國立臺灣大學電機資訊學院電機工程學系

碩士論文

Department of Electronic Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

# 手持裝置應用程式的排名方法和系統實作

# Methods and Systems for Ranking Mobile Application

劉浚宇

Chun-Yu Liu

指導教授：雷欽隆博士

Advisor: Chin-Laung Lei, Ph.D.

中華民國 104 年 7 月

July, 2015

# 誌謝

我要感謝所有幫助過我的人。謝謝栽培我的爸爸媽媽, 謝謝指導我的雷欽隆教授, 謝謝網路安全實驗室的大家, 謝謝過往繼續支持我的大學同學們, 謝謝大家。...

# 摘要

在現在的社會中，手機應用程式已經成為人們不可或缺的一項功能，其中一項很重要的因素是軟體市集的崛起，它使得應用程式的下載變得很方便。然而，這世界上已經存在著不少軟體市集，雖然大多數的人只知道 Google 商店和 Apple 的 app store。手機應用程式可以透過不少方式做行銷，但就我們所知，最有效的行銷手法就是成為軟體市集的推薦。這是很顯然的，因為每個使用者想要下載手機應用程式都必須開啟軟體市集，而每開啟一次就是直接接受了推薦的應用程式作為廣告。另一方面，由於軟體市集是安裝應用程式的入口，因此想要找尋應用程式也會透過此。

在這篇論文中我們提出了一套方法來計算應用程式的熱門程度。我們認為每個應用程式都有它的衰退期，無論它曾是多熱門。因此排序應用程式的熱門程度不單只是其下載量，還要考慮它的衰退情形。這麼做是為了避免有些應用程式曾經風靡一時，卻因為其一時的下載量而歷久不衰，而其實它現在已經不是使用者所喜歡的了。我們對 Google Play 商店的應用程式做統計，觀察下載量的變化，發現大多應用程式都有相似的成長量衰退情形，在不同分類的應用程式又有著不同的衰退速率，這對於手機應用程式的開發商是個很重要的數據分析，可以根據其應用程式種類隨著時間來達成預期的下載量。

# Abstract

In this decade, the popularity of the mobile Internet and the smart phones continue significant growth trend, because the immediacy, connectivity and convenience of the smartphones that are making many changes for the mobile internet user’ s behavior, such easily installing/uninstalling the mobile applications (Apps) from the apps stores. Thus apps installing smartphones have already become necessities in daily life. There are many popular apps stores in the world, for example, Google Marketplace and Apple AppStore are the most well-known apps markets. The convenience of smartphone also brings a great change in the apps ecosystem[16], the downloading ratio of apps, the number of apps users and apps developers are significant increasing in recent years. How to market apps is becoming a critical and large part of mobile internet business. In order to market effectively, to become top-rated marketing apps is the best and the effective approach for the most apps developers [12]. However, apps marketing is hard, the apps ranking and searching strategies are controlled and manipulated by the apps markets. There are few article examining the app ranking system, however the mobile app monetization strategy is dominating the entire mobile internet, for example, the revenue of apps reaches more than $38 billion in 2015.

In this paper, we propose a method to calculate the popularity of mobile applications. We think that no matter how popular they used to be, they own their recession. Therefore, it should not rank apps only by amount of downloads but consider the decays. It can avoid some situation that some applications used to be popular and earned lots of downloads, but actually not users’ favor now. We collect the app data in Google Play store to make statistics, observe the change of downloads. In the experiment, we found most applications exist decay situation in volume growth. And different category of applications have their own decay velocities. This is a great analysis to developer of mobile applications, they can use to predict the downloads. App store can also calculate the real popularity.

# Contents

# List of Figures

# Chapter 1

# Introduction

Flurry [10] is a mobile analytics, monetization, and advertising company. Based their statistics result in 2013, when people using a smart phone. It took 80% of time to use mobile app, 20% of time to use mobile web browser. It means people spent most of their time on mobile app. Also, based on another statistics, the amount of smart phone is larger than tablet. If we browse websites via our smart phone, it will take a lot of time, there are many redundant resources to load, and many extra DOM to show, and more animation need to execute. Mobile app improve this kind of situation, it is much faster, and the smart phone already have some optimization for the UI rendering, that’ s why people spend more time on mobile app.

In 2014, Flurry reported another statistics [11], when people using a smart phone. It took 86% of time to use mobile app, and it’ s even more the 80% in 2013. More and more companies developed their new mobile apps. Based on the change during 2013 and 2014, we can say that mobile apps will conduct the user behavior for using smart phone. Further more, based on the statistics result from Google Play, Amazon Appstore, and Apple App Store. In 2014, Google Play increased more than 200% apps, it means they had double the amount of total apps in their app store. For Amazon Appstore, they increased more than 180% apps. For Apple App Store, they increased more than 150% apps. Based on the increasing rate, after three years, it would have more the 10M mobile apps in Google Play.

Although the number of Apps several times into the original, but innovative services

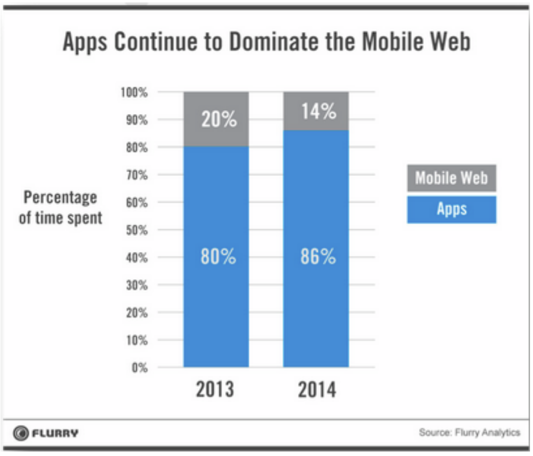


Figure 1.1: Time spending between mobile web and apps

but it is a minority, so most are similar apps. With the increase of similar apps, it is difficult to find a right app in app store. App ranking becomes very important, it will affect the use how to deal with these massive app data.

Today, app stores have their own ranking, usually with popularity rankings, and the “popularity” is depended upon downloads and user comments. Although the popularity is close to the real situation, but there are some problems. For instance, in Google Play store, we can see the high number of downloads, but already unpopular apps in the “Popular” category. These apps ranked higher because they have very high downloads, is simply a momentary trend. Even with a long time, the apps are not popular as before, but the apps still have higher ranks because the high downloads in early. The ranking system is not match the popularity in the real world.

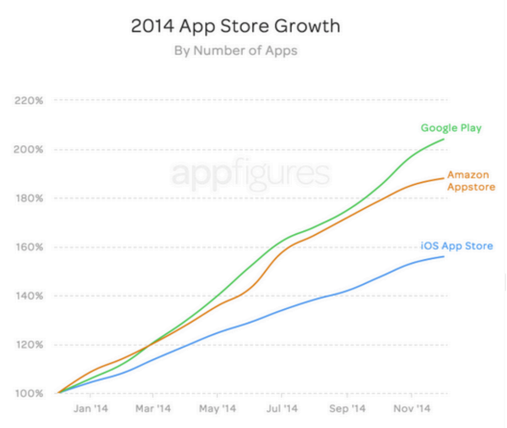


Figure 1.2: 2014 App Store Growth Graph

# Chapter 2

# Background

Today there are so many ranking systems, according to different categories and different rankings, but have one thing in common, to close to the real user preference, popularity. The accuracy of the ranking system can win the user’ s trust, better user experience, and less search cost. The following is a survey conducted for several popular ranking system.

## 2.1 Movies

IMDb[5] is the most popular and authoritative source for movie ranking system. It collects the richness of movie content including cast, production crew, fictional characters, biographies, plot summaries, trivia and reviews. It also is an online database of detail information related to films including bloopers, versions, aspect ratio and sound tracks in the films. Moreover, the authoritative ranking system of IMDb really catches people’s eyes, it is the most authoritative movie ranking system in the world. It offers a rating scale that allows users to rate films, so that the movie ratings of IMDb are reflections of crowd psychology.

Rotten Tomatoes[7] is another widely known film review aggregator. The ranking system is based on the editors and reviewers of Rotten Tomatoes. The reviewing steps are as follows. First, Rotten Tomatoes staff first collect online reviews from writers who are certified members of various writing guilds or film critic associations. Then the staff determine for each review whether it is positive or negative. Note that staff assessment is needed as some reviews are qualitative rather than numeric in ranking. The website keeps track of all of the reviews counted for each film and the percentage of positive reviews is calculated. (Major, recently released films can attract up to 250 reviews.) If the positive reviews make up 60% or more, the film is considered ¨fresh,¨ in that a supermajority of the reviewers approve of the film. If the positive reviews are less than 60%, the film is considered ¨rotten.¨

Metacritic[6] is an online ranking system that aggregates reviews of music albums, games, movies, TV shows, DVDs, and formerly, books. The ranking mechanism is named Metascores that is weighted average approach. In the ranking system, for each product, a numerical score from each review is obtained and the total is averaged. Metacritic's method of scoring converts each review into a percentage that the site decides for itself, before taking a weighted average based on the critic's fame or stature, and listing different numbers of reviews. Three color codes of Green, Yellow and Red summarize the critic's recommendation, giving an idea of the general appeal of the product among reviewers and, to a lesser extent, the public.

There are many research papers about improving the movie searching and ranking system[14].

## 2.2 Books

Sale volume is defined as the quantity of services, such as box office, the download number of app, in a specified period. It can easily reflect the most popular hot services. Therefore, the most of book stores show their book sales ranking for attracting the follow the crowd.

