Vol.2019-MBL-90 No.38
Vol.2019-UBI-61 No.38
2019/3/5

コールドスタート問題を考慮した スマートフォンアプリケーションの利用推定に関する検討

陳　成[1]　前川　卓也[1]　天方　大地[1]　原　隆浩[1]

概要：スマートフォンの普及とともに、アプリケーションストアに登録されているスマートフォン向けアプリケーションの数が大幅に増加している。それに伴い、ユーザのスマートフォンにインストールされているアプリケーションの数も増加しており、ユーザによる起動アプリケーション選択の手間を軽減するために、ユーザが次に利用するアプリケーションを推定する研究が盛んに行われている。本研究ではユーザの利用履歴を利用し、再帰的ニューラルネットワークを用いてユーザの利用アプリケーションの推定を行う。特にコールドスタート問題に着目し、他のユーザ（ソースユーザ）のデータを利用して利用履歴が存在しないユーザ（ターゲットユーザ）の利用アプリケーションを推定する手法を提案する。本研究では、ソースユーザのスマートフォンにインストールされているアプリケーションとターゲットユーザのスマートフォンにインストールされているアプリケーションの間の意味的な類似性を利用し、利用履歴が存在しないユーザとアプリケーションを予測する。

1. Introduction

Because of the recent proliferation of smartphones, the number of available smartphone Apps in App stores is rapidly increasing. This huge number and diversity of Apps enable the installation of a large number of Apps on a user's smartphone. Although the large number of available Apps makes our lives more convenient, it also introduces a new challenge because selecting a particular App from all the installed Apps can be a time-consuming task. To assist a user in selecting Apps in an efficient manner, methods for predicting a next-use App that recommend probable candidates to the user have been studied in the mobile computing, pervasive computing, and recommender system research communities [1].

Mobile Apps are often used in conjunction with other relevant Apps [2]. For instance, when a user uses a smartphone for his or her business work, he or she may initially use a 'word processor' App. When he or she wants to send the edited document to others, the next-used App is likely to be an 'e-mail' App. Considering this feature, a next-use App can be predicted by supervised learning methods based on a user's App usage history.

However, supervised learning-based next-use App prediction poses several cold-start problems, the prediction model cannot be trained immediately after a user firstly installs the next-use App prediction system or install a new App from App store, since no usage history of the user and the App. To alleviate the aforementioned cold-start problems, some studies consider leveraging other users' (source users) usage histories to construct a prediction model for a target user. However, Apps that are installed on the smartphones of source users are different from those installed on the smartphones of the target user, making it difficult to recommend Apps that are not installed on the smartphones of source users to the target user (especially for newly released Apps). In our dataset, 26% of the installed Apps were categorized as unseen Apps.

To cope with the aforementioned problems, we propose an App recommendation method by combining multi-class classification and the semantic representations of a smartphone App. The proposed method permits us to use the source users' usage histories to train a prediction model tailored to a target user to alleviate the cold-start problems. Our concept is to obtain training data in the 1-of-K representations tailored to a target user from the source users' usage histories by using App semantic representations.

[1] 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University

Let us assume that a series of App usages that is obtained from the source users is provided. We obtain a series of K-dimensional vectors in the 1-of-K representation tailored to the target user from the series of App usages, where K is the number of Apps that are installed on a target user's smartphone. We calculate the similarity between each pair of an App installed on a target user's smartphone and an App in the source users' usage histories based on their semantic representations and subsequently convert the App usage history of the source users into a series of the 1-of-K representation of a target user's App based on the calculated similarities. The series of K-dimensional vectors is further used to train a multi-class classifier (next-use App prediction model) for the target user.

2. Related work

2.1 Next-use App prediction

There have been some previous studies related to the next-use App prediction. Zou et al. [3] proposed some light-weighted Bayesian methods to predict a next-use App based on the App usage history. Parate et al. [4] designed an App prediction algorithm that required no prior training and predicted not only the App that will be used next but also the time at which it will be used based on text compression methods. Sun et al. [5] used a prediction model that utilized some App temporal features such as frequency and duration. Liao et al. [6] designed a time-based App predictor, extracting some features from the App usage trace such as an App's usage count in the entire usage trace, usage count in the temporal bucket, and the frequency of App usage. In contrast, we attempted to cope with the cold-start problems in the next-use App prediction. Further, our method, which is based on deep learning, does not require handcrafted features.

