Analysis and Prediction of Application Usage in Android Phones

Shreenath Acharya, Asha Shenoy, Macwin Lewis, Namrata Desai
St. Joseph Engineering College
Mangaluru, INDIA
shree.katapady@gmail.com, asha.92@gmail.com,macwin.lewis@yahoo.com, namratadesai92@gmail.com

Abstract - **Predictive Analytics analyze the present and the historical informations and make future predictions utilizing data mining or machine learning techniques. Predictive models usually check for some patterns and relationships leading to certain behaviours based on the dependent variables. This paper proposes a mechanism named Analysis and Prediction of Application Usage (APAU) in Android Phones for providing recommendations to a smart phone user while selecting applications of their interest like mail checking, messaging and making calls. APAU mainly focuses on identifying usage patterns and investigating the human behaviour during application selections by extracting the generic behavioural patterns to predict and provide useful set of recommendations.**

Keywords – **Trigger, Follower, Machine Learning, Recommendations, Notifications, Prediction, Pattern**

I. INTRODUCTION

It is observed that users have a tendency to use applications in a sequence and also often repeat the same sequence of applications. In these cases, it would be very beneficial to the users when provided with a recommendation system suggesting the most likey next applications they will be interested to use. The user after accepting the application can directly switch to that application, thus avoiding wastage of time and effort. This system initially observes and records the user behaviour in terms of application usage and further recommends applications based on the usage patterns. The system records contextual and activity cues, including location, application usage, calling behaviour and can be considered as both input and output of a prediction framework, in which the future values are predicted based on the current context.

The main aim of APAU is to make application switching in an Android phone smooth, fast and efficient by helping the user conserve time and effort, which could otherwise be wasted on work other than the actual usage. It intends to build a recommendation system that will suggest applications based on the user's usage patterns. The benefits of this work is to make the user to be able to switch between applications faster, reduce effort and maintain focus while performing a task.

The remainder of the paper as follows. Section II explains the related work, section III describes the overall system architecture, section IV explains the system design, section V depicts the implementation, section VI describes the results analysis followed by the conclusion in section VII.

978-1-4673-9745-2 ©2016 IEEE

II. LITERATURE REVIEW

There are many papers from researchers related to the faster application switching in mobile phones.

Abhinav Parate et. al [1] have developed a prototype named Pre-fetch Faster Access to Applications on Mobile Phones(PREPP) for mitigating the effect of network content retrieval time by accurate prediction of application usage and pre-fetching for improving the user interface. Through user and prediction algorithm studies they have shown that the application can be predicted fastly without any modifications to the applications or OSs. The main interface of the application is through the use of a widget on the home screen which displays the set of applications that are being pre-fetched and anticipated to be used next. Concepts such as ‘time to live’ are used along with the context based parameters such as time of day and battery percentages to find the appropriate applications to be pre-fetched. PREPP is a prototype and is not implemented in the real world. It is still under research for android and a few applications are trying to utilize the concept of PREPP. One of the biggest challenges for PREPP is the prelaunch of the applications in android without the system killing the services along.

Tingxin Yan et. al [2] proposed a mechanism known as Fast App Launching for Mobile Devices Using Predictive User Context (FALCON) to overcome the problem of slow app launch. It predicts the application launches before they occur based on the user location and the temporal patterns. The usage of data analysis and cost benefit algorithms in this approach has resulted in stronger predictive performance with lower runtime overhead.

FALCON was implemented in Windows 7.5 phone OS as a patch. While reducing time to launch, it also had side effects of consuming extra memory and slowing down the interface. FALCON utilized online servers to calculate the temporal access patterns and contexts to predict application usage; which requires sharing of information to online servers which is unsafe. Currently FALCON is undergoing further research and is not being actively used in any mobile OS. The biggest challenge for FALCON will be to reduce its online dependency.

