University of Nottingham
UK | CHINA | MALAYSIA

Inferring User Needs & Tasks from App Usage Interactions

Thesis submitted to the University of Nottingham for the degree of **Doctor of Philosophy, Oct 2021.**

Yuan Tian

14295697

Abstract

Mobile devices have become an increasingly ubiquitous part of our everyday life, which are not only used for basic communication. Nowadays, the need for mobile services arises from a broad range of requirements include both single app usage (e.g., check on the weather) and complex task completion (such as planning vacation) which may lead to lengthy operations within distinct apps. Understanding how users interact with apps could provide us great signals for profiling users and help service providers/app developers/smartphone manufacturers to improve user experience and retention. Therefore, in this thesis, we present work towards inferring user needs and tasks from their app usage interactions.

Firstly, we aim to better understand users' behaviour on using one particular app under different contexts. There have been many researchers proposed models for recommending the app user would use next proactively. However, less work has been conducted to enhance the app usage prediction when a new user comes whose information is insufficient for learning. Additionally, besides predicting which app users would use, we aim to further investigate if the app dwell time could also be modelled based on various user characteristics and contextual information. By conducting the comprehensive analysis and experiments, we demonstrate that users' next app and the time spent could be effectively predicted at the

same time.

Other than effectively serving the individual apps that correspond to users' simple needs, we aim to further understand the high-level tasks within users' minds while engaging with different apps. We focus on identifying and characterizing tasks from app usage behaviour and then leveraging the extracted task information for improving mobile services. We first present an automatic method that accurately determines mobile tasks from users' app usage logs based on a set of features. Given the extracted tasks, we further investigate if there are common patterns that exist among all the complex mobile tasks. Finally, we demonstrate that the extracted task information could benefit user profiling in demographics prediction and next task prediction, especially when compared to the traditional app-based methods.

To summarize, in this thesis, we conduct a more comprehensive study on modelling users app usage behaviour. Additionally, we propose to set the stage for evaluating mobile apps usage, not on a per-app basis, but on the basis of users' tasks. Finally, we provide the initial steps in shaping future research on investigating whether and how the extracted tasks could be applied for improving mobile services.

Acknowledgements

First and foremost, I am indebted at my supervisor Dr Ke Zhou for offering me the chance to pursue PhD research at the University of Nottingham. I cannot thank Dr Zhou enough. He immensely helped me shape my research career as it stands today. He taught me insights about doing research, asking the right questions, emphasizing the role of different parts in systematic research and how to best answer research questions. I really appreciate his continuous support of my PhD study and research, for his patience, motivation, enthusiasm and immense knowledge. I would never forget his encouragement "you are improving" and "you are achieving there" when I felt depressed while papers got rejected. Thank you Dr Zhou for being an inspirational guide along my PhD journey.

I want to thank Dr Joel Fischer and Prof Derek McAuley for serving on my PhD committee. I am also honoured to have Dr Natasa Milic-Frayling and Prof Thomas Gärtner as my secondary supervisors. I appreciate their thoughtful comments and feedback on my annual reviews and thesis. Many thanks to Dr Mounia Lalmas, Dr Dan Pelleg and Prof Yiqun Liu, who all contributed to the successful completion of my research works published during my PhD.

I also want to thank other members of the UoN family, including my labmates and good friends, Zeyang Liu, Wen Zheng, Weiyao Meng&Yuzheng Chen, and Feng Chen, who made my time at Nottingham enjoyable. The path towards my PhD wasn't always as cheerful as I would have liked, and I also wish to thank Wenwen Li and Xinyu Zhang for providing support in tough times.

Last but not least, I am very grateful to my parents for their love and unconditional support during these years. All I am I am because of their support, sacrifices and encouragement throughout my life.

Dedicated to my parents, Yali Gou & Shuangke Tian!

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Mobile devices nowadays have become indispensable personal gadgets to support our activities in almost every aspect of our lives [138]. The usage of mobile devices has extended from basic communication needs, e.g. sending SMS and making calls, to many high-level needs that cover almost every aspect of our daily life, e.g., playing games, getting turn-by-turn directions, and accessing news, books, weather, and more [25]. Such needs are mainly supported by mobile applications (apps) which are specifically designed software programs to run on mobile devices like smartphones and tablets. Since the advent of the iPhone in early 2007, users could experience the functionality of pocket-sized mobile devices. Then the mobile devices and the associated various mobile apps are becoming increasingly ubiquitous in our daily life. Nowadays, it is estimated that there are roughly 2 billion smartphone users in the market. Until the first quarter of 2019, Apple's app store had 1.8 million applications and Google's Android market also had around 2.1 million applications [206]. While these apps are used, data logs are typically generated and recorded forming a rich data source of the users' behaviours.

Understanding app usage behaviours is important since it can benefit both mobile users and app makers. For smartphone users, existing studies show that the number of used apps in each user's smartphone ranges from 10 to more than 90, with a median value of about 50 [45]. This large number makes finding a specific app on one's smartphone a non-straightforward task. In particular, as a fundamental interaction with a high frequency of occurrences, it should be made really simple and efficient for an individual to perform. What's more, apps used and the usage pattern can effectively convey lots of personal information. For many people, the mobile apps have been the first thing they checked after getting up and also the last item they viewed before going to sleep. Users install and use apps depending on their needs, interests, habits, etc. This has the potential to provide us with a new lens to better profile users, which could help the service providers/app developers/smartphone manufacturers to provide more satisfying services for improving user experience and retention.

The general app usage patterns have been comprehensively investigated in early works. By leveraging the large-scale app usage dataset, researchers [16, 191] observed the general app usage frequency changed during the day, which grew from 6 am and reached its first peak around 11 am and were most active during the evening (7 pm to 9 pm). Specifically, they also discovered the temporal pattern while specific app usage, e.g., news apps are most popular in the morning and games apps are at night, but communication apps dominate through most of the day. However, only the general pattern analysis is not enough for providing users' with satisfying mobile services. Given the huge amount of various apps installed on users' mobile devices, it is often tedious for users to find the proper app they want to access immediately. Then many researchers tried to recommend the app user is going to open proactively by generating the next app prediction mod-

els [8]. These works are mainly conducted for modelling users' behaviour on choosing one particular app under different contexts [203, 135, 157, 72]. Specifically, besides the important temporal features, they leveraged a wide range of contextual information in a smartphone, including the GPS location, battery, phone settings, accelerometer, latest used app, number of apps launched, etc, and a supervised machine learning model to make personalized app usage predictions [157, 99, 8]. With the knowledge of the most likely next app to be used, both the battery energy consumption and app searching time can be planned in advance and optimized.

Besides predicting the next app users aim to access, if we could know how much time the user would spend while interacting with this app, the app developers or operating systems can further manage the delivery of corresponding contents or services to the end-user by matching their engagement demands. For example, the media apps, like video and news apps, could recommend videos and news with specific lengths based on the predicted time spent to improve user experience. For now, minimal research has been done for analysing mobile app dwell time from a large-scale dataset, let alone modelling users' app dwell time under different contexts. Only the basic aggregated statics on app usage time was reported, e.g, Falaki et al. [45] found that 90% of app usage sessions would be less than 6 minutes and Xu et al. [191] reported that the majority of total network access time for all apps is from 10 seconds to 1 hour for each subscriber in one week. How to effectively predict how long users would stay with an app in different contexts is still an open research question. For example, do the mobile usage contexts (e.g., time of day) in which users access mobile apps impact their dwell time? Answers to such questions could help mobile operating systems and publishers to optimize advertising and service placement. Therefore, to conduct more comprehensive study for inferring

users' app usage needs is the first theme of this thesis. Different from the previous researches, the app usage needs we consider in this thesis is not just which app user want to access, we also take the time user aim to spend on apps into consideration.

Despite the usefulness for inferring users' app usage needs, however, a survey [25] of mining smartphone data for app usage prediction and recommendations carried out in 2017, indicated that the existing approaches for mining smartphone usage data are mostly at its early stage of the syntactic level. There is also a need to interpret human behaviours from app usage data at a higher semantic level, where they stated that the interesting semantic meanings can be the high-level activities performed at different scenarios. For example, to complete a task with a higher semantic level intention like dining out with friends, the user needs to chat with his/her friends on WhatsApp firstly, access Yelp to look for restaurants and book a table. He/she then copies the restaurant address from Yelp to Google Maps to check where the restaurant is, and books a ride on Uber later. This series of app interactions all aim to support the same task for dining out with friends. However, modern mobile devices only effectively serve the individual apps that correspond to simple needs, e.g., weather checking, users get little or no help when their needs transcend the boundary of a single mobile app. Additionally, in the survey of user profiling from their use of smartphone applications [206], the researchers also stated that the existing techniques used for user representation and modelling based on app usage behaviour are rather straightforward, such as simply using app lists or app usage records as features, without considering the relationship between apps or the correlation between apps and users. More sophisticated methods should be used in future research to generate user profiling based on app usage sequences, which include more semantic and

sequential information. Therefore, other than understanding the specific app usage needs, in the second part of the thesis, we aim to further study the high-level tasks users aim to complete while engaging with a series of app usage.

Tasks, which are defined as pieces of work, ranging in scope from specific (e.g., sending an email) to broad (e.g., planning a vacation), are central to all aspects of information access and use [60]. In the context of web search, a search task is defined as a set of queries corresponding to a particular high-level information need, and the queries are not necessarily the same or even similar [78, 86, 61]. For example, the queries "cheap flight" and "hotel booking" may come from the same goal: vacation plan, but they have no words in common [61]. The mobile task we aim to extract from apps usage behaviour has the similar definition as search tasks, where a sequence of app usage is considered as part of a coherent mobile task if they collectively try to achieve a certain goal, as the "Dining Out" task mentioned above. For now, the problem of user task understanding still remains an important problem in modern era mobile app usage. Within the existing research, a primary mechanism for segmenting logged app usage streams is *session*-based, where short inactivity timeouts (30 or 45 seconds) between user actions are applied as a means to demarcate session boundaries [171]. However, the mobile tasks as we illustrated above with users' high-level intentions may span multiple sessions and involve different apps, where the empirically-set short timeout threshold may not be a valid criterion. With this in mind, we investigate how to infer mobile tasks from app usage logs in the second part of this thesis.

Identifying mobile tasks properly could enable app developers, mobile system designers and device manufacturers to better understand user interactions and gauge their satisfaction. Accurate representation of tasks could

be used to provide users with better app suggestions, offer improved personalization, provide better recommendations, and improve user experience. Nowadays, task intelligence has been leveraged in a broad range of application scenarios with software and services we use every day, e.g., search systems, digital assistants, and productivity applications [60]. Mehrotra et al. [120] proposed a task context embedding architecture to learn the representation of queries by leveraging the task context information from historical search logs. They demonstrated that the task information could provide better context for information retrieval (IR) systems to learn from. Zhang et al. [205] proposed task-based recommendation to offer cross-site heterogenous item recommendations on a Web-scale, which could meet users' potential demands better. After extracting tasks from users' app usage log data, how could we leverage the mobile task information for improving user profiling and providing more satisfying services to users then become the last theme of this thesis.

In this thesis, we address three themes concerning users' needs and tasks within app usage behaviour. In the first part of the thesis, we mainly focus on inferring users' specific needs which could be satisfied by single apps. In the second part of this thesis, we study methods for understanding and characterizing users' app usage behaviour in the task space. We develop models for identifying tasks from app usage logs and characterizing the extracted tasks. In the last part of this thesis, we focus on leveraging the tasks information in various applications.

In the next section, we outline the research in this thesis and the questions that are answered within it.

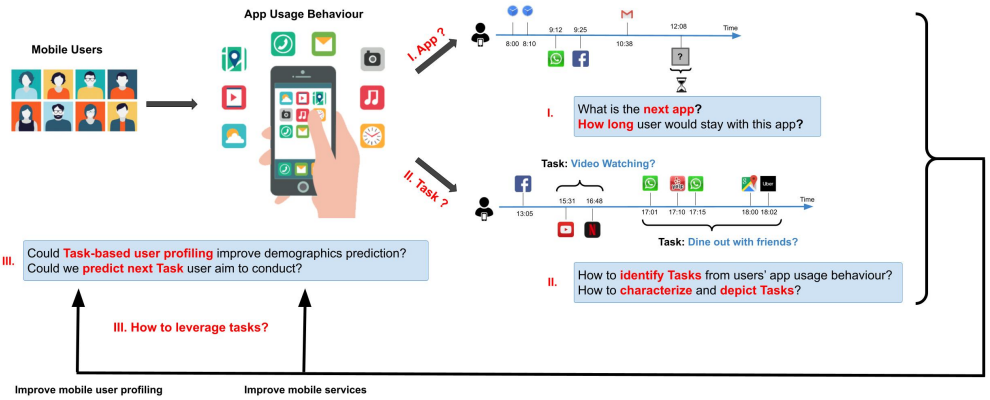


Figure 1.1: Overview of the thesis (the Part I, II, and III state the main content this thesis).

1.1 Research Outline & Questions

The thesis focuses on understanding and characterizing users' needs and tasks from the logged app usage data and leveraging the knowledge learned for enhanced mobile services/applications. An overview of all the research conducted in this thesis is shown in Figure 1.1. The work and research questions can be split mainly into three parts: (1) Inferring users' app usage needs (2) Extracting and characterising mobile tasks (3) Leveraging mobile task information. Specifically, we attempt to answer three big research questions (RQ) within each part of this thesis:

RQ1. How could we improve the methods for inferring users' needs on single apps, especially when compared to the existing models?

RQ2. Other than the traditional approach for modelling users' behaviour based on specific apps, could we measure users' app usage behaviour within task space?

RQ3. How could we leverage the extracted tasks information in different applications for improving mobile services?

Part I: Understanding App Usage Behaviour

Within the problem for predicting the next app user would access, the existing models are mainly based on personalized mechanism [163, 98, 72, 157, 8, 203], where most of them were conducted based on a small number of users (less than 50) within a long period individual's data. However, the personalized prediction model can be very sensitive to the data available and might not perform well for new users. This is generally referred to as the *cold-start* problem, which has not been well studied within the previous works. One potential method to alleviate this issue is by finding *cohorts* of users who share common attributes or experiences with the current user (e.g., demographics, same habit or interests) [195]. Then given a new user, we can leverage the behaviour of other members in the same cohort(s) to enhance app usage prediction. Therefore, to improve the next app prediction specifically in alleviating the *cold-start* issue, we design and conduct experiments to answer the following questions:

RQ 1.1: how could we model users' cohorts based on users' characteristics and logs readily available for mobile app usage?

RQ 1.2: could we employ signals from users who are similar along one or more dimensions, i.e., those in the same *cohort* for improving the prediction performance, especially for alleviating the *cold-start* problem?

After answering the above questions, we would be able to infer users' needs on the next app effectively no matter the sufficient data of individuals are available or not. However, only predict which app users would use is not enough for providing more satisfying services to users. We should also be able to infer how long users would stay while engaging with apps. Real-world mobile app usage behaviour is a complex phenomenon driven by a number of competing factors. For example, a user might be more likely

to engage with certain mobile apps at specific times: John might be more likely to engage with game apps for a longer time at night after work. User characteristics could also make a difference: female users may spend longer time with shopping apps than male users [143]. However, despite that the averaged total time spent on different app categories were reported in [16, 191, 94], there is little research comprehensively analyzed and modeled the dwell time of mobile apps. So we aim to answer that:

RQ 1.3: What are the factors (user characteristics and contexts) that influence the app dwell time?

In answering the question above, we first find that app dwell time is much dependent on the app content itself. For example, checking the weather app is always shorter than playing games. Therefore, it is meaningless to only predict how long a user will stay regardless of which app the user is engaging. Given the inter-dependency between an app and app dwell time, we then aim to address a novel app usage prediction problem:

RQ 1.4: Can we predict which app the user will use next and how long the user will stay on this app simultaneously?

Next app prediction is characterised as the willingness to use an app, whereas engagement (dwell time) is the usage pattern after accessing the app. In this thesis, we consider which app the user will use and how long the user will stay with this app an aggregated measure of users' app usage behaviour.

Part II: Extracting and Characterizing Mobile Tasks

Users' mobile needs span a broad spectrum, which not only include the simple needs, such as weather information checking, can mostly be satisfied via a single app; but also include the tasks that need to access a series of

apps, collect, filter, and synthesize information from multiple sources, e.g., plan to dine out with friends. After improving existing methods on inferring users' app usage needs, which could be simply satisfied via single apps, we turn to another key problem in this thesis, which is to infer users' tasks from their app usage behaviour.

Definition: A set of app usage is considered as part of a coherent *mobile task* if they collectively try to achieve a certain goal. [166]

In the context of web search, there have been many attempts to identify tasks from query logs, relying on a notion of the timeout, lexical characteristics [178] (e.g., number of common words), and topic [69] (e.g., queries expressing car interests: "Honda", "Nissan", and "Ford"). However, how to identify tasks within mobile app usage logs and what features are effective have not been studied. The biggest challenge in identifying mobile tasks is that the apps do not include abundant information as for search queries. Additionally, most of the app usage logs do not provide detailed behaviour information within the apps due to privacy issues. Therefore, a detailed study of the frequency and patterns of real user app usage logs forming mobile tasks manually labelled by annotators is conducted. Then we aim to answer the following research questions:

RQ 2.1: What kind of features can be used effectively to identify mobile tasks from app usage logs?

