A Social Interaction Activity based Time-Varying User Vectorization Method for Online Social Networks

Tianyi Hao and Longbo Huang
Institute for Interdisciplinary Information Sciences, Tsinghua University
haoty14@mails.tsinghua.edu.cn, longbohuang@tsinghua.edu.cn

Abstract
In this paper, we consider the problem of user modeling in online social networks, and propose a social interaction activity based user vectorization framework, called the time-varying user vectorization (Tuv), to infer and make use of important user features. Tuv is designed based on a novel combination of word2vec, negative sampling and a smoothing technique for model training. It jointly handles multi-format user data and computes user representing vectors, by taking into consideration user feature variation, self-similarity and pairwise interactions among users. The framework enables us to extract hidden user properties and to produce user vectors. We conduct extensive experiments based on a real-world dataset, which show that Tuv significantly outperforms several state-of-the-art user vectorization methods.

1 Introduction
Due to the rapid recent development of Internet technologies, online social networks are becoming more and more popular with users. Important information associated with users can be inferred from online social networks, and various user-centric applications have been developed based on information from online social networks, including recommendations and advertisements. As a result, how to efficiently retrieve useful user information from online social networks has become one central question for both industry and academia.

Among the important topics about online social networks, in this paper, we focus on user modeling from an information retrieval perspective. User modeling has been receiving increasing attention in recent years, which aims at developing a deeper understanding about users, so that better analysis and prediction about online user behavior can be achieved. There have been many prior works proposing different approaches for user modeling, for instance, [Amir et al., 2016; Tang et al., 2015]. Different from these methods which are based on user connectivity, we instead focus on utilizing social interaction activities for characterizing online users.

Social interaction activities are widely available and valuable information. People on social media publish messages and photos describing events in life, exchanging ideas and sharing news. From these activities, one can observe many interesting properties of users. Compared to other information, e.g., connectivity, social interaction activities provide more detailed descriptions about users, and preserve time-dependent features. Thus, mining interaction activities should provide us with more accurate modeling for users.

However, efficiently utilizing social interaction data is challenging. First, interaction activities often contain various forms of data, e.g., connectivity and text, and one has to provide a unifying approach to properly fuse different information. Second, the proposed method should be computationally efficient and generate consistent results. Third, the model should respect the self-similar nature of user activities over time. This imposes additional constraints on the modeling solution and further increases the design complexity.

To tackle these challenges, in this work, we propose a novel framework for user modeling based on social interaction activities, called the Time-varying User Vectorization (Tuv). It is based on a novel combination of word2vec [Mikolov et al., 2013a], negative sampling and a smoothing technique for model training. It jointly handles multi-format user data and computes user vectors by taking into consideration user feature variation, self-similarity and pairwise interactions among users. Tuv possesses unique advantages due to its vectorization nature. (i) Vectorization facilitates various analytical operations on users, e.g., similarity computation and clustering. (ii) Time-varying vectors better track evolving user properties, and enable a better user behavior prediction. These features can not only help develop a better understanding about users, but also be used to improve performance of Internet applications including recommendations and advertisements.

The method of user vectorization is not new, e.g., [Perozzi et al., 2014; Tang et al., 2015; Fu et al., 2009]. Our work distinguishes itself from these works in two ways. First, while most works carry out vectorization based on social graphs, we instead focus on utilizing social interaction records, which gives us stronger information about user interactions. We also benefit from text data in interaction records, which contain important information about users not contained in connectivity data. Second, while most works focus on user modeling that is static in time, we handle the problem with time-varying user vectorization, which focuses on modeling user properties over different times. This way, we obtain a deeper understanding about how user properties change in time.
Our contributions are summarized as follows.