Some other studies used sensor data, such as the data from the global positioning system (GPS), for generating predictions. Shin et al. [7] proposed a context model for App prediction that used an extensive variety of contextual information from sensors in a smartphone and constructed a personalized App-prediction model based on naive Bayes. Liao et al. [8] proposes an App usage prediction framework that uses both explicit data from mobile sensors and implicit transitions across App usage. Bazea-Yates et al. [9] collected multiple sensor data from a home screen App Aviate and built a parallel tree-augmented naive Bayes model to generate predictions.

2.2 Cold-start problems in next-use App prediction

Few studies have considered the cold-start problems that have been mentioned in the introduction. Bazea-Yates et al. [9] divided the cold-start problems into App cold-start and user cold-start problems. For the App cold-start problem, where a user installed a new App on his or her smartphone, they used the App usage information obtained from other users for generating predictions. For the user cold-start problem, where a first-time user used the App recommendation system, they used a similar user's model to predict the behavior of the new user. They also compared the installed Apps of new and other users to determine similar users. Natarajan et al. [2] also investigated the App and user cold-start problems. For the App cold-start problem, they assume that a user is more likely to prefer an item belonging to the same genre than an item belonging to a different genre after using a series of the same type of items. In accordance with that assumption, they recommended a new App belonging to the same genre as that of the previously used Apps to a user. For the user cold-start problem, they created a new user's usage history based on uniform distribution over all the Apps. In contrast to the aforementioned studies, we use high-level App semantic information to alleviate the cold-start problem, enabling us to predict the usage patterns of Apps that are not installed on the smartphones of source users.

3. Method

3.1 Preliminaries

We use source users' usage histories to train a prediction model and then generate predictions for a target user. We define an App, source users' U, and target user's u_t as follows:

Definition 1 (App).
A set of Apps is installed on a user's smartphone, i.e., $\mathcal{A} = \{a_{\grave{1}}, a_{\grave{2}}, a_{\grave{3}}, ..., a_{\grave{n}}, ..., a_{\grave{K}}\}$, where $a_{\grave{n}}$ is the n^{th} App in the set. In contrast, the i^{th} used App in a user's usage history can be represented as a_i. When $a_{\grave{n}}$ is the i^{th} used App, a_i becomes equal to $a_{\grave{n}}$. The i^{th} used App a_i by a user is represented as a semantic vector v_{a_i}. The App is also represented as $o^K_{a_i}$ in accordance with the 1-of-K scheme, where K is the number of dimensions of the vector, i.e., the number of Apps installed on the user's smartphone. In addition, we refer to an App that is not installed on the source users' smartphones but is installed

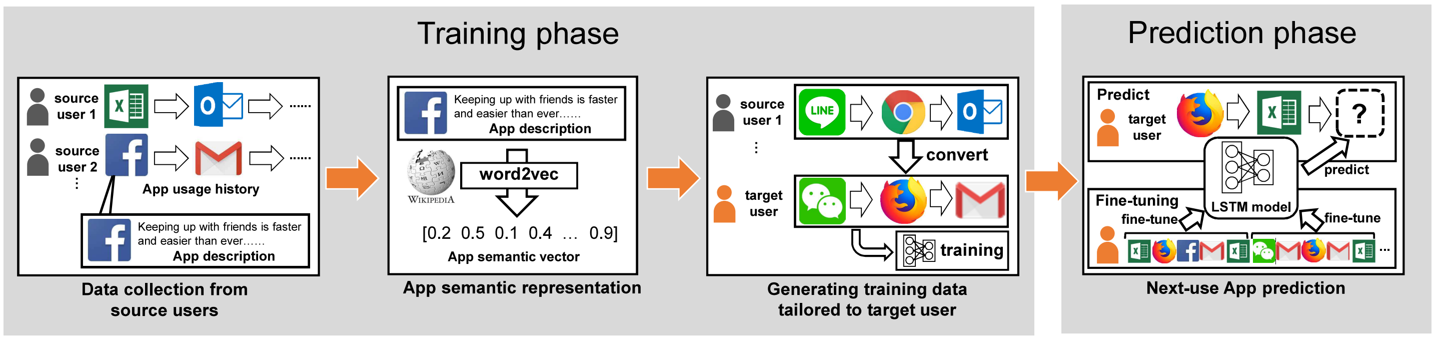

図 1 Overview of the proposed method

on the target user's smartphone as an " unseen App." Furthermore, we refer to an App that is installed on both the source users' smartphones and the target user's smartphone as an "existing App."

Definition 2 (Source users).
A group of N users $U = \{u_1, u_2, u_3, ..., u_N\}$. Each user u_i $(1 \leq i \leq N)$ has an App usage history with length M_i, i.e., $\mathcal{S}_i = \{a_1, a_2, a_3, ..., a_{M_i}\}$.