Garlan et. al [3] have proposed a project named Aura on pervasive computing based on the observation that the bottleneck in modern day computing is the limited resource of human attention. Aura created a greater effectiveness compared to other systems today with hardware technologies such as laptops, handhelds, wireless communication and readily available component software technologies. Aura is based on the belief that the whole is much greater than the sum of parts. The two main capabilities of Aura are supporting user mobility and shielding users from variations in resource availability. It provides the facility of reconfiguration when a user moves from one environment to another and provides a shared white board facility for the user to communicate their ideas through multiple channels support.

A framework developed by Zhung-Xun Liao et. al [4] predicts the mobile applications most likely to be used based on the current status of a smartphone. This framework is a prerequisite for power management, intelligent user experience and faster application launching in smart phones. Based on the analysis of collected log data of real applications usage, it has been estimated that the average number of Apps in a user's smartphone is around 56. As per the increase in the number of Apps, the time taken by the user to use their Apps increase. To ease the inconvenience of searching for Apps and to reduce the delay in launching Apps, it uses the concept of predicting the applications the user actually needs. The IOS and Android systems list the most recently used (MRU) Apps to help users re-launch Apps, but it only works for those Apps which would be re-launched within a short period.

Gediminas Adomavicius et. al [5] provided an insight into the field of recommender systems and described the current generation recommendation methods as content - based, collaborative and hybrid recommendation approaches. They highlighted the limitations of current recommendation methods and discussed possible extensions for improvements on understanding of users and items, integrating contextual information, supporting ratings on multiple criteria and providing more extensible types of recommendations.

Manish Gupta et. al [6] have proposed a method for pattern discovery using sequential data mining capable of implementing efficient recommender systems based on previously observed patterns. This system also helped in improving usability of systems and detection of the events for making strategic product decisions. The usage analysis statistics is based on page access frequency and finding common traversal paths through a web site. This method results in more precision but it may lack in coverage levels when the goal is to generate as many recommendations as possible. In applications such as web pre-fetching this method suits well since the primary goal is to predict user's immediate actions.

III. PROPOSED SYSTEM ARCHITECTURE

Analysis and Prediction of Application Usage is an application that enhances a typical android phone usage. The basic architecture diagram in the Fig.1 illustrates various entities and the interactions within the system. It gives an overall idea of developing the recommendation system and the basic interactions required to accomplish the task of prediction. The system includes several stages of operations which have the following functionalities:

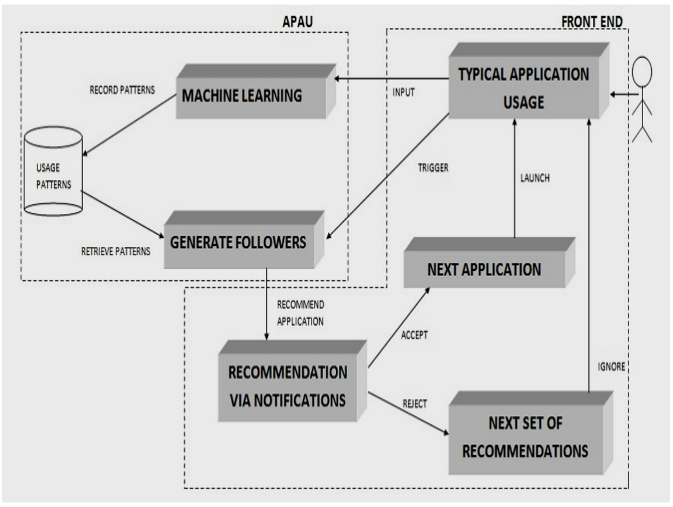

Fig. 1 Architecture Diagram

A. *Initial setup*

The application initially provides recommendations based on usage patterns obtained from a survey. These patterns are stored in the database and are used to make initial predictions.

B. *Machine learning*

The application observes user's usage patterns and makes changes to the initial set of patterns. Machine learning helps the application understand and record the various patterns in the user's usage behaviour.

C. *Modification of patterns*

The patterns would change dynamically over time and would then serve as the base for future recommendations. Eventually, the application would mimic the user's usage patterns to an accurate level, predicting the next application the user would want to use. Thus, providing a personalised version of the application to each user.