- We propose to utilize user interaction activities for time-varying user vectorization in online social networks. Different from most existing approaches, which often rely on static user information, e.g., connectivity graph, our idea enables a more detailed user characterization that keeps track of the dynamic patterns of user features.
- We design a novel framework based on user interaction activities called the Time-varying User Vectorization (Tuv). Tuv jointly handles multi-format data, including user labels, text messages and time information. Moreover, it is able to take into consideration self-similarity of user properties and time-varying relationships among users. Although Tuv is developed for user modeling, it also applies to other tasks with multi-sourced data processing and feature identification.
- We conduct extensive experiments based on a real-world dataset to compare Tuv with several state-of-the-art methods for user vectorization in three applications, i.e., user attention evaluation, time-varying user similarity estimation, and user occupation prediction. Our results show that Tuv is able to accurately capture hidden user properties and outperform existing methods.

2 Related Work

There have been prior works on social network modeling and user vectorization. [Ren et al., 2008] uses SVD for network embedding to detect community structure. [Zhao et al., 2010; 2011] provides a technique to embed social graph nodes into coordinate spaces to preserve node distances within graphs. DeepWalk [Perozzi et al., 2014] generates random walk paths to vectorize social network nodes with the SkipGram model [Mikolov et al., 2013a]. LINE [Tang et al., 2015] defines proximities within a network, and utilizes these metrics for node vectorization. [Tu et al., 2016; Grover and Leskovec, 2016; Liu et al., 2016] are also node vectorization models. Different from our work which uses social interaction activities, all aforementioned works rely on social graphs.

Besides results on social graphs, there are also works utilizing other user information. [Zhang et al., 2017] combines social graphs and user profile information. There are also vectorization models utilizing text information [Amir et al., 2016; Yang et al., 2016]. [Nallapati et al., 2008; Cho et al., 2016] combine network structures with text models, but the texts are associated to local network nodes instead of interaction among nodes. There have also been works on dynamic networks [Fu et al., 2009; Rossi et al., 2013], but most of them focus on network structures instead of social interaction records with multiple forms of data.

To jointly handle text information, we use the word2vec framework from [Mikolov et al., 2013a; 2013b], which is an effective language model for word vectorization.

3 Problem Statement

In this section, we describe the setting of social interaction activities, and formally define the problem of time-varying user vectorization.

3.1 Representing Social Network Data

We describe how we represent social network data, including users and activities, in order to carry out our investigation.

(i) User: We denote the set of users in consideration by $U = \{u_1, u_2, \ldots, u_n\}$, where $n$ is the total number of users.

(ii) Activity: We focus on two social interactions, publishing and forwarding. (a) Publishing is the activity that an author publishes a new message. We denote it as a tuple $A = \langle u, t, m \rangle$, where $u$ is the author, $t$ is the timestamp, and $m$ is the text information. (b) Forwarding is the activity that a user reposts a message from another user. We denote it as a tuple $A = \langle u, t, m, u_a, t_a, u_f, t_f \rangle$. The notations $u$, $t$ and $m$ are similar to a publishing activity. $u_a$ and $t_a$ denote the author and timestamp of the original message. $u_f$ is the user from whom the message is forwarded, and $t_f$ is $u_f$’s forwarding time. If the message is forwarded from the original author, we set $u_f$ to be a null user $u_\emptyset$, and $t_f$ to be a null time $t_\emptyset$. Finally, we denote the set of all activities by $A$.

3.2 Time-varying User Vectorization

Our objective is to find, for each user $u \in U$, a vector representation, which can capture important characteristics of the user and can be used to analyze his relationship with others.

Specifically, we define the active time of a user to be the time period from his first activity to the last. Denote the start time of user $u$’s activities to be $T_u$, and the end time to be $T_u$. Since user behavior does not change very frequently, in order to simplify analysis and computation, we divide time into discrete timeslots of size $\Delta t$. For each timeslot $t$, we define a vector $\vec{v}_u(t)$ for user $u$ at time $t$. Our goal is to find vectors $\{\vec{v}_u(t)\}_{u,t}$ that satisfy the following requirements:

- The resulting vectors can reflect certain personal properties of users, such as occupations. Users with similar properties shall have similar vector representations.
- Relations among different user vectors can reflect relationship (e.g., closeness or similarity) among users.
- The vectors of a user should also not vary too much during consecutive time slots.

