International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

RoloKnow: An Analytic Framework for Mobile Applications and Games

Aman Nigam[1], Hrishikesh Pallod[2], Saurabh Abhale[3], Sidhesh Badrinarayan[4]

[1,2,3,4] Pune Institute of Computer Technology, University of Pune, Pune, India

Abstract: ***The world has moved to smartphones and retaining users on applications is of prime importance to developers. When it comes to engagement and retention, being able to segment users by how often they visit the application, and how long their sessions are, can help developers understand what is working within the application. Techniques like A/B testing and custom event tracking can help consociate the user with the developer. It will also help them analyze the behavior of various users to find the most engaged user and see the commonalities of these user segments from historical data. The usage data is tracked from android devices and is sent to MongoDB which is a NoSQL document oriented database. This data is selectively sent to Hadoop for processing. Final results are stored in MySQL. These results can be viewed by developers through a web interface which has powerful data visualization tools. The developers will also have the freedom to perform application specific analytics.***

Keywords: Smartphones, Retention, A/B testing, MongoDB, Hadoop, MySQL, Data visualization.

1. Introduction

People today are dependent on smartphones owing to the various capabilities that these portable devices offer. Applications make these devices useful. Hence, making simple and powerful applications is of prime importance to application developers across all platforms.

The problem is the lack of a bridge between the developer and the user. A user cannot call or email the developer every time he/she faces a problem with an application. A popular application will have millions of users and it would be impossible for the developer to individually cater to all requests. Hence, a developer needs to follow a data driven approach in application development [1].

RoloKnow is a framework developed to overcome this problem of application developers by bridging the gap between them and their users. Using RoloKnow, developers will get a virtual entry into their users' phones and the way they use the applications, helping developers understand what is working and what is not. A developer would know exactly how popular his application is in the United States, sitting miles away in India or say Japan. RoloKnow has the potential to change the entire process of application development, deployment and management.

2. The 3-Step Solution

RoloKnow provides a 3-Step solution to maximize an application's usability and popularity:

Step 1: Check if the application performing well using RoloKnow's retention reports.
Step 2: If the retention reports are unfavorable, identify the problem using event correlation (funnels) and event-property statistics.
Step 3: Once the probable solutions for the problem are decided, using RoloKnow's A/B testing feature, the developer can let the users decide which solution they prefer.

3. Retention

3.1 Importance

Download statistics cannot be the sole parameter for determining the popularity of an application. A user may download an application but never use it more than once. Retention gives the developer a clear picture of how often users comeback and engage with the application [2]. For any two related activities, retention can give the statistics about how and when a person does one activity and then the other.

3.2 Retention features in RoloKnow

1) *Hourly Active Users (HAU):* This tells us at what hour of the day, an application is used the most. Do your users use the application while having dinner, or do they use it early morning as soon as they wake up. An amateur developer may not realize the importance of such statistics, but a professional will. HAU report shows the developer how the application usage varies over 24 hours. This has been implemented by processing fresh event data every hour to find out the count of unique users.
2) *Daily Active Users (DAU):* This report visualizes how many unique visitors engage with the application per day. DAU can help the developer understand the immediate response to an application update. The daily active users count is not the sum of all the hourly active users. This means if a user uses the application multiple times in 24 hours, it will be counted as only one unique instance for the day. In case of hourly active users, the window of unique user count is limited to just an hour. The script which runs every hour for Hourly Active Users, also maintains a count of new users which have visited that day. This has been done by maintaining an intermediate data structure, which keeps track of the last activity of all unique devices. By comparing the timestamps, new user count for that hour can be calculated. The sum of this new user count for every hour is the daily active user count for

a particular day.

