App Uninstalls Prediction: A Machine Learning and Time Series Mining Approach

Conference Paper · October 2017

DOI: 10.1007/978-3-319-70139-4_52

6 authors, including:

Jiaxing Shang
Chongqing University
68 PUBLICATIONS **800** CITATIONS

SEE PROFILE

Shangbo Zhou
Chongqing University
144 PUBLICATIONS **1,793** CITATIONS

SEE PROFILE

App Uninstalls Prediction: A Machine Learning and Time Series Mining Approach

Jiaxing Shang[1,2,*], Jinghao Wang[1], Ge Liu[1], Hongchun Wu[1,2], Shangbo Zhou[1,2], and Yong Feng[1,2]

[1] College of Computer Science, Chongiqng University, Chongqing, China
[2] Key Laboratory of Dependable Service Computing in Cyber Physical Society, Ministry of Education, Chongqing University, Chongqing, China
{shangjx,shbzhou,fengyong}@cqu.edu.cn, wuhc0217@163.com,
{jinghao_wang,liuge1229}@foxmail.com

Abstract. Nowadays mobile applications (a.k.a. app) are playing unprecedented important roles in our daily life and their research has attracted many scholars. However, traditional research mainly focuses on mining app usage patterns or making app recommendations, little attention is paid to the study of app uninstall behaviors. In this paper, we study the problem of app uninstalls prediction based on a machine learning and time series mining approach. Our approach consists of two steps: (1) feature construction and (2) model training. In the first step we extract features from the dynamic app usage data with a time series mining algorithm. In the second step we train classifiers with the extracted features and use them to predict whether a user will uninstall an app in the near future. We conduct experiments on the data collected from AppChina, a leading Android app marketplace in China. Results show that the features mined from time series data can significantly improve the prediction performance.

Keywords: App Uninstalls Prediction, Time Series, Machine Learning, Data Mining, Mobile Application

1 Introduction

Nowadays, with the rapid growing and developing of mobile devices, mobile apps are playing unprecedented important roles in our daily life. The enormous apps from marketplace provide us with a variety of facilities. For example, one can use the Twitter/Facebook/WeChat apps to communicate and share interesting things with our friends. People use the Taobao/Amazon apps to buy all kinds of stuffs online. Other applications include news reading, shopping, travelling, fit keeping, having fun, vehicle sharing, etc. The research of mobile apps has attracted many scholars in recent years [1–3].

The current research of mobile apps mainly focuses on mining app usage patterns or making app recommendations [4, 5]. However, little attention is paid to the research of people's app uninstall behavior. The motivations behind the

problem are two folds. For app developers, knowing whether their product will be largely uninstalled by users can help them make quick response to the market (e.g. upgrade the product, fix the bugs, etc). For app marketplace providers, if they know someone will give up an app in the near future, they may recommend alternative apps to the user in advance, providing better user experience.

Motivated by this, in this paper we study the problem of app uninstalls prediction based on a machine learning and time series mining approach. Our data is collected from AppChina[3], a leading Android app marketplace in China. Our approach consists of two steps: (1) feature construction and (2) model training. In the first step we extract features from the dynamic app usage data with a time series mining algorithm. In the second step we train classifiers with the extracted features and use them to predict whether a user will uninstall an app in the near future. Experimental results show that the features mined from time series data can significantly improve the prediction performance.

The rest of this paper is organized as follows. Section 2 introduces some related work. Section 3 gives detailed description about the data. We present our approach in Section 4. Section 5 shows the experimental results. Section 6 concludes this paper.

2 Related Work

A majority of the research works focus on app usage pattern mining and recommendation. For example, Pan et al. [4] developed a simple computational model to predict app installation by using a composite network. Shin et al. [5] studied the app usage pattern by predicting the next app to be used by a user and display it on the main screen. Tan et al. [6] proposed an algorithm which automatically determines and predicts each user's frequently-used applications in a fixed time slot. Liao et al. [7] proposed a framework to predict apps to be used regarding current device status. They designed a personalized feature selection algorithm based on minimum description length (MDL) and used kNN classifier for prediction. Xu et al. [8] developed an app usage prediction model that considers different factors such as user history, contextual information, etc. Kim et al. [9] predicted apps that are mostly likely to be used in a given moment with a conditional log-linear model. Lu et al. [10] predicted app usage with considerations of both physical location moving paths and virtual application usage paths simultaneously. Srinivasan et al. [11] designed a middleware running on the phone and discovers frequent co-occurrence patterns indicating which context events frequently occuer together. Baeza-Yates et al. [12] proposed a machine learning model to predicted the next app to be used and display it to the user once the phone is unlocked. Li et al. [13] conducted an empirical analysis of app usage behaviors on an Android app marketplace and studied two types of user behaviors: app management activities and app network traffic