Prior works on user vectorization, e.g., [Perozzi et al., 2014; Tang et al., 2015; Ren et al., 2008] often assume that each user vector $\vec{v}_u$ is a constant vector obtained from structure information of the social networks, e.g., connectivity. We instead focus on the problem of time-varying user vectorization based on user interaction activities, and aim to find a vector function $\vec{v}_u(t)$ whose value characterizes the time-varying nature of user properties. This problem is challenging, because social interaction is a complicated system with various data types, such as pairwise interactions, text messages and timestamps, and a unified framework needs to be designed in order to fuse multi-format data. In our work, we propose a novel framework which takes multi-format data into consideration, and captures the time-varying properties accurately.

4 Method for Vectorization

In this section, we present our time-varying user vectorization (Tuv) method. We start by describing the text data vectorization. Then, we define our objective function and present the algorithm for computing user vectors.
4.1 Vectorization for Text Messages

As we have described, there is a text message \( m \) in each activity record, which contains rich valuable information. Since text messages are often not in numerical form and cannot be directly used in vectorization algorithms, we first preprocess them and convert them into vectors.

For a text message \( m \), we use the sum of word2vec vectors [Mikolov et al., 2013a; 2013b] of the words in it to construct its vector representation \( \vec{v}(m) \). This method has also been adopted in prior works, e.g., [Yu et al., 2015; Arora et al., 2015], and can often produce accurate results for short messages [Adi et al., 2017]. Compared to more complicated models, e.g., [Le and Mikolov, 2014], it is computationally efficient for handling large-scale text messages.

4.2 Objective Function and Optimization

In this section, we explain the design and meaning of user vectors, and present our objective as well as how to solve the Tuv problem. Different from existing works based on graphs, e.g., [Perozzi et al., 2014; Tang et al., 2015], we focus on user interaction activities, since doing so enables the characterization of users’ time-varying properties. This is a challenging task as it requires fusing multiple information types including user connectivity, text messages and timestamps.

Recall that we have two types of interactions: publishing \( A = \langle u, t, m \rangle \) and forwarding \( A = \langle u, t, m, u_a, t_a, u_f, t_f \rangle \). In order to jointly handle them, we unify both representations by changing \( \langle u, t, m, u_a, t_a, u_f, t_f \rangle \) to \( \langle u, t, m, u, t, u_a, t_a \rangle \), where \( u_a \) is the null user and \( t_a \) is the null time. Below, we regard every activity to be in this form.

As discussed above, for each user \( u \), our goal is to find a vector \( \vec{v}_u(t) \) for the user at each time \( t \), which can reflect his properties. Below, one will see that the pattern of messages that a user usually publishes or forwards has a strong relationship with the properties of the user. Thus, we will make full use of the text messages when computing our user vectors.

To this end, we define the “likelihood” that a user \( u \) will publish or forward a message \( m \) at time \( t \) to be a function:

\[
L_m(u, m, t) = S(\vec{v}_u(t) \cdot \vec{v}(m)).
\]

Here \( \vec{v}_u(t) \) is the vector for user \( u \) at time \( t \), \( \vec{v}(m) \) is the vector for text message \( m \), and \( S(x) = \frac{1}{1 + e^{-x}} \) is the sigmoid function. We use the dot product \( \vec{v}_u(t) \cdot \vec{v}(m) \) because we want the user vector to be closer to the text message vectors they often publish or forward. The use of \( S(x) \) ensures that the value of \( L_m(u, m, t) \) is in the range \((0, 1)\). With this definition, \( L_m(u, m, t) \) has a positive correlation with the probability that user \( u \) will publish or forward message \( m \) at time \( t \), so that \( \vec{v}_u(t) \) can reflect user properties related to text messages.