Social recommendations ranking system is based on the social networking. This trend is accompanied by a significant behavioral change: members rely more on the opinion of their peers for the purchase of products or membership to a brand. With social networking designed for content sharing and its versatility which makes it easy to expand its network of friends, amplifies the use of recommendations within the members of your community.

## 2.3 Websites

Page view (i.e., a request to load a single web page) statistics is a fairest indicator for each web site. However, this statistics data can be manipulated by site owners. Therefore, an authoritative web statistics module is adopted by most website owners. The authoritative web statistics module can avoid the cheating of the dupulicated page views by the same user, and can obtain the real user statistics data all over the world. Moreover, the page view is also an important indicator for venture capitals to evaluate the value of an Internet company.

Search engine optimization[13] (i.e., SEO) is the process of affecting the visibility of a website or a web page in unpaid results of a search engine (e.g., Google, the largest search engine in the world). Therefore, the importance of SEO and how to improve it is described by many books and blogs. The higher SEO ranking a website owns, the more popular a website is.

Return rate means the rate of returning visitors who visit a website before and come back to visit the same website again. The return rate indicator is more important than the page view statistics because the page view statistics includes advertiseing, the visits from the the search result, any incautious visits, etc. However, returning visitors in the return rate are the visitors who ever visited a website and may intend to visit the same website again for obtaining the information. Therefore, the return rate is a curical ranking indicator for a website.

## 2.4 Mobile Apps

We often see the app ranking results provided by some data statistics companies, e.g., Google Analytics [3], Flurry [10], App Annie [2], etc. These companies provide their statistics modules and software development kits (SDKs) to developers, so that the developers can install the SDK to easily obtain users’ statistics data, e.g., the user operation flow in an app, the number of clicks of each button, etc. Besides, these companies collect a lot of statistics data from all developers who use their SDK. Thus, these companies can provide an authoritative ranking system. The statistics data of the apps is very useful, including the average usage time, the rate of app uninstallations, and so on.

The owners of app stores own some data including the apps’ downloads, user evaluations, and user feedbacks. They also establish their ranking systems, but the ranking algorithm is not public. As mentioned before, there are some issues in the ranking mechanism of the app stores. And we address these issues in this thesis. Besides, some authoritative technology media websites, e.g., TechCrunch [8], and TechOrange [9], also provide their app statistics reports. Their reports are not only very influential but also advertising oriented, However, they determine the popularity of the apps accoring to data from some statistics companies and the surveys from the community websites.

# Chapter 3

# Approach

## 3.1 System Structure

Reference will now be made in detail to examples of the invention, which are illustrated in the accompanying drawings. Wherever possible, the same reference numbers will be used throughout the drawings to refer to the same or like parts. The example below is directed to search results comprising mobile applications. However, as persons in the art can appreciate, the present invention can be applied to various other search results such as websites, software applications, online games, etc.

### 3.1.1 Network

Figure 3.1 is a conceptual diagram illustrating a system according to an example of the present invention. The system can include a server cluster 100 (or a cloud), application servers 200, and a first user 10, each of which can communicate with each other through network 30, which can be but not limited to the Internet, cellular network or any kind of software define network(s). In this example, the server cluster can include a plurality of servers (e.g., servers 100a-100d shown in figure 3.1), each of which is configured to perform at least a part of the tasks, processes or steps of the present invention, either separately, sequentially, in parallel or a combination thereof to improve the efficiency of the system. A server in the server cluster 100 can include but not limit to a processor, a memory, a communication port and a storage device such as a database. An example of the server cluster can be described with reference to figure 3.2. Here, a personal computer (PC) can also be used as a server in the server cluster 100.

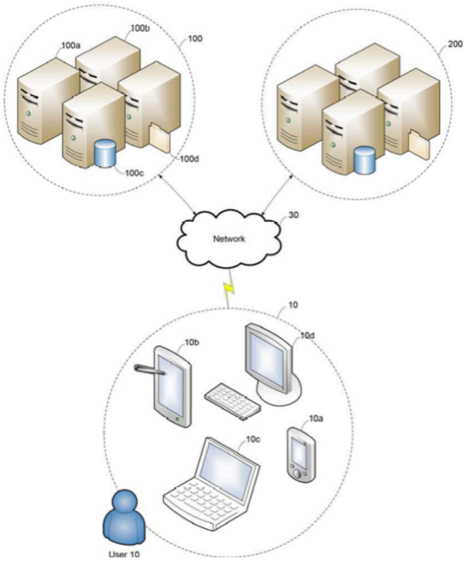


Figure 3.1: A conceptual diagram illustrating a network according to an embodiment of the present invention

### 3.1.2 Database

Figure 3.3(a) and 3.3(b) illustrate data structures of application data that can be provided by an application search engine according to an example of the present invention. Referring to FIGS. 3A and 3B, application data can be stored in a data repository (or database). In this example, the data repository (or the database) can be a relational database (and other file(s) relating to the software application can also be stored in storage with path(s) recorded in the repository/database or a file system). Referring to figure 3.3(a), the data structure (or the data schema) can include a first field 300-1 to store the name of an app (e.g., “Joseph’s App” ), a second field 300-2 to store the application ID used to identify the app (e.g., an application number assigned by the application search engine server 100b to identify data stored in the datebase, which is “00000168” in this example), a third field 300-3 to store the file name of an application package of the app (e.g., “example0000168.apk” ), a fourth field 300-4 to store the location of the app (e.g., a fixed or changing location

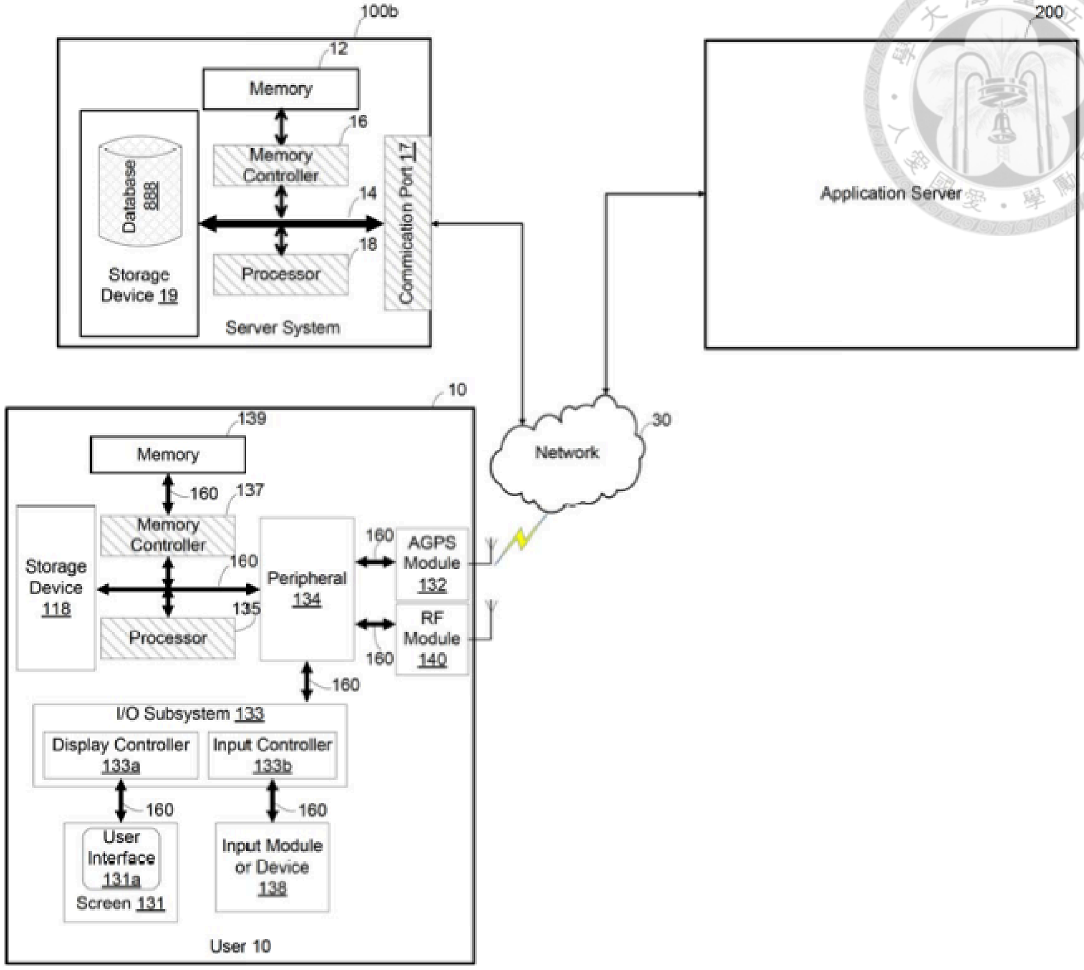
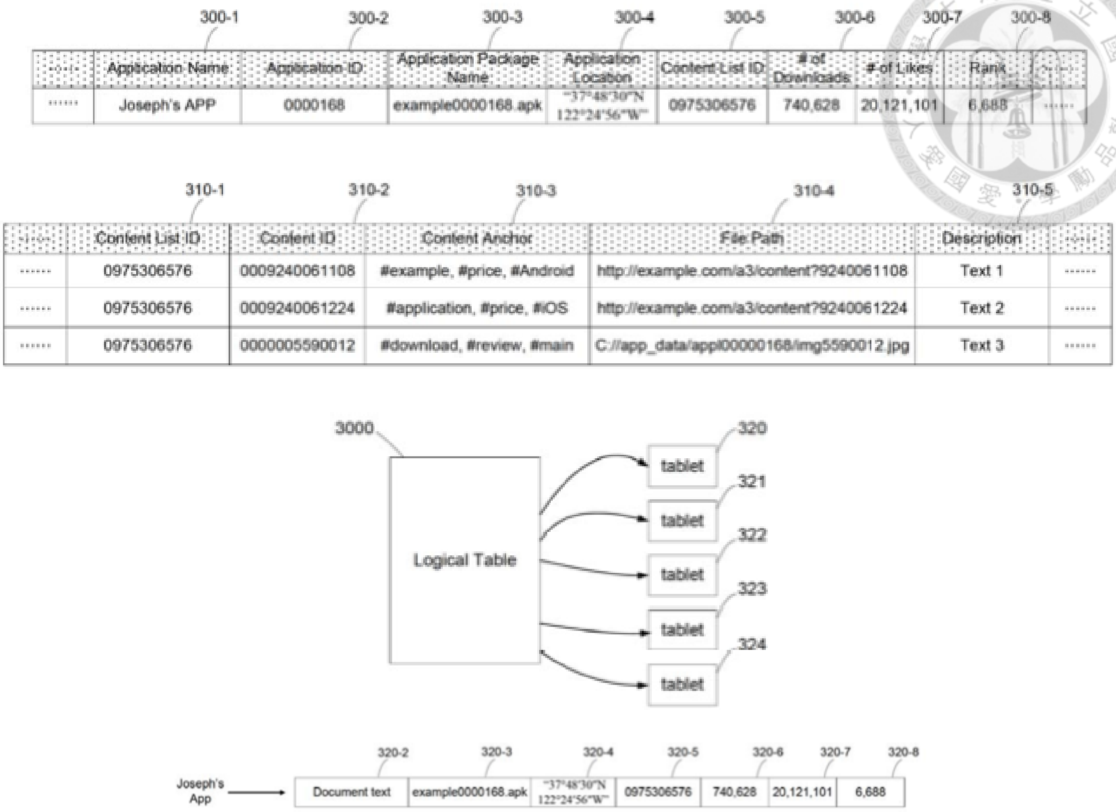