D. *Recommendations via notifications*

The user would be prompted for the next applications he is most likely to use via notifications, thus providing easy access to these recommendations even while using other applications. It is not necessary for the user to return to the home screen to access these recommendations.

E. *Application priority*

The top 3 applications most likely to be used are suggested to the user. Applications are ranked priorities while recording user-specific patterns and dynamically changing the initial set of patterns.

F. *Accept/reject recommendation*

The user would be provided an option to accept or reject the recommendations. On accepting the recommendation, there is an application switch to the chosen application. If the user rejects the recommendation, the system would suggest the next 3 applications based on the priority ranks. In this fashion, a maximum of 12 applications in 4 sets could be suggested to the user, but the attempt is made to rightly predict the next application in the first set of suggestions.

G. *Ignore recommendations*

The user can also choose to ignore the recommendations as they appear as notifications only in the notification bar. The user can continue working without being interrupted by APAU application in any manner.

H. *Contextual prediction*

Parameters like time, location and battery state are considered while predicting applications that the user is likely to use.

I. *Optional disabling of application*

An option to put the application to sleep for a certain time range in the day is provided to the user. The application automatically resumes after this sleep time interval.

IV. SYSTEM DESIGN

When a user opens an application on the phone, the machine learning algorithm identifies the application as a trigger. Machine Learning now performs a database look up to find patterns associated with the trigger. If found, it fetches the patterns from the database and generates the followers for the trigger. Otherwise, Machine learning algorithm stores this application as a new trigger in the database and attempts to build new patterns for it. In this attempt, it fetches high priority applications from the database and generates appropriate follower list. Machine Learning forwards the follower list to the Recommendation System.
The Recommendation system then divides the follower list into sets and sends them in batches to the notification tray in order of application priority and displays it to the user. The user then may choose to accept or reject the recommendation as shown in Fig2.
The user can also ignore recommendations in the sense that the user need not check the notification tray for recommendations if he doesn't wish to do so. In this case, APAU will not interrupt the regular application usage of the user in any manner. If the user chooses to accept a recommendation, the application is opened and a preference update made in the database.
On rejection of a set of recommendations, the user is prompted with the next set of recommendations. Preference updates in the database happen even when rejection of recommendations is encountered. This helps APAU understand the user's pattern of application usage better.
Once an application is opened after the acceptance of recommendation by the user, the machine learning algorithm would then treat this application as a trigger and generate followers for it trying to form a continuous cycle. Even though acceptance of recommendations is important to the system to fulfil its purpose, a rejection also proves to be equally beneficial as it helps the system in understanding the user's patterns and preferences clearly.

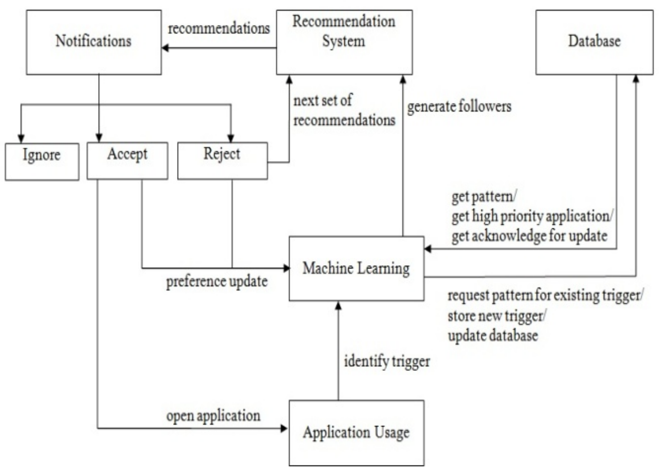

Fig. 2 Flow Diagram

A. *Hardware Interfaces*

The application will run on an Android mobile device or an Android emulator. It does not require other hardware devices or interfaces.

B. *Software Interfaces*

1) Input: The input to the system is application usage behaviour of a user. This is recorded by the application itself. The only input expected from the user is to accept ot reject the recommendations displayed or choose from options provided (Ex: Application sleep time).