In addition to the user-message relationship, pairwise user relationship is also important. To utilize pairwise user relationship to obtain a more integrated model, we introduce two more vectors \( \vec{r}_u(t) \) and \( \vec{s}_u(t) \) for each user \( u \) at time \( t \) to describe the pairwise relationship among users. Based on these vectors, for user \( u \) in activity \( A \), we define the following likelihoods to describe the relationship between pairs of users.

- Original author: The likelihood that user \( u \) at time \( t \) will forward a message authored by \( u_a \) at time \( t_a \) is defined as a function:

\[
L_{\text{au}}(u, u_a, t, t_a) = S(\vec{v}_u(t) \cdot \vec{r}_{u_a}(t_a))
\]

- Last forwarder: The likelihood that user \( u \) at time \( t \) will forward a message last forwarded by \( u_f \) at time \( t_f \) is:

\[
L_f(u, u_f, t, t_f) = S(\vec{v}_u(t) \cdot \vec{s}_{u_f}(t_f)).
\]

Intuitively, \( L_{\text{au}} \) denotes the likelihood that a user forward a particular user’s original post, and \( L_f \) denotes the likelihood that a user forwards a friend’s post. The reason to use \( \vec{r}_{u_a}(t) \) and \( \vec{s}_{u_f}(t) \) instead of \( \vec{v}_u(t) \) is to allow the likelihood to be asymmetric. For a publishing activity, we similarly define the likelihoods as we have extended it to a forwarding one.

Based on the above definitions, we define the overall likelihood for user \( u \) to perform an activity \( A \) to be their product:

\[
L(A) = L_{\text{au}}(u, u_a, t, t_a) \cdot L_f(u, u_f, t, t_f) = S(\vec{v}_u(t) \cdot \vec{r}_{u_a}(t_a)) \cdot S(\vec{v}_u(t) \cdot \vec{s}_{u_f}(t_f)).
\]

Here we remove the \( S(\vec{v}_u(t) \cdot \vec{s}_{u_f}(t)) \) term if \( u_f = u_a \) and \( t_f = t_a \). Figure 1 provides a graphical demonstration of this procedure for generating the likelihood function \( L(A) \).

It remains to find appropriate vectors \( \vec{v}_u(t), \vec{r}_{u_a}(t) \) and \( \vec{s}_{u_f}(t) \), so that the likelihood values best fit our dataset. To do so, we adopt the technique of negative sampling proposed in [Mikolov et al., 2013b], which works as follows. For an activity \( A = \langle u, t, m, u_a, t_a, u_f, t_f \rangle \), in addition to user \( u \), we randomly choose a negative sample \( \tilde{u} \neq u \), following the empirical distribution of users in the activity set \( \mathcal{A} \). With the negative sample \( \tilde{u} \), we define the negative likelihood \( \tilde{L}(A) \):

\[
\tilde{L}(A) = (1 - S(\vec{v}_{\tilde{u}}(t) \cdot \vec{v}(m))) \cdot (1 - S(\vec{v}_{\tilde{u}}(t) \cdot \vec{r}_{u_a}(t_a))) \cdot (1 - S(\vec{v}_{\tilde{u}}(t) \cdot \vec{s}_{u_f}(t_f)))
\]

where \( \tilde{A} \) is attained by replacing user \( u \) with \( \tilde{u} \) in \( A \). Similarly we remove the \( (1 - S(\vec{v}_{\tilde{u}}(t) \cdot \vec{s}_{u_f}(t_f))) \) term if \( u_f = u_a \) and \( t_f = t_a \). Then, given all activities \( A \in \mathcal{A} \), our target likelihood function is given by:

\[
L(A) = \prod_{A \in \mathcal{A}} \left( L(A) \cdot \tilde{L}(A) \right)
\]

Next, we want to also take into account the time-varying and self-similar property of users in the overall objective function. To do so, we assume that for each user \( u \), the vector \( \vec{v}_u(t + \Delta t) \) follows a Gaussian distribution, with its mean
Algorithm 1 Building the task list