Some research works focus on other aspects in studying mobile apps. Fu et al. [14] proposed WisCom: a systems that can analysis user ratings and comments

[3] http://www.appchina.com/

at three different levels of detail. Their method is able to discover inconsistencies in reviews and identify reasons why users like or dislike a given app. Ferdous et al. [15] predicted stress levels at workplace based on smart phone app usage patterns. Ma et al. [16] proposed Monkeydroid which automatically pinpoints whether an app would leak sensitive information. Ding et al. [17] proposed a malware detection method which extracts structural features from Android app function call graph and uses the features to identify malware.

3 Data Collection

Our data is collected from AppChina, a leading Android marketplace in China. The architecture of AppChina is shown in Figure 1.

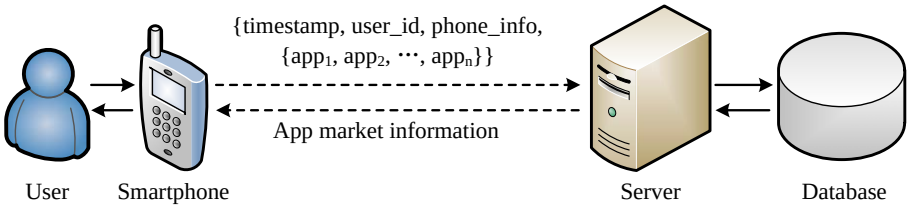

Fig. 1: The architecture of AppChina

After a user has installed the AppChina client on his/her smartphone, he/she can explore new apps, manage existing apps, get app market information from the AppChina server, etc. Meanwhile, each time the user launches the AppChina client, it will collect the user data and send the data to the server. The data contains the following information:

$$\{timestamp, user\_id, phone\_info, app\_list = \{app_1, app_2, \cdots, app_k\}\} \quad (1)$$

where $timestamp$ is the time when the data is collected, $phone\_info$ contains the smartphone information including phone model, screen resolution, dpi, android version, network connection, etc. $\{app_1, app_2, \cdots, app_k\}$ is the app list installed on the user's smartphone at the current timestamp. For each app, we have crawled its information from the AppChina website and build an app profile with the following information:

$$\{app\_id, app\_name, \#ratings, avg\_rating, \#downloads, category\} \quad (2)$$

where app_id and app_name are the id and name of the app, $\#ratings$ and $\#downloads$ are the number of app ratings and downloads, avg_rating is the average rating (ranging from 1 to 5) for the app.

4 Methodology

In this section we present our machine learning and time series mining approach, which consists of two steps: (1) feature construction and (2) model training.

4.1 Feature Construction

From the data as described in Section 3 we cannot directly tell which app is uninstalled or which one is newly installed. However, it is worthwhile to see that the app list $\{app_1, app_2, \cdots, app_k\}$ for each user is dynamic and changes over time, exhibiting time series nature [18]. By mining the time series data, we can infer the app uninstalls and installs. For example, if we recorded an app list $\{Facebook, WeChat, Twitter\}$ for Tom at time t_1, and recorded another app lsit $\{WeChat, Twitter, Weibo\}$ for him at t_2 ($t_1 < t_2$), then it is natural to see that Tom has installed a new app ($Weibo$) and uninstalled an old one ($Facebook$) during the time $(t_1, t_2]$. Without loss of generality, we assume the installation and uninstallation behaviors are performed at time t_2.

Algorithm 1 shows the time series mining algorithm which takes the user app data as input and outputs five-tuples, where each five-tuple defines the relationship between a user and an app. The format of a five-tuple is:

$$\{user\_id, app\_id, t_1, t_2, flag\} \tag{3}$$

where t_1 is the time when the app firstly appears in the user's app list, t_2 is the time when the app was last seen or disappeared from the user's app list, $flag$ is a boolean variable indicating whether the app was uninstalled by the user.

After the five-tuples are generated by Algorithm 1, we can extract features by doing statistical analysis on them. For example, for a five-tuple {*user_id*, *app_id*, t_1, t_2, $flag$}, we can use $t_2 - t_1$ to evaluate the time that the user keeps the app on his/her smartphone. We divide these features into three categories, i.e., *user features*, *app features*, and *correlated feature*, as shown in Table 1.

