<td>design</td>
<td>39,100</td>
<td>33,481</td>
<td>2,015,359</td>
</tr>
<tr>
<td>java</td>
<td>8,703</td>
<td>6,641</td>
<td>289,976</td>
</tr>
<tr>
<td>javascript</td>
<td>10,841</td>
<td>11,254</td>
<td>641,639</td>
</tr>
<tr>
<td>linux</td>
<td>14,835</td>
<td>13,867</td>
<td>615,193</td>
</tr>
<tr>
<td>news</td>
<td>3,807</td>
<td>5,267</td>
<td>133,571</td>
</tr>
<tr>
<td>opensource</td>
<td>7,485</td>
<td>6,923</td>
<td>296,902</td>
</tr>
<tr>
<td>photography</td>
<td>9,910</td>
<td>11,704</td>
<td>399,751</td>
</tr>
<tr>
<td>science</td>
<td>4,790</td>
<td>4,505</td>
<td>130,747</td>
</tr>
<tr>
<td>webdesign</td>
<td>16,171</td>
<td>18,035</td>
<td>1,038,161</td>
</tr>
</tbody>
</table>
Table 2: Results of AUC in the Delicious datasets. The boldface means that the result was significantly best in terms of the two-sided t-test with 95 percent confidence.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Random</th>
<th>N adoptions</th>
<th>N followers</th>
<th>Nearly</th>
<th>Nearly-pos</th>
<th>Nearly-pos/neg</th>
<th>Non-augmented</th>
<th>Augmented</th>
</tr>
</thead>
<tbody>
<tr>
<td>ajax</td>
<td>0.051</td>
<td>0.168</td>
<td>0.200</td>
<td>0.172</td>
<td>0.252</td>
<td>0.133</td>
<td>0.305</td>
<td>0.315</td>
</tr>
<tr>
<td>css</td>
<td>0.034</td>
<td>0.135</td>
<td>0.174</td>
<td>0.156</td>
<td>0.293</td>
<td>0.178</td>
<td>0.332</td>
<td>0.344</td>
</tr>
<tr>
<td>design</td>
<td>0.024</td>
<td>0.136</td>
<td>0.164</td>
<td>0.164</td>
<td>0.244</td>
<td>0.060</td>
<td>0.276</td>
<td>0.276</td>
</tr>
<tr>
<td>java</td>
<td>0.058</td>
<td>0.190</td>
<td>0.210</td>
<td>0.204</td>
<td>0.228</td>
<td>0.113</td>
<td>0.246</td>
<td>0.247</td>
</tr>
<tr>
<td>javascript</td>
<td>0.052</td>
<td>0.134</td>
<td>0.183</td>
<td>0.129</td>
<td>0.240</td>
<td>0.082</td>
<td>0.299</td>
<td>0.304</td>
</tr>
<tr>
<td>linux</td>
<td>0.056</td>
<td>0.171</td>
<td>0.194</td>
<td>0.195</td>
<td>0.266</td>
<td>0.102</td>
<td>0.299</td>
<td>0.298</td>
</tr>
<tr>
<td>news</td>
<td>0.085</td>
<td>0.163</td>
<td>0.247</td>
<td>0.224</td>
<td>0.327</td>
<td>0.306</td>
<td>0.374</td>
<td>0.376</td>
</tr>
<tr>
<td>opensource</td>
<td>0.082</td>
<td>0.154</td>
<td>0.177</td>
<td>0.191</td>
<td>0.235</td>
<td>0.118</td>
<td>0.255</td>
<td>0.256</td>
</tr>
<tr>
<td>photography</td>
<td>0.047</td>
<td>0.169</td>
<td>0.202</td>
<td>0.183</td>
<td>0.223</td>
<td>0.071</td>
<td>0.236</td>
<td>0.240</td>
</tr>
<tr>
<td>science</td>
<td>0.085</td>
<td>0.170</td>
<td>0.189</td>
<td>0.183</td>
<td>0.218</td>
<td>0.143</td>
<td>0.241</td>
<td>0.241</td>
</tr>
<tr>
<td>webdesign</td>
<td>0.034</td>
<td>0.143</td>
<td>0.168</td>
<td>0.170</td>
<td>0.291</td>
<td>0.132</td>
<td>0.336</td>
<td>0.350</td>
</tr>
<tr>
<td>Average</td>
<td>0.055</td>
<td>0.157</td>
<td>0.192</td>
<td>0.179</td>
<td>0.256</td>
<td>0.131</td>
<td>0.291</td>
<td>0.295</td>
</tr>
</tbody>
</table>

Table 3: Averages of runtime taken to extract observers in the design and webdesign datasets. The value in the parenthesis of the augmented method denotes the average time (seconds) for the validation of \((k, \theta)\).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>N adoptions</th>
<th>N followers</th>
<th>Nearly</th>
<th>Nearly-pos</th>
<th>Nearly-pos/neg</th>
<th>Non-augmented</th>
<th>Augmented</th>
</tr>
</thead>
<tbody>
<tr>
<td>design</td>
<td>6.00</td>
<td>16.93</td>
<td>2.57</td>
<td>4.16</td>
<td>4.15</td>
<td>118.60</td>
<td>1206.75</td>
</tr>
<tr>
<td></td>
<td>(1069.31)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>webdesign</td>
<td>3.18</td>
<td>10.11</td>
<td>1.24</td>
<td>2.02</td>
<td>2.02</td>
<td>28.45</td>
<td>904.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(875.98)</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2: #observers-F-measure curve in the webdesign dataset.

significantly better performance than the baselines for all the datasets. While both the two proposed methods had nearly equal performances for seven datasets, the augmented method was significantly better for four datasets and the average of all the datasets. This result suggests that augmentation of the training samples improves performance.

Table 3 shows the averages of runtime taken to extract observers in the design and webdesign datasets. We excluded the result of Random; although it was as fast as other baselines, the performance improvement would be expected depending on the implementation. The proposed methods took more time than baselines: while baselines check training items only once, the proposed methods repeat checking them \(C\) times. The augmented method took the longest time, and the most part was used in the validation of \((k, \theta)\). Although we computed all the \((k, \theta)\)s in parallel, the runtime did not decrease as we expected. This is due to the overhead of the hardware scheduling and the computation of the AUC for selecting the best \((k, \theta)\).

6 Summary and Discussion

In this paper, we formulated a new problem to find observers and proposed a feature selection based framework to address the problem. Our proposed methods outperformed the baselines in real social bookmark datasets.

Let us remark the tradeoff between performance and computational cost for our proposed methods. While the augmented method outperformed the non-augmented method in many datasets, the augmented method requires additional computation caused by the cross validation of \((k, \theta)\). This computation can be reduced by validating \((k, \theta)\)s in parallel. If all the parameters are validated in parallel, the augmented method can run the validation ideally \(v|K||\Theta|\) times faster. However, as described in experiments, the actual runtime will get worse than the expected one due to hardware limitation and additional costs. Thus, when setting large \(|K|, |\Theta|, \) and \(v\) in large-scale data, we suggest using the non-augmented method. Otherwise the augmented method is recommended.

Acknowledgments

KH was supported by MEXT KAKENHI 15K16055.

References

[Armentano et al., 2013] Marcelo Gabriel Armentano, Daniela Godoy, and Analía A Amandi. Followee rec-


