EAI Endorsed Transactions
on Industrial Networks and Intelligent Systems
Research Article ICST.ORG

Mobile Application Search: A QoS-Aware and Tag-Based Approach

Shang-Pin Ma [1,*], Shin-Jie Lee [2], Wen-Tin Lee [3], Jing-Hong Lin [1], and Jui-Hsaing Lin [1]

[1] Department of Computer Science and Engineering, National Taiwan Ocean University, Keelung, Taiwan
[2] Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan
[3] Department of Software Engineering, National Kaohsiung Normal University, Kaohsiung, Taiwan

Abstract

The availability of enormous numbers of mobile applications (apps) is driving demand for the means to search for, recommend, and manage apps. Existing search engines provide basic search functionalities that enable users to find apps by issuing query keywords; however, the ranking of search results does not always satisfy the expectations of users. This study proposes a novel approach to address this issue, called Tag-based and QoS-aware Mobile Application Search and Management (TQMASM). The proposed system provides two functionalities: (1) QoS-aware app search and tag-based app recommendation; and (2) tag-based app management. QoS-aware app search is an objective means of sorting search results by considering QoS (Quality of Service) factors, popularity, and reputation. Tag-based recommendation is used to find apps according to tags annotated by all users. Tag-based management utilizes an hierarchical tree and tag annotation to facilitate the management and usage of apps. A prototype realization of TQMASM was developed to evaluate the feasibility of the proposed approach. Experiment results demonstrate the efficacy of the proposed system in providing a satisfactory ranking of retrieved apps.

Keywords: App Search, App Recommendation, App Management, Quality of Service (QoS), Social Tag.

Received on 23 February 2015, accepted on 7 April 2015, published on 04 June 2015

doi: 10.4108/inis.2.4.e6

1. Introduction

The widespread adoption of mobile applications (apps) has led to exponential growth in the types of mobile applications available [1]. In 2013, the Google Play store officially announced that the over 1 million Android apps had been created [2]. This has driven demand for efficient means with which to search for, recommend, and manage apps.

While existing search engines on Google Play, the iTune Store, and Windows Phone Marketplace provide basic search functions that allow users to find relevant mobile apps by issuing query keywords, the ranking of search results does not always satisfy the expectations of users. Specifically, apps with a higher ranking are not necessarily any better than those with a lower ranking. This can largely be attributed to the fact that current app search engines fail to consider quality of service (QoS) in the search for apps. In Service-Oriented Computing, common QoS factors include availability, reliability, and response time [3]; however, these factors are not applicable to mobile apps because mobile apps are not purely backend functionalities. Unlike server-side web services, mobile apps are user interactive devices, which need to be installed and updated periodically. This paper adopts *popularity* and *reputation* as QoS factors to be included as parameters in the search for mobile apps. Popularity is defined as the degree to which an app is used by the public, which is expressed as long-term performance. Reputation refers to the degree to which an app can be trusted, according to recent reports regarding its quality.

* Corresponding author. Email:shangpin.ma@gmail.com

Another issue is the need to locate the apps that have already been installed on a mobile device. This can be particularly daunting when having to deal with a large number of apps, both installed and uninstalled. This study sought to develop a convenient system for the management and utilization of mobile apps. In the proposed system, the apps arranged by users within a hierarchical tree structure with a tag annotation mechanism, making it far easier for users to fetch and use the apps they require than it would be using the default app list.

The proposed Tag-based and QoS-aware Mobile Application Search and Management (TQMASM) provides two mechanisms: (1) QoS-aware app search and tag-based app recommendation and (2) tag-based app management. QoS-aware app search is an objective means of sorting and ranking search results according to scores of popularity, reputation, and Google ranking. Tag-based app search is a method for the retrieval of apps in accordance with the annotation of tags by all users. Currently, TQMASM focuses primarily on the Android platform.

The remainder of this paper is organized as follows. Related work is presented Section II. Section III outlines the details of the proposed approach. The design and implementation are presented in Section IV. Section V presents the experiments used for the evaluation of the proposed system. Section VI summarizes the benefits and features of the proposed approach and outlines directions for future work.

2. Related Works

In this section, we review several studies related to this topic and compare our approach with these efforts.

2.1. Search Methods Based on App Information

Datta [4, 5] utilized publicly available information to search for apps. This information was categorized as static and dynamic. The former includes the name of the app as well as a description, date of update, price, type, developer name, language and size. The latter includes comments, ratings, version number, ranking and number of installations. The score used for the sorting of apps is calculated according to the ranking of the app in its main category and the average scores of apps created by the same developer.

AppBrain [6] is a specialized search engine for Android apps. AppBrain can let a user login the system with his ID in one of social networking websites, such as Facebook, Google+, or Twitter, and recommend apps to the user based on his social networking data.