2) Output: Based on the usage patterns collected, the system would output recommendations based on context**.**

3) Operating System: The Android operating system, specifically version 4.1 (Jelly Bean) and above will be suitable for running this application/software.

C. *Functional Requirements*

1) Retrieving Input: The software receives two types of inputs: Options being set with appropriate values and the recommendation being accepted or rejected.

2) Real-Time Processing: The software takes the input, processes the data and displays the real-time output. The application always runs in the foreground collecting and analysing usage patterns. On receiving a trigger, the system generates followers and uses these followers to recommend applications to the user. These operations have to take place in real-time.

3) Output: The software must display real time recommendations in the form of simple notifications in the notification tray. This output must be clear and easy to understand.

D. Performance Requirements

1) Real-Time: The software will provide up-to-date information and displays recommendations in real time. The recommendation system takes into account the appropriate context in order to predict applications. The system should not experience any lags and should always be available.

2) System Resource Consumption: The consumption of the device resources should not reach an amount that renders the mobile device unusable. The application should be capable of operating in the background and not hinder the performance of foreground applications.

V. IMPLEMENTATION

APAU has been implemented as an Android application that is compatible with Android OS versions- Jelly Bean and higher.

A. Initial Survey

An application which learns the preferences of the user generally has a lengthy training set or period during which the application is run at a below optimal level. The user preferences vary with each Android device, version and age group. During this initial training period the application can be a nuisance to the user and may be susceptible to errors. To minimize the impact of this training period on usability of the application, an initial dataset was introduced. This dataset was arrived upon after conducting a survey on a small set of android users. The number of applications made for android is too large, and the applications installed on each android device vary with user preferences. As a result this initial survey was conducted based only on common system applications which are present in each android device.
The users surveyed upon were given a set of applications that they would be using and their opinion on three applications they might want to use next were noted. A certain pattern emerged from this survey and we were able to distinguish popular choices for each application. The three most common choices for an application were noted as the initial set of followers for that application.

B. Algorithms for different module

1) procedure machinelearning

```
store current application as trigger
call trigger module
if (user responds)
call database module
end procedure machinelearning
```

2) procedure triggerAndFollower

```
if (trigger exists)
get followers
call recommendation module
else
call databasemodule
get high priority app
call recommendation
end triggerAndFollower
```

3) procedure recommendation

```
get pattern from Data base
display notification
if(accept)
update Database
open application
else
wait for next trigger
end procedure recommendation
```

4) procedure for databaseModule

```
if (trigger search successful)
return pattern
else if (new trigger identified)
store new pattern
return pattern
else if (update request)
update preference field
end procedure database Module
```

VI. RESULTS ANALYSIS

A. RAM Testing

In android, all services running are kept in the background utilising the RAM of the device. If the space available in the RAM is perceived to be extremely low by the device, then Android kills low priority applications and services to free up space in the RAM. Hence, more the capacity of the RAM, more smoothly a service can be executed without the fear of being stopped unexpectedly.
However, if a service is stopped unexpectedly by the system, then on availability of memory, the service is automatically restarted.
Testing was conducted based on the performance of different devices with varying sizes of RAM, with APAU installed. The devices were simulated using the standard ADT emulator.
Different RAM values were used for testing keeping the screen size set at 4.7 inches as well as the Android version set at Android 4.4 KitKat, using a virtual emulating environment. The obtained results after the variation of RAM capacities for the developed application are as depicted in TABLE I.

TABLE I. ANALYSIS OF RAM REQUIREMENTS

Size of RAM (in MB)	Speed in Switching apps	Chances of service being stopped
512	Slow (2 sec - 3sec)	High (if more than 300MB RAM used)
768	Medium(1sec - 2sec)	High (if more than 500MB RAM used
1024	Fast(< 1sec)	Low (if more than 750MB RAM used)

B. Android Version Analysis

The Android platform has different versions of operating systems. Each operating system has a different RAM managing mechanism along with different handlers to threads and API calls. The android interface also changes with each version, being optimized for the right balance of performance and battery life.