Definition 3 (Target user).
A user $u_t \notin U$, who has usage history with length M_t, i.e., $\mathcal{S}_t = \{a_1, a_2, a_3, ..., a_{M_t}\}$. When we wish to predict a next-use App, we do not use all the usage histories to generate predictions because an App that was used a long time ago exhibited little relation to the latent next-use App. In this study, we use a sequence of App usage histories with lengths of k $(1 \leq k \leq M)$, i.e., $s = \{a_{i-k}, a_{i-k+1}, ..., a_{i-2}, a_{i-1}\}$, to predict a next-use App a_i. Further, the next-use App prediction problem can be defined as follows:

Definition 4 (App prediction).
Given a series of App usage histories with length k $(1 \leq k \leq M)$, i.e., $s = \{a_{u_t,i-k}, a_{u_t,i-k+1}, ..., a_{u_t,i-2}, a_{u_t,i-1}\}$, of a target user u_t, the probability that each App $a_{\hat{n}}$ that is installed on this user's smartphone will be a next-use App, i.e., $P(a_{\hat{n}}|s)$, is calculated. We further select the top-N Apps to be the next-use App candidates.

3.2 Overview

An overview of our proposed method is depicted in Figure 1. The proposed method has two phases, training and prediction. In the training phase, we initially compute an App semantic representation, i.e., an App vector, for each App installed on the source users' or target user's smartphones. Further, we utilize the sequences of App semantic vectors from the source users' usage histories to generate training data that are tailored to the target user by leveraging the App semantics. Subsequently, we train a prediction model for the target user based on a deep neural network comprising long short-term memory (LSTM) [10] on the tailored training data. In the prediction phase, we use the App usage history of the user through $i-1$ to predict the target user's i^{th} App usage candidates.

3.3 Semantic representation of App

We assume that we obtain a description of each App from the App store, which is used to build a semantic App vector. An App description describes the features of an App and the functions that are to be provided to users, similar Apps have similar descriptions. Therefore, we construct semantic App vectors from descriptions to calculate the similarity between various Apps. Note that we cannot obtain the descriptions of Apps that are preinstalled or directly installed from the APK files. The titles of such Apps are used in place of descriptions.

In our proposed method, we initially perform the morphological analysis of Japanese descriptions based on [11] because almost all the Apps used in our experiment are Japanese Apps. Therefore, we tokenize a Japanese description, i.e., text segmentation, to extract words from it. Further, we select W keywords of an App that represent the App well from the extracted words using the importance of the words that are computed based on *tf-idf*. The importance of word w is computed using *tf-idf* as follows:

$$tf\text{-}idf(w) = Frequency(w) \cdot \log \frac{N}{|d : d \ni w| + 1},$$

where $Frequency(w)$ indicates the word occurrence frequency of w in the description, and N is the total number of descriptions of all Apps.

Further, we use a word embedding model word2vec [12] to obtain a word vector for each keyword representing the semantics of the keyword. We use an external pre-trained

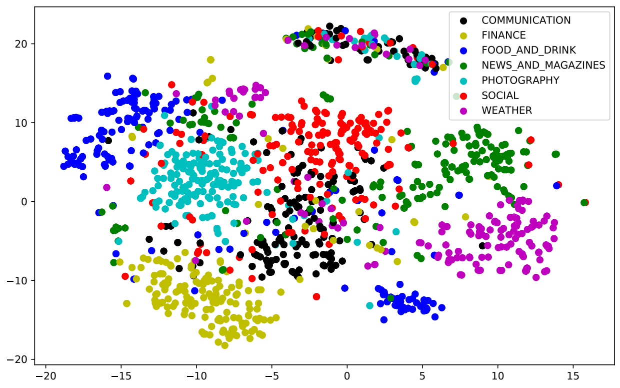

図 2 Visualization of the App semantic vectors

word2vec model on Japanese Wikipedia[*1] to compute a word vector for each keyword. Finally, we compute the mean vector of the W keywords and regard it as the App's semantic vector. We ignore the keywords that are not present in the external Japanese Wikipedia corpus. We use this method to build an App semantic vector for each App on the source users' or a target user's smartphones.

Figure 2 depicts the semantic vectors of approximately 6,000 Apps that are constructed using the aforementioned method that is projected into two-dimensional space using t-SNE [13]. Each vector is colored according to the category that it belongs to on Google Play. We can observe that approximately all of the Apps belonging to the same categories are grouped together and that the distance between Apps belonging to different categories is larger than that for those in the same category.