RQ 2.2: Can we formulate the task identification as a supervised learning problem, which could predict the app usage belong to the same task automatically?

In addressing the above questions, we propose a list of features that move beyond timeouts and demonstrate that they can be used effectively to iden-

tify mobile tasks. However, the laboratory study of mobile tasks is still limited. Besides only shedding lights on the basic statistics of mobile tasks (e.g., number of apps used within a task), we aim to further understand characteristics of mobile tasks in the wild and uncover the common patterns of tasks users aim to complete. By performing the proposed task identification model on a large-scale dataset of app usage logs, which is collected from Verizon Media's Flurry mobile analytics platform, consisting of millions of logs from thousands of users, we aim to gain insights from a large spectrum of mobile tasks.

Definition: The *complex mobile tasks* are the mobile tasks that have more than two different apps involved [167].

Since the complex mobile tasks are more time-consuming and difficult for users to complete, to better improve the current mobile systems and applications, especially in supporting task continuation and task completion, we specifically focus on characterising the complex mobile tasks users aimed to complete with cross-app and multi-topic usage patterns on smartphones. To characterise and depict the common patterns of all complex mobile tasks, we aim to address the following research questions:

RQ 2.3: How to characterize complex mobile tasks based on different attributes?

RQ 2.4: Could we uncover the common patterns that exist in complex mobile tasks by dividing them into natural groups that reflect salient patterns?

We first characterize the complex tasks from three aspects: task context, task complexity, and task content. Then the unsupervised learning approach is employed to derive generic profiles of all complex mobile tasks

based on the extracted characteristics. Given the clustering results, we create the taxonomy for mapping users' complex tasks into different types ranging from "social media browsing" to "dining out" and "family entertainments".

Part III: Leveraging Mobile Task Information

After addressing the above questions regarding the task identification and characterization, we are able to extract all mobile tasks from users' app usage logs automatically, and further assign a type for each complex task. We then aim to investigate how the extracted mobile task information could be leveraged in various applications. Firstly, the mined knowledge about user tasks from log activity data reveals more detailed user intentions and behaviour patterns, which could provide unique signals for user-centric optimization. In the existing research, app interests are traditionally used in constructing mobile user representation, specifically aims at demographics prediction. By leveraging the extracted task information, we aim to answer that:

RQ 3.1: could we represent users not only based on app interests (used apps), but also on tasks, or task types?

RQ 3.2: would the task-based user representation methods benefit demographics prediction?

In answering the above questions, we find that tasks have become a more accurate unit to capture users' goal and behaviour insights, especially when compared to apps. So besides the classic next app prediction problem, we propose to further investigate if we could predict which task user aim to conduct next. With improved speech recognition and information retrieval systems, users are increasingly relying on intelligent assistants (IAs) on

mobile devices to complete their tasks. We believe that if we could know the specific complex task user would conduct in advance, the IAs would work better for tracking the task progress and supporting task completion. For example, under the "Dining Out" task context, if the user only booked the table of the restaurant without checking the navigation or traffic status, the system could push notifications or suggestions to remind users to call a taxi in advance due to the rush hours for having dinner. With this in mind, we aim to further tackle the problem:

RQ 3.3: Could we predict what the next complex task is by measuring users' app usage behaviour within the task space?

This is similar to the *next app prediction problem* [218, 41, 8], but at the level of task.

1.2 Main Contributions

In this section, we summarize the main contributions of this thesis. Our contributions come in the form of analysis, model and empirical contributions.

C1. We demonstrate that the cohort modelling method can effectively enhance the next app category prediction problem, especially in alleviating the user cold-start issues when a limited amount of user interaction data is available.

C2. We conduct the first empirical analysis of mobile app engagement based on dwell time with a large-scale data set collected from thousands of users. By investigating this large scale dataset, we present insights on what information could be indicative factors for inferring

users' app dwell time. Specifically, we consider the influential factors from two aspects: user characteristics (e.g., age, gender, device type, historical preferences) and context (e.g., hour, weekday, last used app, periodic pattern).

C3. Our research investigates a novel problem on simultaneously predicting which app the user will use and how long the user will stay on that app. We propose three different joint learning strategies to solve the novel prediction problem, including sequential, stacking and boosting-based joint models, where the boosting strategy performs best.

C4. We are the first to formally define mobile tasks and formulate the automatic identification of mobile tasks as two supervised machine learning problems. We propose a list of features that move beyond timeouts and demonstrate that they can be used effectively to identify mobile tasks effectively, which include similarity features for extracting common characteristics and log sequence features for capturing semantic relatedness between apps. Our research is an important first step in modelling mobile app usage from the task perspective, which sets the stage for evaluating mobile services, not on a per-app basis, but the basis of user tasks.

C5. We are the first to conduct a comprehensive quantitative study on characterizing complex tasks based on the large-scale commercial mobile app usage logs. A generic mobile app navigation model is proposed to present an accurate picture of the micro-level interactions within this analysis, including how users revisit and switch between different apps.

C6. We propose an unsupervised learning framework to derive generic

profiles of complex mobile tasks. We provide evidence that there actually exist 17 common tasks with 47 sub-tasks, which could be identified solely from their salient properties. Those tasks range from information check, micro documentation work to family entertainment.

C7. We demonstrate that the user representation learned in a task constrained context perform better than the traditionally used app-based representation. The first comprehensive analysis is conducted to reveal the effectiveness of mobile tasks for inferring users' demographics when compared to user modelling solely based on apps independently. We propose to leverage various embedding and hierarchical attention neural architecture for learning the distributed semantic representation of apps and tasks used by a user, which could then be applied to enhance the demographics prediction.

C8. We show that the task types (based on the clustering results in C6) could be used for measuring users' app usage behaviour and demonstrate that the task-based representation of users' app usage behaviour could improve the complex task prediction on smartphones significantly. By modelling users' behaviour only in app space, users' superior intentions could not be well understood.

1.3 Thesis Overview

In this section, we provide an overview of this thesis. We finish this section with reading directions.

The first chapter, to which this section belongs, gives an introduction to the subject of this thesis. This chapter also provides an overview of the

research questions, the contributions and origins of the work. Chapter 2 then introduces the background and related work for all the following six research chapters. The core of this thesis consists of three parts.

In Part I of this thesis, we study users' app usage behaviour for inferring users' app usage needs. In Chapter 3, we focus on predicting the next app category user would use based on the cohort information and show that the cohort-based approach can significantly alleviate the *cold-start* problem, achieving strong predictive performance even with the limited amount of user interactions. A comprehensive overview of correlations between different user/context features and app dwell time is provided in Chapter 4. Based on the comprehensive analysis of users' app dwell time, we further propose several joint prediction models (sequential, stacking, and boosting) for solving a novel app engagement prediction problem – how to predict the next app and how long the user will stay on this app simultaneously?

In Part II of this thesis, we mainly focus on inferring users' tasks from their app usage logs and characterizing the extracted tasks. In Chapter 5, we present a method that accurately determines mobile tasks from users' app usage logs. We show that a set of temporal, similarity and log sequence features used in combination can effectively predict mobile tasks instead of using the traditional time threshold. Since the complex mobile tasks users aimed to complete with cross-app and multi-topic usage patterns on smartphones could play a more important role in improving the current mobile systems and applications in supporting task continuation and task completion. We then focus on characterizing the complex mobile tasks in Chapter 6.

Part III of this thesis revolves around leveraging the extracted task infor-

mation in different applications. Chapter 7 discusses various embedding and deep learning models which leverages task context to learn user representations and validate that task-based user representation with advanced neural models could effectively improve the performance of demographics prediction. In Chapter 8, we validate that the task-based representation of users' app usage behaviour could improve the complex task prediction on smartphones significantly.

Lastly, Chapter 9 concludes the thesis by discussing the main findings, implications for each research question and presents the future work that could be carried out.

Readers familiar with the background on mobile apps can skip the corresponding sections of Chapter 2 and glance through the task-related parts to develop a better understanding of the background needed for the proposed models. Part III assumes a basic understanding of tasks and in particular, a basic understanding of the task extraction technique.

1.4 Origins

We list for each research chapter the publications on which it is based. For each publication, we mention the role of each co-author. The thesis is based on in total 2 journal and 3 conference papers.

- The Chapter 3 is based on *What and How long: Prediction of Mobile App Engagement* [168], to appear on ACM Transactions on Information Systems (TOIS) 2021 by Tian, Zhou and Pelleg. Tian implemented the analysis components and performed the experiments. All authors contributed to the text.

- The Chapter 4 is based on *Cohort Modeling Based App Category Usage Prediction* [165], published at 28th ACM Conference on User Modeling, Adaptation and Personalization (UMAP 2020) by Tian, Zhou, Lalmas, Liu and Pelleg. Tian performed the experiments and analysis while all authors contributed to the text.

- The Chapter 5 is based on *Identifying Tasks from Mobile App Usage Patterns* [166], published at 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2020) by Tian, Zhou, Lalmas, and Pelleg. Tian performed the experiments and implemented the models. All authors contributed to the text.

- The Chapter 6 and Chapter 8 are based on *Characterization and Prediction of Mobile Tasks*, which is currently under review by the ACM Transactions on Information Systems (TOIS). It is an extension work of *Identifying Tasks from Mobile App Usage Patterns* [166]. Tian implemented the analysis and performed experiments. All authors contributed to the text.

- The Chapter 7 is based on *Inferring Users' Demographics From Mobile Tasks*, which is currently under review by the ACM International Conference on Web Search and Data Mining (WSDM 2022). Tian implemented the models, performed the analysis and experiments. All authors contributed to the text.

Chapter 2

Background

In this chapter, we introduce the background and the most relevant prior works to this thesis. Therefore, we aim to cover three major parts: (1) understanding apps usage behaviour; (2) understanding tasks; (3) understanding mobile users based on app usage behaviour. Within the entire literature review, we mainly focus on the works that are most relevant while briefly discussing the broad background. We recommend readers refer to other relevant books or surveys [25, 155, 206] if more details are needed.

2.1 Understanding App Usage Behaviour

A mobile application, also referred to as a mobile app or simply an app, is a type of software application designed to run on a mobile device, which can be a smartphone, tablet, or even a watch. Apps are generally downloaded from app stores, which are a type of digital distribution platform. The three biggest app stores are Google Play [57] for Android, App Store [6] for iOS, and Microsoft Store [123] for Windows 10. There are 2.87 million apps

available for download on the Google Play Store. The Apple App Store has 1.96 million apps available for download [14]. The popular categories of mobile apps range from games, social networking, utilities, business, food&drink, music, sports, travel to education, etc [56]. The existing studies show that the number of installed apps in each user's smartphone ranges from 10 to more than 90, with a median value of about 50 [45]. The average smartphone owner uses 10 apps per day and 30 apps each month [14]. It was recently reported that mobile apps are expected to generate over 935 billion dollars in revenue by 2023 [14]. Having a better understanding of mobile app usage could be helpful to provide users with better personalized services and recommendations. If the companies and businesses want to compete and claim their share of this multi-billion dollar industry, they also need to have a better understanding of exactly how people are using mobile apps.

In this section, we review previous work in detail on app usage pattern analysis and how to model users' app usage behaviour aims at predicting their future behaviour.

2.1.1 App Usage Pattern Analysis

2.1.1.1 Large-scale descriptive analysis.

Early works focused on conducting large-scale descriptive analysis of general patterns within app usage behaviour [54, 191, 16]. The AppAware project [54] showed end-users "which apps are hot" by aggregating worldwide occurrences of app installation events. However, since AppAware only gathers the installation, update, and uninstallation of an app, the system is not aware of the actual usage of a specific app. Xu et al. [191] are the first

attempt in addressing the lack of how, where and when smartphone apps are used at the population scale. They comprehensively investigated the diverse usage patterns of smartphone apps via network measurements from a national level tier-1 cellular network provider in the United States. They identified traffic from distinct marketplace apps based on HTTP signatures and present aggregate results on their spatial and temporal prevalence, locality, and correlation. Other than identifying users' app usage behaviour from the interpretation of HTTP signatures, Bohemer et al. [16] conducted a large-scale app usage behaviour based on the logged information, which was from over 4,100 users of Android-powered mobile devices. They collected the app data by a tool *AppSensor*, which was implemented as a background service within Android and was installed by end users. Both Bohemer et al. [16] and Xu et al. [191] discovered the temporal pattern of app usage, e.g., news apps are most popular in the morning and games are at night, but communication apps dominate through most of the day. Furthermore, communication apps are almost always the first used upon a device's waking from sleep. They also found that, in general, the app usage frequency changed during the day, which grew from 6 am and reached its first peak around 11 am, and were most active during the evening (7 pm to 9 pm). Additionally, they observed that some apps have a high likelihood of co-occurrence on smartphones – that is, when a user uses one app, he or she is also likely to use another one. For example, a Browser is opened quite frequently following the use of a News app. The connection between Lifestyle and Shopping apps is also quite strong, with Lifestyle apps frequently leading the user to enter into a Shopping App.

With mobile shopping surging in popularity, people are spending ever more money on digital purchases through their mobile devices. Kooti et al. [85] then analyzed a large data set consisting of more than 776M digital

purchases made on Apple mobile devices that include songs, apps, and in-app purchases. They found that 61% of all the spending is on in-app purchases and that the top 1% of users are responsible for 59% of all the spending. These big spenders are more likely to be male and older, and less likely to be from the US.

2.1.1.2 Small-scale behaviour analysis.

Other than the large-scale descriptive analysis of app usage, later researchers mostly focused on conducting the small-scale study to uncover more detailed app usage behaviour from various aspects.

Pielot et al. [141] reported an in-situ study involving 15 mobile phones users, where they collected one-week real-world notifications through a smartphone logging application. They found that users have to deal with 63.5 notifications on average per day, mostly from messengers and email. Whether the phone is in silent mode or not, notifications were typically viewed within minutes. Their findings implied that reducing interruptions and deferring notifications may work in a professional context. Ferreira et al. [46] conducted a 3-week study by collecting app usage data from 21 participants to explore how they manage their time interacting with the apps. They showed that approximately 40% of app launches last less than 15 seconds and happen most frequently when the user is at home and alone. They defined these events as app micro-usage: brief bursts of interaction with apps. Jones et al. [79] presented a revisitation analysis of smartphone app use. This analysis was based on app launch logs from 165 users within three months. Their analysis revealed distinct clusters of applications and users which share similar revisitation patterns.

Except from only focusing on analyzing specific app usage patterns, several

works were conducted for uncovering the correlations between app usage and other services/events. Carrascal et al. [26] tried to explore why certain interactions between apps usage and mobile search occur. They reported on a 2-week, mixed-method study involving 18 Android Users. They showed that when people engage with mobile search they tend to interact with more mobile apps and for longer dwell time. Additionally, they found that certain categories of apps are used more intensely along with mobile search, e.g., Shopping & Retail and Entertainment apps were all used more intensively when people engaged with mobile search, both in terms of app launches and duration of app usage. Furthermore, Van et al. [172] studied how app usage behaviour was disrupted by major events. For example, a significant increase in news and finance app usage was observed after the EU referendum vote in the UK, and an increase in the news during the Wimbledon final was also discovered.

2.1.2 App Usage Behaviour Prediction

While the studies above have successfully explored mobile app usage, they do not provide a clear answer as to how to practically leverage the findings in order to support improved app usage behaviour prediction.

2.1.2.1 Next App Prediction

Explicit and implicit features. Within the prior works, app usage prediction mostly refers to the problem of predicting the next app that will be used for a given user and at a given time [25]. Existing next app prediction literature is mostly conducted around 2012 to 2013 since it is becoming harder to find an app on one's smartphone due to the increasing number

of apps available and installed on smartphones. The next app prediction models are built during this period are mostly based on two types of features, ***explicit*** and ***implicit*** features [131, 99, 216], with various predictive modelling methodologies. The ***explicit*** features are the readily available information extracted from the phone, include (1) location and other data sources that can imply locations or any forms of data can be translated into locations [136, 202, 216]. E.g., GPS, WiFi access points (APs) and cell tower IDs, etc. (2) time [100, 196], e.g., hour of the day, day of the week. (3) phone settings, e.g., ring volume, silent mode, vibrate mode, airplane mode, etc. As stated in [72] phone setting features can be used to effectively filter out unlikely app candidates in the given circumstances. (4) phone status, e.g., battery level, screen brightness, phone movement and phone carrying position identified based on accelerometer and proximity sensor readings, etc. The ***implicit*** features instead refer to the subtle derived app statistics, which require feature engineering and data modelling. For example, they can be the distribution of app usage duration, the probability of each app being the first to be used, the correlation statistics of app usage sequences, etc.