Figure 3.2: A diagram illustrating a network according to an embodiment of the present invention

where the app is set, such as a coordinate like “37°48′30″N 122°24′56″W”), a fifth field 300-5 to store the content list ID (i.e., an ID assigned to identify the list of content of the app referred in figure 3.3(b)), a sixth field 300-6 to store the download count for the app, a seventh field 300-7 to store the number of likes for the app (e.g., the number of times a “like” button is pressed by users to approve the content of the app), and an eighth field 300-8 to store the number of check-in for the app (e.g., users executing the app to check-in to specific locations). Figure 3.3(b) represents one of the data structures for content list ID stored in 300-5 referred in figure 3.3(a). The structure includes a first field 310-1 to identify content list IDs, a second field 310-2 to store all the content IDs corresponding to 310-1, a third field 310-3 to identify content achors, a fourth field 310-4 to store file paths, and a fifth field 310-5 to identify content location. Those skilled in the art understand that



| ...... | Application Name | Application ID | Application Package Name | Application Location | Content List ID | # of Downloads | # of Likes | Rank | ...... |
|---|---|---|---|---|---|---|---|---|---|
| ...... | Joseph's APP | 0000168 | example0000168.apk | "37°48'30"N 122°24'56"W" | 0975306576 | 740,628 | 20,121,101 | 6,688 | ...... |

| ...... | Content List ID | Content ID | Content Anchor | File Path | Description | ...... |
|---|---|---|---|---|---|---|
| ...... | 0975306576 | 0009240061108 | #example, #price, #Android | http://example.com/a3/content?9240061108 | Text 1 | ...... |
| ...... | 0975306576 | 0009240061224 | #application, #price, #iOS | http://example.com/a3/content?9240061224 | Text 2 | ...... |
| ...... | 0975306576 | 0000005590012 | #download, #review, #main | C://app_data/appl00000168/img5590012.jpg | Text 3 | ...... |

| | | | | | | |
|---|---|---|---|---|---|---|
| Document text | example0000168.apk | "37°48'30"N 122°24'56"W" | 0975306576 | 740,628 | 20,121,101 | 6,688 |

Figure 3.3: data structures according to an embodiment of the present invention

some fields shown in figure 3.3(a) and 3.3(b) are only examples of the present invention and at least some of the fields can be optional. Moreover, the data structure in the data repository/database is not required to include the exact same fields as shown in figure 3.3(a) or 3.3(b). Moreover, the logical table 3000 or other data structure/schema (e.g., HFS or those used in big data schemes) can also be used to implement the database of the present system.

### 3.1.3 Search Engine for Searching Apps

Figure 3.4 is a diagram illustrating a system of searching application data generate by apps (i.e., "In-App"). Referring to figure 3.4, the In-App search framework can include a scheduler, an app robot (or a plurality of app robots, hereinafter the "app robot" or the "app robots") and a data harvest framework. The app robot(s) can be configured to control and operate apps. The scheduler can be configured to schedule the time to activate a corresponding app robot to run an app. The data harvest framework can be configured to gather and extract the application data generated by apps and their corresponding application server(s), and store the collected application data in the data repository (or the database). The main purpose of the scheduler is to activate the app robot(s) based on specific scheduling policy and logic from the data repository. Other factors can be programmed such as the execution frequency of the apps, the running pool of the apps, the running states of the apps and etc. The scheduling policy and logic can also be developed by an artificial intelligence approach. The data harvest framework can be configured to gather and extract application data from an app’ s corresponding applicaiton server(s), and store the application data in the data repository (or the database). At least two approaches can be used to harvest the application data in the system framework of the present invention: the system hooking approach or the network analyzer approach (we can apply either one of them or both). There are two types of the system hooking approach: the user-level hook and the kernel-level hook. Both can be used to intercept and trace native API calls. A hook system usually comprises two parts: a hook server and a driver. A hook server is responsible for injecting the driver into targeted application processes. The difference between a user-level hook and a kernel level hook is that user-level hook may suffer library dependency problems when library updates (for example, such as moving from Android 4.3 to 4.4). A network analyzer approach monitors the network traffic and extracts the application data from end-to-end data flows. For example, a data gathering proxy can be set up between the app and its application server. In this example, any data transmission of the app can be intercepted and cached in the data gathering proxy. The cached data may be restored in data repository later.

## 3.2 Half-Life Method

There are various half-life mathematical models[17] that can be used in the current invention such as but not limited to exponential or geometric decay. We use the exponential half-life mathematical model to illustrate the current invention. Assume that $N(t)$ is the activating count for an app, and t is the time (e.g., in days). Consider the exponential decay model $N(t) = N_0 e^{-\lambda t}$, that is, $N(0) = N_0$ when $t = 0$ (when the number of the app

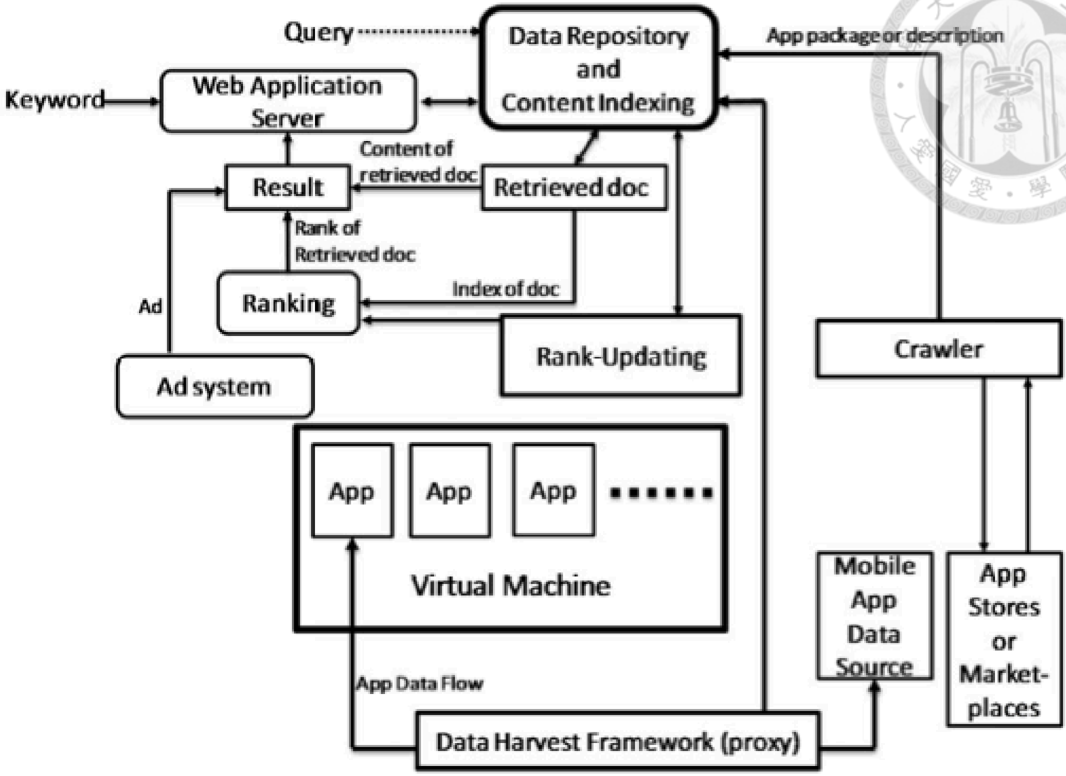


Figure 3.4: System of searching application data according to embodiments of the present invention

