Time-aware GCN: Representation Learning for Mobile App Usage Time-series Data

Kohsuke Kubota*
NTT DOCOMO, INC.
kousuke.kubota.xt@nttdocomo.com

Keiichi Ochiai
NTT DOCOMO, INC.
ochiaike@nttdocomo.com

ABSTRACT

With the expansion of smartphones, most users are using many apps (applications) on their smartphones. As the way users use their apps would reflect their personality, understanding their app usage is increasingly becoming an interesting problem. In order to understand their app usage, which consists of time-series data, we can use sequential models such as N-gram and long short-term memory (LSTM) for considering sequence characteristics. However, it is still challenging to reduce the impact of internal factors (*e.g.*, their feelings) and external factors (*e.g.*, notifications on their smartphones) on the differences in the order of apps used in a short term. In this paper, we propose a novel method for representation learning of app usage, called Time-aware Graph Convolutional Networks (*T-GCN*), to address the problem mentioned above. We evaluated the performance of T-GCN with the large-scale real-world dataset on the app usage prediction task. The results demonstrate that T-GCN achieves 3.6% higher accuracy than the LSTM model in accuracy@10.

CCS CONCEPTS

• **Information systems** → **Data mining**; • **Human-centered computing** → **Mobile computing**.

KEYWORDS

Smartphone App, App Usage Prediction, Graph Convolutional Networks

ACM Reference Format:
Kohsuke Kubota and Keiichi Ochiai. 2020. Time-aware GCN: Representation Learning for Mobile App Usage Time-series Data. In *KDD '20: The Second International Workshop on Deep Learning on Graphs: Methods and Applications, August 23–27, 2020, San Diego, CA*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/XXXXXXX.XXXXXXX

1 INTRODUCTION

Smartphones are rapidly spreading all over the world. As reported in [1], the number of smartphone users would reach 3.8 billion by 2021. On such prevalent smartphones, most users use numerous apps in various scenes such as watching videos and communication. According to [15], the average number of apps installed on a user's smartphone is 56, and some users have 150 apps. Users' app usage would reflect their personality, lifestyle, and context when observed for a relatively long-term. We can thus estimate users' personality and predict their behavior by using embeddings reflecting their state. Such estimate and prediction would enable optimal advertisement delivery and behavioral support for users. Understanding users' app usage on their smartphones, therefore, is increasingly becoming an interesting problem worth studying.

Previous studies have attempted to estimate users' state and predict app usage by creating features from app usage records. Ochiai et al. [19] detected that users have problems with their smartphones by app usage records. Shin et al. [22] predicted users' next app usage based on the naïve Bayes model and reported that the last app has essential influences on next app prediction. Because app usage records consist of time-series data, estimating users' state and predicting app usage are strongly dependent on the recent data of sequence. Therefore, it is becoming quite significant to consider the sequential relationship between app usage records.

For app usage records, which are time-series data, a straightforward method to represent the records is to use a sequential modeling technique such as N-gram [2] and the long short-term memory (LSTM) model [9]. However, the short-term order of users' app usage would be affected by internal factors (*e.g.*, notifications on their smartphones) and external factors (*e.g.*, their feelings). An example of internal factors is someone who looks at Twitter and Facebook every morning. One day, he/she launches Twitter first and then Facebook. Another day, he/she may launch Facebook first and then Twitter depending on his/her feelings. Users' long-term app usage patterns would almost remain unchanged, whereas short-term app usage patterns would change subject to their feelings. As for external factors, an example may be someone who has a behavioral pattern of using a news app and a note-taking app in this sequential order. Sometimes, however, he/she launches a messaging app to read a received notification before opening the note taking app. Many apps have notification functions and would cut in the sequences of apps. In such cases, the order of short-term app usage would be different from users' intentions. N-gram and LSTM, therefore, may not be suitable for extracting features of app usage because they are designed to accurately learn app usage patterns in the sequential order.

For generating app sequence representations, we assume cooccurrence within a certain time period is also important. In this paper, we propose a novel representation learning method for app usage, called Time-aware GCN (*T-GCN*). This method effectively extracts cooccurrences from app usage records while alleviating the short-term difference in the order mentioned above. First, we

KDD '20, August 23–27, 2020, San Diego, CA

ACM ISBN 978-1-4503-XXXX-X/YY/MM…$15.00
https://doi.org/10.1145/XXXXXXX.XXXXXXX

formulate a next app prediction as graph classification, and we apply GCN to the app usage graph for extracting graph representations. Next, we propose a method to construct the app usage graph considering temporal information from app usage records. Finally, we evaluate T-GCN through a large-scale real-world dataset on a next app prediction task. The experimental results show that T-GCN achieves 3.6% higher accuracy than the LSTM model in accuracy@10.