Input: The set of user activities, \( A \), the set of all users, \( U \)
Output: The task list \( T \)
1: Initialise \( T \) with empty list
2: for \( A \in A \) do
3:   Generate negative sample \( \bar{A} \), and add \((A, \bar{A})\) to \( T \)
4: for \( u \in U \) do
5:   for \( t \in \{T_u, T_u + \Delta t, \ldots, T_u - \Delta t\} \) do
6:     Add \( (u, t, t + \Delta t) \) to \( T \)
7: return \( T \)

Algorithm 2 Optimization of the objective function

Input: The task list, \( T \)
Output: The vectors, \( \bar{v}_u(t), \bar{r}_u(t), \bar{s}_u(t) \)
1: Initialize \( \bar{r}_u(t), \bar{s}_u(t) \) for all \( u, t \) with random vectors
2: \( \eta = \eta_0, \bar{v}_u(t) = 0 \) for all \( u, t \)
3: for each iteration \( i \leq i_{\text{max}} \) do
4:   Sort \( T \) with a uniform random order
5: for each task in \( T \) do
6:   if the task is \((A, \bar{A})\) then
7:     \( \bar{c} = 0 \)
8:   for \( a \in \{A, \bar{A}\} \) do
9:     \( g_e = \eta_1(a = A) - S(\bar{v}_u(t) \cdot \bar{r}_u(t)) \)
10:    \( \bar{c} = \bar{c} + g_e \bar{v}_u(t) \)
11:   \( \bar{r}_u(t) = \bar{r}_u(t) + g_e \bar{r}_u(t) \)
12:   \( \bar{s}_u(t) = \bar{s}_u(t) + g_e \bar{s}_u(t) \)
13: end for
14: end if
15: end for
16: \( \eta = \max(\lambda \eta, \eta_{\text{min}}) \)
21: return \( \bar{v}_u(t), \bar{r}_u(t), \bar{s}_u(t) \)

5 Experiments and Applications

In this section, we describe experimental results of our Tuv model, obtained based on real-world social network data. We present three applications of our method and show that Tuv outperforms existing methods.

5.1 Data and Settings

We collect data from Sina Weibo (http://weibo.com/), a Chinese social media with over 100 million active users every day. We first identify and fix the user set, and then crawl their interaction records with Python scripts. We collected data for 7,370 verified users (with IDs being verified by Sina Weibo) as our user set, with 252,560 directed following edges among them. For interaction activities, there are 17,387,066 records in total, including 9,829,429 publishing records and 7,557,637 forwarding records. The time duration is from August 2009 to June 2017.

After data collection, we begin to implement our framework. As described in Section 4.1, we compute the vectors of all the text messages in advance. Since most messages in Sina Weibo are in Chinese, we first use the THULAC tool [Sun et al., 2016] to split the messages into words. After that, we compute the vector \( \bar{v}(m) \) for each message \( m \) following Section 4.1. After obtaining the sentence vectors, following the algorithms in Section 4.2, we compute the vector representations \( \bar{v}_u(t) \) and user relationship vectors \( \bar{r}_u(t) \) and \( \bar{s}_u(t) \). The detailed parameters for \( \Delta t \) being one year long are 100 dimensions for vectors, \( i_{\text{max}} = 30, \eta_0 = 0.1, \eta_{\text{min}} = 1 \times 10^{-5}, \lambda = 0.518, \sigma^2 = 0.09 \). For \( \Delta t \) being one season long, we re-
place them with $\lambda = 0.527$, $\sigma^2 = 0.00225$. Based on our experiments, the performance of our model is not very sensitive to the parameters, and the performance stays similar within a reasonable range of values. These values are chosen in this range to optimize the performance of the evaluation.

In the following, we present three applications of the resulting user vectors computed above, and compare them with other results obtained using other methods. We will see that $Tuv$ achieves better performance.