Appolicious [7] is an app search engine for both apps in Android and iOS platforms. Appolicious is integrated with Facebook services to provide personal recommendation based on the information of app installations of the user, the user's friends, and the communities the user has joined.

Unlike most existing methods, TQMASM emphasizes the popularity and reputation of an app to re-rank search results according to an objective QoS score.

2.2. App Search Methods Based on App Usage

Shi and Kamal [8] and Yan et al. [9] presented app search methods based on app usage, with the underlying assumption of the inefficiency of using publicly available information on the App Marketplace. Reasons behind this inefficiency include the following: (1) Few users actually rate apps; (2) ratings can be faked; and (3) ratings become obsolete in a short period of time.

Yin et al. [10] devised an AT (actual value and tempting value) model to predict whether a user would install a new app by comparing apps the user has installed and ones the user may be interested in but did not download.

Karatzoglou at al. [11] proposed a Djinn model based on context parameters (e.g. time and location) to perform the collaborative filtering of implicit data based on tensor factorization. The aim of the Djinn model is recommending apps to the user based on the current context.

Despite the fact that methods based on app-usage can be used to retrieve apps according to personal data, we sought to enable the same functionality without compromising the privacy of users. Thus, TQMASM is based on publicly available data related to apps, such as meta data, data gleaned from social networks, and the public tags submitted by users. In addition, to avoid bias in app data, we focused on long-term quality (popularity) as well as recent performance (reputation) in order to enhance the validity of all data used for evaluation.

2.3. Tag and Category

Social bookmarking enables users to add, annotate, edit, and share bookmarks associated with web documents. The folksonomy [12] indicates the mechanism used for the classification of tags built up through user tagging. This results in the formation of a user-generated taxonomy, as opposed to an authoritative hierarchical taxonomy. Noll et al. [13] employed the techniques in social bookmarking and tagging to re-rank web search results. Karlson et al. [14] provided a method to solve the problem of navigating through search results when using a mobile device. This was achieved by devising a hybrid model capable of performing iterative data filtering based on the navigation and selection of hierarchically-ordered categories (facet navigation) in conjunction with incremental text entry to further narrow search results.

The development of TQMASM enables users to manage their apps by classifying and then accessing them according to hierarchically-ordered categories as well as tags.

3. Proposed Approach

In this section, we provide an in-depth description of the proposed TQMASM approach to app management.

3.1. Tag-Based App Tree Management

This study adopted tags and hierarchically organized categories, which are commonly used in text retrieval systems, to facilitate the management of mobile apps. The proposed "app tree" enables users to list an app in a given category, regardless of whether it has been installed. Additional categories can also be introduced by the user in order to expand and customize the hierarchical structure. For example, a user could list "Facebook" and "Line" within a category named "Social Network", which is itself filed within the category "Life". Apps could also be listed in multiple categories should the user deem this convenient when browsing for apps.

User seeking to describe an app using more specific terms can also attach tags to a given app. For instance, the app Evernote could be tagged using the terms "personal", "note" or "inspiration".

Furthermore, the app tree can be applied across multiple mobile devices and desktops, which makes it possible to edit the app tree using a large desktop screen and then browse for apps via this app tree on a mobile phone.

Figure 1 presents the desktop UI wherein an app named "Amazon Shopping" is assigned to the category "Life". Any app can be moved by dragging and dropping into any category.

Figure 2 presents the desktop UI wherein the tag "shopping" is added to the app "Amazon Shopping". The new tag is added by right-clicking the app and inputting the tag content.

Figure 3 presents the UI designed for browsing apps on mobile devices using the app tree. The user can check recent information related to an app or access the app directly.