3.4 Generating training data tailored to a target user

The procedure of generating the training data that is tailored to a target user is summarized in Algorithm 1. An input of the algorithm is a sequence with length M of the App usages of a source user, and an output is a sequence of App usages with length M tailored to a target user. After an App usage history with length M tailored to the target user is obtained, we extract $M-(k+1)$ sequences of App usages with length $k+1$ and subsequently train the next-use App predictor on the sequences. We initially prepare the conversion matrices that convert the usage of an App of the source user into the usage of an App installed on the target user's smartphone (lines 3-9). Using the matrices, we probabilistically generate multiple sequences of App usages tailored to the target user from the sequence of App usages of the source user (lines

Algorithm 1: Generating training data tailored to a target user

```
Input: $\mathcal{T}$: a set of Apps from a target user, $\mathcal{S}$: a
       sequence of App usages from a source user, $\theta$: a
       parameter used to control the quantity of
       unseen Apps in training data
1  $\mathcal{D} \leftarrow \emptyset$
2  /* Extract a set of Apps of source user */
3  $\mathcal{A}_s \leftarrow \textsc{ExtractAppSet}(\mathcal{S})$
4  /* Divide target user Apps into 2 groups */
5  $\mathcal{G}_e \leftarrow \mathcal{T} \cap \mathcal{A}_s$ /* A set of existing Apps */
6  $\mathcal{G}_u \leftarrow \mathcal{T} \cap \bar{\mathcal{G}}_e$ /* A set of unseen Apps */
7  /* Creating App conversion matrices */
8  $\boldsymbol{M}_e \leftarrow \textsc{GenerateConversionMatrix}(\mathcal{T}, \mathcal{A}_s, \mathcal{G}_e)$
9  $\boldsymbol{M}_u \leftarrow \textsc{GenerateConversionMatrix}(\mathcal{T}, \mathcal{A}_s, \mathcal{G}_u)$
10 /* Generating $P$ sequences of training data
     from $\mathcal{S}$ */
11 repeat
12     /* Initialize App conversion table */
13     $T \leftarrow \emptyset$
14     for $\forall a_s \in \mathcal{A}_s$ do
15         $\boldsymbol{M} \leftarrow$ zero matrix
16         $x \leftarrow \textsc{Uniform}(0,1)$ /* Randomly generate
             $x \in [0,1]$ */
17         if $x < \theta$ then
18             $\boldsymbol{M} \leftarrow \boldsymbol{M}_u$
19         else
20             $\boldsymbol{M} \leftarrow \boldsymbol{M}_e$
21         /* Generate mapping from a source App to
             a target App */
22         $a_t \leftarrow \textsc{ConvertApp}(\boldsymbol{M}, a_s)$
23         /* Add mapping from a source App to a
             target App */
24         $T[a_s] \leftarrow a_t$
25     /* Convert $\mathcal{S}$ using conversion table $T$*/
26     $\mathcal{D}_p \leftarrow \textsc{ConvertSequence}(\mathcal{S}, T)$
27     $\mathcal{D} \leftarrow \mathcal{D} \cup \{\mathcal{D}_p\}$
28 until repeat the processing $P$ times
Output: $\mathcal{D}$: a set of training sequences tailored to a
        target user
```

11-28). To generate each output sequence, we probabilistically construct a conversion table that describes a mapping from an App of the source user to an App of the target user (lines 13-24). Because a mapping from an App of the source user to an App of the target user that is computed from the App semantics is not entirely accurate, we probabilistically generate a variety of multiple sequences in accordance with the similarities between Apps to train

[*1] http://www.cl.ecei.tohoku.ac.jp/~m-suzuki/jawiki_vector/

a robust next-App predictor.

3.4.1 Generating conversion matrices

In this study, we generate two conversion matrices that are used to control the number of usages of unseen Apps that are included in the training data to be generated, i.e., sequences of App usages tailored to the target user, because the number of usages of unseen Apps to be included in the training data can be typically less than that of the existing Apps. A conversion matrix is used to obtain the candidates for target user Apps converted from an App of the source user as follows.