peaks and will start to decay exponentially). Different apps can have different half-lifes depending on their popularity over time, and the ranking of the apps are determined by their half-lifes. In addition, ranking using half-life can be done in combination with popularity and/or other half life-related characteristics of the search result such as quantity of popularity peak, the point of its popularity lifespan the search result occupies at time of ranking, etc... Further, ranking can also be done in combination with other ranking methods such as the number of search query keyword matches, etc···Figure 3.5 illustrates an embodiment of the present invention. The system for ranking apps can include an app popularity monitor 502, a multiplexer 504, a half-life calculator 506 and a ranking calculator 508. The app popularity monitor 502 can determine the popularity of an app based on its download count, user count, activating count or combination thereof. Based on a number of predetermined reference rates (for examples growth/decay rate of number of users, execution rate and/or other rates), the app popularity monitor 502 can also generate triggers. For example, when the growth rate of number of users and the execution rate for an app fall below certain predetermined reference rate, the app popularity monitor 502, through its multiplexer 504, will trigger the half-life calculator to start calculating the half-life of the app. In another embodiment, the half-life calculator 506 can directly monitor the popularity of an app during a predetermined period of time, for example, monitoring an app’ s daily download count for a period of at least one month. Once the daily download count decays to approximately half of its regular value or the value when the half-life calculator 506 is triggered, the half-life of the app is determined.

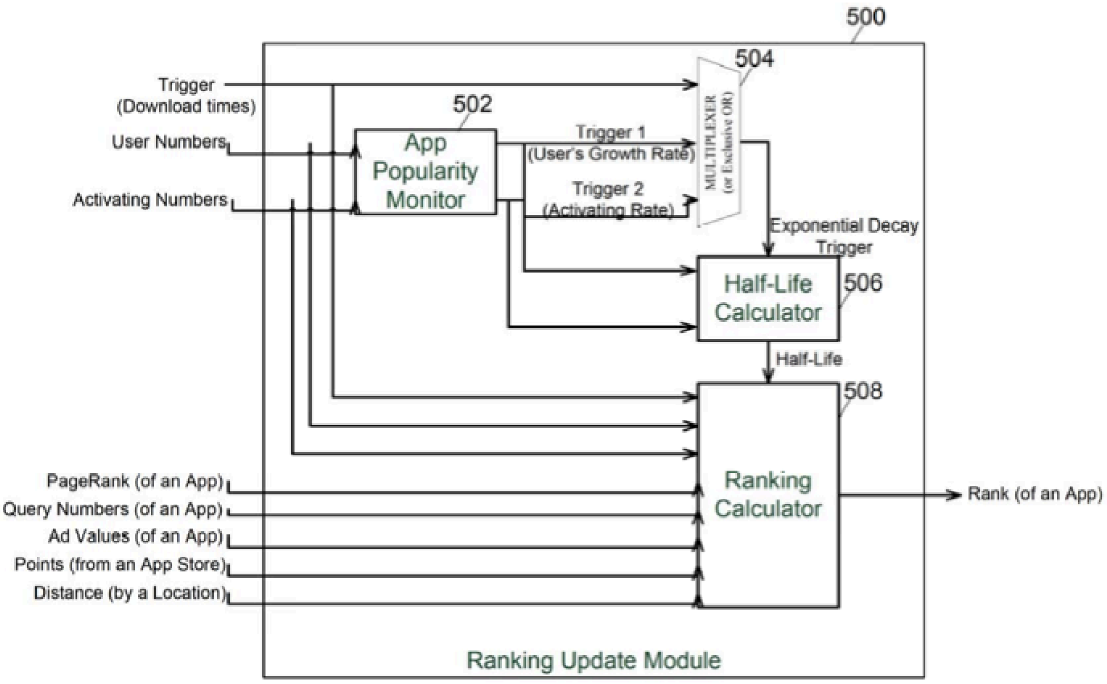


Figure 3.5: A block diagram of the ranking algorithm for ranking software applications according to an embodiment of the present invention

When an app’ s half-life is determined by the half-life calculator 506, such information is sent to the ranking calculator 508. The ranking calculator 508 can calculate the rank of the app according to its half-life. Those skilled in the art understand that the ranking calculator 508 can also rank using popularity and/or other half life-related characteristics of the search result such as quantity of popularity peak, the point of its popularity lifespan the search result occupies at time of ranking, etc... Further, ranking can also be done in combination with other ranking methods such as the number of search query keyword matches, etc···. In one emodicment, the system of the present invention can further include at least a memory (not shown in the figures), flash (not shown in the figures), storage (not shown in the figures) or combination thereof configured to provide a script (not shown in the figures) to be embedded in the webpages where the app can be downloaded. The script is able to collect information such as the app’s download count, user count or activating count to be passed to a ranking update module when the app is downloaded from the webpate where the script is embedded. In another example, the system of the present invention can further include at least a memory, flash, storage, or combination thereof configured to provide an sdk (not shown in the figures) to be embedded in an app. The sdk can collect information when the app is downloaded to a client (not shown in the figures). The information can include the the app’s download count, user count or the activating count. Such information is transmitted to the ranking update module on a server by executing the sdk on the client.

Figure 3.6, 3.7 and 3.8 illustrate methods of ranking apps according to an embodiment of the present invention. Referring to figure 3.6, the method for ranking apps can include steps 602-610. In step 602, the app popularity monitor determines an app’s popularity based on its download count, user count, activating count or combination thereof. In step 604, based on a number of predetermined referenced rates, the app popularity monitor can decide if an app’s popularity has peaked. If yes, proceed to step 606. If not, proceed to step 610. In step 606, the half-life calculator 506 can calculate an app’s half-life after the app’s popularity peaks. In step 608, the rank of an app is calculated based on its half-life. In step 610, the rank of an app is further calculated based on the popularity and/or other half life-related characteristics of the search result such as quantity of popularity peak, the point of its popularity lifespan the search result occupies at time of ranking, etc... Further, ranking can also be done in combination with other ranking methods such as the number of search query keyword matches, etc⋯.. Referring to figure 3.7, the method can further include providing a script to be embedded in the webpages where app can be downloaded in step 702, and transmissing information about the app to the ranking update module by executing the script in step 704, wherein the information can include the app’s download counnt, user count or activating count. Once the popularity of the app peaks, it will generate an exponential decay trigger to the half-life calculator

506 in step 708. The method described and illustrated in 3.8 is similar to those described and illustrated with reference to figure 3.7, except that, for example, it includes a step of providing an sdk to be embedded in the app in order to collect information of the app on a client, as shown in step 710. It will be appreciated by those skilled in the art that changes could be made to the examples described above without departing from the broad inventive concept thereof. It is understood, therefore, that this invention is not limited to the particular examples disclosed, but it is intended to cover modifications within the spirit and scope of the present invention as defined by the appended claims. Further, in describing representative examples of the present invention, the specification may have presented the method and/or process of the present invention as a particular sequence of steps. However, to the extent that the method or process does not rely on the particular order of steps set forth herein, the method or process should not be limited to the particular sequence of steps described. As one of ordinary skill in the art would appreciate, other sequences of steps may be possible. Therefore, the particular order of the steps set forth in the specification should not be construed as limitations on the claims.

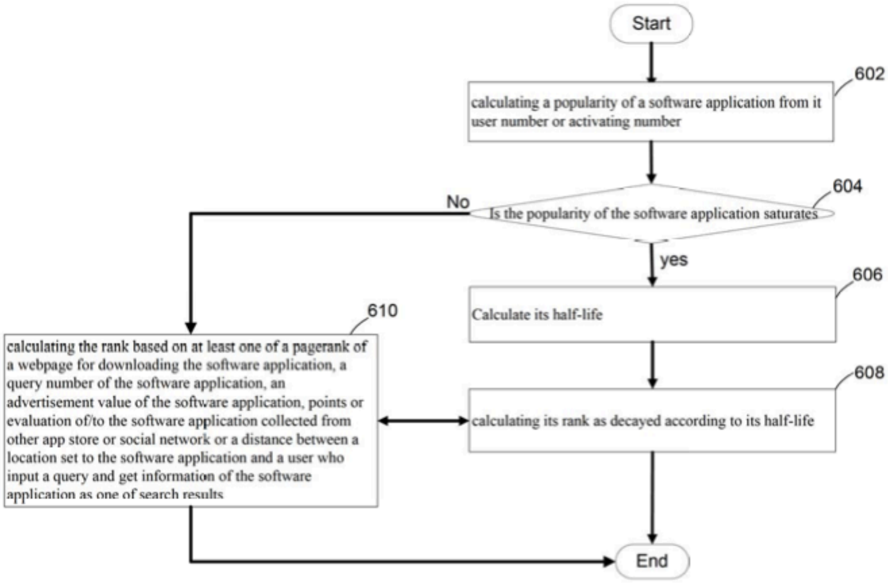

Start
602
calculating a popularity of a software application from it user number or activating number
604
No
Is the popularity of the software application saturates
yes
606
Calculate its half-life
610
calculating the rank based on at least one of a pagerank of a webpage for downloading the software application, a query number of the software application, an advertisement value of the software application, points or evaluation of/to the software application collected from other app store or social network or a distance between a location set to the software application and a user who input a query and get information of the software application as one of search results
608
calculating its rank as decayed according to its half-life
End