### 5.2 User Attention Evaluation

Our first application is user attention evaluation, which is used to describe the level at which a user pays attention to another user. In an online social network, when user $u_1$ pays attention to user $u_2$, $u_1$ has a larger probability of reading and forwarding $u_2$’s messages. User attention evaluation is an important problem and has a lot of applications including friend recommendation and message ranking.

To evaluate user attention, we conduct experiments on predicting whether a user will forward a message from another user at certain time. We use $80\%$ of data to train the $Tuv$ model and compute pairwise similarities for all users. Then, from the remaining $20\%$ of data, we randomly choose $8000$ records to generate $4000$ positive samples and $4000$ negative ones. Then we use $4000$ samples among them to train an SVM [Cortes and Vapnik, 1995] (with input from $Tuv$ model and output to be $+/-$ labels generated) and use $4000$ samples to test. Then we use the SVM accuracy to evaluate our model.

The positive and negative samples are generated in this way. For each record $A = (u, m, u_a, t_a, u_f, t_f)$, we generate a positive sample $(+1, (u, t, u_a, t_a))$, meaning that the message is forwarded by $u$. Then, we randomly select another user $\tilde{u} \neq u$ to generate a negative sample $(-1, (\tilde{u}, t, u_a, t_a))$, meaning that the message is not forwarded by $\tilde{u}$.

We use the LIBSVM library [Chang and Lin, 2011] to do the classification. Each input vector contains $I_1 = \{\vec{v}_u(t), \vec{r}_u(t), \vec{s}_u(t)\}$, $I_2 = \{\vec{v}_{u_a}(t_a), \vec{r}_{u_a}(t_a), \vec{s}_{u_a}(t_a)\}$, together with the pairwise cosine similarities between vectors from these two sets $I_3 = \{\cos(\vec{v}_{\tilde{u}_1}(t), \vec{v}_{\tilde{u}_2}(t)) | \tilde{u}_1 \in I_1, \tilde{u}_2 \in I_2\}$.

For comparison, we select several existing user vectorization methods. The input vectors are also user embedding vectors and their cosine similarities. The baselines include:

- **DeepWalk** [Perozzi et al., 2014]: A method generating node vector representation from a social graph with random walks.
- **LINE** [Tang et al., 2015]: A large-scale information network embedding scheme based on social graph.
- **Weighted LINE with user activities** (LINE-w): Instead of the social graph, use the social interaction records as the input to the LINE framework, and the edge weights are social interaction frequencies.
- **LINE-w-time**: Regard each user at each time slot as a node, and apply LINE-w.

We use accuracy, precision and recall as the metrics, and the results are shown in Table 1. We see that social interaction based methods outperform graph structure based methods. Among social interaction methods, the accuracy of $Tuv$-year outperforms than all other methods. For precision and recall, there is only one exception that static LINE-w has a better precision, but it has a much lower recall. The results demonstrate that $Tuv$ achieves a good performance in user attention evaluation.

### 5.3 Time-varying User Similarities

Our second application is computing similarities between users. To carry out our evaluation, we use the cosine similarity of user vectors to denote the similarity between users. For $u_1$ and $u_2$ at time $t$, the similarity level is computed as:

$$\text{sim}(u_1, u_2, t) = \cos(\vec{v}_{u_1}(t), \vec{v}_{u_2}(t)).$$

Note that this is a feature not possessed by other methods without a time component, e.g., connectivity based similarity [Perozzi et al., 2014; Tang et al., 2015; Liu et al., 2016].

With the user vectors we computed, for each user $u$ at time $t$, we identify its closest neighbors with the largest similarity values. According to our observation, these neighbors are usually close friends, users who have deep working relations with the user, or users who have similar properties.

Take the blog ID fbb0916 (http://weibo.com/fbb0916) for example. We select five neighbors and plot the similarity values among them from 2014 to 2017 by seasons in Figure 2. It shows that our vectorization for users can efficiently capture time-varying user properties, including their relationship with other users in online social networks, and the pattern changes over time.