Figure 1. UI Design: Assigning apps to designated categories

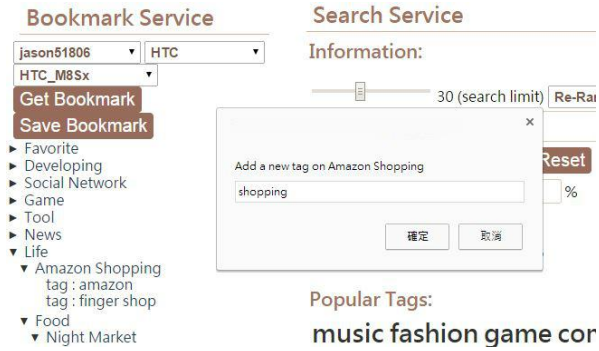

Figure 2. UI design: Adding tags to apps

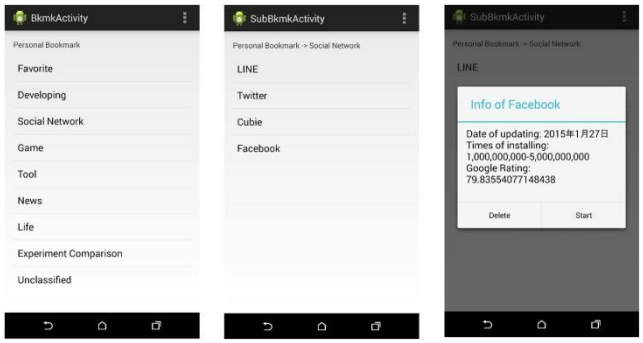

Figure 3. Browsing apps using the app tree

3.2. QoS-Aware Tag-Based App Search

As mentioned above, we devise a novel method with which to evaluate the quality of service (QoS) of apps according to popularity and reputation. We then rank the apps retrieved from Google Play according to the calculated QoS score in order to provide a more reasonable ranking of search results. In the following, we explain the popularity score, the reputation score and the Google ranking score in detail.

3.2.1 Popularity Score (PS)

Most users would prefer to choose apps that have been installed by many other users as well as those that are regularly updated [5]. We therefore sought to obtain a popularity score capable of representing long-term performance as well as the freshness of an app. We integrated the number of times of installation (TI) as well as the period elapsed since the most recent update (PUD) to determine a popularity score, using a method based on fuzzy theory [15].

The first step in this calculation involves converting crisp sets into fuzzy ones; therefore, we defined the fuzzy sets of PUD and TI separately. We designated five fuzzy sets (PUD1, PUD2, PUD3, PUD4, and PUD5) to represent five time ranges over a 24-month period, and five fuzzy sets (TI1, TI2, TI3, TI4, and TI5) to express different ranges for the number of installation.

The operations in fuzzy reasoning include Intersection (see Eq. (1)) and Union (see Eq. (2)). To produce the popularity score, we first used the Intersection operation to

compose the TI sets and PUD sets according to the aggregation rules presented in Figure 4. The fuzzy sets VL (very low), L (low), M (medium), H (high), and VH (very high) indicate different popularity rankings. In cases where there are more than one membership function for a given fuzzy set related to popularity, we use the Union operation to produce a single fuzzy degree, which represents the popularity of an app. We then use the center of gravity (COG) method to perform defuzzification [15] in order to obtain a crisp number capable of representing popularity (as shown in Eq. (3)).

$$\mu A \cap B(x) = min[\mu A(x), \mu B(x)] \quad (1)$$

$$\mu A \cup B(x) = max[\mu A(x), \mu B(x)] \quad (2)$$

	TI1	TI2	TI3	TI4	TI5
PUD5	VL	VL	L	L	M
PUD4	VL	L	M	M	H
PUD3	L	L	H	H	H
PUD2	M	M	H	VH	VH
PUD1	H	H	VH	VH	VH

VL: Very Low, L: Low, M: Medium, H: High, VL: Very High

Figure 4. Aggregation rules for TI and PUD

$$\text{PS(N)} = \text{COG} = \frac{\sum_{x=a}^{b} \mu A(x)x}{\sum_{x=a}^{b} \mu A(x)} \quad (3)$$

In the following, two apps are used as examples to illustrate the proposed popularity scoring method. PUD and TI of the first app are 6 months and 100,000~500,000, whereas PUD and TI of the second app are 16 months and 1,000,000~5,000,000.

Figure 5 illustrates the PUD membership functions of the two apps. In this example, the red dashed line is the first app with 0.8 membership degree for PUD1, 1.0 for PUD2, and 0.2 for PUD3. The blue dotted line is the second app with 0.4 membership degree for PUD3, 1.0 for PUD4, and 0.6 for PUD5.

Figure 6 presents the TI membership functions of the two apps. In this example, the red dashed line is the first app with 0.4 membership degree for TI2, 1.0 for TI3, and 0.8 for TI4. The blue dotted line is the second app with 0.6 for TI3, 1.0 for TI4, and 0.4 for TI5.

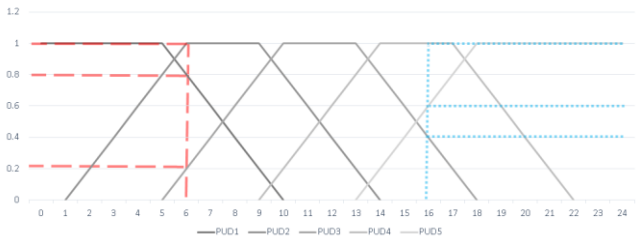

Figure 5. Fuzzy sets of PUD (period since update date)