4.2 Model Training

After the features are extracted, the next step is to build training and testing sets. We have collected 50 days data, from June 1st, 2012 to July 31st, 2012. The data of the former 40 days are used to train the classifiers. We then use the classifiers to prediction whether a user will uninstall an app from his/her current app list within the following 10 days. If a user uninstalles an app, the example is marked as positive, otherwise it is negative. The training set contains 220,789 examples (110,181 positive and 110,608 negative), while the testing set contains 222,466 examples (111,016 positive and 111,450 negative).

Features: We use two groups of features to train our classifiers. The first group includes 8 features: {$\#apps$,$user\_avg\_time$, $\#ratings$, avg_rating, $\#downloads$, $\#users$, $category$, run_time}, the second group includes 12 features: {$\#apps$,

Algorithm 1 Time series data mining

Require:
Users' dynamic app usage data:
$\{timestamp, user\_id, phone\_info, app\_list = \{app_1, app_2, \cdots, app_k\}\}$
Ensure:
Five-tuples: $\{user\_id, app\_id, t_1, t_2, flag\}$ (indexed by $\{user\_id, app\_id\}$)
Initialize: five-tuple set $S = \Phi$
Group all users' dynamic app usage data according to $user\_id$.
for each user u **do**
  Sort u's app usage data according to $timestamp$.
  for each app usage data d of u **do**
    for each app a in $d.app\_list$ **do**
      if $\{u, a\}$ not in S **then**
        Add five-tuple $\{u, a, d.timestamp, d.timestamp, 0\}$ to S.
      else
        Five-tuple $tp = S.find(u, a)$
        $tp.t_2 \leftarrow d.timestamp$
      end if
    end for
  end for
  for each tp in $S.find(u)$ and $tp.flag = 0$ and $tp.app\_id \notin d.app\_list$ **do**
    $tp.t_2 \leftarrow d.timestamp$, $tp.flag \leftarrow 1$ //Mark the app as uninstalled
  end for
end for
return Five tuple set S

Table 1: Features extracted by time series data mining

Category	Feature	Description
User features	*model*	The model of the smartphone, e.g., Huawei
	os_version	The version of Android operation system
	resolution	The screen resolution of the smartphone
	dpi	The dpi of the spartphone
	#apps	The number of apps that the user has ever used
	#user_uni	The number of apps that the user has ever uninstalled
	user_un_ratio	The user's uninstallation ratio: *#user_uni/#apps*
	user_avg_time	The average time that an app stays on the user's smartphone
App features	*#ratings*	The number of ratings of the app
	avg_rating	The average rating of the app
	#downloads	The number of downloads of the app
	category	The category of the app
	#users	The number of users who have ever used the app
	#app_uni	The number of users who have ever uninstalled the app
	app_un_ratio	The app's uninstallation ratio: *#app_uni/#users*
	app_avg_time	The average time that a user keeps the app
Correlated feature	*run_time*	The time that the app stays on the user's smart phone

#user_uni, *user_un_ratio*, *user_avg_time*, *#ratings*, *avg_rating*, *#downloads*, *#users*, *#app_uni*, *app_un_ratio*, *category*, *run_time*}. The difference between the two feature groups is that the second feature group considers the uninstallation ratio while the first group does not.

Classifiers: We consider 8 classifiers in this paper, i.e., (1) Linear Regression (**LR**), (2) Support Vector Machine (**SVM**), (3) Naive Bayes (**NB**), (4) Decision Tree (**DT**), (5) k-Nearest Neighbors (**kNN**, (6) Random Forest (**RF**), and (7) Gradient Boosting Decision Tree (**GBDT**). All the classifiers are implemented with the Python scikit-learn [19] toolbox.

5 Experimental Results

In this section we will introduce the experimental study, including the evaluation metrics, experimental environment and the results.

Evaluation metrics: We evaluate the prediction performance of our approach with four metrics, i.e., (1) **Precision**, (2) **Recall**, (3) **F1-Score**, (4) **Accuracy**.

Experimental environment: The experiments are carried out on a computer with 2.5GHz AMD A10-5750M CPU and 8GB memory. Our code is implemented in Python 2.7 programming language.

Results of different classifiers: The prediction performance of different classifiers is shown in Figure 2. Most of the classifiers perform better in terms of precision than recall. Specifically, the SVM classifier exhibits the best performance in precision on both the two feature groups, while the Naive Bayes performs the worst in that metric. When evaluated by the overall prediction performance, i.e., f1-score and accuracy, the GBDT classifier performs best in the 8 feature group while the GBDT, LR and SVM classifiers perform best on the 12 feature group. In general, the GBDT classifier exhibits better performance than other classifiers.