Some researchers proposed the next app prediction models only based on explicit features. Yu et al. [202] proposed the Latent Dirichlet Allocation (LDA) based personalized context-aware app recommendation approach by leveraging individual user's contextual information, including day of the week, period of a day (i.e., morning, noon, afternoon, evening, and night), hour of the day, profile type, and location. While some researchers proposed to predict the next app user would access based on modelling the implicit features. Tan et al. [163] and Liao et al. [98] all tried to predict the next app based on mined temporal profiles for each app. Tan et al. [163] treated users' app usage patterns as periodic time series cycles and predict

the app will be used based on a prediction algorithm with fixed cycle length. Liao et al [100] investigated three implicit app usage features from the apps usage trace to determine the top k apps with the highest usage probability, which are a global feature, a temporal feature and a periodicity feature. The global feature was defined as the probabilities of the apps being used, which was aggregated over all available temporal periods. The temporal feature is useful for those apps regularly used at a specific time, e.g., the user usually set an alarm at around 23:00. The periodical feature is useful for those apps that have a significant period, e.g., the user checks email around every 3 hours. Zou et al. [218] proposed three Bayes models on next app prediction, which only relied on the last one or two apps user accessed before. They also pointed out that the latest used app and time are more effective than location and other context information in the next app prediction.

Most of the next app prediction models are proposed by combining both the explicit and implicit features [157, 72, 196, 203, 8]. Shin et al. [157] leveraged a wide range of contextual information in a smartphone, including the GPS location, time, battery, phone settings, accelerometer, latest used app, number of apps launched, etc, to make personalized app predictions based on naïve Bayes model. Liao et al. [99] adopted the KNN (K-Nearest Neighbors) classification model to predict the next app through the proposed personalized feature selection algorithm. They claimed that they need to use different sets of features to predict their app usage. They extracted explicit features from hardware sensors, including data from environmental sensors: time, location, wifi-signal; device sensors: space, free ram and battery levels; personal sensors: acceleration, speed and heading. They then estimated the distribution of transition probability among apps and design an Apps Usage Graph to model both app usage order and

transition intervals for extracting the implicit feature. Yan et al. [196] proposed to consider the app session triggers, like the usage possibility of SMS, Phone and Web Browser before user accessing each app, and combined it with time and location features to predict the next app. Baeza et al. [8] combined various explicit features and implicit features in a parallelized Tree Augmented Naive Bayes [50] (i.e., PTAN) for next app prediction. The used explicit features include location features like latitude, longitude, GPS accuracy, location semantics (either home or work), time-based features, and phone status features such as the status of charging and audio cable connection. The implicit features were a set of temporally adjacent app usage information.

Modern next app prediction models. After 2017, instead of relying on various features for app usage modelling, more advanced models are proposed to predict the next app. Yu et al. [201] developed a technique that leveraged transfer learning to predict which apps are most popular and estimated the whole usage distribution only based on the Point of Interest (POI) information of that particular location. They stated that their model outperformed by about 25.7% over the existing state-of-the-art approaches. With the deep learning approaches becoming increasingly popular, some researchers proposed different neural models for predicting the next app, including DNN [207] and LSTM [193], etc. Zhao et al. [207] proposed the *AppUsage2Vec* method, which measured the contribution of each app to the target app by introducing an app-attention mechanism. The user personalized characteristics in app usage are learned by a module of dual-DNN (Deep Neural Network) [89]. Furthermore, they encoded the top-k supervised information in the loss function for training the model to predict the app most likely to be used next. Xu et al. [193] proposed a generic prediction model based on Long Short-term Memory (LSTM) to

covert the temporal-sequence dependency and contextual information into a unified feature representation for next app prediction and stated that it outperforms other models.

2.1.2.2 Other App Usage Behaviour Prediction

Besides predicting the next app user would access immediately, some previous attempts have been made to predict the app installation [105], attention-based app engagement [113] and spending in apps [85], etc.

Liu et al. [105] took the initiative to analyze a very large app management log, which covers 5 months of 17 million anonymized users' detailed downloading, updating, and uninstallation activities. They presented a surprising finding that the metrics commonly used to rank apps in app stores do not truly reflect the users' real attitudes. They identified behavioural patterns that could be regarded as indicators of users' preferences on apps, which can be integrated by machine learning algorithms for predicting the ratio of positive ratings of the apps. Mathur et al. [113] tried to model and predict users' attention-based engagements (two levels: focused attention and felt involvement) in the context of smartphones. They conducted the work based on physiological measures (headset readings) within 10 participants to detect engagement of a usage session using a Random Forest classifier. Their research only focused on inferring the general engagement under mobile context, which has no works regarding which app users are currently using. Kooti et al. [85] tried to model the in-app purchasing behaviour in multiple steps. They trained a classifier to predict whether the user will make a purchase from a new app or continue purchasing from the existing app. Based on the outcome of the previous step, they also attempted to predict the exact app, new or existing, from which the next

purchase will come. Their results uncovered new insights into spending habits in the mobile digital marketplace.

2.2 Understanding Tasks

From the prior works in understanding users' app usage behaviour above, we can find that the current mobile devices and existing research works only focus on how to effectively serve the individual apps that correspond to users' simple needs, e.g., weather checking, users get little or no help when their needs transcend the boundary of a single mobile app with a high-level task in mind. We now turn to present the prior works related to another theme of this thesis: understanding tasks based on users' app usage interactions. Tasks, which are defined as pieces of work, ranging in scope from specific (e.g., checking weather) to broad (e.g., planning a vacation), are central to all aspects of information access and use [60]. Helping users complete tasks is crucial for a number of applications, such as search systems, digital assistants, and productivity applications. App usage logs have been extensively studied to generate insights that would improve mobile user experience. However, the problem of user task understanding is still an important problem, which hasn't been well studied in the modern era mobile app usage.

2.2.1 Mobile sessions v.s. mobile tasks.

The mobile "task" we define in our thesis is more similar to the so-called "task" in search context [78, 86, 61], which consist of a set of apps (queries) corresponding to a particular high-level intention, and the apps (queries) are not necessarily the same or even similar. For example, Whatsapp, Yelp,

GoogleMap and Uber apps from different functionalities would be engaged to support the same task for "dining out with friends". For now, little research has explored methods to understand and identify mobile tasks based on app usage logs, let alone to support users in task continuation and task completion.

Within the context of mobile app usage, mobile sessions first allow us to look beyond individual apps, preserve semantic associations between apps usage and maintain the context of user activity. Some researchers also regarded a mobile session as a "task" initially while conducting the research [171]. Strategies for session identification from mobile log data have been extensively studied. A smartphone usage session is commonly defined based on a threshold value of potential idle or standby time between apps usage. Carrascal et al. [26, 16] defined a session as an interaction sequence without turning off the display for more than 30 seconds. Also common is the definition of a phone usage session based on active screen usage, which considers the time between the screen on and screen off as one session [64]. Since most of the studies applied different arbitrary thresholds in their session analysis, where the assumption of session segmentation was not empirically validated, Van Berkel et al. [171] conducted a systematic assessment of smartphone usage gaps. They proposed various metrics related to usage sessions and evaluate various machine learning approaches to classify gaps in usage. They built a Constant Classifier that takes as input a constant threshold T (in milliseconds) and classifies a usage session as a continuous session if the time attribute is less than T, or as a new session otherwise. They finally stated that solely relying on the use of phone standby time analysis to classify smartphone usage gaps is not reliable. But if a researcher insists on using a constant arbitrary threshold, they encouraged to consider using a 45-second threshold for segmenting

app usage streams into sessions (best accuracy: 68%). Rather than sessions, our work aims to study the method for extracting tasks from app usage logs, which could organize logs based on high-level user intentions as the search tasks. The task with the same user high-level intention may span multiple sessions and involve different apps, where the empirically-set short timeout threshold may not be a valid criterion [171].

2.2.2 Task based Intelligence

After extracting tasks from users' log data [178], their attributes such as priority [127], duration [183], task context [22], task taxonomies and dependencies [103, 77] could all be inferred and then further deployed in a broad range of application scenarios with software and services we use every day. In this section, we review task intelligence leveraged in search, recommendation, and personalization.

Some researchers studied how to leverage the task context information for query suggestion [24] and ranking documents in Web search [189]. The context of a search query often provides a search engine with meaningful hints for answering the current query better. Mehrotra et al. [120] proposed a task context embedding architecture to learn the representation of queries by leveraging the task context information from historical search logs. They demonstrated that the task information could provide better context for information retrieval (IR) systems to learn from.

Tasks are also leveraged for providing more satisfying recommendations [205, 169]. Zhang et al. [205] proposed task-based recommendation to offer cross-site heterogenous item recommendations on a Web-scale, which could meet users' potential demands better. For example, the user may turn to

Amazon for the dress worn by an actress after watching a video on Youtube. They also stated that task-based recommendation would be one of the key components to the next generation of universal web-scale recommendation engines. Recently, intelligent assistants are becoming increasingly popular. They can serve various purposes from entertainment (e.g. playing music), home automation to work management (e.g., timers, reminders). Trippas et al. [169] investigated the task duration, continuity and regularity, which allow them to provide design recommendations for how intelligent assistants could support work tasks, both on-demand and proactively.

Both White et al. [182] and Mehrotra et al. [121] showed that tasks can indeed be used for improving personalization. White et al. [182] proposed an enhancing personalized search system by mining and modelling task behaviour. They mainly focused on solving the challenge for unseen queries and for new search scenarios. They indicated that building richer models of users' current and historic search tasks can help improve the likelihood of finding relevant content and enhance the relevance and coverage of personalization methods. They described a method whereby they mined historic search-engine logs to find other users performing similar tasks to the current user and leverage their on-task behaviour to identify Web pages to promote in the current ranking. Later, Mehrotra et al. [121] introduced a task-based user modelling approach for behaviour targeting, where users are represented by their actions over a task space. Specifically, given a web search log, they extracted search tasks performed by users and then constructed a user-task association matrix. They borrowed insights from Collaborative Filtering [152] to learn a low-dimensional factor model wherein the interests/preferences of a user are determined by a small number of latent factors. They evaluated the performance of their proposed approach on Collaborative Query Recommendation where the goal is to recommend

queries to a user based on queries issued by similar users. The experiment results showed that the proposed task-based user modelling approach performed better than the bag-of-words (query) based representation, which demonstrated that search tasks can serve as potent user modelling tools.

Users increasingly prefer to complete tasks with mobile devices in their daily life. In this thesis, we not only propose the task identification model based on app usage. Additionally, we go beyond task extraction and present novel applications of leveraging mobile task information in different applications, including user modelling and complex task prediction.

2.3 Profiling Mobile Users

Profiling smartphone users well is the key to improving mobile user experiences. It can help us to improve mobile devices, applications, and services, e.g., targeted advertisements and personalized recommendations, etc. In this section, we review the related studies in the literature for profiling users from apps usage patterns on smartphones. A large volume of app data could be collected, such as what apps are installed on smartphones, how apps are used, and the basic information of apps, e.g, app categories, app descriptions, and app reviews. The app data used in the existing mobile user profiling method could be roughly divided into three classes: (1) installed app list (2) app usage records (3) app metadata.

Installed app list. Users decide to install apps depending on their needs, preferences and tastes. Thus, the apps that a user has installed intuitively could be good indicators of their needs and interests. What's more, the list of apps installed on a smartphone is relatively more accessible. Many previous studies have shown that installed app lists could reflect users'

needs or tastes to a certain degree. Zhao et al. [208] stated that they are the first work to explore mining user attributes from installed app lists. They developed an attribute-specific representation to describe user characteristics and then modelled the relationship between a user attribute and an app list. Their approach achieved the average equal error rate of 16.4% for 12 predefined user attributes, including phone price and size preference, and 10 underlying user attributes reflected by niche app, e.g., movie fan, beauty shopping, own a car or not, etc. Xu et al. [192] provided an approach for user profiling based on readily available information like a user's snapshot of app installation. They indicated that by leveraging their approach, demographics and personality traits become predictable for everyone who uses a smartphone without the pains of answering a survey. However, by installing an app, the user may simply want to try the app out, and may never use it again. According to the statistics in [150], only 10% of apps were used 80% of the time, suggesting that a lot of apps are downloaded but not used regularly. Even for the same app, its usage can be different across users like frequency and intensity. Thus to make it more accurate in profiling user characteristics, app usage records are used, which could report the way how users interact with apps, such as when an app is launched or killed, how long and how often it is used.

App usage records. Compared with installed app lists, there is temporal context, i.e., time information, accompanied with the app usage records, recording when one user uses which apps. Zhao et al. [209] discovered 382 distinct kinds of users from more than 10,000 individuals by clustering users who are represented based on the average usage weight of each app category in different time periods. They named the users as night communicators, screen checkers, evening learners and young parents, etc. They also showed that the way users engage with their apps is related to their

demographics. Jones et al. [79] identified three distinct clusters of users based on their app revisitation patterns: *checkers* who exhibit brief but quick revisit patterns, *waiters* who are split between short-medium revisitation and long revisitation, and *responsives* who exhibit sometimes brief and sometimes long revisit patterns.

App metadata. App metadata refers to the basic information of apps including icons, the description that introduces an app function, the category that one app belongs to, reviews (comments), and the number of downloads, etc. App metadata could help us understand one app and further infer why one user installs or uses it. App metadata could usually be crawled from app store websites, such as Google Play [55]. Compared to installed app lists and app usage records, app metadata always suffers the issue of sparsity, e.g., only very few apps can receive useful feedback from users [101] and the description of some apps is rather limited, consisting of just a few words reviews [51]. So researchers suggest combining other types of app data together, such as installed app lists or app usage records, to compensate for the sparsity of the app metadata. Seneviratne et al. [154] showed that by representing users based on the features extracted from installed app list, app costs, app distribution in different categories, and tf-idf weights within app descriptions, etc, user's gender, a demographic attribute that is frequently used in targeted advertising, can be instantly predicted with an accuracy around 70%.

Part I

Understanding App Usage Behaviour

Chapter 3

Cohort Modeling based App Category Usage Prediction

We start the Part I of this thesis, aiming to better understand users' app usage behaviour, especially for improving the existing methods on app usage behaviour prediction. In this chapter, we start our investigations by studying how to enhance the behaviour prediction for new users when insufficient information is available for learning. We propose to use cohorts modelling method for enhancing the prediction. This chapter addresses our research questions **RQ 1.1** and **RQ 1.2**, as specified in Section 1.1.

RQ 1.1: how could we model users' cohorts based on users' characteristics and logs readily available for mobile app usage?

RQ 1.2: could we employ signals from users who are similar along one or more dimensions, i.e., those in the same *cohort* for improving the prediction performance, especially for alleviating the *cold-start* problem?

Various stakeholders in the mobile industry [126, 10] are keen to understand how users interact with different apps, including phone operators,

manufacturers, advertising companies, and service providers. It has been shown in the past that many contextual features, such as time, location, last used app and other device signals, can be used to predict app usage [163, 98, 72, 157, 8]. Personalization of app usage prediction has been investigated in many prior studies [163, 98, 72, 157, 8], which is typically learned by using an individual's data. The ability to tailor prediction results to a particular individual enables a wealth of opportunity to better satisfy their particular needs. Personalized models are typically learned from observed usage behavior and context information (such as temporal/periodic pattern and sensor signals), which are either used directly [157, 8] or converted into a different representation (e.g., graph) to build models and improve personalization tasks [99, 218].

Despite the value of personalized models, one drawback is that they require lots of user historical interaction information to become effective. Every time a new user comes, a new prediction model needs to be trained for a period until it can predict users app usage correctly. The personalized prediction model can be very sensitive to the data available and might not perform well for new users. This is generally referred to as the *cold-start* problem. One way to alleviate this problem is by finding *cohorts* of users who share common attributes or experiences with the current user. Given a user, we can leverage the app usage behavior of other members of their cohort(s) to enhance prediction by providing signals if insufficient information is available for this user.

Modeling aggregate user behavior in existing online behaviour prediction approaches is commonly performed with collaborative filtering (CF) techniques [49], where groups of similar users (based on factors such as liking the same item [148] or previous used apps [131]) has been shown to work well. However, CF only exploits usage history information and explicit user

rating feedback, ignoring the context information when people use various apps.

In this chapter, different from CF, we propose using cohorts to enhance *app category usage prediction* by exploiting various dimensions, including contexts, user characteristics and user interactions. Our method creates predefined cohorts covering three aspects: demographics (e.g., age, gender), psychographics (e.g., interests, way of living) and behavioral patterns (e.g., engagement frequency, revisitation patterns). Rather than limiting ourselves to these pre-defined sets, we also propose cohorts modeling methods that assign users to a combination of cohorts. We demonstrate through extensive experiments with a large-scale app usage log data that our cohort modeling methods can yield significant improvements over a personalized prediction model.

Finally we demonstrate that compared to existing approaches, our proposed cohort modeling method can significantly alleviate the *cold-start* problem, as it can achieve strong predictive performance for new users, even with limited amount of user interactions available. Moreover, users' interpretable cohort information can provide more transparency and expose the reasoning behind the prediction, which has been shown to be useful in improving the effectiveness of such recommendations [204]. Note that we do not directly compare our proposed approach with CF in this work given it is difficult to model all the dimensions we consider into the CF framework. Rather, our main focus is to demonstrate the effectiveness of our cohort-based approach, compared to personalized models, especially for the cold-start scenario. To our knowledge, we are the first to extensively utilize users' cohort information in predicting large-scale mobile app category usage.

