Chapter 1
INTRODUCTION

Nowadays, mobile applications (apps), which enable us to perform more and more tasks through smart phones, tablets or other terminals, are becoming increasingly prevalent and ubiquitous in our daily life. To accelerate the development of mobile apps, more and more third-party services, which can be consumed as backend services of mobile apps, are available in Mobile-Backend-as-a-Service (MBaaS) or Platform-as-a-Service (PaaS) systems (e.g., Parse [12], Kinvey [13] and IBM Bluemix [14). Through consuming mobile backend services, developers can release mobile apps with rich user experiences much more rapidly than before. However, with the market growth of mobile backend services, more and more equivalent services with same or similar functionalities (e.g., nearby restaurant recommendation service) are released in MBaaS or PaaS systems. When facing more and more choices, it turns to be a more and more challenging process for mobile developers to discover mobile backend services with high ROIs. Many existing solutions have been proposed to recommend appropriate Web services for web or desktop developers (e.g., based on keyword matching, popularity ranking, consumer application clustering, and etc.), while few of them considers the unique characteristics of mobile backend services. Therefore, their recommendation results are not effective enough for mobile developers. Although most mobile backend services are implemented and provided as traditional or RESTful Web services, they indeed have multiple unique characteristics over traditional Web services from the perspective of their consumers (i.e., mobile apps). For example, since various of device data (e.g., screen resolution, network connection mode), sensor data (e.g., GPS sensor) and user data (e.g., user's orientation) in mobile devices can provide good hints to improve results of mobile backend services (e.g., nearby restaurant recommendation), if a mobile backend service makes good use of mobile data, the service probably will be more suitable for the consumption of mobile apps. However, to the best of our knowledge, there is no existing work that explicitly focuses on such unique characteristics of mobile backend services. Therefore, directly applying existing solutions for the recommendation of mobile backend services is not effective enough.

Chapter 2
MBaaS and Its Characteristics

2.1 MBaaS

Mobile Backend as a Service, also known as 'MBaaS', is an efficient computing architecture that connects mobile applications to cloud computing services. These platforms help you to reduce the time that is required to build the mobile applications. MBaaS allows the developers to focus on complex and core features instead of the low-level tasks. MBaaS platforms are also preferred over Mobile Enterprise Application Platforms (MEAPs).

2.2 Characteristics of MBaaS

2.2.1 Mobile Data Sensitivity:

Mobile backend services usually rely on mobile data (e.g., mobile device characteristics, real-time data collected by mobile device sensors, or user data stored in mobile devices) to improve their analysis results. However, such personal data are very sensitive for users.

2.2.2 Data Traffic Efficiency:

To avoid high charges of users' mobile data traffic, mobile backend services should not introduce large data transmission volume.

2.2.3 Responsiveness:

Since mobile users usually have little tolerance for slow responses, mobile backend services need to be implemented in right ways (e.g., asynchronous mode) to improve their responsiveness.

2.2.4 Adaptiveness:

Since the user quantity of a mobile app is not predictable and can potentially have an explosive growth in seconds or minutes, mobile backend services should also be adaptive enough to react to such unpredictable changes.

Chapter 3

PROPOSED SYSTEM

Based on the summarized unique characteristics, we propose an intelligent recommendation approach for mobile backend services. In the approach, given a specific natural-language-based feature requirement description from mobile developers, our approach first applies Natural Language Processing (NLP) based techniques to identify feature-relevant mobile backend services, and then prioritize them based on ranking scores that will be flexibly computed based on metrics corresponding to unique characteristics of mobile backend services as well as interests or feedbacks of mobile developers. Note that, although currently we propose and implement four metric analyzers for the above-mentioned unique characteristics of mobile backend services, the approach supports the extending of metric analyzers. For example, if we find a new unique characteristic of mobile backend services in the future, we can implement a new metric analyzer with certain interfaces defined by the approach for the new characteristic. Once new analyzer is provided, the approach can automatically use new metrics to prioritize recommendation results.