Figure 3.6: Flowcharts illustrating methods of ranking software applications according to an embodiment of the present invention

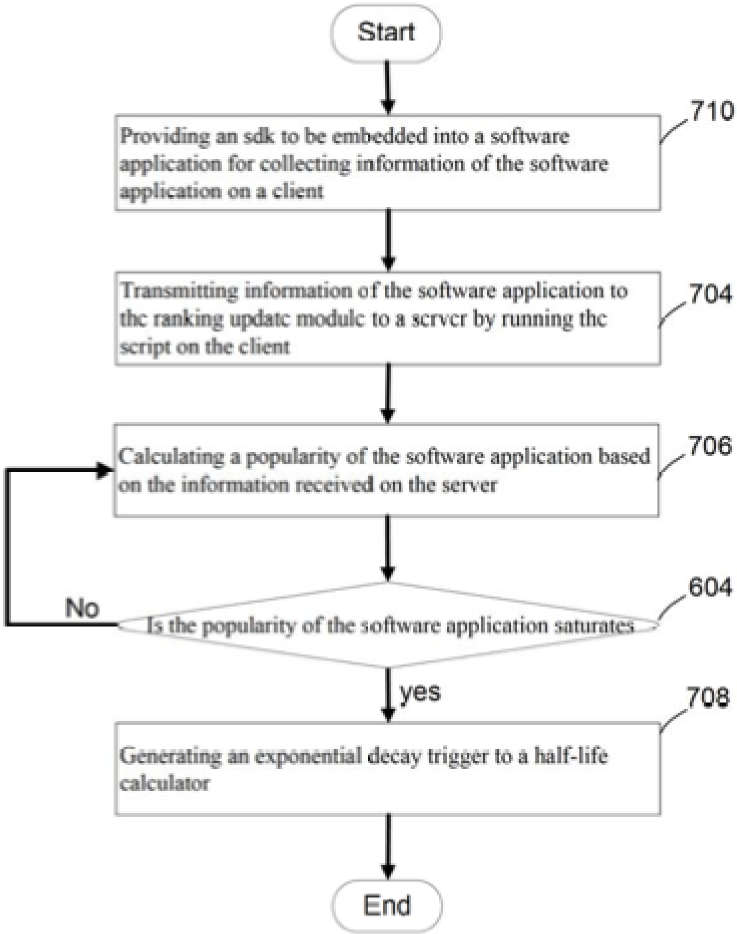

Start
710
Providing an sdk to be embedded into a software application for collecting information of the software application on a client
704
Transmitting information of the software application to thc ranking updatc modulc to a scrvcr by running thc script on the client
706
Calculating a popularity of the software application based on the information received on the server
604
No
Is the popularity of the software application saturates
yes
708
Generating an exponential decay trigger to a half-life calculator
End

Figure 3.7: Flowcharts illustrating methods of ranking software applications according to an embodiment of the present invention

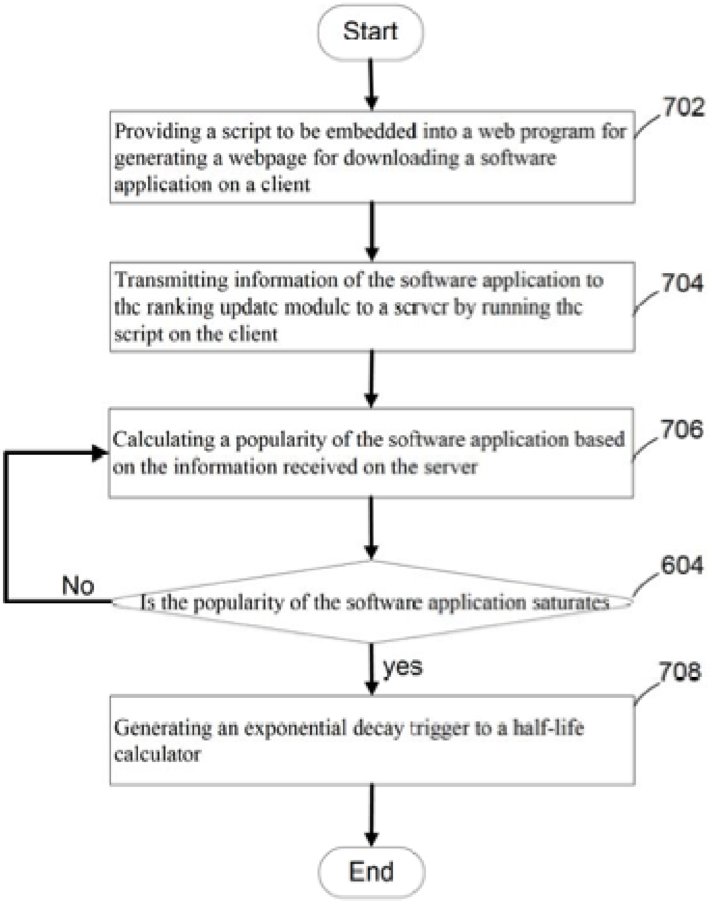

Start
Providing a script to be embedded into a web program for generating a webpage for downloading a software application on a client
702
Transmitting information of the software application to thc ranking updatc modulc to a scrvcr by running thc script on the client
704
Calculating a popularity of the software application based on the information received on the server
706
No
Is the popularity of the software application saturates
604
yes
Generating an exponential decay trigger to a half-life calculator
708
End

Figure 3.8: Flowcharts illustrating methods of ranking software applications according to an embodiment of the present invention

# Chapter 4

# Experiments

## 4.1 Materials

To verify the half-life calculator, we design a distributed crawler to retrieve the app data from Google Play. According to the app information, we can know the amount of change in apps. We use the half-life calculator to predict some values in the next section, and comparing to the actual values we retrieve. In the first experiment, we select downloads as popularity basis. In the second, we choose user ranks.

## 4.2 Crawler

A web crawler[15] is an internet robot which systematically browses the World Wide Web. When the crawler come to visits a website, it collects contents inside and spreads by the outlinks in this website. To do so, the crawler can retrieve all the websites which connect to the World Wide Web by links. Google search engine is a good example for this. Their crawlers crawl every day so that they can keep the data almost synchronize with original website. The app crawler is similar to the web crawler. An app detail page has outlinks such as “similar apps” and “developer apps” and can link to other app detail page. As web crawler, we start from the portal (Google Play homepage [4]), visit all the apps recursively. Otherwise, there are over 1 million applications in Google Play store so we have to collect those data efficiently. Distributed crawler is our solution. An app crawler first gets an app detail link and retrieve the app data, save to database if it has modified or updated. Then inserts all the outlinks to database so that other crawlers can work simultaneously. Figure 4.1 shows the app crawler' s flow chart.

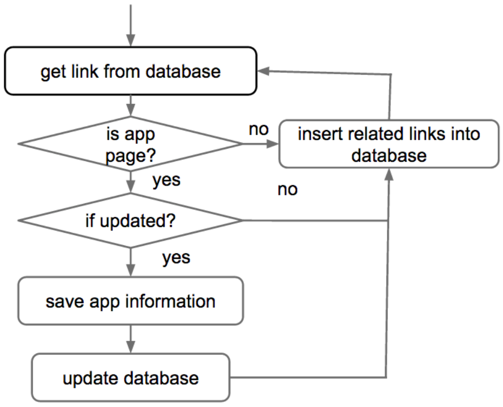


Figure 4.1: Flow chart of the distributed app crawler

## 4.3 Experiments

We do those experiments on Amazon Web Server [1], 10 machines are used to collect app data distributedly. We use the last two values we retrieve to calculate the half-lifes decay function to predict the next value. Then compare the predicted value to the actual value. Figure 4.2 shows an app "ShotZombie" , which calculates the every week download changes prediction and compares to the actual values. Then we get two lines, compute the correlation coefficient of two.