TABLE II. BEHAVIOUR OF APAU ON DIFFERENT ANDROID VERSIONS

Version	**Observed Behaviour**	**Remarks**
2.2 (Froyo) and 2.3 (Gingerbread)	Notifications are not displayed properly	The custom notifications API was introduced in a later version
4.0 (Ice Cream Sandwich)	Speed is slow	The process management system is old and deprecated, hence speed observed is slow as compared to higher versions
4.1,4.2,4.3 (Jelly Bean)	Optimal functionality	None
4.4 (KitKat)	Optimal functionality	None

A test was conducted to observe the behaviour of APAU on different versions of Android. The results were noted after running APAU on virtual devices run by the emulator. The RAM was set at 1GB and screen size at 4.7 inches running a virtual ARM v7 Cortex processor. The obtained results are shown in the TABLE II.

C. Limitations and their consequences

Although APAU has been successful in benefitting the user by providing an easy interface to application switching, there may be certain limitations which could be taken care by little effort/patience from the user while utilizing it in real time as described below.

As APAU launches a service that runs in the background continuously, battery consumption might be affected. Of course, the application's benefits overweigh this minor issue. When battery levels are low or battery consumption is of concern, the application can be temporarily shut down; the application can be re-launched later.

As a typical notification can only accommodate 3 buttons, more than 3 applications cannot be recommended for an instance. This limitation is due to the Android system allowing only up to 3 buttons on a single notification.

Another limitation is that it might take a little extra time for APAU to initially understand application usage of the user and to recommend the right application patterns. The user is expected to be patient during the training period of the application.

VII. CONCLUSION

In present day's software engineering, the software has to be simple, user friendly and must be able to incorporate any further improvement. Analysis and Prediction of Application Usage is simple and was built with the intent to help users in faster application usage by providing shortcuts to applications that the user would want to use next. It saves time in navigation and also helps the users in performing tasks simultaneously without having to return to the home screen.

The application keeps track of preferences of the applications being used and the order in which the users use them. This provides fairly accurate estimates of which application the user would require next. By using concepts of machine learning and storing values in a database, APAU has been made efficient enough to process the system conditions and return recommendations at an optimal speed. The main objective of APAU, i.e. providing a set of recommendations of applications for the user to use next is successfully implemented while minimizing the load on system resources.

APAU can be improvised by taking into consideration parameters like time and location so that recommended applications are more relevant to the user's context of use. It can also be made much more effective by pre-launching the high priority followers so as to further reduce the inter-application time switching.

REFERENCES

[1] A. Parate, M. Bohmer, D. Chu, D. Ganesan and B. M. Marlin, "*Practical prediction and pre-fetch for faster access to applications on mobile phones*", UbiComp, pp 275-284, Zurich, Switzerland, 2013.

[2] T. Yan, D. Chu, D. Ganesan, A. Kansal and J. Liu, "*Fast App Launching for Mobile Devices using Predictive User Context*"**,** MobiSys '12 Proceedings of the 10th International Conference on Mobile Systems, Applications and Services, pp 113-126, ACM New York, ISBN: 978-1-4503-1301-8, 2012.

[3] D. Garlan, D. P. Siewiorek, and P. Steenkiste. "*Project Aura: Toward distraction-free pervasive computing*", IEEE Pervasive Computing, 1: 22–31, 2002.

[4] Z. Liao, S. Li, W. Peng, Philip and Yu, "*On the Feature Discovery for App Usage Prediction in Smartphones*", CoRR abs/1309.7982 (2013).

[5] G. Adomavicius, and A. Tuzhilin, "*Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions*", IEEE Transactions on Knowledge and Data Engineering, Vol. 17, No. 6, pp 734-749, 2005.

[6] M. Gupta and J. Han, "*Applications of Pattern Discovery using Sequential Data Mining: Applications and Studies*", 1-23, IGI Global, 2012.