3) *Weekly Active Users:* A window of a day or a month may be too small or large for a developer and so this feature can help the developer understand the application usage changes through all weeks of a month. If there is a huge drop in the number of users between week 1 and 2 then the developer can think of different ways to attract users.
4) *Monthly Active Users (MAU):* For a major update or change in the marketing strategy, a developer needs to analyze a larger window of user behavior. Monthly Active Users is the count of unique users for a month. This has been implemented by maintaining a count of new users using the application every day through the month. The sum of these new users over the month is the monthly active user count. The backend for this feature involves processing of the data structure maintained by DAU every day, with the processed data for the current month.

4. Event Correlation (Funnels)

With the help of this feature, a developer can understand the problem in the application. The problem can be identified by allowing the developer to visually compare multiple events and see where the user retention reduces [3]. The developer is given freedom to compare any number of events of the application. In other words, event statistics provide event specific retention reports. For instance, in a photo-sharing application, a user is required to perform multiple events to publish a photograph, which may include editing, adding filters, among others. If the retention reports after the final step are not positive, the problem lies in one of the intermediate steps. This can be identified using funnels.

5. Event Property Statistics

Many-a-times along with tracking events, particular event properties are also required by a developer. For example, in a bike racing game, let us assume a curve in the race track and a booster that a racer can use for that curve. A developer may want to know how many players actually use the booster and how many do not. So the event is 'race_curve' and the property is booster with a boolean value yes or no.

Similarly any property and any number of properties can be tracked along with an event. Using this feature, developers can get an in-depth understanding of their application.

6. A/B Testing

6.1 Importance

Often a situation arises when a developer has two ways to carry out the task. Instead of guessing which might work better, A/B testing allows the end users to make this decision. When in execution, the application will contact the server and decide whether the object should run in 'A' mode or 'B' mode. A conversion ratio is maintained, which is a ratio of the number of times the event under A/B is successful to the number of times the object is displayed in A/B. This feature can be extensively used in beta phase to test various features in different scenarios. Before the final release, the developer will have statistics good enough to make proper decisions.

6.2 Example

Let us take an example of a song library application. For such an application, the download button will directly affect the revenue. So the developer needs to make the button prominent. A/B Testing can help the developer in making a calculated decision of which color to finally use in the application. Figure 1 explains this example graphically. 65% of users pressed the green download button while only 20% pressed the blue one. This indicates that the green download button is more prominent. Hence, the developer would chose green as the final color for his download button.

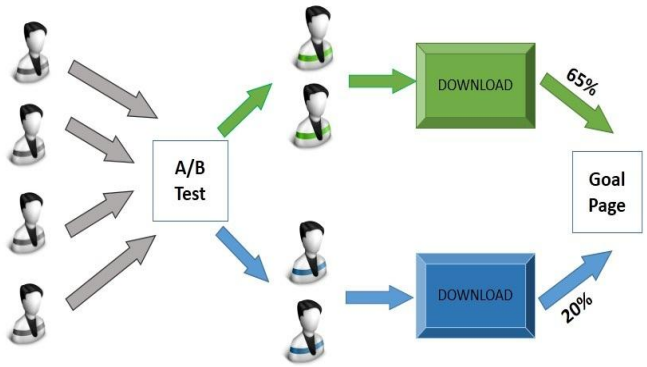

Figure 1: A/B Testing Overview

6.3 A/B Testing Implementation in RoloKnow

While the application is in execution, it will contact the server to decide which scenario the application needs to run in. The usage is tracked and cached. This includes the 'developer_id', 'app_id', 'item_name', 'activity_name', 'a', 'total_a', 'b', 'total_b'.

The variable 'total_a' is the count of how many times the object was displayed in 'A' mode, while 'a' is the count of how many times the user actually completed the step in 'A' mode. Similarly, for 'total_b' and 'b'.

7. Property Statistics

RoloKnow is currently equipped to show an application's usage statistics for the following properties:
1) Country
2) Network Carrier
3) OS Version
4) Mobile Model
5) Application Version.

7.1 Country Statistics

Country stats will give an overview of the regions where the application is popular across the world. As people in different regions have different mindsets, this information will help the developer prioritize the next update and also know his audience better.