$$\hat{\boldsymbol{p}}^{|\mathcal{T}|} = \boldsymbol{M}\boldsymbol{o}_{a_s}^{|\mathcal{A}_s|}, \qquad (1)$$

where $\boldsymbol{M}$ is the conversion matrix, $\boldsymbol{o}_{a_s}^{|\mathcal{A}_s|}$ is a $|\mathcal{A}_s|$-dimensional vector of App a_s represented using the 1-of-K scheme, $\mathcal{A}_s$ is a set of Apps installed on the source users' smartphone, and $\mathcal{T}$ is a set of Apps installed on a target user's smartphone. In addition, $\hat{\boldsymbol{p}}^{|\mathcal{T}|}$ is a $|\mathcal{T}|$-dimensional vector that depicts the probability of converting each App of the target user from a_s. Here, we explain the procedures for generating the conversion matrices. After obtaining a set of Apps installed on the source users' smartphone $\mathcal{A}_s$ (line 3), we obtain a set of existing Apps $\mathcal{G}_e$ (line 5) and a set of unseen Apps $\mathcal{G}_u$ (line 6). Using $\mathcal{G}_e$, we generate a conversion matrix for the existing Apps $\boldsymbol{M}_e$ (line 8). $\boldsymbol{M}_e$ is used to convert an App of the source user that is not installed on the target user's smartphone into an existing App. Further, we generate a conversion matrix for unseen Apps $\boldsymbol{M}_u$ using $\mathcal{G}_u$ (line 9). $\boldsymbol{M}_u$ is used to convert an App of the source user that is not installed on the target user's smartphone into an unseen App. A conversion matrix $\boldsymbol{M}$, which is a $|\mathcal{A}_s| \times |\mathcal{T}|$ matrix, can be described as follows.

$$\boldsymbol{M} = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1|\mathcal{A}_s|} \\ m_{21} & m_{22} & \cdots & m_{2|\mathcal{A}_s|} \\ \vdots & \vdots & \ddots & \vdots \\ m_{|\mathcal{T}|1} & m_{|\mathcal{T}|2} & \cdots & m_{|\mathcal{T}||\mathcal{A}_s|} \end{bmatrix}$$

Using $\mathcal{G}$ ($\mathcal{G}_e$ or $\mathcal{G}_u$), a value of each element m_{ij} in the matrix is calculated according to Algorithm 2.

When an App of the source user a_j is also installed on the smartphone of the target user, the mapping from a_j to an App of the target user a_i is calculated deterministically (lines 4-8) (For example, when a_i is equal to a_j, a_j is definitely converted into a_i (line 6)). Otherwise, the mapping is calculated based on the similarity between a_i and a_j (lines 10-15). After a_i and a_j are converted into semantic word vectors (lines 11-12),

Algorithm 2: GenerateConversionMatrix

```
Input: $\mathcal{T}$: a set of Apps from a target user, $\mathcal{A}$: a set of
       Apps from a source user, $\mathcal{G}$: a set of unseen
       Apps or existing Apps
1  $\boldsymbol{M} \leftarrow |\mathcal{A}| \times |\mathcal{T}|$ zero matrix
2  for $i = 1$ to $|\mathcal{T}|$ do
3      for $j = 1$ to $|\mathcal{A}|$ do
4          if $a_j \in \mathcal{T}$ then
5              if $a_i = a_j$ then
6                  $m_{ij} \leftarrow 1$
7              else
8                  $m_{ij} \leftarrow 0$
9          else
10             if $a_i \in \mathcal{G}$ then
11                 $v_{a_i} \leftarrow$ Word-Embedding($a_i$)
12                 $v_{a_j} \leftarrow$ Word-Embedding($a_j$)
13                 $m_{ij} \leftarrow NormalizedSimilarity(v_{a_i}, v_{a_j})$
14             else
15                 $m_{ij} \leftarrow 0$
Output: Conversion matrix $\boldsymbol{M}$
```

$NormalizedSimilarity(v_{a_i}, v_{a_j})$ in the 13th line calculates the semantic similarity between a_i and a_j based on cosine similarity as follows:

$$NormalizedSimilarity(v_{a_i}, v_{a_j}) = \frac{Similarity(v_{a_i}, v_{a_j})}{S},$$

where

$$Similarity(v_{a_i}, v_{a_j}) = \frac{v_{a_i} \cdot v_{a_j}}{||v_{a_i}|| \cdot ||v_{a_j}||}$$

and $S = \sum_n Similarity(v_{a_n}, v_{a_j})$. Note that we calculate $\boldsymbol{M}$ for existing and unseen Apps as $\boldsymbol{M}_e$ and $\boldsymbol{M}_u$, respectively. For $\boldsymbol{M}_e$, m_{ij} is zero when a_i is an unseen App (line 15). This indicates that a_j (that is not installed on the target user's smartphone) is converted only into existing Apps. In contrast, for $\boldsymbol{M}_u$, m_{ij} is zero when a_i is an existing App (line 15). This indicates that a_j is converted only into unseen Apps.