The remainder of this chapter is organized as follows. Section 3.1 provides some background. Section 3.2 introduces the large-scale dataset used in our study, which is also leveraged in the following chapters. Section 3.3 presents our approaches for extracting informative and interpretable user cohorts based on different aspects of users' characteristics and app usage behaviors. In Section 3.4, we introduce our approach of using cohort modelling for the purpose of app category usage prediction. Section 3.5 details experiments carried out to the proposed prediction problem, including demonstrate the effectiveness in alleviating the user cold-start problem. Finally, Section 3.6 outlines our conclusion.

3.1 Related Work

Most of the previous research work on app usage prediction (exact app or app category) is based on both the temporal patterns and sensor signals collected, and only the personalized prediction model is explored (refer to Section 2.1.2.1) [163, 98, 100, 72, 157, 8, 196]. In those works, the number of participants is small or from one social community (e.g. college students), where the cold-start problem hasn't be well studied. In this chapter, we propose to predict users' app category usage based on cohorts and show that this method help in alleviating the user cold-start problem.

A basic design consideration for next-app predictive modeling is whether we should learn a generic model based on multiple users' data or a personalized model based on each user's data. The majority of existing next app prediction models as we list above [157, 202, 163, 72, 218, 99] are relying on the personalized mechanism since most studies are conducted based on a small number of users (less than 50) within a long period. As for per-

sonalized models, they are typically learned by using an individual's data. With sufficient data for model training, personalized models were reported to have better accuracy than the generic models [157]. However, training a personalized model often encounters *cold-start* and data sparseness issues, especially at beginning of data collection for an individual. Cold start here refers to the frequent scenario that at the beginning of data collection for each individual, there is typically insufficient data to train a personalized model for immediate interpretation of his/her live smartphone data feeds accurately. The underlying assumption for the generic model is that a group of users share common app usage patterns in a similar context. Do et al. [41] trained both generic and personalized models based on all the users' data. Their experimental results showed that the generic models performed well against personalized models especially when each user's training data was very small such as for a duration of less than 3 weeks.

In the previous research, Collaborative filtering algorithm (CF) [1, 159] was used to find people with similar interests and leverage their activities and preferences to provide relevant recommendations. However, the app usage prediction problem differs from traditional collaborative filtering settings, such as the Netflix rating prediction problem, in many aspects. First, user interaction with items such as apps is *brief* and *repetitive* in nature, whereas items like movies and books are usually watched/read once. Second, the user feedback of app usage is inherently implicit in the form of item clicks, as opposed to explicit feedback like ratings or comments. Additionally, app usage has a temporal ordering of clicks within sessions. Lastly and most importantly, app usage prediction must be made available *dynamically* as the user interacts with the system. The cold-start problem also exists in CF recommendation systems [214, 151, 102, 34] and Natarajan et al [131] tried to solve it by clustering users based on their (sparse) one-step item

transition probabilities.

For profiling mobile users, besides the demographics, e.g. age and gender, some researchers looked at modeling smartphone users based on their app usage behaviour. Jones et al. [79] identified three distinct clusters of users based on their app revisitation patterns: *checkers* who exhibit brief but quick revisit patterns, *waiters* who are split between short-medium revisitations and long revisitations, and *responsives* who exhibit sometimes brief and sometimes long revisit patterns. Zhao et al. [209] identified 382 distinct kinds of users from more than 10,000 individuals. In their work, users are represented by the average usage weight of each app category in different time periods. Furthermore, Li et al. [93] reported how the choice of mobile device models impact app selection, revealing the significance of device models on app usage.

To summarize, although many studies have been conducted on mobile app usage prediction and mobile user modeling, no existing works have been conducted on modeling the user cohort for the purpose of app category usage prediction. In addition, less research has been undertaken on the user cold-start problem. In this chapter, we aim to investigate whether users' app category usage behaviour can be predicted based on users' cohorts information. If yes, we then further validate that if the proposed framework can help the user cold-start issue.

For simplicity, we will often refer to app usage to mean app category usage in the rest of this chapter, unless otherwise stated.

Table 3.1: Top 10 popular app categories.

App Categories	%Logs	App Function Examples
productivity	28.6%	mail, calendar, notepad
social	10.6%	SNS, dating
tools	8.3%	caculator, screen lock, light
communication	6.8%	SMS, IM, free video calls
entertainment	5.1%	TV player, streaming video
utilities	3.3%	network, cleaner
sports	3.1%	live sports, sports news
music	2.7%	music player
lifestyle	2.5%	diary, discount, recipe
arcade	2.4%	games

3.2 Flurry Dataset

In this section, we introduce the dataset used in our study, which is collected from Flurry mobile analytics platform, a library that mobile developers integrate into their apps to measure app usage and allow in-app advertising. We collected a sample of logs from a week in March 2017 of more than 1.3 million logs with over 9K different apps and 12K users from the United States. Each log consists of the user's general app usage information, such as demographics, timestamp, app category, app id and time spent. Each app belongs to one of 45 categories ranging from social, communication to business, etc. Table 3.1 shows the most popular app categories and their proportions in all the logs, where the app categories are consistent with the Google Play App taxonomy [56]. Generally, users are more likely to interact with app categories like productivity, social, tools, communication, and entertainment apps nowadays. We also illustrate several examples of the most popular functionalities of the corresponding apps within each category. Table 3.2 shows various statistics of our dataset. Either Android or iOS operates the devices. 51.8% of app users are female, and most logs (over 70%) are generated by users between 25 and 54 years old [1]. Since the

[1]Users have been classified into five age ranges in our dataset: 13-17,18-24,25-34,35-54 and 55+.

Table 3.2: Overall statistics of the dataset.

OS	**%Logs**	**%Users**	**Device**	**%Logs**	**%Users**
android	78.0%	57.5%	Phone	94.3%	90.0%
ios	22.0%	42.5%	Tablet	5.7%	10.0%
Age	**%Logs**	**%Users**	**Gender**	**%Logs**	**%Users**
13-17	5.7%	9.0%	female	44.1%	51.8%
18-24	13.9%	18.4%	male	55.9%	48.2%
25-34	32.2%	30.3%			
35-54	43.2%	36.5%			
55+	5.0%	5.8%			

dataset is also leveraged in the following chapters, we refer to the Flurry dataset to mean the same dataset presented here in the rest of the thesis.

3.3 Cohort Modeling

One goal of this section is to extract informative and interpretable user cohorts based on different aspects of users' characteristics and app usage behaviors. The cohorts are used to draw "portraits" of users, which we believe will help predicting app category usage. A user cohort is a group of people who share common characteristics or experiences within a defined time-span. From previous work, three major dimensions have been used to classify users [59]: demographics, psychographics and behavioral. Demographics is the most popular dimension; it includes age, gender, occupation, education, religion, race, and location. Psychographics brings a better understanding of the users as a person by measuring psychological aspects, such as the way of living (lifestyle), interests and opinions [217]. Finally, the behavioral dimension focuses on the actual behavior of users, and includes spending/consumption habits, session frequency, usage rate, and loyalty status. In this work, we use these three dimensions to develop user cohorts based in a one-week time window. The taxonomy of user

cohorts is detailed in Table 3.3.

3.3.1 Demographics

The first user cohort is based on user demographics. In our case, these are age, gender, and operating systems. Some studies have shown that age and gender have an important impact on how users use apps on their smartphones [209]. For example, male users may be more engaged with sports apps and female users use more shopping apps. We group users into two gender cohorts: male and female, and five age cohorts: 13-17, 18-24, 25-34, 35-54 and 55+. Li et al. [93] reported how the choice of device models can impact the adoption of app stores, app selection and abandonment, online time, and data plan usage. Their work revealed the significance of device models against app usage, and suggest taking into account the device models as an essential factor in app recommendation tasks. We use operating system and group users into three cohorts accordingly: Android, iOS, and others.

3.3.2 Psychographics

Besides demographics, the psychological characteristics of a user, such as specific needs, preferences, and interests may also be a strong driver of app usage. For example, a young man interested in cooking may tend to use more recipe apps even if this category of apps is not broadly popular within his demographic group. Such differences between individuals and the communities to which they belong might be reflected in app usage. Thus, considering the psychographics of users may provide important insights in predicting app usage. We define user cohorts from three aspects

Table 3.3: User Cohort Taxonomy

Taxonomy	**Features**	**Cohort Modeling Dimension**	**Cohort Dimension Illustration**	**#Label**
Demographic	"Physical" Attributes	Gender	Female, Male	2
		Age	13-17, 18-24, 25-34, 35-54, 55+	5
	Technographics	Operation System	iOS, Android and Others	3
Psychographic	Interests	Absolute App Category Interests	Absolute Top K Preferred App Categories	45-961
		Relative App Category Interests	Relative Top K Preferred App Categories	45-1.5k
	Way of Living	Get up Time	Late-riser, Normal, Early-bird	3
		Bed Time	Night-bird, Normal, Early-to-bed	3
		Nocturnal phone use	Heavy-use, Normal, Light-use	3
	Community	Temporal App Interests	night communicators, evening TV watchers, weekend morning gamers, etc.	114±7
Behavioural	Engagement	Time Spent	Tourists, interested, average, active and VIP	5
		Access Frequency	Tourists, interested, average, active and VIP	5
	Revisitation	Revisitation Patterns	Checkers, Waiters, Responsives	3

of users' psychographics (see Table 3.3): interests, the way of living, and communities.

Interests Users' specific interests may be indicative of future intent on app usage. For example, users who love sports might potentially access sports apps and consume more sports-related content than others. Therefore, it is important to consider users' interests when predicting app usage. To group users based on their app preferences, keeping the most popular app categories for each user based on their historical app access frequency is a straightforward way of doing this. However, previous researchers [94, 139] have found that the app popularity distribution follows Zipf's law, which indicates that only a few apps have high installation/usage whereas many apps have low installation/usage. Users grouped by their *absolutely* top app categories may lead to a skewed distribution meaning that most users may be classified into a few cohorts, mostly highly popular apps, such as social-networking, productivity, and communication.

Other than selecting the *absolutely* top app categories, we employ another strategy to select the *relatively* most popular app categories for each user by normalizing across all users. This strategy can properly represent users' app category preferences as well as keeping users' specific preference characteristics. Specifically, the top k apps for user u are selected based on the popularity score $P(a, u)$ for each app a, which is calculated based on usage frequency of that app category for user u and the usage frequency of the corresponding app category for all users:

$$P(a, u) = \frac{f(a, u)}{\sum\limits_{u_i \in U} f(a, u_i)} \tag{3.1}$$

where app $a \in A$, and A is the set of all the app categories engaged by u. $f(a, u)$ represents the usage frequency of the app category a for user u in

a given period while U is the set of all users. Given this popularity score $P(a, u)$, we can select the top k app categories to represent the interests of user u. Through this normalization, the "niche" popular app categories are used for representing that user.

Table 3.4: Description of the cohorts generated based on different selected app category sets (% User Identified: percentage of users are uniquely identified by the selected app categories; # Cohorts: number of cohorts generated; Avg. # of Users: average number of users in each cohort; Std: standard deviation of the number of users in each cohort.)

Interests Rep.	**% User Identified**	**# Cohort**	**Avg. # of Users**	**Std.**
Absolutely Top 1	0%	45	65	142.67
Absolutely Top 2	4.0%	364	8	18.62
Absolutely Top 3	18.5%	961	3	5.01
Relatively Top 1	0%	45	62	37.82
Relatively Top 2	4.7%	582	5	5.21
Relatively Top 3	33.5%	1522	2	1.71

Selecting appropriate k for the top k app categories can be also crucial for the user representation. Consistent with prior results [181, 170, 8], we find that if we deem k as the number of all the app categories used by that user, 77.3% of the users can be uniquely identified (i.e., each of those users belongs to a user cohort that consists of exactly that one user). To empirically evaluate this, we present the results of the cohorts generated by varying k from 1 to 3, as shown in Table 3.4. We can observe that by selecting k equal to 3, most of the user cohorts consists of only 2-3 users whereas many users can be uniquely identified (i.e., many cohorts only consist of exactly one user). Therefore, we enumerate different interests representation, setting $k \leq 3$ in the rest of the work.

Way of Living We focus on when a user gets up or goes to bed, and how actively the user uses the phone during the night (midnight to 6 AM).

Get-up & Bed time. Murnane et al. [128] found that users' smartphone app usage patterns vary for individuals with different body clock types. In

this work, we focus on when a user gets up or goes to bed [209] identified the get-up time and bed time by the charge cycle of smartphone batteries; however, this information is not available in our dataset. Following a similar methodology to [212], we assume that the users start to "stop" using the phone before getting to sleep and pick up the phone when they get up. If there is an idle time of phone usage for longer than 4 hours at night (i.e., the idle time starts between 8 PM and 5 AM; ends between 4 AM and 1 PM), we identify it as the sleeping time. We then use the timestamp of the start and end of this sleeping time respectively as the "go-to-bed" time T_b and "get-up" time T_g.

Nocturnal Phone Usage. This measures how actively a user uses the phone during the night. Following the methodology in [212], the total duration of all the active periods of app usage during night time (between midnight and 6 AM) D_n is computed as the feature for representing nocturnal phone usage.

After obtaining those variables (go-to-bed time T_b, get-up time T_g and nocturnal usage duration D_n), we need to further group them into user cohorts. We are particularly interested in those traits that make the users different from others. Following from [212], we first normalize those discrete features using z-score. By assuming those features follow the Gaussian distributions, we then calculate the mean and standard deviation for each of those features. As shown in prior work [212], those features far away from the mean of the feature with more than one standard deviation (std) can be utilized for representing the special user traits. Therefore, as shown in Table 3.3, for each "way of living" feature, a pair of semantic labels is generated for two ends of the feature distribution, i.e., lying outside of the interval of (mean $\pm$ std). For example, based on the distribution of get-up time T_g for all users, if a user gets up within the time period of mean $\pm$ std,

we will label his/her cohort as "normal". Otherwise, we will label him/her as "later-riser" if the user gets up later than the timestamp of "mean+std" and as "early-bird" if he/she gets up earlier than the timestamp of "mean-std".

Communities Several studies have clustered users into different communities based on their temporal app usage patterns. For example, Zhao et al. [209] identified 382 distinct kinds of users using their clustering method based on the usage frequency of different app categories during specific time periods. Within their proposed clustering method, they identified different types of users and ultimately label them with a community label, such as night communicators, evening learners and car lovers. Different from *interests* described in Section 3.3.2, the *communities* capture the more fine-grained temporal app usage patterns.

In our work, we utilize the same methodology to assign each user to different communities. Based on our dataset, each user is represented by a vector C_v of 45 (categories) $\times$ 4 (time periods) $\times$ 2 (weekends and workdays) for a total of 360 dimensions. By applying the best performing k-means-MeanShift hybrid clustering algorithm described in [209], we obtain a total of 114 $\pm$ 7 clusters.[2] This k-means-MeanShift clustering algorithm combines the benefits of multiple standard clustering algorithms, is computationally feasible, and finally is able to automatically determine the ultimate number of clusters. We find that our clustering results are similar to the findings in [209] that many meaningful communities exist in our clusters, such as night communicators, evening TV watchers and weekend morning gamers.

[2]Since we use 5-fold cross validation in our performance evaluation, different number of clusters are generated in different folds.

3.3.3 Behavioural

The third dimension is based on behavioral patterns. We consider two aspects of users' behavioral characteristics, which we show in Table 3.3: engagement and revisitation.

Engagement Within the context of web browsing, Lehmann et al. [92] created five types of user groups based on their frequency of visiting the site over a month: tourists, interested, average, active and VIP users. They identified that the proportion of specific types of users based on their engagement will be different across websites. In this work, we also hypothesize that users with different engagement levels may behave differently in terms of app usage. Additionally, we measure not only users' engagement in terms of how frequently they access apps, but also the more fine-grained total time spent (i.e., total dwell time). The latter represents the total duration of users accessing mobile apps in the given period.

Following the five-level engagement level definition in [92], we propose the following strategy for defining users' engagement cohorts based on their mobile app behavior patterns. We group users into five different engagement cohorts of duration and frequency, respectively, by using the quantiles at 20%, 40%, 60%, 80% and 100% as the breakpoints, resulting in those five cohorts: tourists, interested, average, active and VIP.

Revisitation Jones et al. [79] present a revisitation analysis of smartphone use. They propose that users could be clustered into three different types based on their revisitation patterns, where a revisitation curve for a particular user is constructed by considering the in-between duration in launching *any* app on their phone. They grouped the users based on the revisitation curves into three user cohorts: *checkers* (users exhibiting

brief revisit patterns lightly skewed towards fast revisitation of less than 4 hours), *waiters* (users exhibiting longer revisit patterns longer than 16 hours), and *responsives* (users exhibiting a hybrid of brief and long revisit patterns).