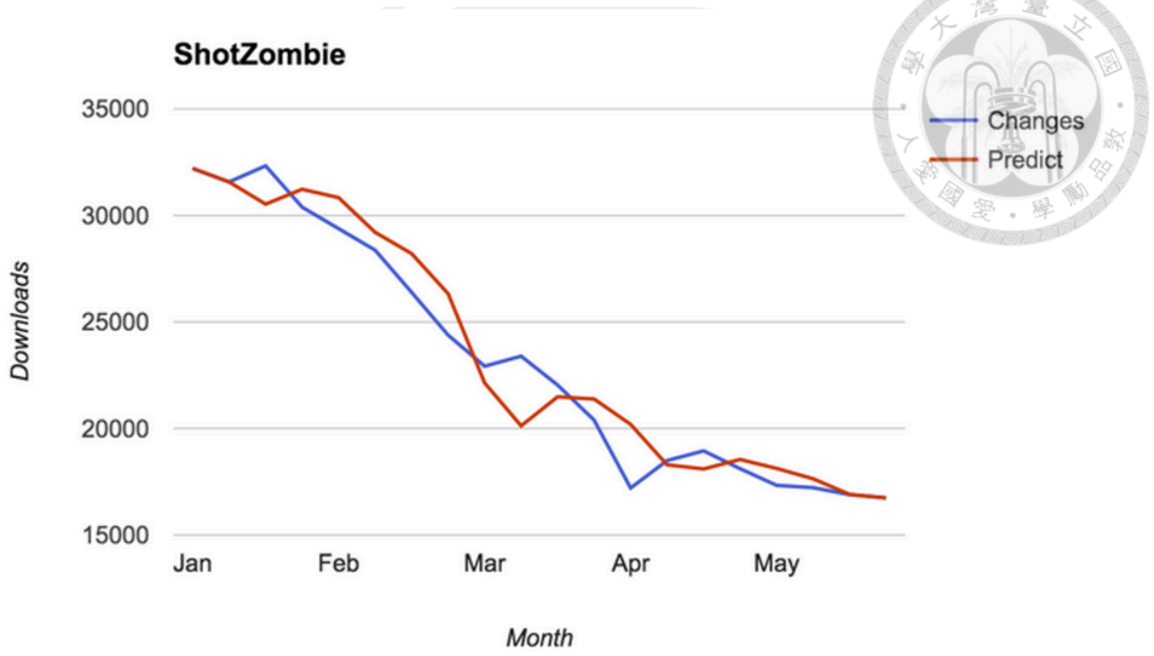


Figure 4.2: Half-lift decay function prediction on ShotZombie

### 4.3.1 Experiment I

First experiment targets the popularity in downloads. Downloads are the most objective factor about popularity. It directly reflects the users' decision. So we collect downloads of applications(Which present the apps eccept games) and games and classify into categories. And another side, we compare all apps to over 500k downloads apps. As demonstrated in Figure 4.3 and 4.4, we got average correlation coefficient on different categories. Those present games and applications, over 500k downloads and all. In the results, we get accurate prediction on more than 500k downloads apps in either applications or games. Otherwise, games are more accurate than applications.

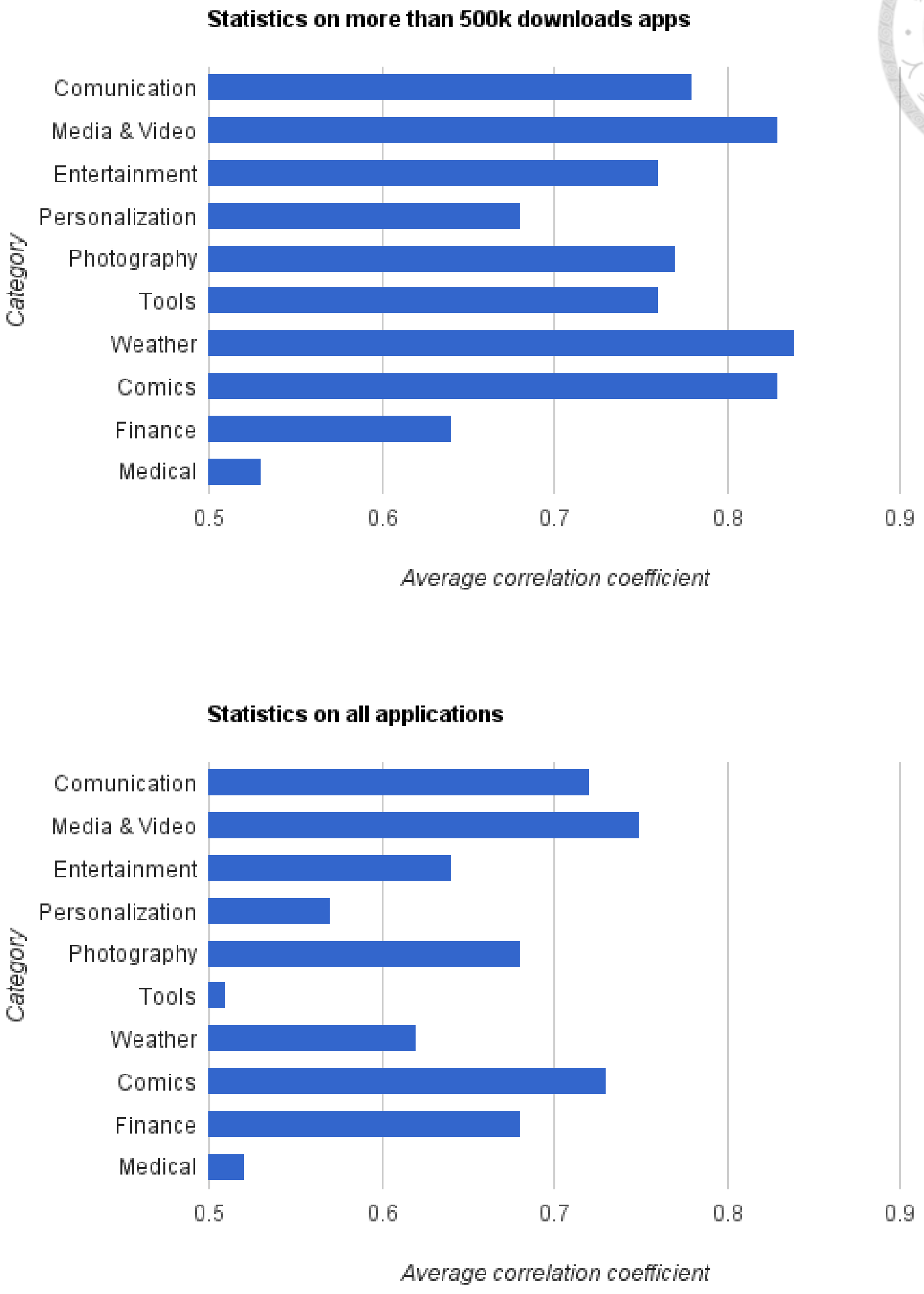


Figure 4.3: Experiment I - Applications

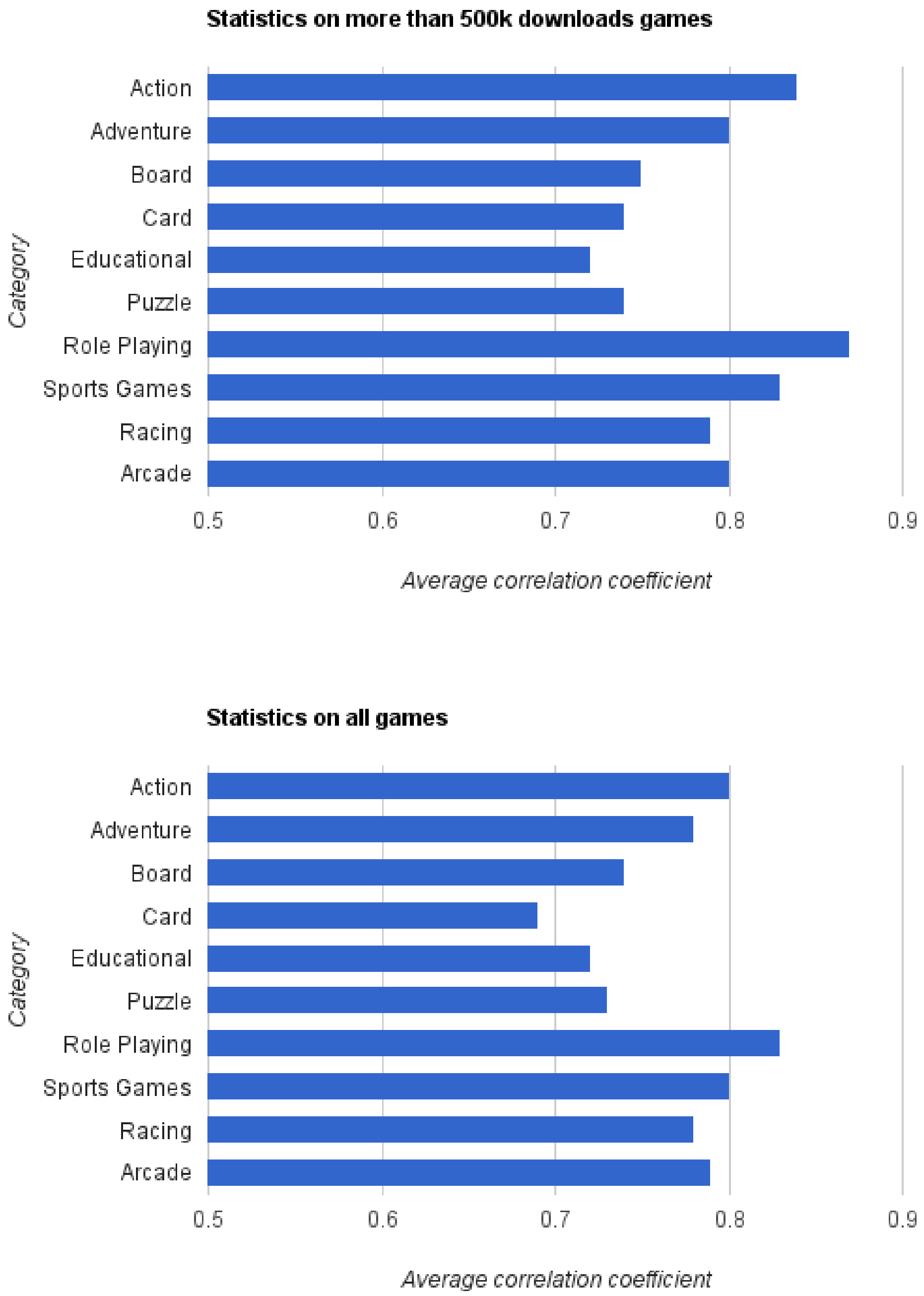


Figure 4.4: Experiment I - Games

### 4.3.2 Experiment II & III

Second experiment targets the popularity in user ranks. The user rank in Google Play store is a short comment which user can give a score from 1 to 5 to an app. As we observe for a while, the second and third stars are seldom. Which means that people usually give the best or worst rank to apps. Another guess is people don’ t give rank if they have no idea about this app. What we concern here is popularity, the negative comment are not the factor. So in second experiment, we test at least 4 and at least 5. Other factors is just like the previous experiment. Figure 4.5 and 4.6 shows the result about user rank at least 4, 5 in Figure 4.7 and 4.8. Likewise, we get more accurate predictions on more than 10k ranks apps both in games and applications. And user rank at least 5 is bether than at least 4.