Our contributions are demonstrated as follows:

- We formulate an app usage prediction as graph classification.
- We propose a method to construct an app usage graph considering a time lag from app usage records and propose a novel representation learning method for app usage, called Time-aware GCN (*T-GCN*).
- We evaluate a large-scale real-world dataset of app usage on a next app prediction task. We demonstrate that T-GCN achieves 3.6% higher accuracy in accuracy@10 than the LSTM model, which is widely used in sequence modeling.

2 PROPOSED METHOD

In this section, we introduce T-GCN. We first give out the definitions for key terms used in this paper and then define the problem. Finally, we describe T-GCN in detail.

2.1 Preliminaries

Definition 1 (App Usage Sequence) We extract a set of apps over a sliding window with overlapping from a series of apps for each user ordered by time. We define n apps as the app usage sequence S. We can select either or both the number of apps and the time range for window extraction. Also, we allow the window setting to be in the same way or to be changed for each user.

Definition 2 (App Usage Graph) We construct the app usage graph $G = (V, E)$ from S. Let V be the set of apps in S, and E be the set of edges. We link the edges between apps within time lag Δt_{th}.

Definition 3 (Next App prediction as Graph Classification) Given a set of app usage graphs $\{G_1, \cdots, G_N\} \subseteq \mathcal{G}$ and a set of next app $\{Y_1, \cdots, Y_N\} \subseteq \mathcal{Y}$, we aim to learn the classifier f with training data (h_G, Y_G), where h_G denotes the representation of an entire graph G_N.

2.2 T-GCN

This subsection describes T-GCN in detail. We first construct an app usage graph from an app usage sequence. Because the order of app would change as discussed in Section 1, we deal with the app usage graph with an undirected graph. We link apps temporally close to each other in order to capture temporal usage features. Ochiai et al.[20] set an edge for two consecutively used apps. However, we speculate that we would acquire better representations by utilizing more temporally close apps information. Therefore, we set edges from all nodes within the time lag Δt_{th}. Specifically, we construct an app usage graph from n apps and edges between apps within Δt_{th}.

We formulate a next app prediction as graph classification and then learn representations of app usage graphs. Figure 1 shows the process of the next app prediction with T-GCN. We generate a representation of the entire graph with T-GCN, and we predict the next app using one-layer MLP with the representation as input.

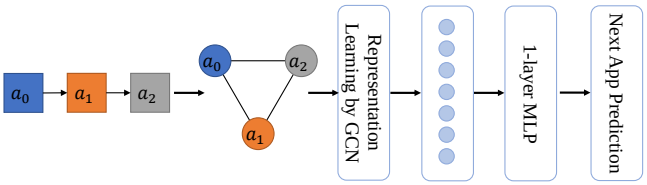

Figure 1: The process of next app prediction with T-GCN. We construct an app usage graph with apps a from an app usage sequence. We generate a representation of the entire graph with T-GCN, and we predict the next app using one-layer MLP with the representation as input.

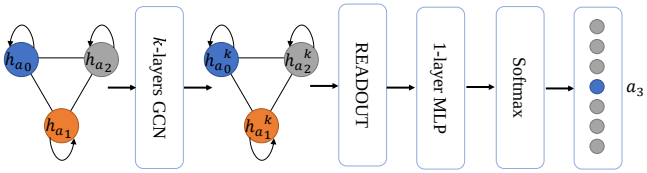

Figure 2: T-GCN architecture. First, we apply k GCN layers to an app usage graph for acquiring each node features h_a. Next, we generate a representation of a node in the entire graph with *READOUT* function. Finally, we get a score with linear transformation and predict a next app by obtaining a probability with softmax function.

We introduce how GCN generates representations for graph data. At training time, aggregate function is trained to aggregate representations of a node's neighborhood normalized by the number of its adjacent nodes. At inference time, the trained model is utilized to generate a representation of a node in the entire graph by repeatedly aggregating representations of its neighbors. After k iterations, a representation of the entire graph is acquired by summarizing representations of nodes. Formally, the procedure can be described by previous studies[12][23] as follows:

$$h_v^{(k)} = \text{ReLU}\left(W \cdot \text{MEAN}\left\{h_u^{(k-1)}, \forall u \in \mathcal{N}(v) \cup \{v\}\right\}\right) \quad (1)$$

$$h_G = \text{READOUT}\left(\left\{h_v^{(k)} \mid v \in G\right\}\right) \quad (2)$$

where h_v^k is the representation of node v at the k-th iteration, $\mathcal{N}(v)$ is a set of neighboring nodes to v, and MEAN denotes the element-wise mean.