Following [79], we use an exponential scale for revisit interval bins, which are 1, 2, 4, 8, 16 and 32 minutes; 1, 2, 4, 8, 16, 32 hours (i.e., 1.3 days), 64 hours (i.e., 2.6 days), 128 hours (i.e., 5.3 days), and above (i.e., >5.3 days). A revisitation curve characterizes a user by its 15-dimensional vector R_v, where each dimension corresponds to the frequency of revisits within the corresponding bin. These curves are like a "signature" of users' behavior in launching mobile apps. We iteratively apply k-means for a varying number of clusters and use within-groups sum of squares to plot the variations as a function of the different number of clusters. We then pick the "elbow" of the curve as the optimal number k of clusters. Based on this simple k-means method, we identify a substantial trichotomy of user cohorts within their revisitation patterns: checkers (which accounts for 39.6% of users), waiters (13.5%) and responsives (46.8%).

3.4 App Category Usage Prediction

In this section, we introduce our approach of using cohort modeling for the purpose of app category usage prediction. We start by formalizing the prediction problem, and then discuss how to assign the cohort information when a new user comes. To assign users into cohorts of different granularity, we also describe how we build combined user cohorts. Finally, we evaluate the performance of the proposed prediction method, including for the user cold-start problem.

3.4.1 Problem Formulation

In our work, we aim to predict the app category a new user will use based on their cohort information. Our aim is to overcome the data sparsity issue and to guarantee efficient real-time prediction. Each user can be characterized by his/her cohort information, i.e., demographics, psychographics, and behavioral patterns. For each user, given the current time (time of day and day of week) and his/her cohorts information, we want to predict the app category he/she will use. This prediction task can be formalized as a multi-class prediction problem. There are $K = |A|$ classes for this prediction task, where A is the set of all app categories in the dataset.

The **app category usage prediction problem** is formally defined as follows: Given a list of app categories $\{a_1, a_2, ..., a_i\}$, the users' cohorts information C and temporal context T, the problem of app usage prediction is to find an app category $\hat{a}$ that has the highest probability of being used under C and T. Specifically, we aim to solve:

$$\hat{a} = \underset{a_i \in A}{\operatorname{argmax}} P(a_i|C, T)$$

3.4.2 User Cohorts Assignment for New Users

Since the user cohort based approach aims at addressing the user cold-start problem, we split the training and test set based on users instead of log entries. In the training set, the users are assigned to the specific cohorts based on their usage logs. For example, the community cohorts are generated based on the clustering results of users in the training set. During the test stage, for all test users whom have not been seen by the system before, we assign those users to existing cohorts in the training set and

then proceed to the prediction task. During the assignment process, we need to compare the new users with all "old" users in the vector space, so we first introduce the representative vectors for representing users' cohorts information as vectors. For scalar based cohort: e.g., demographics, the representative vector refers to the one-hot encoded vector of the categorical scalar. For cluster-based cohorts: e.g., community and revisitation, the representative vector is the vector used in clustering. We then propose three assignment approaches to assign the test/new users to the most accurate cohort within all the different cohorts taxonomies as described in Section 3.3.

Nearest centroid classifier (NCC) If we want to determine which existing cohort a new user belongs to, the straightforward way is to find the nearest cohort. Therefore, we propose to use the nearest centroid classifier [185] as the first assignment methodology, which is a classification model in machine learning that assigns observations to the class of samples whose centroid is closest to the observation. In our scenario, given existing users' representative vectors $\{(\overrightarrow{x}_1, y_1), ..., (\overrightarrow{x}_n, y_n)\}$ with cohort labels $y_i \in Y$, we compute the per-cohort vectors:

$$\overrightarrow{\mu_l} = \frac{1}{|C_l|} \sum_{i \in C_l} \overrightarrow{x}_i$$

where C_l is the set of indices of samples belonging to cohort label $l \in Y$; the assignment function for the cohort label assigned to a new user $\overrightarrow{x'}$ is:

$$\hat{y} = \underset{l \in Y}{\operatorname{argmin}} \| \overrightarrow{\mu_l} - \overrightarrow{x'} \|$$

K-nearest neighbor classifier (KNNC) The second approach is to apply the k-nearest neighbor (KNN) [47] rule, which is one of the most

straightforward non-parametric techniques in pattern classification. The basic idea of k-nearest neighbor classifier is: an object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. Here we set $k = \sqrt{n}$, where n is the amount of unique cohort labels. Similar to NCC, representative vectors are used when calculating the distance between two users. More specifically, given a new user x' and a similarity metric d based on Euclidean distance, KNN classifier performs the following two steps: (1) It runs through the whole training set computing d between the new user x' and each user in the training set. We state the k users in the training set that are nearest to x' as the set C. (2) It then estimates the conditional probability for each cohort label, that is, the fraction of users in C with that given cohort label:

$$\hat{y} = \underset{j \in Y}{\operatorname{argmax}} P(y = j | X = x') = \frac{1}{k} \sum_{i \in C} I(y^{(i)} = j)$$

where $I(x)$ is the indicator function which evaluates to 1 when the argument x is true and 0 otherwise. Finally, the new user x' gets assigned to the cohort label with the highest probability.

Classifiers trained based on the existing cluster labels (RF) Given the NCC assignment, some of the labels will be assigned based on the centroids of clustered results. However, assigning new points based on distance in a clustering algorithm is complex because the results of a clustering algorithm may be imperfect; they only present a snapshot of a (hopefully good) segmentation within the current data. With more data coming in, the cluster may change. Therefore to make the assignment more robust given a particular clustering segmentation, we can train an additional classifier where the resulting clusters are treated as different classes. In that

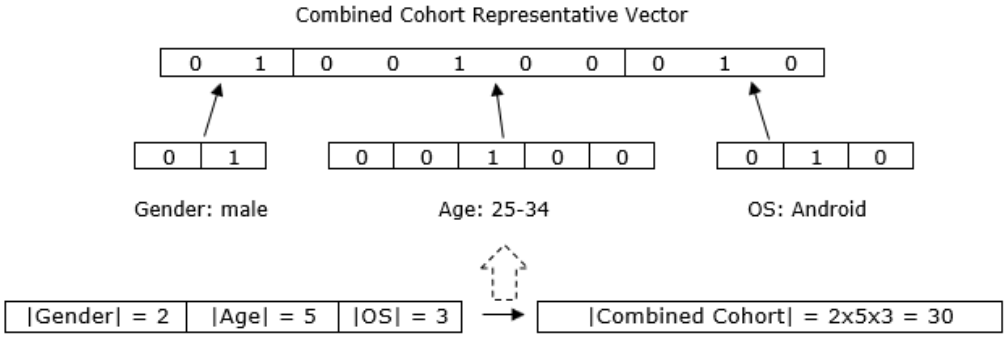


Figure 3.1: Structured representative vector of combined user cohorts: combining age, gender and operation system groups would potentially result in 30 different cohorts.

way, we can account more intuitively for the non-robustness of the clustering labels. Besides, as we expect the clustering to reflect "some structure", it is a cheap and straightforward way to encapsulate that structure. Following this, the classifier learns $P(c|x)$ based on users' representative vectors and corresponding cluster labels. When a new user x' appears, we can directly predict which class the new user belongs to instead of assigning him/her based on distance or neighbors. So here, we propose to use an additional classifier based on the Random Forest algorithm, which is efficient in assigning the cohort label to new users:

$$\hat{y} = \underset{c \in Y}{\operatorname{argmax}} P(c|x')$$

3.4.3 Combination of Multiple User Cohorts

Besides predicting users' next app category solely based on one type of cohorts, we also want to consider multiple types of cohorts together. For example, as shown in Figure 3.1, a combined cohort with different types of demographic cohort features could be generated to describe a user. This combined cohort would have 30 different labels since there are potential 30

different compositions based on age, gender, and operating system cohort labels. For instance, we can observe one of the combined cohorts in Figure 3.1: "Android male user cohorts aged 25-34". It is possible to generate a new cohort based on a combination of any cohort dimensions listed in Table 3.3.

The cohort assignment for new users within these combined cohorts follow the same methodologies described in Section 3.4.2. The representative vectors x are updated by concatenating the original vectors x_i of each selected cohort:

$$x = [x_1, x_2, ..., x_i], i \in S_c$$

where S_c is the set of selected cohorts to be combined. Figure 3.1 illustrates the generation of such new representative vector.

3.5 Experimental Results

In this section, we firstly empirically demonstrate how our proposed cohorts can be used to improve the prediction of users' app category usage. Secondly, we investigate whether our proposed approach can help addressing the user cold-start issue when compared with other prediction mechanisms.

3.5.1 Experimental Setup

We apply 5-fold cross-validation to evaluate each model. At each time, we split all the users into training, validation and test set: the logs of three-fold of the users are used as the training set, one-fold is validation set and the remaining one-fold users are used as the test set.

Table 3.5: Measurements of next app category prediction based on different cohorts information. All the results are statistical significant ($p < 0.01$) using the two tailed t-test compared to the temporal context only baseline. Bold scores state the best performance in each cohort.

User Cohorts	# Cohorts	Assignment	Measurements			
			acc	pre	rec	F1
Context Baseline						
Hour + Weekday	-	-	0.416	0.173	0.416	0.244
I. Single Cohort						
(a). Demographics						
Age	5	-	0.441	0.197	0.441	0.272
Gender	2	-	0.443	0.197	0.443	0.272
Operating system (OS)	3	-	0.447	0.234	0.447	0.291
(b). Psychographics						
Interests: Top1-Absolutely	45	-	**0.686**	**0.666**	**0.686**	**0.665**
Interests: Top2-Absolutely	364	NCC	0.559	0.515	0.559	0.496
Interests: Top3-Absolutely	961	KNNC	0.467	0.387	0.467	0.362
Interests: Top1-Relatively	45	-	0.555	0.528	0.555	0.510
Interests: Top2-Relatively	582	NCC	0.547	0.494	0.574	0.485
Interests: Top3-Relatively	1522	KNNC	0.418	0.393	0.418	0.402
Way of Living: Sleep Time	3	-	0.440	0.203	0.440	0.270
Way of Living: Get-up Time	3	-	0.439	0.206	0.440	0.271
Way of Living: Nocturnal	3	-	0.443	0.197	0.443	0.272
Communities	115	NCC	0.572	0.550	0.572	**0.517**
(c). Behavioural						
Time Spent	5	-	0.444	0.222	0.444	0.286
Frequency	5	-	0.444	0.222	0.444	0.286
Revisitation Pattern	3	NCC	0.444	0.219	0.444	0.283
II. Combinatory Cohort						
(d). Demographics						
Age + OperatingSys	15	NCC	0.443	**0.255**	0.443	**0.298**
Gender + OperatingSys	6	NCC	0.447	0.240	0.447	0.291
Age + Gender	10	NCC	0.436	0.206	0.436	0.271
Age + Gender + OperatingSys	20	NCC	0.436	**0.255**	0.436	**0.298**
(e). Psychographics						
Getup + Nocturnal	6	NCC	0.437	0.236	0.437	0.280
Sleep + Get-up + Nocturnal	17	NCC	0.431	**0.248**	0.431	**0.291**
Top1-Absolute + Community	363	NCC	0.682	0.662	0.682	0.660
Top1-Absolute + Nocturnal	80	NCC	0.681	0.661	0.681	0.659
Top1-Relative + Community	489	NCC	0.577	0.565	0.577	0.543
Top2-Absolute + Community	1034	KNNC	0.572	0.529	0.572	0.514
Top1-Relative +Nocturnal	88	NC	0.548	0.519	0.548	0.505
(f). Behavioural						
Time Spent + Revisitation	12	NCC	0.439	0.244	0.439	0.295
Frequency + Revisitation	12	NCC	0.439	0.244	0.439	0.294
(g). Across Taxonomies						
Age + OS + Revisitation	69	NCC	0.430	0.290	0.430	0.329
Age + OS + Time Spent	96	NCC	0.426	**0.292**	0.426	**0.332**
Age + OS + Top1-Absolute + Time Spent	649	NCC	0.600	0.552	0.600	0.565

3.5.2 App Usage Prediction

Our goal is to predict which app category the user will use next. We use a set of state-of-the-art algorithms to build models for our prediction problem: (1) XGBoost (XGB) [30], as an example of ensemble learning method; (2) K Nearest Neighbours (KNN) [33], as an example of the non-parametric method for classification; (3) L2-regularized Logistic Regression(LR) [62], as an example of linear classifier. The parameters in each model, e.g., K in KNN, the number of used trees, the maximum depth of the trees and the learning rate are tuned on the validation sets.

The features include the user cohorts and the temporal context (different hours of a day and days of a week). Additionally, we use the prediction model only based on context features as our benchmark, which takes all the users as they are "the same" (from one cohort). We report four metrics with 5-fold cross-validation: accuracy (acc), precision (pre), recall (rec), and F1-measure (F1). We test the prediction performance of the proposed user cohorts based on each cohort taxonomy individually and then when combined. We consider several combinations.

Table 3.5 presents the results. For the combined user cohorts (Table 3.5.II), since there are a large number of compositions across different cohort taxonomies, we only report the results of the top performing combinations, on which we test the combinations among any two, three or four different cohorts. Note that we also only report those combinations for which we observe a performance boost compared to using any individual cohort feature. For the different cohort assignment methods we employ for new users (see Section 3.4.2), we only report the one with the best performance.[3]

[3]We use '-' in Table 3.5 to denote such scenario when there is no single winner for the three assignment approaches (i.e., all those methods result in the identical cohort assignment).

Firstly, we find that compared to the context-only baseline approach, all the cohort based models achieve better performance on all metrics; all those improvements are found to be statistically significant ($p < 0.01$) through a two-tailed t-test. This demonstrates that not surprisingly, incorporating user characteristics on top of the temporal context help to improve app category usage prediction.

Secondly, we investigate more specifically the gains obtained by the single cohort models (Table 3.5a-c). We can observe that in general the psychographic cohort models (Table 3.5b) perform better than the demographic (Table 3.5a) and behavioural (Table 3.5c) cohort models by a large margin. When looking at the psychographic cohort models, we can find that the user interests cohort based on the "absolutely top 1 app category", and "communities" are the best predictive models. This indicates that users that have common interests or belong to the same communities may behave more similarly in their app usage behavior, which is not only constrained to their past, but also their future app usage. However, compared to the baseline, we observe only marginal improvements on the "way of living" (Table 3.5b), behavioral (Table 3.5c) and demographic (Table 3.5a) cohort models. This is not surprising as most of those models contain only a small number of cohorts (3-5) and are not sufficiently discriminative.

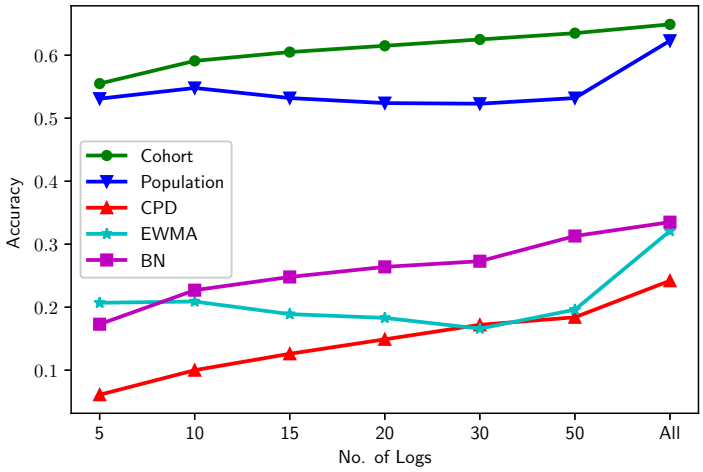
Finally, when examining the combined cohorts (Table 3.5d-g), we find that they generally perform better than when using any one of them respectively. For example, all of the "way of living" cohorts outperform any of them when individually used. When combining demographics and revisitation behaviour patterns (Table 3.5g), we observe an increase of 10% performance improvement, compared to using demographics only (Table 3.5d). However, it is worth noting that combining "interests" with any other cohorts (Table 3.5e) would result in only marginal improvements and some-

times even inferior performance. This implies that enriching the cohorts with additional information might not always necessarily help. Another interesting observation is that most of the time, the simple Nearest Centroid Classifier (NCC) cohort assignment approach outperforms KNCC and RF approaches.

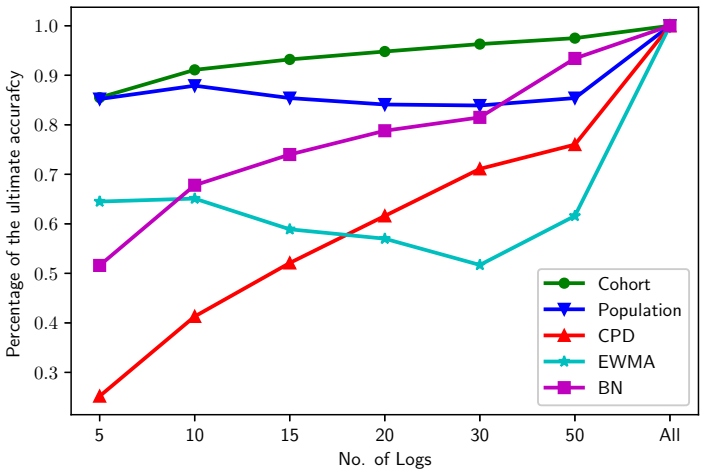
3.5.3 User Cold-Start Problem

In this section, we focus on analyzing whether the cohort-based prediction model can help solving the user cold-start problem. We adopt the best performing cohort model (see Table 3.5) for the rest of the experiments.