3.4.2 Generating a conversion table

We probabilistically generate a conversion table (dictionary) T using $\boldsymbol{M}_e$ and $\boldsymbol{M}_u$ for each iteration, which enables us to create a variety of training sequences toward robust next-App estimation. T describes a mapping from each App of the source user to an App of the target user. For each App of the source user a_s, we randomly select a conversion matrix $\boldsymbol{M}$: $\boldsymbol{M}_e$ or $\boldsymbol{M}_u$ (lines 16-20 in Algorithm 1). In this study, we use a large θ value to increase the number of usages of unseen Apps that are to be included in the training sequences. Using $\boldsymbol{M}$, we determine a mapping from a_s to an App of the target user (line 22 in Algorithm 1) in accordance with Algorithm 3.

Algorithm 3: ConvertApp

```
Input: $\boldsymbol{M}$: a conversion matrix, $a$: an App from a
        source user
1 $\boldsymbol{o}_a \leftarrow$ 1-of-K($a$) /* 1-of-K representation of $a$ */
2 /* Compute conversion probabilities according
    to Equation 1 */
3 $\hat{\boldsymbol{p}} \leftarrow \boldsymbol{M}\boldsymbol{o}_a$
4 $\hat{a_t} \leftarrow$ Roulette-Wheel-Selection($\hat{\boldsymbol{p}}$)
  Output: an App $\hat{a_t}$ similar to $a$
```

First, an App installed on the source users' smartphone a is represented using the 1-of-K scheme (line 1). Further, we compute the probability with which each App of the target user is converted from a according to Equation 1 (line 3). Finally, an App of the target user that is similar to a is probabilistically chosen based on roulette wheel selection (line 4). In the roulette wheel selection, an item is randomly selected according to the probability that is associated with each item. In this study, the probability of the i^{th} App (similarity between a and the i^{th} App) is stored in the i^{th} element of $\hat{\boldsymbol{p}}$. A mapping from a_s to a_t obtained by ConvertApp() is further added to T (line 24 in Algorithm 1).

3.4.3 Converting a sequence of App usages

Using the conversion table T, we convert the sequence of App usages by a source user $\mathcal{S}$ into a training sequence tailored to the target user (line 26 in Algorithm 1) according to Algorithm 4.

Algorithm 4: ConvertSequence

```
Input: $\mathcal{S}$: a sequence of App usages from a source
        user, $T$: a conversion table
1 $\mathcal{S}_t \leftarrow$ array with length $|\mathcal{S}|$
2 for $s = 1$ to $|\mathcal{S}|$ do
3     /* Obtain mapping from $a_s$ */
4     $a_t \leftarrow T[a_s]$
5     $\mathcal{S}_t[s] \leftarrow a_t$
  Output: a sequence of App usages tailored to a target
          user $\mathcal{S}_t$
```

We convert the s^{th} usage of an App of the source user a_s into an App of the target user a_t using the conversion table T (line 4).

3.5 Predicting the next-use App using a neural network

In accordance with the procedure described in Section 3.4, we obtain multiple training sequences with length $k+1$ of App usages in which the $k+1^{th}$ usage is an answer, i.e., the next-use App. Thus, we train a neural network to output the $k+1^{th}$ used App when a sequence of usages from 1 to k is provided as input. Note that each App usage is represented in the 1-of-K scheme with the number of dimensions being the number of Apps installed on the target user's smartphone. Because the App usage history is the time-series data, we choose the LSTM model to generate predictions. In accordance with the data structure of the training data, we adopt a two-layer many-to-one multilabel classification LSTM model [14] whose input and output are a sequence and a fixed-size vector, respectively. In our case, the input of the network is a series of App usage histories represented in the 1-of-K scheme, and the output is a vector consisting of the class probability of each App. The network comprises two LSTM layers with 300 nodes using a rectified linear units (ReLU) activation function and an output softmax layer. To reduce overfitting, we use dropout, a simple regularization technique where randomly selected nodes are ignored during training. We train the network using backpropagation based on Adam to minimize the categorical cross-entropy between estimates and the ground truth. In the prediction phase, an App usage sequence with length k obtained from the target user is provided as input to the network, and the network outputs the probability with which each App of the target user will be a next-use App. Finally, the Apps with the top-N probabilities are chosen as the next-use App candidates.