Figure 2 shows T-GCN architecture. First, we apply k GCN layers to an app usage graph for acquiring each node feature. Next, we generate a representation of a node in the entire graph with *READOUT* function. Finally, we get score with one layer MLP and predict a next app by obtaining probability with softmax function. Besides, we adopt permutation-invariant functions for *READOUT* function to eliminate the effect of the order of apps in a graph. We compare both summing-up and averaging representations of all nodes, which are permutation-invariant functions.

3 EXPERIMENTS

In this section, we explain the experiments we conduct using a large-scale real-world app usage dataset to demonstrate the performance of T-GCN.

3.1 Dataset

We use a large-scale real-world dataset of app usage [1], which contains 1,000 users, and 2,000 apps [24]. The dataset was collected from a mobile cellular network in Shanghai. One record corresponds to an HTTP request or response by user access. Each record consists of the anonymized ID of each mobile device, timestamp, used app ID, etc.

We conduct preprocessing to the dataset in order to evaluate T-GCN. Specifically, we first delete the duplicate app ID within 10 seconds. Next, we remove the users with less than 100 records and those with more than 1,000 records for excluding outlier user cases. This dataset includes 252 users and 1,111 apps. Finally, we extract the top 100,000 records. We split the dataset into three sets: 80% train set, 10% validation set, and 10% test set. We also generate an app usage sequence from 10 apps over a sliding with a step size of three.

3.2 Performance Metric

We adopt *accuracy@k*, which was used by the previous work [6] for app usage prediction, to evaluate performance of models. We select the top k apps, calculating the probability of belonging to classes by models. If the correct classification appears among k selected classifications, we take it as one, otherwise zero, and let this value be *hit@k* for a single test case. accuracy@k is calculated by the average of the hit@k computed using all data as follows:

$$\text{accuracy@k} = \frac{\#hit@k}{|R_{test}|} \tag{3}$$

where $\#hit@k$ is the number of hits in the test set, and $|R_{test}|$ is the number of all test cases. We conduct an evaluation by setting $k = 5,\ 10,$ and 20.

3.3 Baselines

In order to evaluate the performance of T-GCN, we compare T-GCN with the two methods: LSTM and T-GCN with building app usage graphs using an existing method[20]. Because Almeida et al. [3] used LSTM in predicting human behavior, we apply the method to app usage prediction as a baseline model. Also, Ochiai et al. [20] also proposed a building app usage graph method, which sets an edge for two consecutively used apps. Thus, we select T-GCN with the method as the baseline model, called T-GCN with a single edge (T-GCN-SE).

3.4 Hyper-Parameter Setting

We extract an app usage sequence using a sliding window, where the three apps overlap, with both n apps and edges within Δt_{th}. We select the window setting for all users. Table 1 shows the parameters of models. We employ grid search to optimize performances and select the models of best accuracy for the validation set. We also train models with Adam optimizer[11], adopting early stopping with the patience of three [21].

[1] This dataset is available in http://fi.ee.tsinghua.edu.cn/appusage/.

Table 1: Parameters of models

Parameter	Value
Epochs	30
The number of layers	1
Hidden layer units	varied in {16, 32, 64}
Dropout rate	0.5
Learning rate for Adam optimizer	0.001
β_1 for Adam optimizer	0.9
β_2 for Adam optimizer	0.999
Weight decay for Adam optimizer	0
Batch size	32
READOUT function	varied in {mean, sum}

3.5 Results

We first examine how the Δt_{th} between apps for setting edges affect the performance of T-GCN. Figure 3 shows the accuracy@10 of T-GCN we obtained while changing Δt_{th} from 100 seconds to 3,600 seconds. We set the parameters as follows: hidden layer units to 64, and *READOUT* function to sum. As shown in Figure 3, setting edges within 1,800 seconds performs the best at the window because the curve of accuracy is converging. These results indicate that setting edges within a longer time than some width is essential to capture intricate app transition patterns. We therefore link the edges between apps within 1,800 seconds.