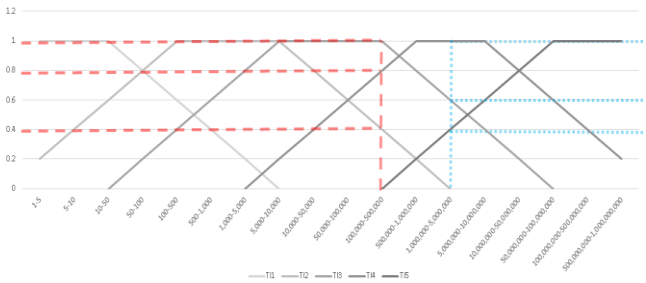

Figure 6. Fuzzy sets of TI (number of times of installation)

In Figure 4, red indicates the cells to which the first app is applied, and blue text indicates the cells to which the second app is applied. The purple text indicates the cells to which both apps are applied. For example, in the first app, to calculate the membership degree of popularity set H, the intersections of membership degree pairs are calculated first: "TI3 and PUD3", min[1.0, 1.0] = 1.0; "TI4 and PUD3", min[0.8, 1.0] = 0.8; and "TI3 and PUD2", min[1.0, 0.8] = 0.8. We then apply the union operation: max[1.0, 0.8, 0.8], from which we determine that the membership degree of H is 1.0. For the second app, we can use the same method to calculate the union operation: max[0.4, 0.4, 0.4, 0.4], from which we determine that the membership degree of H for the second app is 0.4.

Figure 7 represents the fuzzy popularity scores for the two apps. The area with light bars and the area with zigzags represent the score of the first app with 0.4 L, 0.4 M, 1.0 H, and 0.8 VH. The area with small grids and the area with zigzags represent the score of the second app with 0.6 L, 1.0 M, and 0.4 H.

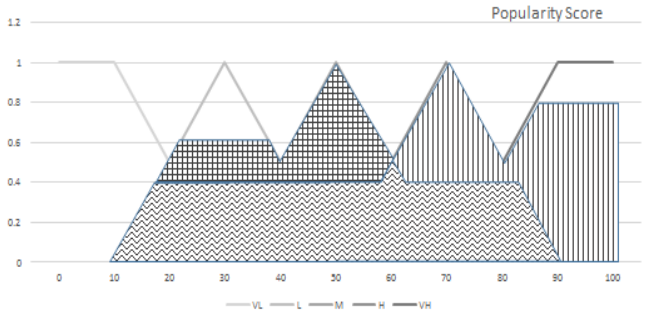

Figure 7. Popularity score

The application of COG defuzzification makes it possible to calculate the popularity score of the first app as 64.12 (vertical bar and zigzag areas), and the second app as 48.04 (small grid and zigzag areas). Although the TI score of the second app is approximately 10 times higher than that of the first app, it has not been updated for more than one year. Thus, the second app obtained a lower popularity score than did the first one. It should be noted that if an app is not updated regularly, it is very possible that it will be unable to meet the current requirements of users or may include defects that have not been adequately addressed.

3.2.2 Reputation Score (RS)

The reputation of an app is another important concern in the search for apps. In evaluating the reputation of an app, this study valued the most recent ratings more highly than past performance. In this way, the reputation score avoids ranking highly apps with good average scores but poor recent performance.

We applied the REAL (Risk-Enabled Reputation Model) method [16] to evaluate the reputation of mobile apps. The application of the REAL method progresses through three steps. Step 1 involves recording the ratings of an app for use as the basic component of the reputation model. Step 2 involves the discovery of active ratings that could be used for further calculations. Step 3 involves calculating the reputation of the app.

The application of REAL makes it possible to detect recent bad ratings related to an app in the form of active ratings. For example, the rating of the Facebook app on Google play was 3.8 on June 10, 2014, and its reputation score was 3.31. This app earned a reputation score lower than its average rating due to a number of poor reviews related to an inconvenient user interface in newer versions, which have been posted since its initial rating in June 2014. In this manner, the reputation score is able to avoid the situation in which previous high ratings would otherwise overshadow recent bad performance, which results in a more reasonable score with which to represent the reputation of an app.

3.2.3 Google Ranking Score (GRS)

Popularity and reputation scores can be used to determine the quality of an app; however, the relevance between a query term and the retrieved apps is also important. Google Play provides a high degree of precision in the retrieval of apps based on query terms; however, it provides inappropriate rankings with regard to popularity and reputation. Thus, we integrated the proposed popularity and reputation scores with Google ranking in order to enhance search results.