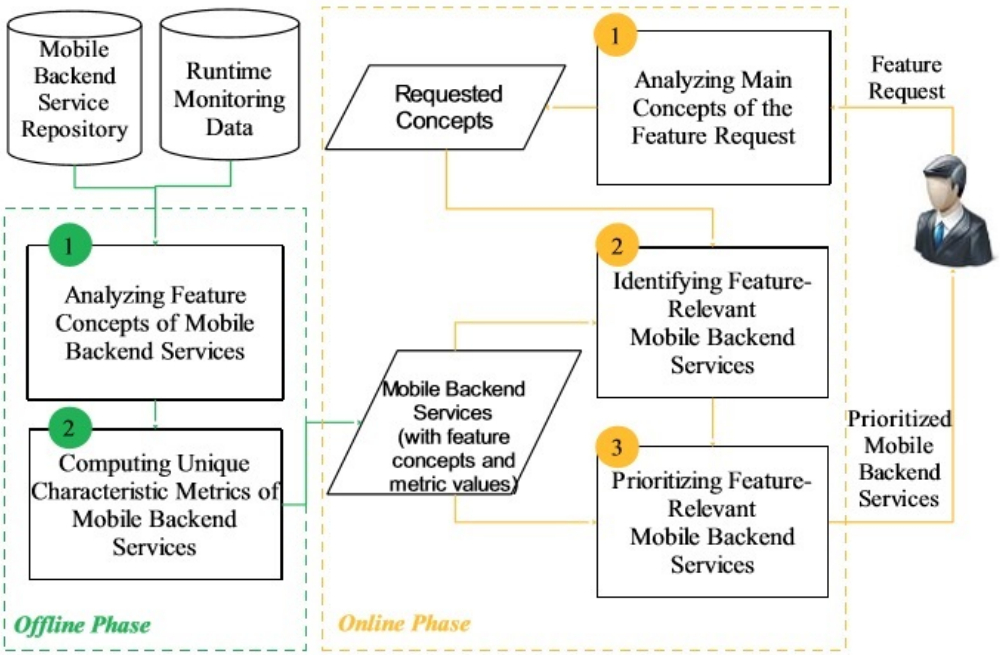

Fig. 3.1 Approach Overview

In general, our approach includes two phases: the offline phase and the online phase. In the offline phase, we first analyze feature concepts of mobile backend services based on their specifications. Then we compute unique characteristic metrics of mobile backend services based on their specifications stored in their registered repository and their runtime monitoring data captured by their hosting platform. Section 3.1 describes the availability of the inputs required by this phase. Section 3.2 and Section 3.3 describes the details of the first step and the second step of this phase, respectively. In the online phase, given a feature request from a mobile developer, we first apply NLP-based techniques (e.g., topic analysis) to analyze main concepts of the feature request. Then, through matching the main concepts of the feature request with feature concepts of mobile backend services in the repository, we identify out the feature relevant mobile backend services. After that, we prioritize the identified mobile backend services based on their unique characteristic metrics as well as preferences of the mobile developer. Finally, the prioritized mobile backend services will be recommended to the mobile developer. Section 3.4 describes the analysis process of main concepts of feature requests from mobile developers. Section 3.5 describes the identification process of feature relevant mobile backend services. Section 3.6 describes the prioritization process for the identified feature relevant mobile backend services.

3.1 Availability of Inputs in the Offline Phase

The inputs of this phase includes a mobile backend service repository, which provides limited mobile backend services in the form of RESTful Web Services with their specifications (e.g., feature description, input parameters with descriptions, and output parameters with descriptions), and also runtime monitoring data, which captures performance or runtime data (e.g., request timestamp, response time, input data size, output data size of each consumption) during actual consumption scenarios.

With the rapidly growth of MBaaS or PaaS platforms, more and more backend services are being deployed on and managed by those platforms. Since that almost each of such platforms provides a catalog of its managed services, the input “mobile backend service repositories” of this phase in our approach can be easily extracted from such catalogs. Besides that, since such services are totally managed and controlled by such platforms, it will be very easy and natural for such platforms to monitor the runtime of

Their managed service instances. During the monitoring process, the input "runtime monitoring data" required by our approach can also be easily captured.

3.2 Analyzing Feature Concepts of Mobile Backend Services

This step accepts a mobile backend service repository, which includes specifications (i.e., service description, input parameter description, and output data description) of the mobile backend services registered in it, as the input and returns a limited list of feature concepts for each mobile backend service in the repository as outputs. In this step, for each mobile backend service, we first merge all of its description information (i.e., service description, input parameter description and output description) into a single paragraph. Then, we successively apply NLP techniques (e.g., Apache OpenNLP) for word segmentation, Part-Of-Speech (POS) tagging, and word stemming. After that, we compute frequency values of all entities (i.e., noun words) and actions (i.e., verb words). Based on their frequency values, we prioritize entities and actions involved in such paragraph to identify top N (e.g., 5) of them as the feature concepts of the mobile backend service

3.3 Computing Unique Characteristic Metrics of Mobile Backend Services

In this step, we compute four unique characteristic metrics for each mobile backend service: mobile data sensitivity, data traffic efficiency, responsiveness, and adaptiveness. In our approach, this step can be easily extended: once we identify some new unique characteristic of mobile backend services, we can implement a new unique characteristic metric anlayzer with specified interfacts and then easily applied it into this step (i.e., adding some configuration parameters).