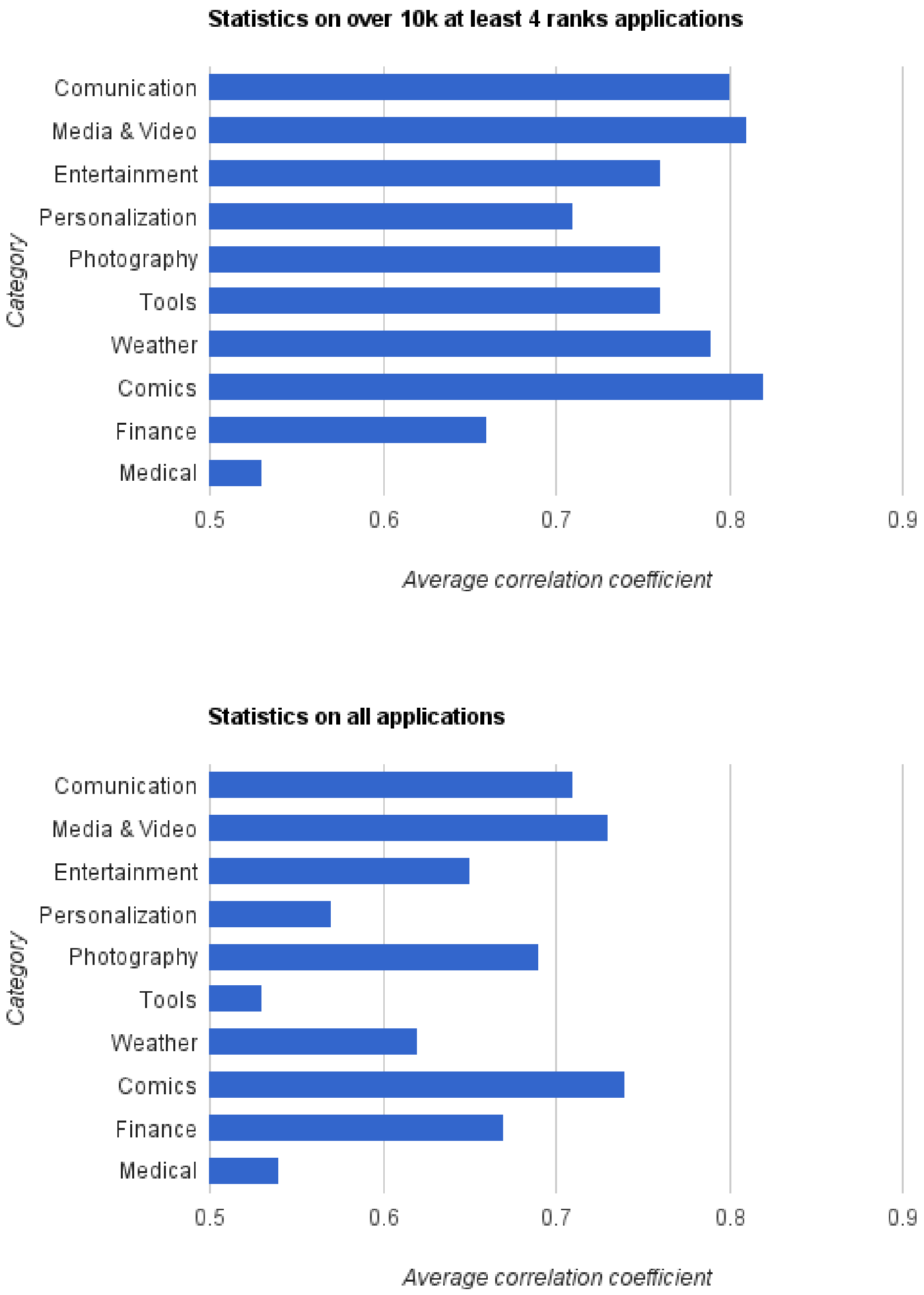
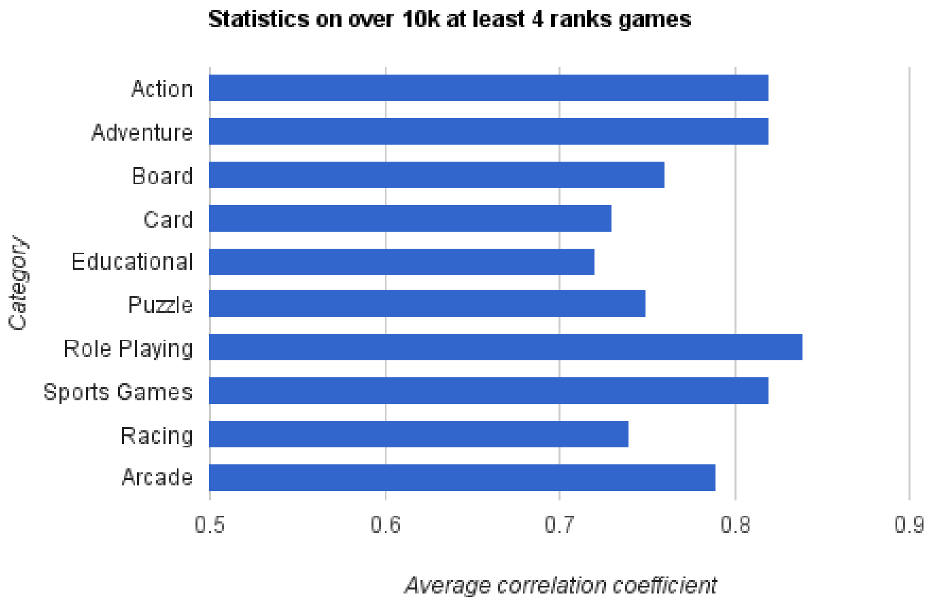


Figure 4.5: Experiment II - Applications



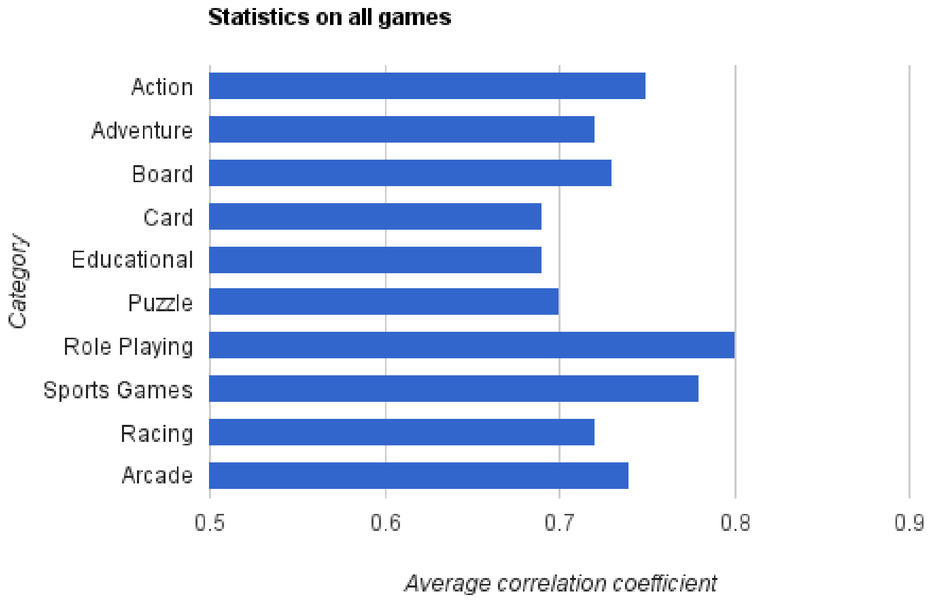
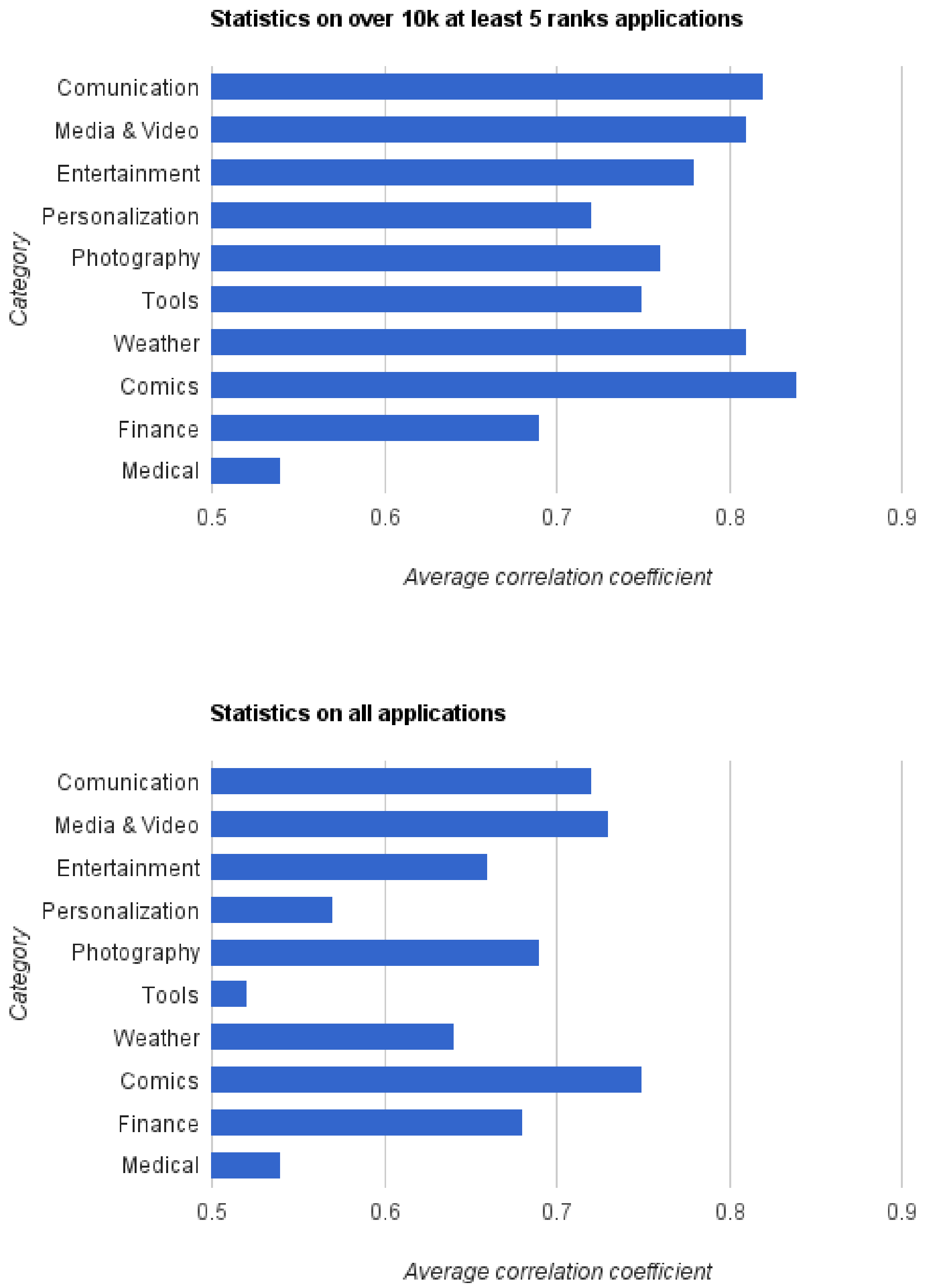