The Google Ranking Score (GRS) is defined as follows. We first set two constants (MAX_SCORE: 90 and RANK_GROUP: 3). We then set a variable called *reduction*, which is calculated using Eq. (4). Note that the value of (N / RANK_GROUP) is rounded down to the integer. GRS is calculated using Eq. (5).

$$\text{reduction}(N) = 0.3 \times \left(1 + \frac{N}{\text{RANK\_GROUP}}\right) \quad (4)$$

$$\text{GRS}(N) = \text{GRS}(N-1) - \text{reduction}(N) \quad (5)$$

The rationale behind the concept of reduction is based on the observation that apps with a high ranking (such as ranks 1~6) are usually highly related to the query keywords, whereas apps with low ranking (such as ranks 25~30) may be entirely unrelated to the keyword. Thus, we define the *reduction* score as the score to be subtracted. Basically, the lower the ranking is, the higher the *reduction* score is. According to this definition, the app with the highest ranking in Google Play is subject to no reduction and therefore obtains a score of 90 (MAX_SCORE). The reduction of the N^{th} app is 0.3 * (1 + (N / 3)), and its score is equal to the previous score with the reduction score subtracted. For example, the reduction of the second app is 0.3, such that the final score is 89.7. In another example, the score of the 29th app is 43.8 and the reduction of 30th app is 3such that the score of the 30th app is 40.8.

3.2.4 Aggregating Popularity Score and Reputation Score into the QoS Score

Examining the long term as well as recent performance makes it possible for TQMASM to provide a reasonable ranking of search results for a given query. The QoS score of mobile app *a* is calculated by aggregating the popularity, reputation, and Google ranking scores based on weights *w1, w2 and w3* (as shown in Eq. (6)).

$$\text{QoS}(a) = w1 * \text{PS}(a) + w2 * \text{RS}(a) + w3 * \text{GPS}(a) \quad (6)$$

3.3. Tag-Based App Recommendation

We also provide an app recommendation mechanism based on annotated tags from all users. The basic idea is that if the user is willing to put an app into his app tree, then the app is probably worth recommending.

To realize the recommendation feature, we used the commonly used information retrieval technique, TD-IDF (term frequency - inverse document frequency), for the ranking of apps according to extracted indices. In order to build tag indices, TQMASM gathers all tags and category names from the app trees of all users and store them as tag indices. TF (term frequency) refers to the importance of a tag index for an app. TF is obtained by calculating the proportion of the tag index for an app to the total amount of all tag indices for an app. IDF (inverse document frequency) indicates the degree to which a tag can be distinguished for all apps. IDF is obtained by calculating the proportion of the total number of apps to the number of apps annotated with this tag, and taking the logarithm of the proportion. The TF-IDF value is calculated by multiplying TF and IDF.

1	Raohe	Night Market	Somgshan	Wu Fen Pu
TF	3/22	11/22	6/22	6/22
IDF	log(205/2)	log(205/9)	log(205/6)	log(205/1)

2	Night Market	Taiwan
TF	4/5	1/5
IDF	log(205/9)	log(205/5)

Figure 8. TF-IDF score of candidate apps

Figure 8 displays the TF-IDF information of two illustrative apps. There are 205 apps in the app repository. The app #1 is "Real Raohe Night Market" ("正港饒河夜市" in Chinese), and app #2 is "Night Market Plan" ("夜市通"

in Chinese). The first candidate app was tagged "Raohe" ("饒河" in Chinese, the name of the Night Market) 3 times, "Night Market" 11 times, "Songshan" ("松山" in Chinese, a region name) 6 times, and "Wu Fen Pu" ("五分埔" in Chinses, a place name) 6 times, and the second one "Night Market Plan" was tagged "Night Market" 4 times and "Taiwan" 1 times. There are 2 apps annotated with "Raohe", 9 apps with "Night Market", 6 apps with "Songshan", 1 apps with "Wu Fen Pu", and 6 apps with "Taiwan" .

Issued query terms are "Raohe", "Night Market", "Wu Fen Pu", "Songshan", and "Taiwan". The detailed data of TF and IDF are shown in Figure 8. The final TF-IDF scores of these two candidate apps are 1.47 and 1.41 respectively.

$$Score1 = \log(205/2)*(3/30) + \log(205/9)*(11/30) + \log(205/6)*(6/30) + \log(205/1)*(6/30) = 1.47$$

$$Score2 = \log(205/9)*(4/5) + \log(205/5)*(1/5) = 1.41$$

Figure 9 presents the retrieved results for the keyword "movie" using tag-based app recommendation of TQMASM. The apps are ranked by the TF-IDF scores. Recommended apps are closely related to the keyword "movie" .