7.2 Network Carrier Statistics

This information will help developers know which network carrier is preferred by their application users. Carrier billing, wherein the in-application purchases are directly build to the network carrier, is a popular practice today. By knowing which carrier is most popular the developer can collaborate with them and come up with some schemes which will benefit both parties.

7.3 OS Version Statistics

OS stats will help the developers understand which android version is most popular among the users. If a significant number of users use an older version of an operating system, like Gingerbread, updating the application with features not supported by Gingerbread will lead to reduction in the number of users.

7.4 Mobile Model Statistics

Android is a fragmented market. A developer may find it difficult to analyze the popularity of his application based on the make of the phone. This feature will help the developer make sure the devices that use their applications the most, are always compatible with the updates.

7.5 Application Version Statistics

If many users use an older version of an application, then the developer cannot remove support for that version. To take call on which versions to support, the developer can use this feature.

8. Data Visualization for Web Interface

For an analytics tool, it is very important to show the data in an interactive and understandable format. Using RoloKnow's powerful data visualization tools, a developer will be able to compare different statistics using a donut chart, understand the regions where the application is working using an interactive world map and also use histograms for other analyses [5].

9. Architecture

RoloKnow works in four major steps:

1) Smartphones to MongoDB: All the required user data that the developer needs is sent through the devices to MongoDB through a web server. Here the data is in its original form [6].

2) MongoDB to Hadoop: Selective data from MongoDB is sent to Hadoop for processing [7]. Here, Pig is used to carry out the MapReduce tasks. Pig is a high-level platform for creating MapReduce programs used with Hadoop. The language for this platform is called Pig Latin [8].

3) Hadoop to MySQL: The processed and meaningful data needs to be stored. For this RoloKnow uses MySQL. All the churned data is stored here for further use.

4) MySQL to Web Interface: Numbers are difficult and sometimes monotonous to analyze. That is why, all the meaningful data is shown to the developer using RoloKnow's web interface with powerful data visualization capabilities [5]. This is the final stage of the process and the developer gets meaningful results. Figure 2 below shows the architecture of RoloKnow.

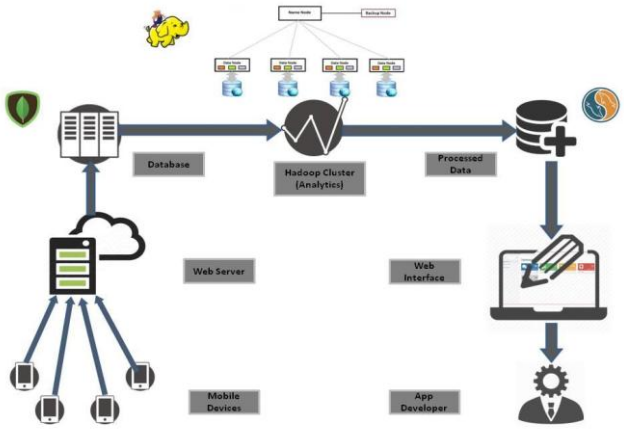

Figure 2: Architecture of RoloKnow

10. Implementation

10.1 Client-side Library

Client Library- A library, 'roloknow.jar', will be provided to the developer, to deploy in the application along with the developer id and an application id. The task of this library is to track event data, store it locally and send to MongoDB. Flushing of data has been implemented using a thread. If the user is not connected to Wi-Fi, the thread is put to sleep for larger amounts of time to save user data [4].

Steps to use the library in android studio-

1) Copy 'rolokow.jar' to 'app/libs' directory of your application.
2) Right Click and select 'Add to Library'.
3) Open build.gradle file and check if 'compile('app/libs/roloknow.jar')' is added under dependencies.
4) Perform a clean build.
5) Now, this library can be used with a few basic function calls. RoloKnow (this, dev_id, app_id).start() will initialize the library with the developer id and the application id. It will also start a thread to carry out flushing.
6) Event can be tracked using the function call given below in Figure 3. To add any additional data in the form of name value pairs, a json array can be passed as an argument to this function call.