To compute mobile data sensitivity, we pre-summarize a list of mobile-data-related words or phases (e.g., geolocation, latitude, longitude, screen resolution, network connection mode, Wi-Fi, gravity, acceleration, contact, call history and etc.). For each mobile backend service under analysis, through checking that whether names and descriptions of its input parameters contain any word in the pre-summarized list, we identify whether the service consumes mobile data.

If more input parameters of a mobile backend service are related to mobile data, we would set a higher value as the mobile data sensitivity of the service.

The other three metrics (i.e., data traffic efficiency, responsiveness, adaptiveness) are computed based on the runtime monitoring data captured by hosting platforms for mobile backend services.

To compute the data traffic efficiency of a mobile backend service under analysis, we use the monitored sizes of input/output data of its historical consumptions to compute the average transmission data size of the service. If a mobile backend service has a larger average transmission data size, we would set a lower value as the data traffic efficiency of the service.

To compute the responsiveness of a mobile backend service under analysis, we use the monitored response time of its historical consumptions to compute the average response time of the service. If a mobile backend service has a longer response time, we would set a lower value as the responsiveness of the service.

To compute the adaptiveness of a mobile backend service under analysis, we first use the monitored starting timestamps of its historical consumptions to compute consumption amount per second. Then, we compute the average response time of the consumptions in each second. If, as consumption amount per second grows, the corresponding average response time of related consumptions increases more and more largely, we will set a lower value as the adaptiveness of the service.

3.4 Analyzing Main Concepts of a Feature Request

This step accepts a natural language based feature request description as the input, and returns a list of main concepts extracted from the feature request as outputs. In this step, for any given feature request, we successively apply NLP techniques (e.g., Apache OpenNLP) for word segmentation, POS tagging, and word stemming. After that, we compute frequency values of all entities (i.e., noun words) and actions (i.e., verb words). Based on their frequency values, we prioritize entities and actions involved in such feature request to identify top N (e.g., 5) of them as the main concepts of the feature request.

3.5 Identifying Feature-Relevant Mobile Backend Services

With main concepts of a feature request, in this step, we identify out the feature relevant mobile backend services by matching them with feature concepts of mobile backend services in the repository. If there are more shared feature concepts between a mobile backend services with the feature request, we would set a higher value as the relevancy of the mobile backend service to the feature request. After computing relevancy values, we identify top N (e.g., 100) mobile backend services based on their relevancy values as the feature-relevant mobile backend services.

3.6 Prioritizing Feature-Relevant Mobile Backend Services

In this step, we prioritize feature-relevant mobile backend services based on their metrics computed in the step described in Section 3.3. For each feature-relevant mobile backend service S, we compute a ranking score Rs based on the following formula:

$$R_S = \sum_{i=1}^{n} W_{m_i} * V_{m_i}^{s}$$

In this formula, $\mathbf{m_i}$ means the $\mathbf{i^{th}}$ metric of each mobile backend service, $\mathbf{w_m}$ means the weight of $\mathbf{m_i}$ for each mobile developer, $\mathbf{Vm_i^s}$ means the value of the metric m_i of the mobile backend service **S**. To compute the ranking score $\mathbf{R_s}$ for each mobile backend service (e.g., S), we first compute a weighted value for each metric and then add them together as the ranking score $\mathbf{R_s}$. In our approach, weight values of metrics are initially set to 1 and would updated based on mobile developers feedback (e.g., if a mobile developer prefers to some certain metrics, corresponding weights of those metrics would be increased for the developer).

Chapter 4

EXPERIMENTAL EVALUATIONS

In this section, we use open data, which is available from the world-leading API registration platform ProgrammableWeb [1], to evaluate the effectiveness of the prototype implemented based on the proposed approach. Based on the category tags (e.g., Coupons, Discounts, Weather, Analytics and etc.) that are provided by ProgrammableWeb and have potential to serve as mobile backend services, we first filter out two hundreds of Restful services and manually extracted out their description information and specifications to construct a service repository. Due to the missing of the runtime monitoring data of these services, we implement a simulator to randomly generate runtime monitoring data for each service. For the input of the online phase, we prepare a feature request based on a real requirement from an IBM internal mobile project: “enable users to get real-time weather information in his or her current location”.