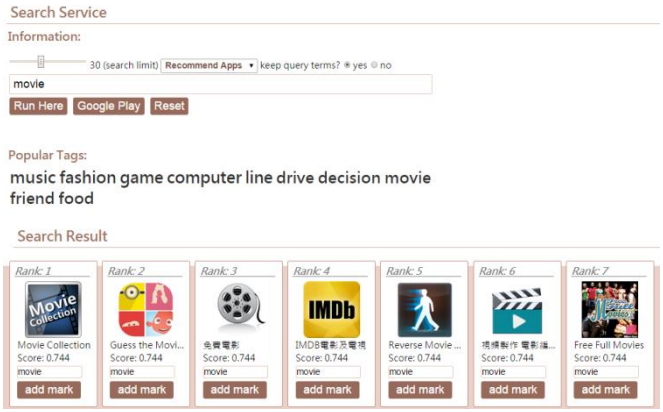

Figure 9. Results of tag-based recommendation

4. Design and Implementation

Figure 10 presents the architecture of the prototype system used to realize TQMASM. The prototype was developed using Android (mobile client-side), HTML and jQuery (web client-side), and Java Servlet technology (server-side). App data are extracted from Google Play.

Mobile App UI (MAU) is the client-side module used to provide all required user interfaces for the mobile device. Users can create their own app tree and utilize apps on the MAU. App Manage Service (AMS) is the controller used to invoke the App Tree Handler, which creates and modifies the app tree and tags. The App Tree Database contains category and tag data of all users. Desktop Web UI (DAU) is the user interface used to search for and manage apps. App Search Services (ASS) are the controllers used to coordinate the Popularity Scoring Model (PSM), Reputation Scoring Model (RSM), and Tag-based Search Model (TSM). RSM generates a score according to an app's recent ratings. PSM produces a score according to an app's TI (number of installation) and PUD (period elapsed since last update). TSM builds tag indices and calculates TF-IDF scores to facilitate the recommendation of apps. Member Service (MS) enables users to log into the system.

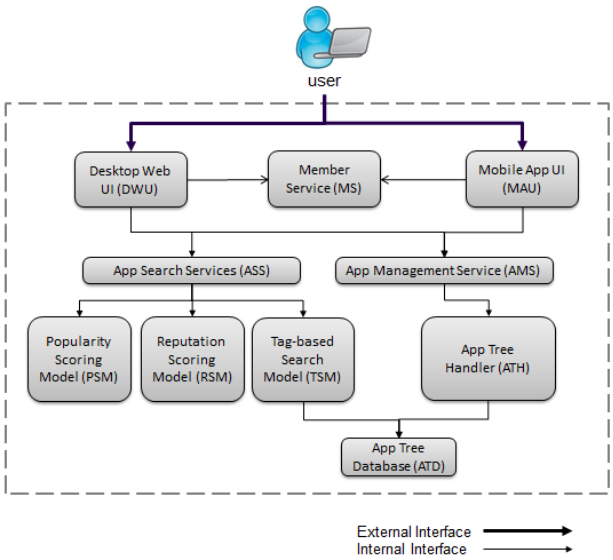

Figure 10. Architecture of the prototype system

This implementation was realized as an Android app and a Web application hosted on a desktop computer with the following configuration: Intel Core 3.07GHz with 8G RAM, 500G hard disk, and Windows 7 (64bit).

Figure 11. Prototype system: QoS-aware search

In the following, we present a simple case to illustrate the search functionality of the app search. In this case (see Figure 11), the user enters the search term "note" to find apps with note services. The ranked search results returned two apps with very different rankings: (1)"Evernote" has a higher ranking (from rank 6 to rank 1), and (2) "Class Note Lite" has a lower ranking (from rank 2 to rank 12). The first app is much fresher, has been installed far more frequently, and has a better rating than the second app. Evernote is widely recognized as the best app for keeping, managing, and sharing notes, but it is not in the first rank using Google Play. In contrast, Class Note Lite provides many functionalities; however, many users feel that the user interface is not satisfactory and the app has not been updated for a long time (more than 7 months). The revised ranking

of these apps appears more reasonable than the original ranking.

5. Evaluation of proposed system

Experiments were conducted to evaluate the effectiveness of the QoS-aware app search mechanism. We chose not to conduct app-tree-based management and tag-based app recommendations for the following reasons: (1) App-tree-based management is merely a design concept, which makes it difficult to identify a quantitative evaluation method; (2) Tag-based app recommendation is not widely used at present, which makes it impossible to deduce meaningful results due to limited test data. Thus, we focused on evaluating the performance of the QoS-aware app search by comparing five methods used to search for apps:

(1) Google Play: Official mobile app marketplace provided by Google.
(2) PS+RS: QoS score comprising popularity score and reputation score, but not including the Google Play score.
(3) PS+GRS: QoS score comprising popularity score and Google ranking score, but not including the reputation score.
(4) RS+GRS: QoS score comprising reputation score and Google ranking score, but not including the popularity score.
(5) TQMASM: Proposed TQMASM approach, integrating the popularity, reputation, and Google ranking scores.