Table 2 shows the performance comparison with models averaged over three different random seeds. We link the edges between apps within 1,800 seconds for T-GCN-SE and T-GCN. As shown in Table 2, T-GCN achieves 1.7%, 3.6%, and 3.0% higher accuracy than LSTM, respectively. T-GCN also achieves 1.1%, 1.1%, and 0.8% higher accuracy than T-GCN-SE. These results show that T-GCN performs best in terms of all metrics: accuracy@5, accuracy@10, and accuracy@20. We conduct statistical hypothesis testing on accuracy. First, we utilize the F-test, which checks the equality of two variances of the hits. Since we could not reject the null hypothesis for a significance level of 0.05, we assume that the two variances are equal, and therefore we select the independent Student's t-test for checking if the difference in the accuracy between LSTM and others is statistically significant. The result of the t-test shows the notes of Table 2. As shown in Table 2, we reject the null hypothesis for a significant level of 0.05 between LSTM and T-GCN-SE, and for a significant level of 0.01 between LSTM and T-GCN.

4 RELATED WORK

We first review related work about app usage prediction in Section 4.1 and representation learning using deep learning in Section 4.2.

4.1 App usage prediction

Several studies have attempted to predict users' next app usage. Utilizing features of app usage records is essential for app usage prediction. Manually generating features of app usage for such contextual information [10, 14, 16, 22] is a general approach to the task.

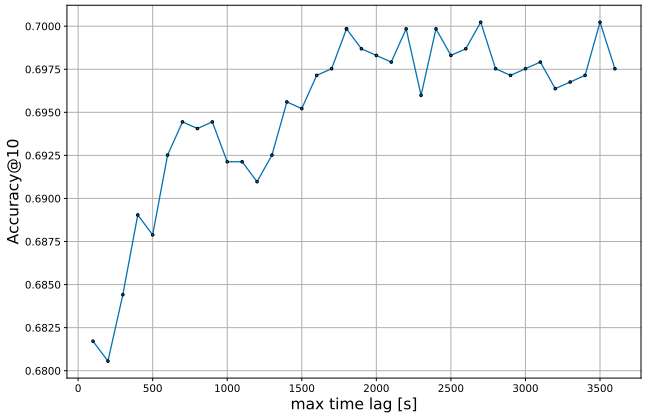

Figure 3: Accuracy@10 of T-GCN with different max time lag between apps

Table 2: Performance comparison with models

Model	Accuracy@5	Accuracy@10	Accuracy@20
LSTM	0.526	0.660	0.771
T-GCN-SE	0.534	0.685*	0.793*
T-GCN	**0.543**	**0.696****	**0.801****

Notes: t-test for checking the difference in accuracy between LSTM and the others.
* $p < 0.05$
** $p < 0.01$

Sequence modeling for app usage is important for app usage prediction. The methods based on the Markov model are proposed for modeling app transitions [13, 18]. The representation learning methods for sequence relationships based on Word2Vec and Doc2Vec are proposed [4, 25].

Graph-based methods are proposed for app usage prediction. Chen et al. [5] proposed a method based on the node and path similarity of a bipartite network of apps and users. Chen et al. [6] proposed a heterogeneous graph embedding algorithm to map contextual information such as location, time, and attribution into the common latent space.

4.2 Representation Learning using Deep Learning

Many studies have attempted to generate representations using deep learning in order to capture data features. For example, Convolutional Neural Networks (CNN) are used to extract representations for image [17]. LSTM is utilized to capture temporal relationships among sequential data [7].

Graph Neural Networks have been proposed to generate representations for graph data such as citation networks [12] and social networks [7]. Ochiai et al. [20] proposed a representation learning method for user state estimation by utilizing GCN. Deng et al. [8] also proposed a method predicting social events with GCN using word relation graphs.

Our study differs from previous works in terms of the following reasons: effective construction of graphs for generating representations from app usage records, which consist of time-series data.

5 CONCLUSION

This paper presented Time-aware GCN (T-GCN), which is a novel representation learning method for app usage based on GCN. We formulated an app usage prediction as graph classification. We proposed a method to construct the app usage graph considering temporal information from app usage records. We evaluated a large-scale real-world dataset of app usage on a next app prediction task. The evaluation validated the effectiveness of T-GCN compared with LSTM. The results demonstrated that T-GCN achieves 3.6% higher accuracy than LSTM in accuracy@10.

In future work, we plan to explore the following directions: We will 1) compare T-GCN with LSTM varying window size of sequence; 2) evaluate the app usage prediction task increasing the amount of data, because the dataset which we use may be small and lead to overfitting; 3) evaluate other tasks using app usage such as estimating users' stress and detecting smartphone usage trouble; 4) apply other time-series tasks such as predicting next items on e-commerce and visitor prediction on restaurants. Another direction would be to consider contextual information. As app usage is affected by when and where users use their apps, we would like to incorporate contextual information into T-GCN.