Figure 4.6: Experiment II - Games


Statistics on over 10k at least 5 ranks applications

Statistics on all applications

Figure 4.7: Experiment III - Applications

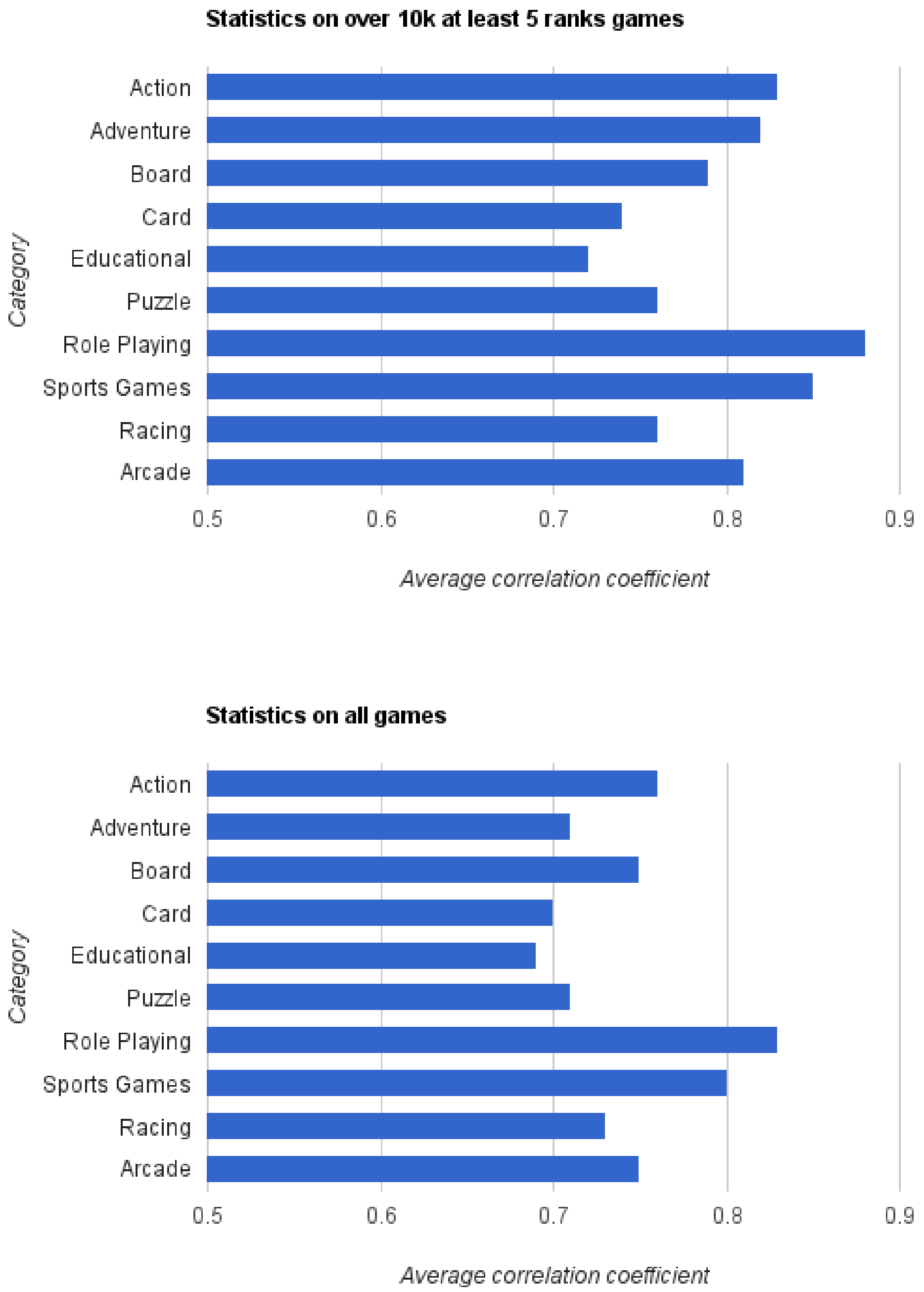


Figure 4.8: Experiment III - Games

# Chapter 5

# Conclusion

According to the experiments, half-life decay function can use to predict an app's lifecycle of popularity. An interesting point is that the popularity may influence accuracy degree. There are better results if just tests the popular apps. So accuracy is relative to popularity(either in high downloads or user ranks). Another point is the games or applications with high adhesive capasity get great accuracy. It implies the adhesive capasity is highly relative to keeping popularity. In this paper, we just individually consider the two factors: downloads and user ranks. But there are many factors about apps may influence the popularity. I think conbining some factors may get better prediction. Here we present this model and in the future we may find a better way or factors to increase the accurate rate of ranking update module.

# Bibliography

[1] Amazon web services (aws), a collection of remote computing services, also called web services, make up a cloud-computing platform offered by amazon.com. http://aws.amazon.com/.

[2] App annie, the app analytics and app data industry. https://www.appannie.com/cn/.

[3] Google analytics. http://www.google.com/analytics/.

[4] Google play, originally the android market, is a digital distribution platform operated by google. https://play.google.com/store.

[5] Imdb. http://www.imdb.com/.

[6] Metacritic. http://www.metacritic.com/.

[7] Rotten tomatoes. http://www.rottentomatoes.com/.

[8] Techcrunch, an online publisher of technology industry news. http://techcrunch.com/.

[9] Techorange. http://buzzorange.com/techorange/.

[10] Flurry, a mobile analytics, monetization, and advertising company. http://www.flurry.com/, 2005.

[11] Apps solidify leadership six years into the mobile revolution. http://flurrymobile.tumblr.com/post/115191864580/apps-solidify-leadership-six-years-into-the-mobile, 2014.

[12] Carare O. The impact of bestseller rank on demand: Evidence from the app market. *INTERNATIONAL ECONOMIC REVIEW*, 53(3):717–742, Jul 2012.

[13] Meng Cui, Songyun Hu. Search engine optimization research for website promotion. *Information Technology, Computer Engineering and Management Sciences (ICM), 2011 International Conference*, 4(2):100–103, Sep 2011.

[14] Seung-Taek Park , David M. Pennock. Applying collaborative filtering techniques to movie search for better ranking and browsing. *KDD '07 Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 7:550–559, Aug 2007.

[15] Shkapenyuk, V. , Suel, Torsten. Design and implementation of a high-performance distributed web crawler. *Data Engineering, 2002. Proceedings. 18th International Conference*, 18:357–368, Mar 2002.

[16] T. Petsas, A. Papadogiannakis, M. Polychronakis, E. P. Markatos, T. Karagiannis. Rise of the planet of the apps: a systematic study of the mobile app ecosystem. *IMC '13 Proceedings of the 2013 conference on Internet measurement conference*, 13(13):277–290, Oct 2013.

[17] Tsuchiya K, Sugita M. A mathematical model for deriving the biological half-life of a chemical. *Nordisk hygienisk tidskrift*, 52(2):105–10, 1971.