Ten users were invited to issue three queries via TQMASM (total 30 queries), and actually install and use the top 15 apps obtained for each query. This meant that each user was required to install and browse approximately 45 apps. According to their experience, the users were asked to provide ratings ranging from 0 to 100 for each of the installed apps. The rating guidelines were as follows: (1) Worst: 0~20 points; (2) Bad: 21~40 points; (3) Medium: 41~60 points; (4) Good: 61~80 points; and (5) Excellent: 81~100 points.

The measurement indicator used in this research is the Kendall tau metric [17, 18], a method used for the calculation of the non-parametric correlation coefficient (See Eq. (7)). The calculated coefficient ranges from -1 to 1 with -1 referring to the most negative correlation, 0 referring to zero correlation, and 1 referring to the most positive correlation. We analysed the correlation between the user rankings and the rankings provided by the above five app search alternatives. For further analysis, we respectively calculated the average τ values of 30 queries for the top 5, top 10, and top 15 apps retrieved in order to evaluate the performance of the proposed system.

$$\tau_n = \frac{c-d}{\frac{1}{2}n(n-1)} \qquad (7)$$

where *c* is the number of concordant pairs, *d* is the number of discordant pairs, and *n* refers to the number of included apps.

Based on the above experiment configuration, two kinds of experiments were conducted: parameter setting and performance evaluation. For parameter setting, we compared the sum of the average top-5 correlation, average top-10 correlation, and average top-15 correlation for all possible parameter combinations in Eq. (6). In other words, all combinations of *w1* (for popularity score), *w2* (for reputation score), and *w3* (for Google ranking score) were considered in this experiment. According to our results in Figure 12, the best parameter setting is 4:2:4 (total correlation of 0.866); i.e., *w1* (for popularity score) is 0.4, *w2* (for reputation score) is 0.2, and *w3* (for Google ranking score) is 0.4.

For the evaluation of performance, we compared the proposed TQMASM approach with the four other above-mentioned alternatives (Google Play, PS+RS, PS+GRS, and RS+GRS) resents the final experiment results. The proposed TQMASM approach exhibits performance far exceeding all other methods for the top-5, top-10, and top-15 retrieved apps. These findings indicates that (1) the proposed TQMASM yields search rankings better suited to the expectations of users; (2) The popularity score (PS) improves the degree of satisfaction users have for search rankings, whereas the reputation score cannot be applied independently; and (3) the integration of the popularity score (PS), reputation score (RS), and Google ranking score (GRS) is necessary to obtain appropriate search rankings.

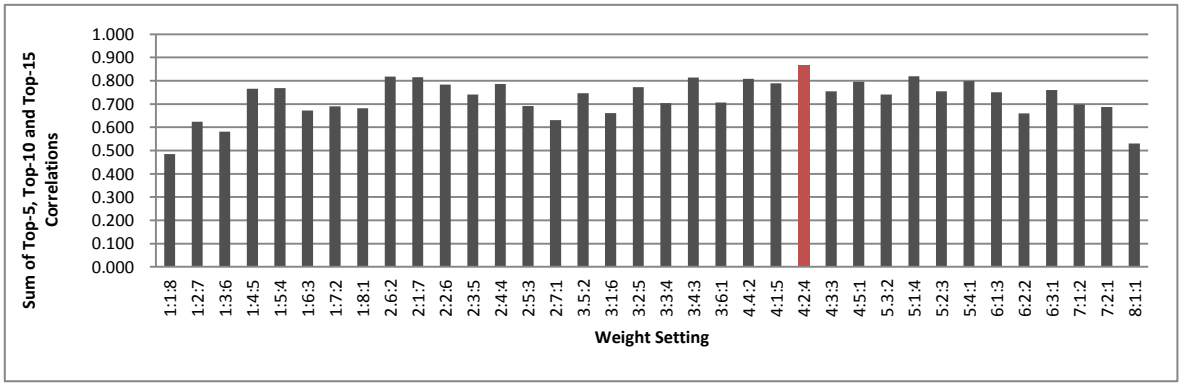

Figure 12. Parameter settings

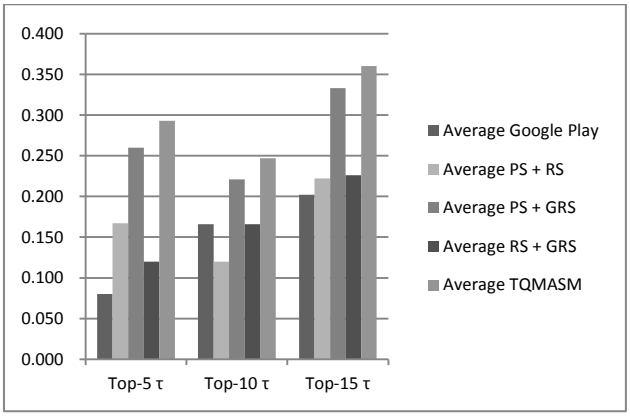

Figure 13. Experiment results

6. Conclusion

This paper presents a novel approach to the management of mobile apps, referred to as Tag-based and QoS-aware Mobile Application Search and Management (TQMASM). This system includes three main functionalities:

- Objective search ranking, which takes into account the popularity as well as the reputation of an app
- Retrieval mechanism to find mobile apps according to tags annotated by all users
- Category-based app management mechanism to facilitate the management of and access to mobile apps

Our future research plans include (1) devising an automatic or semi-automatic tag annotation method to reduce the effort required for categorization; (2) periodic analysis of app trees to enable the active recommendation of relevant newly-published mobile apps; and (3) integrating the QoS-aware app search approach with tag-based recommendation to improve the precision of app searches.

Acknowledgements

This research was sponsored by Ministry of Science and Technology in Taiwan under the grant MOST 103-2221-E-019-039.

References

[1] S.-P. Ma, J.-S. Jiang, and W.-T. Lee. *Service Brick Composition Framework for Smartphones*. in *20th Asia-Pacific Software Engineering Conference (APSEC 2013)*. 2013.

[2] *Android's Google Play beats App Store with over 1 million apps, now officially largest*. 2013; Available from: http://www.phonearena.com/news/Androids-Google-Play-beats-App-Store-with-over-1-million-apps-now-officially-largest_id45680.

[3] S.-P. Ma, C.-L. Yeh, and P.-C. Chen, *Service Composition Management: A Risk-Driven Approach*. Journal of Universal Computer Science, 2014. **20**(3): p. 302-328.

[4] A. Datta, K. Dutta, S. Kajanan, and N. Pervin, *Mobilewalla: A Mobile Application Search Engine*, in *Mobile Computing, Applications, and Services*, J. Zhang, J. Wilkiewicz, and A. Nahapetian, Editors. 2012, Springer Berlin Heidelberg. p. 172-187.

[5] A. Datta, S. Kajanan, and N. Pervin, *A Mobile App Search Engine*. Mobile Networks and Applications, 2013. **18**(1): p. 42-59.

[6] *AppBrain*. Available from: http://www.appbrain.com/.

[7] *Appolicious*. Available from: http://www.appolicious.com/.

[8] K. Shi and K. Ali, *GetJar mobile application recommendations with very sparse datasets*, in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2012, ACM: Beijing, China. p. 204-212.

[9] B. Yan and G. Chen, *AppJoy: personalized mobile application discovery*, in *Proceedings of the 9th international conference on Mobile systems, applications, and services*. 2011, ACM: Bethesda, Maryland, USA. p. 113-126.

[10] P. Yin, P. Luo, W.-C. Lee, and M. Wang, *App recommendation: a contest between satisfaction and temptation*, in *Proceedings of the sixth ACM international conference on Web search and data mining*. 2013, ACM: Rome, Italy. p. 395-404.

[11] A. Karatzoglou, et al., *Climbing the app wall: enabling mobile app discovery through context-aware recommendations*, in *Proceedings of the 21st ACM international conference on Information and knowledge management*. 2012, ACM: Maui, Hawaii, USA. p. 2527-2530.

[12] S. Hayman. *Folksonomies and tagging: New developments in social bookmarking*. in *Ark Group Conference: Developing and Improving Classification Schemes*. 2007.

[13] M.G. Noll and C. Meinel, *Web search personalization via social bookmarking and tagging*, in *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*. 2007, Springer-Verlag: Busan, Korea. p. 367-380.

[14] A.K. Karlson, G.G. Robertson, D.C. Robbins, M.P. Czerwinski, and G.R. Smith. *FaThumb: a facet-based interface for mobile search*. in *Proceedings of the SIGCHI conference on Human Factors in computing systems*. 2006: ACM.

[15] X. Liu, *Parameterized defuzzification with maximum entropy weighting function-Another view of the weighting function expectation method.* Math. Comput. Model., 2007. **45**(1-2): p. 177-188.

[16] J. Lee, S.-J. Lee, H.-M. Chen, and C.-L. Wu, *Composing web services enacted by autonomous agents through agent-centric contract net protocol.* Information and Software Technology, 2012. **54**(9): p. 951-967.

[17] R. Nelson, *Kendall tau metric.* Encyclopaedia of Mathematics, 2001. **3**: p. 226-227.

[18] R. Fagin, R. Kumar, and D. Sivakumar, *Comparing top k lists*, in *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*. 2003, Society for Industrial and Applied Mathematics: Baltimore, Maryland. p. 28-36.