ACKNOWLEDGMENTS

REFERENCES

[1] 2020. *Number of smartphone users worldwide from 2016 to 2021.* https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/.
[2] Ryan Aipperspach, Elliot Cohen, and John Canny. 2006. Modeling human behavior from simple sensors in the home. In *International Conference on Pervasive Computing*. Springer, 337–348.
[3] Aitor Almeida and Gorka Azkune. 2018. Predicting human behaviour with recurrent neural networks. *Applied Sciences* 8, 2 (2018), 305.
[4] Ricardo Baeza-Yates, Di Jiang, Fabrizio Silvestri, and Beverly Harrison. 2015. Predicting the next app that you are going to use. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. 285–294.
[5] Hui Chen, Ke Yu, and Xiaofei Wu. 2018. Predicting App Usage from Mobile Internet Traffic Data Based on Node and Path Similarity. In *Proceedings of the 2nd International Conference on Telecommunications and Communication Engineering*. 245–250.
[6] Xinlei Chen, Yu Wang, Jiayou He, Shijia Pan, Yong Li, and Pei Zhang. 2019. CAP: Context-aware app usage prediction with heterogeneous graph embedding. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 1 (2019), 1–25.
[7] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).
[8] Songgaojun Deng, Huzefa Rangwala, and Yue Ning. 2019. Learning Dynamic Context Graphs for Predicting Social Events. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1007–1016.
[9] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[10] Ke Huang, Chunhui Zhang, Xiaoxiao Ma, and Guanling Chen. 2012. Predicting mobile application usage using contextual information. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. 1059–1065.
[11] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[12] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[13] Vassilis Kostakos, Denzil Ferreira, Jorge Goncalves, and Simo Hosio. 2016. Modelling smartphone usage: a markov state transition model. In *Proceedings of the 2016 ACM international joint conference on pervasive and ubiquitous computing*. 486–497.

[14] Z. Liao, P. Lei, T. Shen, S. Li, and W. Peng. 2012. Mining Temporal Profiles of Mobile Applications for Usage Prediction. In *2012 IEEE 12th International Conference on Data Mining Workshops*. 890–893.

[15] Zhung-Xun Liao, Shou-Chung Li, Wen-Chih Peng, S Yu Philip, and Te-Chuan Liu. 2013. On the feature discovery for app usage prediction in smartphones. In *2013 IEEE 13th International Conference on Data Mining*. IEEE, 1127–1132.

[16] Zhung-Xun Liao, Yi-Chin Pan, Wen-Chih Peng, and Po-Ruey Lei. 2013. On mining mobile apps usage behavior for predicting apps usage in smartphones. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 609–618.

[17] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. 2011. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International conference on artificial neural networks*. Springer, 52–59.

[18] Nagarajan Natarajan, Donghyuk Shin, and Inderjit S Dhillon. 2013. Which app will you use next? collaborative filtering with interactional context. In *Proceedings of the 7th ACM conference on Recommender systems*. 201–208.

[19] Keiichi Ochiai, Kohei Senkawa, Naoki Yamamoto, Yuya Tanaka, and Yusuke Fukazawa. 2019. Real-time on-device troubleshooting recommendation for smartphones. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2783–2791.

[20] K. Ochiai, N. Yamamoto, T. Hamatani, Y. Fukazawa, and T. Yamaguchi. 2019. Exploiting Graph Convolutional Networks for Representation Learning of Mobile App Usage. In *2019 IEEE International Conference on Big Data (Big Data)*. 5379–5383.

[21] Lutz Prechelt. 1998. Early stopping-but when? In *Neural Networks: Tricks of the trade*. Springer, 55–69.

[22] Choonsung Shin, Jin-Hyuk Hong, and Anind K Dey. 2012. Understanding and prediction of mobile application usage for smart phones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. 173–182.

[23] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).

[24] Donghan Yu, Yong Li, Fengli Xu, Pengyu Zhang, and Vassilis Kostakos. 2018. Smartphone app usage prediction using points of interest. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 4 (2018), 1–21.

[25] Sha Zhao, Zhiling Luo, Ziwen Jiang, Haiyan Wang, Feng Xu, Shijian Li, Jianwei Yin, and Gang Pan. 2019. Appusage2vec: Modeling smartphone app usage for prediction. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 1322–1333.