4. Evaluation

4.1 Dataset

We collected an App usage dataset from 20 participants using our developed Android App that was installed on their smartphones. Each participant in the dataset has 9,430 usage history data items on an average collected for approximately one hundred days. The average duration of the participants' data collection period was 82.5 days. The average number of installed Apps is 52.8. To obtain the App semantic representations, we retrieved a description of each App from Google Play. For Apps that were not available on Google Play, we used the titles of the Apps instead of descriptions. Table 1 presents the total number of unseen Apps and existing Apps and their usage

表 1 Statistics of our dataset

	total # of Apps	total # of usage sequences
Unseen Apps	383	14,052
Existing Apps	1,064	174,549
All Apps	1,447	188,601

sequences. The usage sequence of an unseen App is the usage sequence with length k whose next-use App, i.e., answer, is an unseen App ($k = 5$). In contrast, the usage sequence of an existing App is the usage sequence whose next-use App, i.e., answer, is an existing App.

4.2 Evaluation methodology

4.2.1 Evaluation measure

To evaluate the App prediction methods, we employ a top-N prediction accuracy metric in our experiment. If the *any* App in a set of N candidate Apps is actually the next-use App, we regard the next-use App to be accurately predicted. The prediction accuracy metric can be defined as **Accuracy@N** $= \frac{\sum_{i=1}^{T} H_i^N}{\sum_{i=1}^{T} A_i}$, where H_i^N is the number of accurately predicted next-use Apps for a test user i when N candidates are provided, A_i is the total number of test data for test user i, and T is the number of test users.

4.2.2 Methods

We evaluate the following methods.

• **MFU** (most frequently used): The top-N candidates are the most frequently used N Apps in the training data.

• **RankSVM**: A ranking method proposed in [15], which is a learning-to-rank algorithm for query-based document search, is used to obtain the top-N next-use Apps. RankSVM is a pair-wise ranking algorithm that computes a ranking list based on a pair of candidate items.

• **One-hot**: The neural network architecture is identical to that of the proposed method. However, the training data are not tailored to a target user. Each App is represented in the 1-of-K scheme where K is the size of the set of all Apps installed on the smartphones of source/target users.

• **Word2vec**: The neural network architecture is similar to that of the proposed method. Each App that is used in the input and output of the network is represented by a semantic App vector. In the prediction phase, we calculate the cosine similarity between an output vector and a semantic vector for each of the Apps installed on a target user's smartphone. The top-N candidates are the N Apps of the target user with the top-N cosine similarities. The loss function that is used to train the network is the mean squared error.

• **Proposed**: This is the proposed method.

We use "leave-one-participant-out" cross validation to evaluate the aforementioned methods. Therefore, we consider one participant to be a target user and the remaining participants as source users.

4.3 Results

Figure 3 depicts the transitions of the Accuracy@N for the five methods when N is varied. The poor performance of MFU indicates the difficulties associated with next-use App prediction. Because the Apps used by the participants are diverse, it is difficult to predict the next-use Apps using only information related to the frequency of usage. The performance of RankSVM is poor because the model cannot deal with the time-series data. In addition, the considerably high dimensionality of the input vectors of the model can also degrade the performance. The Accuracy@N for Word2vec does not exhibit a substantial alteration when N is varied. This result indicates that the candidate Apps provided by Word2vec are not diverse. Word2vec outputs a semantic vector and selects candidate Apps according to the similarity between a semantic vector of an App and the output vector, resulting in the low diversity of the App candidates, i.e., including only Apps similar to the output vector.

Proposed achieved the optimal performance and outperformed One-hot by approximately 3%. Figure 4 depicts the Accuracy@N of the five methods for the existing Apps. The figure exhibits that the performances of Proposed and One-hot are nearly identical. This is because the neural network architectures of these methods are identical. In contrast, as depicted in Figure 5, with respect to the Accuracy@N of unseen Apps, Proposed considerably outperformed the other methods. Surprisingly, Proposed achieved 56% accuracy when $N = 10$ even though the usage history of these unseen Apps by source users is not available at all. One-hot and MFU could not predict the use of unseen Apps at all because of their architectures. While their accuracies are poor, RankSVM and Word2vec could also predict the usage of these unseen Apps. This is because these methods compute the App candidates based on the semantics of Apps. However, Proposed, the architecture based on the 1-of-K scheme, works effectively for this multi-class classification task.