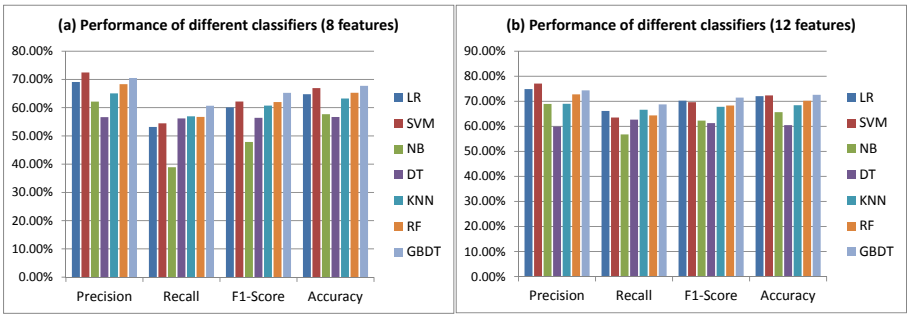

Fig. 2: The prediction performance comparison of different classifiers

Performance of different feature groups: The prediction performance of different feature groups is shown in Figure 3, from which we see that for all the classifiers and evaluation metrics, the 12 feature group significantly outperforms the 8 feature group. The most significant difference appears on Figure 3(b), i.e., the performance evaluated by recall. For example, the recall value of the Naive Bayes classifier increase from 0.389 to 0.568, about 46% increment. Given that the 12 feature group includes the unintallation ratio features which are mined through our time series data mining algorithm, the results indicate that our machine learning and time series mining approach can significantly improve the app unintalls prediction performance.

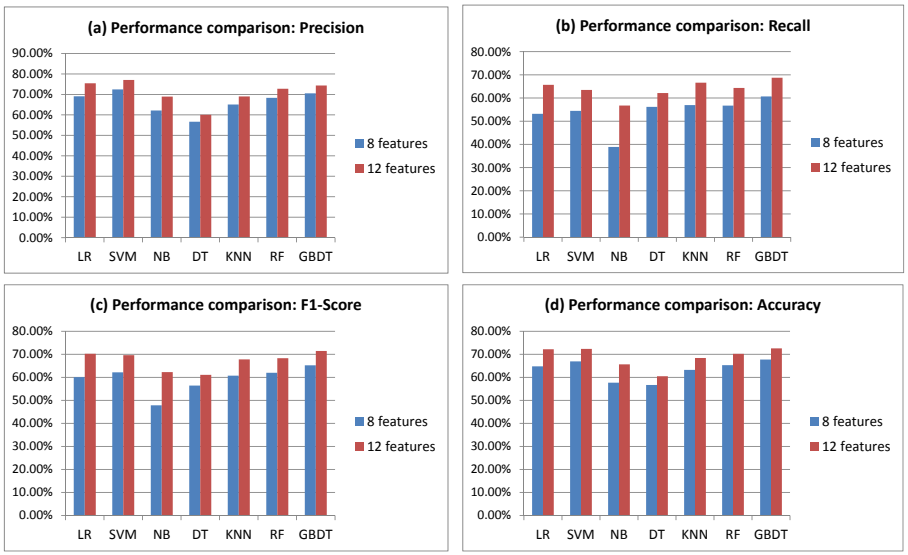

Fig. 3: The prediction performance comparison of different feature groups

6 Conclusion

In this paper, we study the problem of app uninstalls prediction based on a machine learning and time series mining approach. Our approach consists of two steps: (1) feature construction and (2) model training. In the first step we extract features from the dynamic app usage data with a time series mining algorithm. In the second step we train classifiers with the extracted features and use them to predict whether a user will uninstall an app in the near future. We conduct experiments on the data collected from AppChina, a leading Android app marketplace in China. Results show that the features mined from time series data can significantly improve the prediction performance.

Acknowledgements. This work was supported in part by National Natural Science Foundation of China (No. 61702059), China Postdoctoral Science Foundation (No. 2017M612913), Fundamental Research Funds for the Central Universities of China (No. 106112016CDJXY180003), Graduate Student Research and Innovation Foundation of Chongqing City (No. CYS17024), Frontier and Application Foundation Research Program of Chongqing City (No. cstc2017jcyjAX0340, cstc2015jcyjA40006), Social Undertakings and Livelihood Security Science and Technology Innovation Funds of Chongqing City (No. cstc2017shmsA20013).