The baseline approaches we compare against are both personalized and population-based prediction models. Although there are many personalized models (see Section 2.1.2.1), some of them are not applicable because they use additional information. We select CPD [163], EWMA [163], and BN [218] as our comparative baselines for personalized models as they can be used with our dataset. CPD (Cumulative Probability Distribution) computes the probabilities of each used app in all the specific time slots based on historical app usage time series for that user, and selects the app with the highest probability at the prediction time based on its time slot. EWMA (Exponentially Weighted Moving Average) replaces the cumulative probability in CPD with exponentially weighted moving average [75] so that the newer data points have higher influence in the prediction. BN [218] is a Bayesian Network model that relies on both app usage and time context and calculates a linear combination of the user's last used app and the second last used app for the final prediction. Regarding the population-based prediction models, following [41], we generate the baseline model based on our available predictive features and utilize random forest to combine all



(a) Accuracy Comparison



(b) Convergence Speed Comparison

Figure 3.2: Performance comparison among different prediction models with limited amount of historical user logs. Three personalized baseline models: CPD [163], EWMA [163] and BN [218], one population-based (generic) baseline model [41] and our proposed cohort model.

those features for the next app category prediction. We combine a set of extensive features that could be extracted from our dataset, which include hour, weekday, last used apps, historical popularity of users' app usage in different time windows, one hour, one day and all history, and periodicity (intervals between app usage) [100].

To explore whether the user cohorts based methodology perform better

especially for new users (for which there are limited interaction data), we randomly select 20% of the users in our dataset, and use them to simulate the new users by increasing the number of interaction data (logs) available we consider for each user. Specifically, we extract different amount of logs from 5 to 50 for each user to simulate various severity of the cold-start problem. We hypothesize that models that handle well the cold-start problem tend to perform competitively even with very limited amount of user logs.

Figure 3.2 presents the prediction performances of all baseline models and our proposed cohort-based prediction model, given different amount of historical logs available for the test users (x-axis). The results are averaged across all the test users. Firstly, we can observe in Figure 3.2a that when the amount of user interactions is limited, all personalized models perform worse (accuracy is below 35%) than the population baseline models and the cohort models. CPD is the worst for the personalized model when there are logs, followed by EWMA and BN. However, both population and cohort models can achieve over 50% accuracy even with the limited amount of historical user data. Secondly, we observe from Figure3.2b that only the cohort-based model achieves over 90% of the best accuracy when only 10 entry logs are considered. The performance steadily increases as more and more logs are available. This demonstrates that our proposed user cohort based model outperforms both the personalized models and the generic model for the user cold-start problem.

3.6 Conclusion

In this chapter, our goal is to identify meaningful user cohorts information to help with the app category usage prediction problem. We show

that besides personalized prediction approaches, users' app category usage behavior can be predicted based on cohorts information. Based on our proposed taxonomies of user cohorts modelling, we find that psychographics (interests and community) perform best. Additionally, we identify that our proposed user cohorts based prediction outperforms both the personalized and population-based models on the user cold-start problem.

Through our study, we demonstrate the value of cohorts, especially for new users. This is promising as cohorts information could be used not only on their own but also in combination with other signals as they become more present. For a new user without much interaction data, general cohorts information such as interests or community could be collected, e.g., a user could label themselves as car lovers or young parents. Users' app category usage could be predicted with relatively high accuracy using this basic cohort information. The cohort labels can also be utilized to explain the prediction model, enabling the recommendation to be more transparent and interpretable [204]. After improving the existing app usage prediction methods, especially for solving the *cold-start* issue for new users, we are able to infer users' preference on the next app no matter the users have sufficient data or not. In the next chapter, we further aim to understand how long user will stay with apps.

Lastly, we need to acknowledge that there are several limitations of this work, which we would like to address in future work. Firstly, our dataset only consists of relatively short-term app usage and it would be interesting to study signals that could relate to long-term characterizations of user cohorts. Secondly, we mainly focus on next mobile app category prediction in our work. Our method is general while it is worthwhile extending this to further investigate our cohort-based methods on next app prediction [8]. Finally, the cohort taxonomy we define in our work is only a first step

proof-of-concept, and can be refined with more fine-grained cohorts when relevant interaction or user profile information become available.

Chapter 4

What and How long: Prediction of Mobile App Engagement

After solving the *cold-start* problem in app usage prediction for new users by leveraging the cohort-modelling approaches in the previous chapter, we now would be able to infer users' needs on "what" app users would use effectively no matter the sufficient data of individuals are available or not. However, only predict the next app user would use is not enough for providing more satisfying services to users. In this chapter, we further investigate if we could predict "how long" user would stay with an app and then investigate a novel app engagement prediction problem, where we consider which app user will use and how long the user will stay with this app an aggregated measure of users' app usage behaviour. This chapter addresses our research questions **RQ 1.3** and **RQ 1.4**, as specified in Section 1.1.

Properly monitoring user engagement is one of the key ingredients of success to improve user experience and retention. Users are generally engaged with an app when they appreciate the mobile app content to which they have given their attention. One way user engagement has been measured

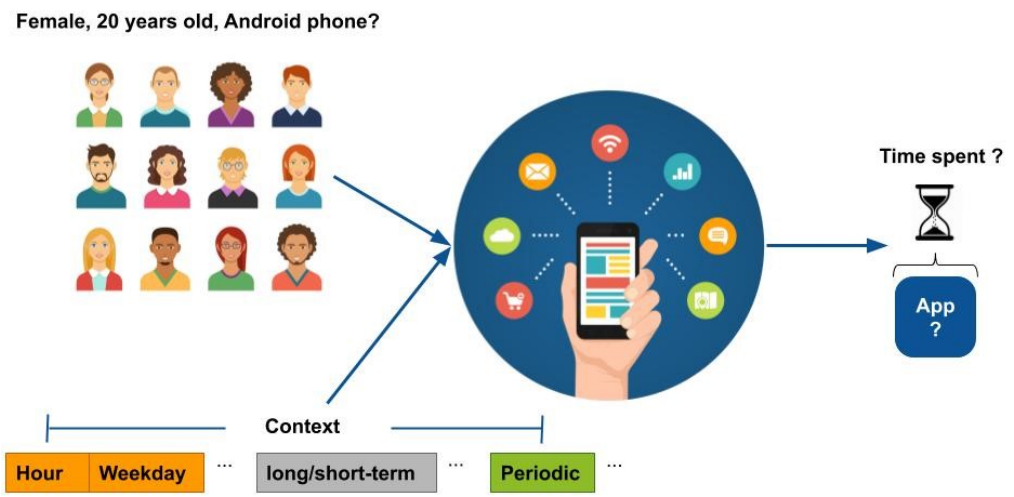


Figure 4.1: Overview of the next app and app dwell time prediction.

at a large scale is by tracking how long users spend with content, e.g., the time spent on a webpage. Studies on user engagement in the contexts of desktop-based systems [173] and websites [39, 200] have shown that simple metrics such as dwell time are meaningful and robust in modelling user engagement. More importantly, these studies have shown that with an awareness of engagement, users' experience with a system can be substantially improved which in turn leads to user growth, user retention, and increasing revenue streams.

However, even many past works in mobile computing have investigated how individuals download, install and use different apps on their mobile devices [203, 135, 157, 72], to our knowledge, few studies have examined how to effectively predict how long users would stay with a mobile app. Real-world mobile app usage behaviour is a complex phenomenon driven by a number of competing factors. Game apps, in general, have a higher probability to be used for a long period, whereas weather app is, not surprisingly, shorter. Intuitively, there are certain times in a day when a user might be more likely to engage with certain mobile apps: John might be more likely to engage with game apps for a longer time at night after work. User characteristics

could also make a difference: female users may spend longer time with shopping apps than male users [143]. Despite that the averaged total time spent on different app categories were reported in [16, 191, 94], there is little research that comprehensively analyzes the dwell time of mobile apps. This motivates the research question **(RQ 1.3)**:

- *What are the factors (user characteristics and contexts) that influence the dwell time a user spends on an app?*

As far as we know, there have been many works conducted for modelling users' behaviour on choosing one particular app under different contexts. However, they do not describe how users engage with that app. Attention is a scarce resource in the modern world. For instance, a user may become immersed in the video watching or quickly abandon it – the distinction of which will be clear if we know how much time the user spent interacting with this app given different contexts. Next app prediction is characterised as the willingness to use an app, whereas engagement is the usage pattern after accessing the app. Though most researchers have focused on measuring which app user will use [99, 100, 8, 193], the engagement of apps is still not well understood. Hence, we consider which app user will use and the engagement level of how long the user will stay with this app an aggregated measure of users' app usage behaviour. Furthermore, app engagement (dwell time) is much dependent on the app content itself. As we mentioned above, checking the weather app is always shorter than playing games. So it is meaningless to predict how long a user will stay regardless of which app user is engaging. Given the inter-dependency between an app and app dwell time, we aim to predict the next app as well as app engagement (dwell time) as shown in Figure 4.1, which is another research question **(RQ 1.4)** we seek to answer:

- *Can we simultaneously predict which app user will use next and how long the user will stay on this app?*

Answers to these questions have a profound impact on the success of an app, as engagement awareness can radically improve users' experience with digital services [149].

To answer **RQ 1.3**, we first demonstrate how different factors affect users' app dwell time by presenting the first population-level analysis of how different contexts affect the app usage duration. The analysis is based on a large-scale mobile app usage dataset with more than 1.3 million logs from over 9k unique apps and 12k users, The data was collected from Flurry mobile analytics platform (Flurry dataset introduced in Section 3.2) over a period of one week. Specifically, we consider the influential factors from two aspects: user characteristics (e.g., age, gender, device type, historical preferences) and context (e.g., hour, weekday, last used app, periodic pattern). We find that, for example, users between 20 and 40 years old are more likely to have a shorter dwell time than teenagers and older people. Both the demographics and device type have an influence on how long user stay with mobile apps. Furthermore, we also establish that the app dwell time differs significantly across different time in a day. More importantly, we observe that users' app dwell time maintains periodic patterns and follows historical trends. For example, some users spend a similar length of time to regularly check the shopping apps every day (i.e., after every 24 hours). Additionally, users have different historical engagement habits on their app usage duration. For example, some users always prefer staying long on social apps, while others tend to only check for a short while.

Based on the comprehensive analysis of users' app dwell time, we are able to conduct the study of how users' app dwell time can be inferred from these

features. We then set to answer **RQ 1.4** – how to predict the next app and how long the user will stay on this app simultaneously? Based on past work on next app prediction [99, 100, 8], we propose several joint prediction models (sequential, stacking, and boosting) for such app engagement prediction, whereas the dwell time is represented as discrete levels (light, medium, and intensive) defined based on different app categories. Additionally, different from personalized models in prior work [99, 100, 8], our models leverage the community-behaviour patterns by extracting predictive features from the entire user population.

To the best of our knowledge, this is the first empirical study on inferring predictive features at the scale of millions of logs for app dwell time, assessing how user characteristics and context features impact the dwell time with a mobile app. Our proposed predictive models are empirically validated to be effective for this novel task.

The rest of this chapter is organized as follows. We first review the related work in Section 4.1. Section 4.2 presents fundamental analysis that demonstrates the characteristics of the data for conducting our study. Section 4.3 introduces all the features we extracted from our dataset and provides insights on how these features could impact app dwell time prediction. Section 4.4 presents our methodology for predicting the next app and app dwell time simultaneously, and we propose three joint prediction strategies to solve this problem. We then show the experimental results of our proposed models and analyze the effectiveness of different models in Section 4.5. We discuss the implications and potential limitations of this work in Section 4.6 and conclude in Section 4.7.

4.1 Related Work

4.1.1 Engagement Measurements

Approaches to measuring user engagement of online services can be divided into three main groups: (a) self-reported engagement, (b) cognitive engagement, and (c) online behaviour metrics [92]. In the first group (a), questionnaires and interviews are used to elicit user engagement attributes or to create user reports and to evaluate engagement [83]. The second type of approach (b) uses task-based methods and physiological measures to evaluate the cognitive engagement (e.g., facial expressions, vocal tone, and heart rate) using tools such as eye-tracking [43], heart rate monitoring, brain readings from headset [113], swipe on the screen [134] and mouse tracking [71]. However, these two types of methods have known drawbacks, e.g., reliance on user subjectivity of the self-reported engagement, and only be able to measure a small number of user interactions of the cognitive engagement.

The third type of approach (c), adopted by the web-analytics community, has been studying user engagement through online behaviour metrics that assess users' depth of engagement within a site, e.g., the time spent on a webpage. Studies on user engagement in the contexts of desktop-based systems [173] and websites [39, 200] have shown that simple metrics such as dwell time are meaningful and robust in modelling user engagement. For example, the time spent on a resource has been validated as an effective metric for measuring user engagement in the context of web search [2, 12], and recommendation tasks [198]. Kelly et al. [80, 49] consider dwelling times as an indicator of page relevance or user satisfaction during search engine interactions. Yi et al. [198] recommend designing dwell time based

user engagement metrics and claim that this would enable them to extract better user engagement signals for training recommendation systems thereby optimizing for long term user satisfaction.

Therefore, we also propose to use the metric of time spent on mobile apps (dwell time) as our user engagement metrics within a large-scale dataset. For now, minimal research has been done for modelling mobile app dwell time from a large-scale dataset; only the basic aggregated statics on app usage time was reported. Falaki et al. [45] found that 90% of app usage sessions would be less than 6 minutes and Xu et al. [191] reported that the majority of total network access time for all apps is from 10 seconds to 1 hour for each subscriber in one week. Li et al. [94] reported usage time for different app categories in total and found that communication apps account for 49% cellar time against all apps. Böhmer [16] found that the Libraries & Demos apps (default Updater, Google Services Framework, etc.) have the longest average usage time from opening to closing. However, these summarized basic statistics of app usage time can not provide an in-depth understanding of what factors could influence the dwell time user spend on an app and whether we could predict how long user would stay with an app. In our work, we conduct the first empirical analysis of dwell time during app usage based on a large-scale data set collected from thousands of users; and we are the first to assess how different kinds of features (e.g., demographics, device type, hour of day, and last used app) impact the mobile app dwell time.

4.1.2 Dwell Time Prediction

Much research work has been conducted on predicting which app user will use (a.k.a. next app prediction introduced in Section 2.1.2.1). However, as

far as we know, no research has been done for predicting how long users will stay with an app. In this chapter, by leveraging a large-scale data set of users' app usage logs, we could be able to model users' engagement (dwell time) within specific apps by exploiting user characteristics, temporal context, and short/long-term behavioural patterns. Most importantly, we investigate the challenges of simultaneously predicting which app user will use and how long the user will stay with this app.

Although no research has been conducted on the app dwell time prediction, some researchers investigated the dwell time modelling and prediction in other areas, e.g., session-based recommendation (SBR) systems [15, 215, 179], videos watching [188], news and non-news pages reading [153, 66], and media streaming [175].

The SBR tasks aim to predict the next click/buying/dwell time based on users' interactions in a session, e.g., buying one item after viewing several products (within a commerce site). They stated that dwell time should be used as a proxy to user satisfaction of the clicked result since clicked through or not is not enough to identify the satisfaction of the user. In their scenario, researchers aim to predict which link/product user would click among a list of recommended similar results (under specific search query). Additionally, they assumed that the longer user stays with the clicked result, the more satisfied the user will be with the result. However, in our dwell time prediction problem, a user would use an app occasionally with no recommendation context, and the longer user stays with this app does not directly mean the user is satisfied with it or not. The dwell time may be affected by the specific app (e.g., a weather app or game app), or whether the user accessed it during commuting or at night while they have more leisure time. Therefore, predicting how long a user will stay with an app is to predict the usage pattern while using the app. It would allow

the service provider to optimize the user experience along with its business goals, the apps can be tuned to be more exploratory or exploitative based on the expected length of the usage. The SBR tasks can be solved by item-to-item and matrix factorization methods [63, 129], Markov Decision Process (MDP) based technique [164]. Recently, deep learning methods, Recurrent Neural Networks (RNN) have emerged as powerful methods of modelling sequential data in SBR [15, 215, 179].

The dwell time modelling of the other online services, like video watching [188], news/non-news page reading [153, 66], and media streaming [175] have more similar characteristics of apps usage, i.e. how long user would stay could be affected by the category of the content or the original length of the video or news. Additionally, the underlying motivation for dwell time prediction of these services is also the same as our work. They consider popularity and engagement as different measures of online behaviour. Although popularity describes the human behaviour of choosing one particular item, it does not describe how users would engage with this item. i.e., popularity is characterized as the willingness to click a video, whereas engagement is the video watching pattern after clicking. By knowing how long user would stay, we could be able to provide users with more satisfying content/services that increase long term user engagement and as a side-benefit. It also allows the service providers to optimize the user experience along with its business goals. Wu et al. [188] conducted a large-scale measurement study of engagement on 5.3 million videos over a two-month period and measured a set of engagement metrics (e.g., watch time, watch percentage) for online videos. They predicted dwell time from video context, topics, and channel reputation, etc. Seki et al. [153] and Homma et al. [66] all clarified the characteristics of relationships between dwell time on news/non-news pages reading in order to discover which features are

effective for predicting the dwell time, including (1) Dwell time by Device: desktops and mobies; (2) Dwell time by access time; (3) Dwell time by if users visited from inside or outside the site; (4) Dwell time by click and non-click: if the user clicked links in the page; (5) Dwell time by scroll depth. Vasiloudis et al. [175] explored the prediction of session length in a mobile-focused music streaming service. They predicted the length of a session using contextual and user-based features including gender, age, subscription status, device, network type, duration of the user's last session, and time elapsed since the last session.