In the evaluation, based on the repository and the simulated runtime monitoring data, we run the prototype to pre-analyze feature concepts of services in the repository and also compute four metrics for each service. Then we run the prototype to recommend mobile backend services based on the feature request. We manually examine the top results to verify the effectiveness of our approach. Through examining details of the top 5 recommended mobile backend services, we find that the prototype is effective to recommend the services that own more unique mobile characteristics with higher priorities.

To further demonstrate the effectiveness of the approach, we use real-world consumption data of these services (e.g., number of mashups built on each service for mobile users) from ProgrammableWeb to evaluate the effectiveness of the approach. In the evaluation, we only use the metric “mobile data sensitivity” to prioritize services (as other three metrics are not computed based on data from ProgrammableWeb). Through examining the top 5 recommended mobile backend services, we find that the prototype indeed recommends the services with more actual consumers (i.e., existing mobile services or apps built on them) with higher priorities.

Chapter 5

RELATED WORK

Since service recommendation is a hot research topic, many existing recommendation solutions for web services have been proposed. However, none of them consider unique characteristics of mobile backend services.

Meng et al. [3] develop a keyword-matching-based service recommendation method named KASR. Li et al. [4] propose a recommendation method which is based on the topic-level semantic matching. Tang et al. [5], Ahmed et al. [16], and Zheng et al. [17] consider non-functional properties like QoS to recommend services. Cao et al. [18] exploit interests of users based on their service consumption history to recommend right ones. Xu et al. [19] present a social-aware service recommendation approach. In the approach, they build a coupled matrix model to identify social relationships among potential users, topics, mashups, and services and then use it to recommend services.

Chapter 6

CONCLUSION

We first systematically studied and summarized out four unique characteristics of mobile backend services (i.e., Mobile Data Sensitivity, Data Traffic Efficiency, Responsiveness, and Adaptiveness). Based on such unique characteristics of mobile backend services, we proposed an extendable recommendation approach for mobile backend services to help mobile developers to efficiently identify what they really need. Through empirical evaluations we conducted on hundreds of mobile backend services, we demonstrated the effectiveness and efficiency of the proposed approach.

REFERENCES

[1]. ProgrammableWeb, URL: http://www.programmableweb.com/.

[2]. André Ribeiro, and Alberto Rodrigues da Silva, Survey on CrossPlatforms and Languages for Mobile Apps, International Conference on the Quality of Information and Communications Technology.2012.

[3]. Meng S, Dou W, Zhang X, et al. KASR: A Keyword-Aware Service Recommendation Method on MapReduce for Big Data Application, TPDS, 2013.

[4]. Li C, Zhang R, Huai J, et al. A Probabilistic Approach for Web Service Discovery Services Computing, 2013.

[5]. Tang M, Jiang Y, Liu J, et al. Location-aware collaborative filtering for QoS-based service recommendation, IEEE International Conference on Web Services (ICWS), 2012.

[6]. Ahmed W, Wu Y, Zheng W. Response Time based Optimal Web Service Selection, IEEE Transactions on Parallel and Distributed Systems, 2013.

[7]. Zheng Z, Ma H, Lyu M R, et al. Collaborative web service QoS prediction via neighborhood integrated matrix factorization, IEEE Transactions on Services Computing, 2013.

[8]. Cao B, Liu J, Tang M, et al. Mashup Service Recommendation based on User Interest and Social Network, IEEE International Conference on Web Services (ICWS), 2013.

[9]. Xu W, Cao J, Hu L, et al, A Social-Aware Service Recommendation Approach for Mashup Creation, IEEE International Conference on Web Services (ICWS), 2013.

[10]. Chen L, Hu L, Zheng Z, et al. Wtcluster: Utilizing tags for web services clustering, Service-Oriented Computing, Springer Berlin Heidelberg, 2011.

[11]. StackOverflow, URL: http://stackoverflow.com/search?q=mobile+backend+service

[12]. Parse, URL: https://www.parse.com/

[13]. Kinvey, URL: http://www.kinvey.com/

[14]. IBM Bluemix, URL: https://console.ng.bluemix.net/

[15]. Rapidvalue, URL: http://www.rapidvaluesolutions.com/