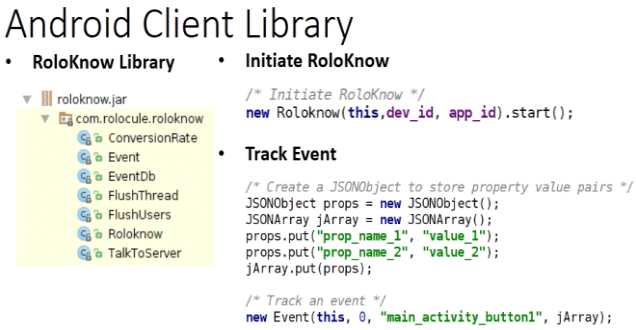

Figure 3: Steps to include RoloKnow library in an application

10.2 Hadoop Analytics

RoloKnow uses Pig in Hadoop to perform analytics on event data. It calculates hourly, daily, weekly and monthly active users. In addition, it performs analytics to find correlation between events and also finds property statistics as mentioned above.

10.2.1 Algorithm for DAU and HAU calculation

1) Load the latest event details from MongoDB into P;
1) Group P by (app_id, developer_id, device_id) and calculate latest timestamp for each group and store into Q;
2) Extract the data for the current hour from Q and store into 'hour_data';
3) From 'hour_data' generate unique count of 'device_id's for every (app_id, developer_id) and store into 'unique_count';
4) Load 'last_activity', which is a data structure maintaining most recent timestamps for unique (app_id, developer_id, device_id) into R;
5) Extract the data from R for the current day and store it in 'day_data';
6) Perform a natural join on 'hour_data' and 'day_data' and group the result by (app_id, developer_id). Store the result into 'common_data';
7) For each (app_id, developer_id) in 'common_data' generate count of 'device_id's and store into 'common_count';
8) Subtract count of 'device_id's in common_count from unique_count to get the new user count for the hour and store the result into 'FINAL'

'FINAL' will have developer_id, app_id, date, hour, unique_user_count, new_user_count. 'unique_user_count' is the count of unique users for a particular hour. Sum of new_user_count through the hours of the day is the daily active user (DAU) count. This data will be stored in MySQL to be used by the web interface.

10.2.2 Mathematical Representation

The above algorithm can be mathematically represented as,

$$U = [H] \quad \ldots(1)$$

$$N = U - [(H \bowtie D)] \quad \ldots(2)$$

$$DAU = \sum_{i=0}^{23} N \quad \ldots(3)$$

where, **N** is the new user count for an hour,
H is a list of devices active in a particular hour,
D is a list of devices that have previously accessed the application on a particular day under observation,
U is the count of hourly active devices,

(1) gives the hourly active user count, (2) gives the count of new users for a particular hour in a day and (3) gives the daily active user count for a day (24 hours).

11. Conclusion

RoloKnow is a product which can cater to the entire developer base and would become an integral part of application development. With the gap between the users and developers bridged with RoloKnow, the cycle of application development will surely become easier and faster. It is platform independent; therefore, any application developer would feel the need to use RoloKnow. There are going to be millions of users and their billions of events. RoloKnow will be a one-step solution for this.

References

[1] Oskar Wirén, "Data Driven Development for Mobile Applications", Teknisk- Naturventenskaplig, pp. 43-45, August 2013.
[2] Website: https://www.mixpanel.com/retention
[3] Website: https://www.mixpanel.com/funnels
[4] Website: https://developer.android.com
[5] Website: http://docs.amcharts.com/3/javascriptcharts
[6] Website: http://docs.mongodb.org/manual/
[7] Website: https://hadoop.apache.org/docs/stable/
[8] Website: https://pig.apache.org