In order to evaluate the accuracy of our user similarities, we compute the message-based similarity among users as the ground truth. Specifically, it is defined as the Jaccard similar-

### Table 1: Prediction result of user attention and user similarities.

<table>
<thead>
<tr>
<th>Method</th>
<th>attention (%)</th>
<th>prec.</th>
<th>rec.</th>
<th>RMSE</th>
<th>$R^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepWalk</td>
<td>72.6</td>
<td>78.2</td>
<td>62.7</td>
<td>0.0591</td>
<td>0.341</td>
</tr>
<tr>
<td>LINE</td>
<td>74.0</td>
<td>78.2</td>
<td>70.6</td>
<td>0.0567</td>
<td>0.393</td>
</tr>
<tr>
<td>LINE-w</td>
<td>83.4</td>
<td><strong>87.5</strong></td>
<td>78.0</td>
<td><strong>0.0494</strong></td>
<td><strong>0.539</strong></td>
</tr>
<tr>
<td>LINE-w-season</td>
<td>77.0</td>
<td>75.4</td>
<td>80.1</td>
<td>0.0649</td>
<td>0.206</td>
</tr>
<tr>
<td>LINE-w-year</td>
<td>82.8</td>
<td>81.9</td>
<td>84.4</td>
<td>0.0573</td>
<td>0.381</td>
</tr>
<tr>
<td>Tuv-season</td>
<td>82.3</td>
<td>80.8</td>
<td>84.7</td>
<td>0.0588</td>
<td>0.349</td>
</tr>
<tr>
<td>Tuv-year</td>
<td><strong>85.0</strong></td>
<td>82.5</td>
<td><strong>88.9</strong></td>
<td>0.0563</td>
<td>0.402</td>
</tr>
</tbody>
</table>

![Figure 2: Similarities with fbb0916 for five users over seasons.](image-url)
ity [Levandowsky and Winter, 1971], i.e.,

\[ \text{nsim}(u_1, u_2, t) = \frac{|S(u_1, t) \cap S(u_2, t)|}{|S(u_1, t) \cup S(u_2, t)|}, \]

where \( S(u_1, t) \) and \( S(u_2, t) \) is the set of messages that \( u_1 \) and \( u_2 \) published or forwarded at time \( t \). We use 80% of all messages to train our Tuv model, and the remaining 20% to compute the Jaccard similarities. Then, we use linear regression to evaluate the correlation between the Tuv similarities \( \text{nsim}(u_1, u_2, t) \) and the ground-truth \( \text{nsim}(u_1, u_2, t) \), since a stronger correlation indicates a better accuracy of our Tuv method. We use 4, 000 pairs of users to train the linear regression model, and use 4, 000 pairs to test it. We also compare our method with the existing baselines.

The result is shown in Table 1. The metrics include the root-mean-squared error (RMSE) and the coefficient of determination \((R^2)\). From the result, we see that except for LINE-w, our method Tuv-year reaches the best performance. However, the method LINE-w does not capture the time-based property. For the time-varying version of LINE-w, i.e., LINE-w-time, our method Tuv shows better results than LINE-w-time in both year-long and season-long timeslots.

### 5.4 Occupation Prediction

Our third application is occupation prediction. Occupations are important information related to user status. For verified users on the website, there is verification information, for instance, “music producer” and “writer”, which we use to obtain their true occupations. Among the 7370 users, up to 5,990 users take the eight most common occupations, such as music, actor, and business. We use 3,990 of them as training data, and the other 2,000 as testing data. Assuming that vectors for user \( u \) at different time slots are \( \bar{v}_u(T), \bar{v}_u(T + \Delta t), \ldots, \bar{v}_u(T + (k - 1)\Delta t) \), we compute the general user vector \( \bar{v}_u^{*} \) of the user as the average of all the above vectors.