References

1. Rehman, M., Liew, C., Wah, T.: Frequent pattern mining in mobile devices: A feasibility study. In: 6th IEEE International Conference on Information Technology and Multimedia (ICIMU), pp. 351-356. IEEE Press, Putrajaya (2014)
2. Rehman, M., et al.: Mining personal data using smartphones and wearable devices: A survey. Sensors 15, 4430-4469 (2015)
3. Cao, H., Lin, M.: Mining smartphone data for app usage prediction and recommendations: A survey. Pervasive and Mobile Computing 37, 1-22 (2017)
4. Pan, W., Nadav, A., Alex, P.: Composite social network for predicting mobile apps installation. In: 25th AAAI International Conference on Artificial Intelligence (AAAI), pp. 821-827. AAAI, San Francisco (2011)
5. Shin, C., Hong, J.H., Dey, A.K.: Understanding and prediction of mobile application usage for smart phones. In: 14th International Conference on Ubiquitous Computing (UbiCom), pp. 173-182. ACM, Pittsburgh (2012)
6. Tan, C., Liu, Q., Chen, E., Xiong, H.: Prediction for mobile application usage patterns. Nokia MDC Workshop 12, (2012)
7. Liao, Z.X., Li, S.C., Peng, W.C., Philip, S.Y., Liu, T.C.: On the feature discovery for app usage prediction in smartphones. In: 13th IEEE International Conference on Data Mining (ICDM), pp. 1127-1132. IEEE Press, Dallas (2013)
8. Xu, Y., Lin, M., Lu, H., Cardone, G., Lane, N., Chen, Z., Campbell, A., Choudhury, T. : Preference, context and communities: a multi-faceted approach to predicting smartphone app usage patterns. In: 17th ACM International Symposium on Wearable Computers (ISWC), pp. 69-76. ACM, Zurich (2013)
9. Kim, J., Mielikäinen, T.: Conditional log-linear models for mobile application usage prediction. In: European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD), pp. 672-687. Springer, Nancy (2014)
10. Lu, E.H.C., Lin, Y.W., Ciou, J.B.: Mining mobile application sequential patterns for usage prediction. In: IEEE International Conference on Granular Computing (GrC), pp. 185-190. IEEE Press, Hokkaido (2014)
11. Srinivasan, V., Moghaddam, S., Mukherji, A., Rachuri, K.K., Xu, C., Tapia, E.M.: Mobileminer: Mining your frequent patterns on your phone. In: ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiCom), pp. 389-400. ACM, Seattle (2014)
12. Baeza-Yates, R., Jiang, D., Silvestri, F., Harrison, B.: Predicting the next app that you are going to use. In: 8th ACM International Conference on Web Search and Data Mining (WSDM), pp. 285-294. ACM, Shanghai (2015)

13. Li, H., Lu, X., Liu, X., Xie, T., Bian, K., Lin, F.X., Mei, Q., Feng, F.: Characterizing smartphone usage patterns from millions of Android users. In: ACM Internet Measurement Conference (IMC), pp. 459-472. ACM, Tokyo (2015)
14. Fu, B., Lin, J., Li, L., Faloutsos, C., Hong, J., Sadeh, N.: Why people hate your app: Making sense of user feedback in a mobile app store. In: 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 1276-1284. ACM, Chicago (2013)
15. Ferdous, R., Osmani, V., Mayora, O.: Smartphone app usage as a predictor of perceived stress levels at workplace. In: 9th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth), pp. 225-228. IEEE Press, Istanbul (2015)
16. Ma, K., Liu, M., Guo, S., Ban, T.: MonkeyDroid: detecting unreasonable privacy leakages of android applications. In: 22nd International Conference on Neural Information Processing (ICONIP), pp. 384-391. Springer, Istanbul (2015)
17. Ding, Y., Zhu, S., Xia, X.: Android malware detection method based on function call graphs. In: 23rd International Conference on Neural Information Processing (ICONIP), pp. 70-77. Springer, Kyoto (2016)
18. Yu, S., and Abraham, Z.: Concept Drift Detection with Hierarchical Hypothesis Testing. In: Proceedings of the 2017 SIAM International Conference on Data Mining, pp. 768-776. SIAM, Texas (2017)
19. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. Journal of Machine Learning 12, 2825-2830 (2011)