Given the task similarity between these engagement prediction studies and our focus (app engagement prediction), we selected two of them [175, 188] with the most features that could be extracted from our dataset as the baselines for comparing the performance of our proposed model regarding the app engagement prediction. We all have similar engagement characteristics and the motivation for engagement prediction, e.g., the dwell time is originally correlated with the content category and more exploratory or exploitative service could be provided based on the expected length of the usage.

In summary, for the traditional next app prediction task, we selected four models from previous works that could be fitted to our dataset as baselines, include three works [163, 157, 218] based on the temporal pattern and contextual features and one recent work based on neural approach (LSTM) [193]. For the joint prediction problem of predicting the next app and engagement level together, since there is no existing similar joint prediction work (app dwell time has a high dependency on which app user is engaging), we added those two recent works [175, 188] for predicting dwell time as baselines. The only difference is they solely predict how long a user will stay based on the specific item, without predicting on which item the user

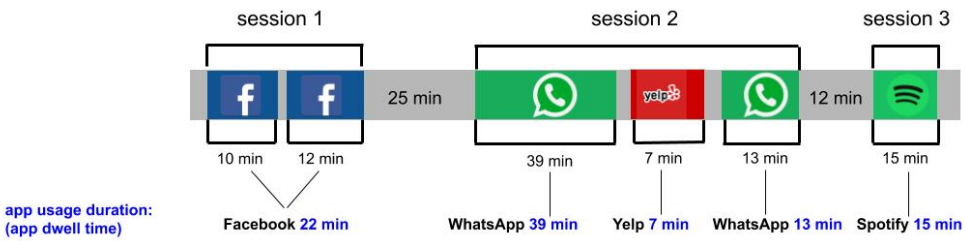


Figure 4.2: Visualisation of app usage logs and the corresponding app usage duration marked in blue. Five minutes of inactivity is used for segmenting mobile sessions.

will engage with. In order to make them comparable baselines to our work, we assumed the ground truth of the next app is known for those two prior works [188, 175] and leveraged them specifically for predicting engagement of the known next app. This demonstrates the upper bound of those approaches (oracle performance) regarding the joint prediction problem.

4.2 Data Context

Our study is based on the Flurry dataset. Besides the general statistics introduced in Section 3.2, in this section, we present more fundamental analysis that demonstrates the characteristics of the data for conducting our study.

Following previous work [26], we consider a five-minute range of inactivity as the signal of ending a session as shown in Figure 4.2. If the user leaves an app but revisits any app within 5 minutes the *session* continues; otherwise, the *session* ends. In our work, we aim to predict which app the user will use next and how long the user will stay with that app. The app usage duration we aim to predict is calculated as follows (as shown in Figure 4.2): we aggregate the consecutive app usage duration of the same app within

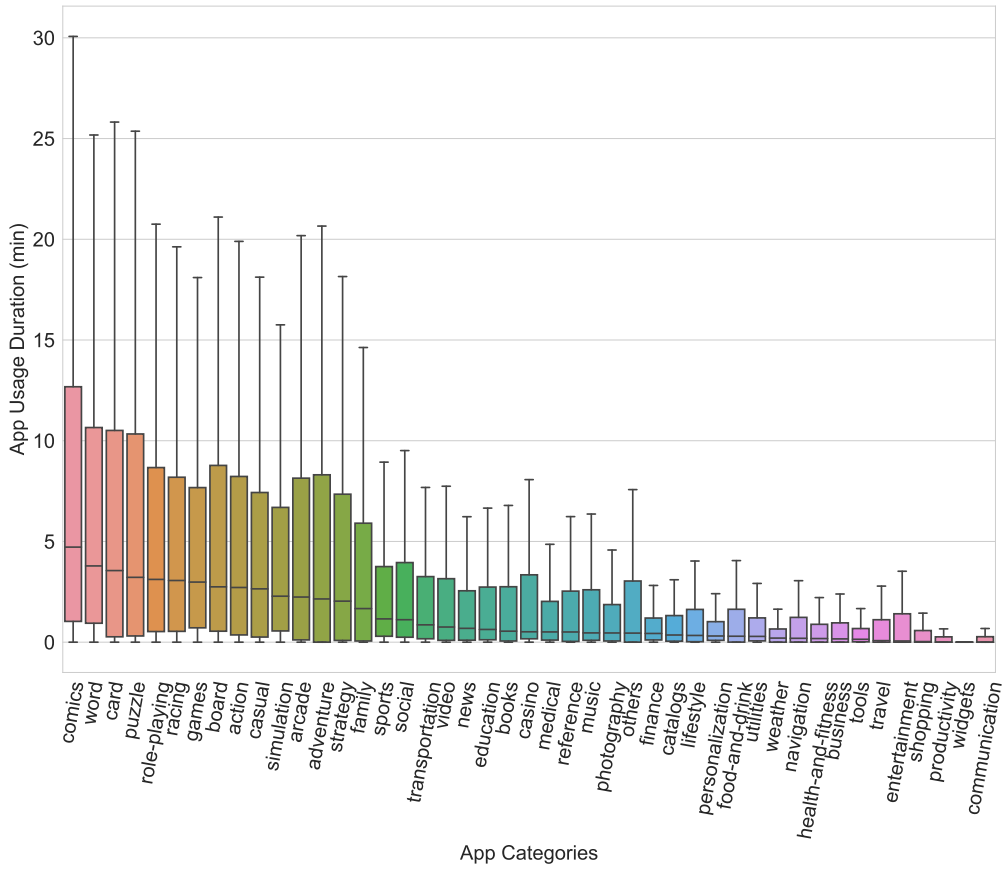
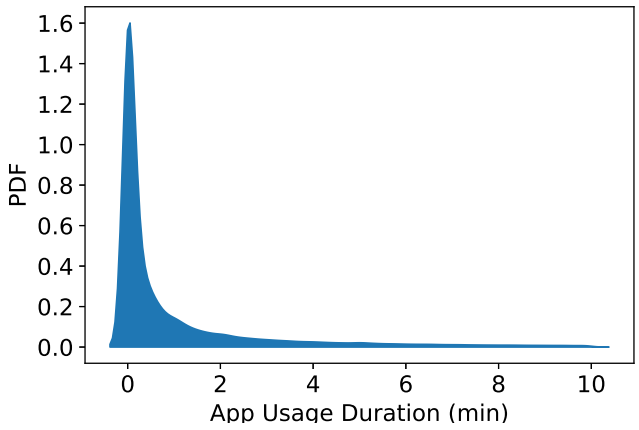
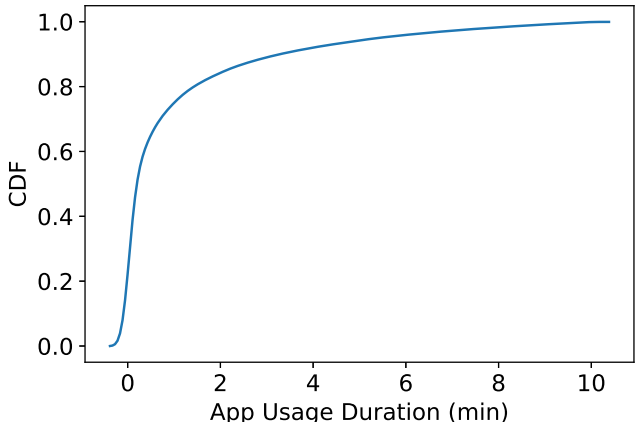


Figure 4.3: Overall average app usage duration for different app categories (different colors shown in the figure is only used to distinguish the different app categories).

each session. The app usage duration can be also referred to as app *dwell time*. We adopt this definition as the unit of our analysis for the rest of the thesis. Note that we only consider user-triggered events; i.e. we do not include events that are triggered by background refresh when conducting our analysis. To reduce bias from users with a low level of engagement, we restricted our sample to those users who interacted with apps from at least five different categories. All the data was anonymized by removing all personally identifiable data prior to processing.



(a) PDF (Probability Density Function)



(b) CDF (Cumulative Distribution Function)

Figure 4.4: PDF and CDF of app usage duration across all apps

4.2.1 Distributions of App Usage Duration

We start by presenting the average app usage duration for different app categories in Figure 4.3, which ranges from less than 1 minute to over 10 minutes. We find that users always spend longer time when they are engaging with some app categories for relaxation, such as comics and games apps (i.e., card, word, puzzle, and board). It also states that different app categories initially have different lengths of app usage duration, e.g., game apps have a much longer duration than communication apps. On average for all the app categories, our users' app dwell time lasted about

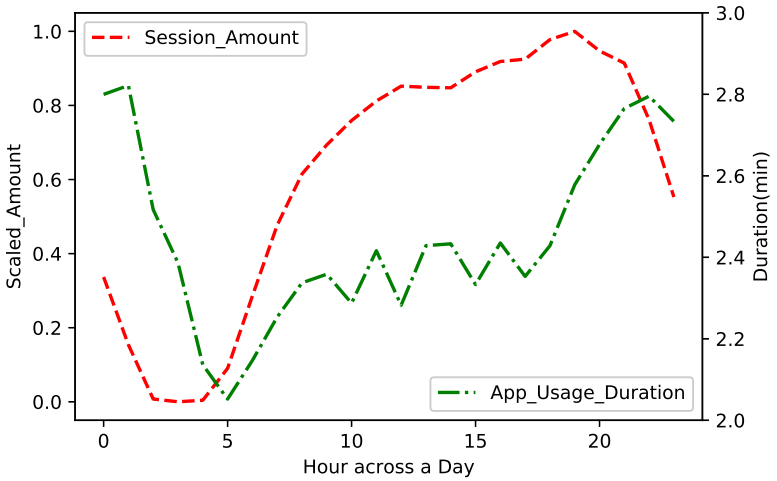


Figure 4.5: App usage patterns across a day.

2.5 minutes. In Figure 4.4, we show the probability density function (PDF) and cumulative distribution function (CDF) of app usage duration. We can find that most of the app usage (93.7%) is less than 10 minutes and 80% of them only last less than 2 minutes.

We then look at how users engage with their apps throughout the day. Figure 4.5 plots the scaled amount of sessions (using min-max normalization) and average app usage duration each hour. We can find that the total app usage (in terms of session amount) is at its maximum in the afternoon and evening, peaking at around 7 PM. This aligns with findings reported in [172, 16]. Figure 4.5 also shows the average app dwell time regarding the different hours of the day. Generally, the average app usage duration across the day is between 2 minutes and 3 minutes. The duration increases after 6 PM and drops after 1 AM. This might be due to people have more leisure time during the non-working hours. We indeed find that the app categories associated with longer duration at late night are game and tool (system cleaner/VPN) apps.

4.2.2 Engagement Level Definition

To evaluate the performance of app usage duration prediction more intuitively, we propose to represent the time length of each duration as a discrete value, i.e., we classify the app usage duration into three engagement levels: light, medium, and intensive. Additionally, as we have found in Figure 4.3 that different app categories initially have different lengths of app usage duration, so it may not be reasonable to label the engagement levels without differentiating app categories. Specifically, to assign the corresponding engagement level of each app usage duration: (1) we first calculate the quantiles (33% and 67%) of the duration for different app categories respectively; (2) then we assign the level label to each duration based on their corresponding app category quantiles. For example, the 33% and 67% quantiles of weather apps are 6 seconds and 23 seconds respectively. If the current usage duration of the weather app is 10 seconds, then its engagement level is medium. For the game apps whose 33% and 67% quantiles are 1.3 minutes and 5.4 minutes respectively. When a game app usage duration is longer than 5.4 minutes, its engagement level is intensive.

4.3 Inferring Users' App Dwell Time

To model the app engagement accurately, we need to uncover what information could be indicative factors for users' app dwell time. This is the main focus of this section.

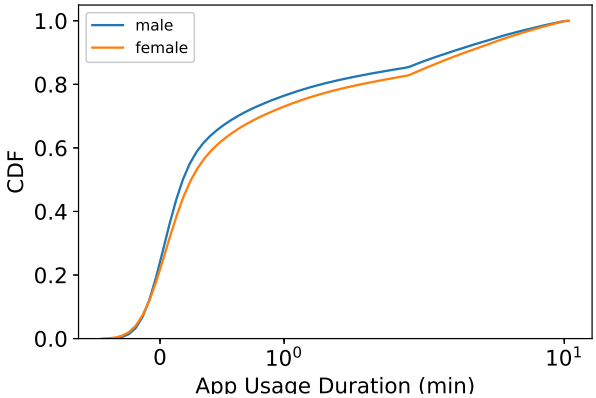
Table 4.1: Top 5 app categories with the biggest gender effects – the higher the gender effect is, the bigger difference exists in the app usage duration between male and female users. The gender group G_1 has longer app usage duration.

App Category	**Gender Group G_1**	**Gender Effect**
widgets	female	3.30
medical	female	2.54
entertainment	female	2.10
communication	female	1.64
shopping	female	1.63
App Category	**Gender Group G_1**	**Gender Effect**
casino	male	1.81
video	male	1.64
health-and-fitness	male	1.60
finance	male	1.30
navigation	male	1.27

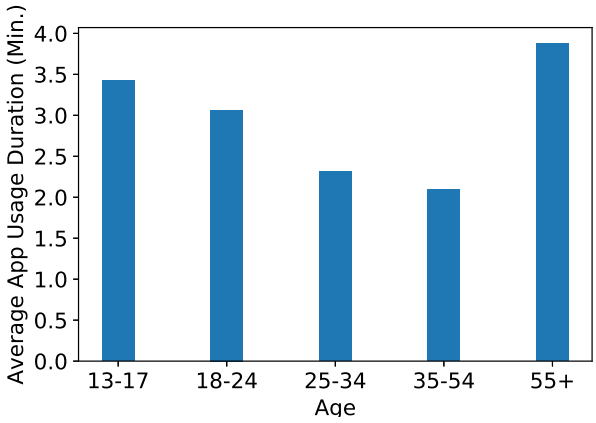
4.3.1 User and Device Characteristics

User Demographics Prior studies [209, 154, 85] have shown that demographics (age and gender) have a significant impact on how users select apps or make in-app purchases on their smartphones. However, little is known about the correlation between users' demographic information and the time spent when using an app. Establishing such a relationship is the main focus of this subsection. By analyzing our user mobile app usage data, we demonstrate that on average, the app usage duration for female users is 2.8 minutes, which is longer than the male users who spent 2.2 minutes. To show the general pattern of usage duration across all apps, Figure 4.6a presents the CDF of app usage duration for both the male and female users. We find that the duration distributions are similar between male and female users.

To investigate the effect of age, we demonstrate in Figure 4.6b that age affects app usage duration more significantly. Users between 25 and 54 years spend less time while engaging with apps than those who are teenagers



(a) Duration vs. Gender



(b) Duration vs. Age

Figure 4.6: Effect of gender and age on app usage duration.

or seniors. This scenario is consistent with results from the previous user studies [46, 141]. Ferreira et al. [46] pointed out that micro-usage (less than 15 seconds) is popular across their participants aged between 22 and 40 years old. Pielot et al. [141] found that their participants (aged between 24 and 43 years old) need to deal with the large volume of notifications coming from personal communication due to high social expectations and the exchange of time-critical information.

Besides these findings of general patterns across all apps, we hypothesize that the impacts of demographics would vary across different app categories. Therefore, we further explore if users' app usage duration with each app category would be affected by their demographics. To measure the impacts brought by users with different genders in a more generalized manner, we calculate the *gender effect* for each app category of the two genders. First, for each app category C_a, we calculate the average app usage duration of female and male users respectively, i.e., $D(C_a, G_i)$ where $G_i \in \{male, female\}$. Secondly we calculate the *gender effect*, $GE(C_a)$, for each app category C_a as:

$$GE(C_a) = \frac{D(C_a, G_1)}{D(C_a, G_2)}$$

where $GE(C_a)$ measures the *gender effect* of the app category C_a. The higher the effect is, the bigger difference exists in the app usage duration of the male and female users within this app category. The gender group in the numerator has a longer dwell time on this app category. Table 4.1 shows the top 5 app categories with the highest gender effects. We can find the female users have a longer app usage duration than male users with apps like widgets, medical, entertainment, communication, and shopping. On the other side, male users stay longer with the casino, video, health-

Table 4.2: Top 5 app categories with the biggest age effects – the higher the age effect is, the bigger difference exists in the app usage duration between users with different age. The age group A_i has longer app usage duration when compared with other users.