5. Conclusion

This study proposed a new method for predicting the

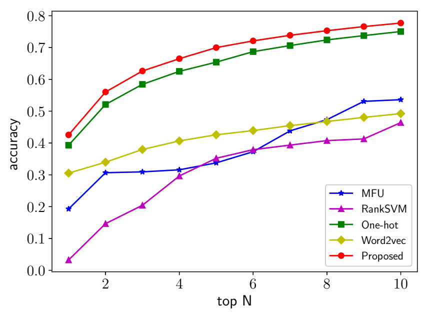

図 3 Transitions of Accuracy@N when N is varied

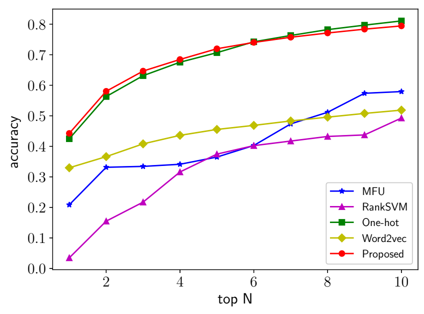

図 4 Transitions of Accuracy@N for existing Apps when N is varied

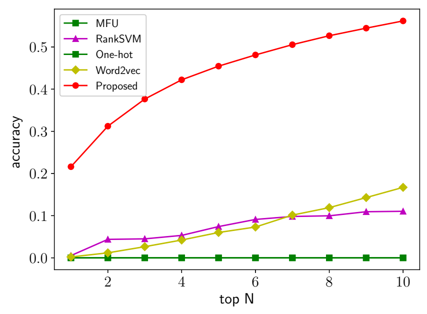

図 5 Transitions of Accuracy@N for unseen Apps when N is varied

next-use mobile Apps based on the App usage history of a target user using the training data collected from other users (source users). The proposed method makes use of the semantic representations of Apps to predict the usage of Apps that are not installed on the smartphones of source users by a target user. Our experiment that was conducted using the actual App usage data revealed that the proposed method achieved an accuracy of 56% (Accuracy@N; $N = 10$) while predicting the usage of Apps that were not installed on the smartphones of source users.

参考文献

[1] H. Cao and M. Lin, "Mining smartphone data for app usage prediction and recommendations: A survey," *Pervasive and Mobile Computing*, vol. 37, pp. 1 – 22, 2017.
[2] N. Natarajan, D. Shin, and I. S. Dhillon, "Which app will you use next?: Collaborative filtering with interactional context," in *the 7th ACM Conference on Recommender Systems (RecSys '13)*, 2013, pp. 201–208.
[3] X. Zou, W. Zhang, S. Li, and G. Pan, "Prophet: What app you wish to use next," in *the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication (UbiComp '13)*, 2013, pp. 167–170.
[4] A. Parate, M. Böhmer, D. Chu, D. Ganesan, and B. M. Marlin, "Practical prediction and prefetch for faster access to applications on mobile phones," in *the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '13)*, 2013, pp. 275–284.
[5] C. Sun, J. Zheng, H. Yao, Y. Wang, and D. F. Hsu, "Apprush: Using dynamic shortcuts to facilitate application launching on mobile devices," in *ANT/SEIT*, 2013.
[6] Z.-X. Liao, Y.-C. Pan, W.-C. Peng, and P.-R. Lei, "On mining mobile apps usage behavior for predicting apps usage in smartphones," in *the 22Nd ACM International Conference on Information & Knowledge Management (CIKM '13)*, 2013, pp. 609–618.
[7] C. Shin, J.-H. Hong, and A. K. Dey, "Understanding and prediction of mobile application usage for smart phones," in *the 2012 ACM Conference on Ubiquitous Computing (UbiComp '12)*, 2012, pp. 173–182.
[8] Z. X. Liao, S. C. Li, W. C. Peng, P. S. Yu, and T. C. Liu, "On the feature discovery for app usage prediction in smartphones," in *2013 IEEE 13th International Conference on Data Mining (ICDM '13)*, 2013, pp. 1127–1132.
[9] R. Baeza-Yates, D. Jiang, F. Silvestri, and B. Harrison, "Predicting the next app that you are going to use," in *the Eighth ACM International Conference on Web Search and Data Mining (WSDM '15)*, 2015, pp. 285–294.
[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
[11] T. Kudo, K. Yamamoto, and Y. Matsumoto, "Applying conditional random fields to Japanese morphological analysis," in *the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP '04)*, 2004, pp. 230–237.
[12] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26*, 2013, pp. 3111–3119.
[13] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.
[14] Z. C. Lipton, D. C. Kale, C. Elkan, and R. C. Wetzel, "Learning to diagnose with LSTM recurrent neural networks," *CoRR*, vol. abs/1511.03677, 2015. [Online]. Available: http://arxiv.org/abs/1511.03677
[15] T. Joachims, "Optimizing search engines using clickthrough data," in *the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '02)*, 2002, pp. 133–142.