Figure 3 shows the vector distribution of the users in three most common occupations, obtained by projecting the 100-dimensional vectors onto a 2-d space by the PCA method [Pearson, 1901]. We can see that the user vectors for each occupation concentrate around a particular area, showing that our vectorization model captures hidden features of users.

After obtaining the average vector representation \( \bar{v}_u^{*} \) of the users, we use SVM [Cortes and Vapnik, 1995] for training and predicting. For each user \( u \), the input is the average vector \( \bar{v}_u^{*} \). For comparison, except for the above methods, we also include the average text message vectors, i.e., the average of all text message vectors published or forwarded by a user, to represent the user. The prediction accuracies of Tuv and the baselines are shown in Table 2. For every scheme evaluated, we also created the “x+text” version of it, by concatenating the average text message vectors with the resulting vectors. Doing so leads to a better prediction accuracy for all schemes.

Note that the \( F_1 \) scores [Yang, 1999] are common accuracy measures for multi-class classification. In Table 2, “normal” refers to the algorithm with input being vectors generated by Tuv (or DeepWalk, LINE); “only text” refers to the case when the input are average sentence vectors, and “x+text” refers to the case when the input are vectors obtained by concatenating the Tuv vectors and average sentence vectors.

![Figure 3: 2-d vector distribution of users in the three most common occupations under Tuv. Here we set \( \Delta t = 1 \) year.](image)

<table>
<thead>
<tr>
<th>Method</th>
<th>( F_1 )-mic</th>
<th>( F_1 )-mac</th>
<th>( F_1 )-mic</th>
<th>( F_1 )-mac</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only text</td>
<td>-</td>
<td>0.668</td>
<td>0.520</td>
<td></td>
</tr>
<tr>
<td>DeepWalk</td>
<td>0.616</td>
<td>0.391</td>
<td>0.655</td>
<td>0.493</td>
</tr>
<tr>
<td>LINE</td>
<td>0.631</td>
<td>0.400</td>
<td>0.640</td>
<td>0.466</td>
</tr>
<tr>
<td>LINE-w</td>
<td>0.619</td>
<td>0.402</td>
<td>0.659</td>
<td>0.465</td>
</tr>
<tr>
<td>Tuv-season</td>
<td>0.650</td>
<td>0.495</td>
<td>0.662</td>
<td>0.528</td>
</tr>
<tr>
<td>Tuv-year</td>
<td><strong>0.668</strong></td>
<td><strong>0.517</strong></td>
<td><strong>0.681</strong></td>
<td><strong>0.543</strong></td>
</tr>
</tbody>
</table>

Table 2: Prediction result of occupations in \( F_1 \)-micro and \( F_1 \)-macro.

We see that Tuv outperforms the methods of DeepWalk, LINE and LINE-w for both the normal versions and the “x+text” versions. It is interesting to see that all “x+text” versions perform better, showing that text message vectors contain strong information for user occupations. It can also be observed that Tuv achieves a significant improvement after merging with the average text message vectors, while other baseline methods do not provide an obvious improvement.

### 6 Conclusion

In this paper, we propose a novel framework, called time-varying user vectorization (Tuv), for studying user relationships in online social networks. Tuv is different from previous works on user vectorization in that it focuses on utilizing user interaction activities, and can reflect changes of user properties over time. Our model takes into account text messages for all activities, original author and last forwarder for forwarding activities and time information. We have also considered the time-varying effect of user vectors. Our experimental results show that our Tuv scheme is efficient in characterizing user features and is suitable for various applications important to Internet business, including user attention evaluation, user similarity computing and occupation prediction.

### Acknowledgments

This work is supported in part by the National Natural Science Foundation of China Grant 61672316 and Grant 61303195, the Tsinghua Initiative Research Grant, and the China Youth 1000-Talent Grant.
References


[Yang et al., 2016] Yi Yang, Ming-Wei Chang, and Jacob Eisenstein. Toward socially-infused information extraction: Embedding authors, mentions, and entities. EMNLP, 2016.