App Category	Age Group A_i	Age Effects	App Category	Age Group A_i	Age Effects
shopping	13-17	2.75	widgets	18-24	2.35
entertainment	13-17	1.81	entertainment	18-24	2.20
social	13-17	1.28	business	18-24	1.65
photography	13-17	1.27	tools	18-24	1.41
sports	13-17	1.24	adventure	18-24	1.41
App Category	**Age Group A_i**	**Age Effects**	**App Category**	**Age Group A_i**	**Age Effects**
food-and-drink	25-34	1.59	lifestyle	35-54	1.29
video	25-34	1.30	casino	35-54	1.27
medical	25-34	1.30	word	35-54	1.23
music	25-34	1.29	travel	35-54	1.22
transportation	25-34	1.27	action	35-54	1.21
App Category	**Age Group A_i**	**Age Effects**			
productivity	55+	2.16			
entertainment	55+	2.11			
board	55+	2.00			
puzzle	55+	1.90			
books	55+	1.88			

and-fitness, finance, and navigation apps.

Similarly, we calculate the *age effect* brought by different users on the app dwell time. Since there are five age groups in our dataset, we choose the average app usage duration of all users as the reference value to calculate the *age effect*. Therefore, for each app category C_a, we calculate the average duration of users belonged to different age groups respectively, i.e., $D(C_a, A_i)$ where $A_i \in \{$13-17, 18-24, 25-34, 35-54, 55+$\}$. Then we calculate the *age effect*, $AG(C_a)$, for each app category C_a as:

$$AG(C_a) = \frac{D(C_a, A_i)}{D(C_a, A)}$$

which measures the *age effect* of the app category C_a. A represents the users across all ages who have engaged with the app category C_a. The higher the effect is, the bigger difference exists in the app usage duration of the users with corresponding ages A_i. The age group in the numerator has

a longer dwell time on this app category when compared with other users. Table 4.2 shows the app categories with top *age effects*. It is interesting to find that teenage users have a longer duration in the shopping apps. Users between 25 and 34 prefer to stay longer with the food-and-drink apps than other users. It is not surprising to find that the older users over 55, will spend a much longer time when using the apps of entertainment, games (board/puzzle), and books apps since they have more spare time. For the longer time spent in productivity apps, this may result from that old users are not as efficient in the operations with the productivity apps (e.g., Microsoft Office, file managers).

Device Li et al. [93] ever analyzed the influence brought by different mobile device models on users' online time and they found that users rely less on the cellular network as the price of the device model increases. However, there has been no empirical research conducted for the impacts on dwell time with different apps brought by device characteristics. In this section, we compare app usage duration across different device types. There are mainly two different types of devices in our dataset: tablet and phone. We use the similar methodology of calculating the *gender effect* for each app category to calculate the *device effect* with the two device types. First, for each app category C_a, we calculate the average usage duration of phone users and tablet users respectively, i.e., $D(C_a, T_i)$ where $T_i \in \{phone, tablet\}$. Secondly, we calculate the *device effect*, for each app category C_a as:

$$DTE(C_a) = \frac{D(C_a, T_1)}{D(C_a, T_2)}$$

The higher the *device effect* $DTE(C_a)$ is, the bigger difference exists in the app usage duration of the users with different device types. Table 4.3 shows the app categories with the most significant difference across the different devices. We can find that the tablet users have a longer dwell time on

Table 4.3: Top 5 app categories with the biggest device effects – the higher the device effect is, the bigger difference exists in the app usage duration between the phone and tablet users. The users with device type T_1 have longer app usage duration.

App Category	Device Type T_1	Device Effects
productivity	tablet	4.95
books	tablet	3.29
business	tablet	3.00
board	tablet	2.91
travel	tablet	2.82
App Category	**Device Type T_1**	**Device Effects**
navigation	phone	8.04
shopping	phone	6.02
weather	phone	1.29
family	phone	1.22
personalization	phone	1.19

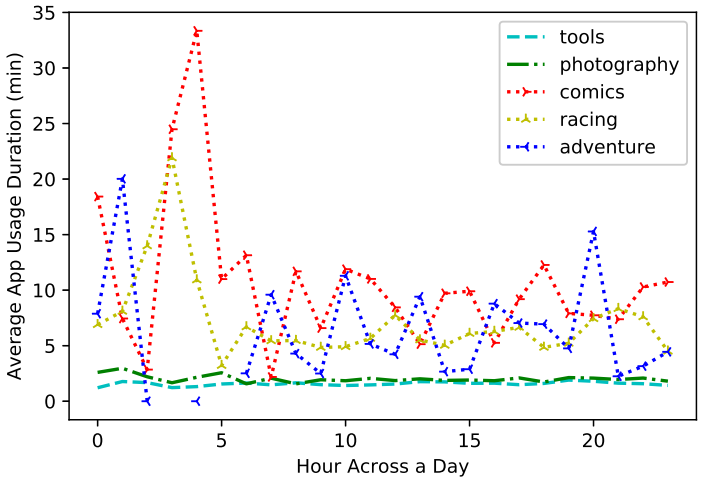


Figure 4.7: Temporal patterns of average app usage duration for different app categories.

productivity, books, business, board, and travel apps. On the other hand, phone users have a longer dwell time with navigation, shopping, weather, family, and personalization apps.

4.3.2 Temporal Patterns

The temporal usage pattern is always an essential factor for inferring which app user will use, where we usually model the possibility of using different apps at a specific time [72, 157, 98]. From our descriptive analysis in Figure 4.5, we can find that the average app usage duration with all app categories across a day is about 2 to 3 minutes, and no sharp fluctuations exist. We use the index of dispersion [35] as a normalized measure of the dispersion for the app usage duration distribution across a day, which is defined as the ratio of the variance σ^2 to the mean μ: $D = \frac{\sigma^2}{\mu}$. In general, the index of dispersion for all apps is 0.043. To validate that whether the temporal pattern varies on different app categories, we further explore the index of dispersion of the app usage duration across a day for different app categories respectively. Table 4.4 shows the app categories with the smallest and biggest index of dispersion in app usage duration. We can find that the usage duration of most game apps will be significantly different across a day. However, for some functional apps like productivity, tools, utilities, and photography, whenever the user accesses them, the app usage duration does not change much. Figure 4.7 illustrates the distribution of average usage duration across a day for several app categories, where we can find that besides the different variances, the temporal patterns could be significantly different with various app categories (each app category has its own specific temporal pattern). For example, comics apps have a longer usage duration around 4 AM; the usage duration of racing apps is peaking around 3 AM; longer time is spent in adventure apps around 8 PM and 1 AM.

Table 4.4: Index of dispersion of average app usage duration distribution across a day.

App Category	Index of Dispersion	App Category	Index of Dispersion
tools	0.02	comics	4.18
productivity	0.02	adventure (game)	3.86
utilities	0.02	strategy (game)	2.51
music	0.04	racing (game)	2.05
photography	0.05	family	1.59
news	0.06	transportation	1.35
widgets	0.07	medical	1.03

4.3.3 Short-term context

4.3.3.1 Last Used App

Previous studies identified that some apps were often used together [94, 191, 16]. For example, Li et al. [94] showed that some apps are installed together, whereas other studies [191, 16] found that some genres of apps are highly likely to be accessed sequentially. Therefore, we aim to explore whether the last used app could also impact the next app usage duration. To avoid the biases of the original differences of app usage duration across different app categories and measure the impacts more intuitively, we quantify the app usage duration into the corresponding engagement level defined as in Sec 4.2.2. Figure 4.8 shows several examples for illustrating the different patterns between specific last apps and the engagement levels of next apps. Each cell in the table represents the transition probability P_{ij} from the last used app a_i to the next app a_j at a given engagement level e_{a_j}. By having P_{ij} for all the three engagement levels, we calculate its standard deviation as σ_j^i. This represents the extent of how much the last app a_i would have very different transition probabilities on different engagement levels of the next app a_j. The higher σ_j^i, the more likely the last app a_i would result in a specific engagement level of the next app a_j. For a given app a_j, we average σ_j^i across all the last used apps a_i as $\sigma_j = \sum_{i=1..N} \sigma_j^i/N$, where N is the

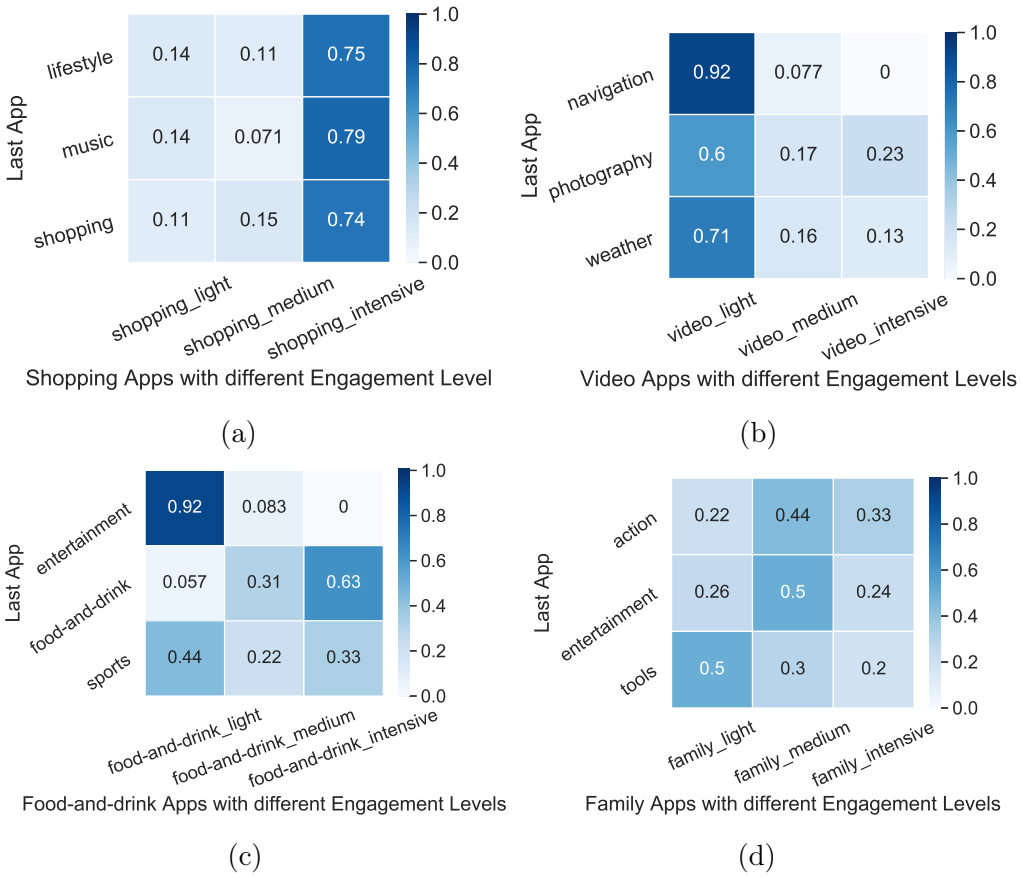


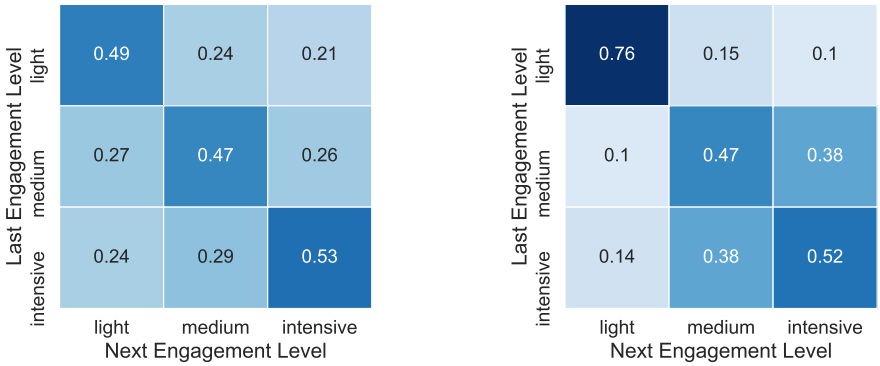
Figure 4.8: Correlation between last used app and the next app's engagement level (discrete representation of app usage duration: light, medium and intensive). The darker color means higher probabilities that the next app will be engaged with the corresponding engagement level.

amount of unique last used apps. For all the apps, the σ_j ranges between 0.11 and 0.32, whereas the median is 0.15. For the apps with higher σ_j, i.e., shopping apps ($\sigma_j \approx 0.32$) and video apps ($\sigma_j \approx 0.29$), the last used app could lead them to be used with specific engagement level (intensive for shopping and light for video). For example, for the video apps (as shown in Figure 4.8b), if they are used after using navigation apps, the time spent on the video app is much more likely to be short. This could be explained by users who are on their commute to work. After checking the navigation, users may need to wait for the bus or subways, so they could have time for enjoying a short video. Additionally, we also show the app categories with average σ_j and lower σ_j, i.e., food-and-drink apps ($\sigma_j \approx 0.16$) and

family apps ($\sigma_j \approx 0.11$), whereas the last used app could not impact the engagement level of next app usage significantly. For example, no matter which app is used before the family apps (as shown in Figure 4.8d), the time spent on the family apps do not differ much.

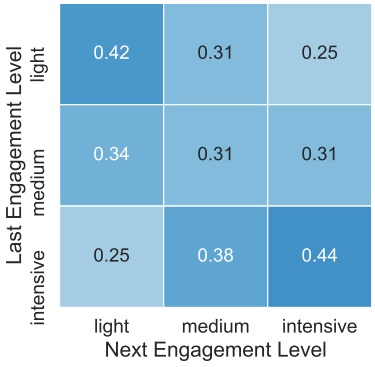
4.3.3.2 Last Engagement Level

Besides the last used app, we hypothesize that the time spent on the last app (last engagement level) could also impact how long the user will stay with this app. To focus solely on the impact of the last engagement level, rather than the impact of last used app, we calculate the transition probability between the engagement levels of the last and next usage *within the same app*. We find four typical patterns across all different app categories. We illustrate these correlation patterns between the last and next engagement levels in Figure 4.9. Across all 45 app categories, 37.8% (17/45) of them follow the pattern of maintaining a higher probability that the next engagement level is as same as the last engagement level (i.e., higher probability in the diagonal of the heatmap), such as the books apps shown in Figure 4.9a. Besides this common pattern, we also find that 44.4% (20/45) of the app categories have a bidirectional (increasing/decreasing) trend with the closest level, like the food-and-drink apps (i.e., engagement level transition between medium and intensive). The last common pattern is illustrated by comics apps, 13.3% (6/45) of app categories have a similar transition probability across all levels. The remaining pattern is illustrated by travel apps, where no other app categories have similar patterns with them. We can find that there is an apparent increasing trend between engagement level light and medium for travel apps, which could state that users may be more likely to get addicted to planning for a vacation within a travel app.

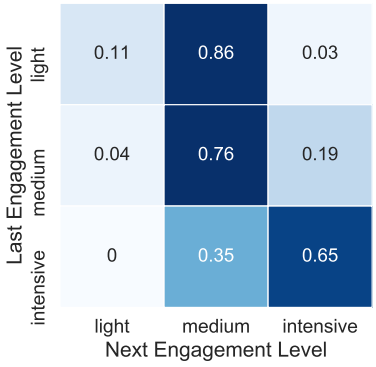


(a) Books Apps

(b) Food-and-drink Apps



(c) Comics Apps



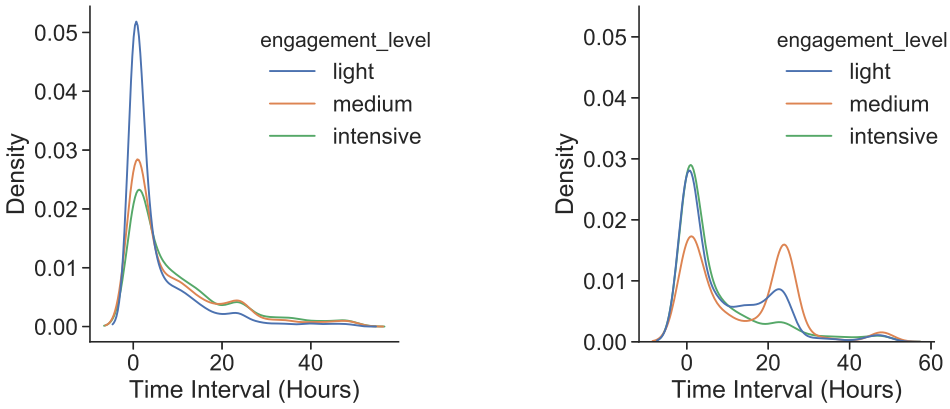
(d) Travel Apps

Figure 4.9: Transition probability of the last engagement level and next engagement level of the same app usage. The darker color means higher probability.

4.3.4 Long-term Context

4.3.4.1 Periodic Pattern

Some apps are used repeatedly after every specific period. For example, users may check the mail apps every hour or play with a game app every 3 hours [98, 163]. To validate whether the periodic pattern also exists in the app dwell time (i.e., after a specific interval, users may stay with an app again for the similar length of time as before that interval), we first quantify the interval between the two accesses of the same app at the hour level, e.g., 28 min $\approx$ 0 hour, 1.6 hours $\approx$ 2 hours. We then generate the histogram of interval time length with different engagement levels respectively for each



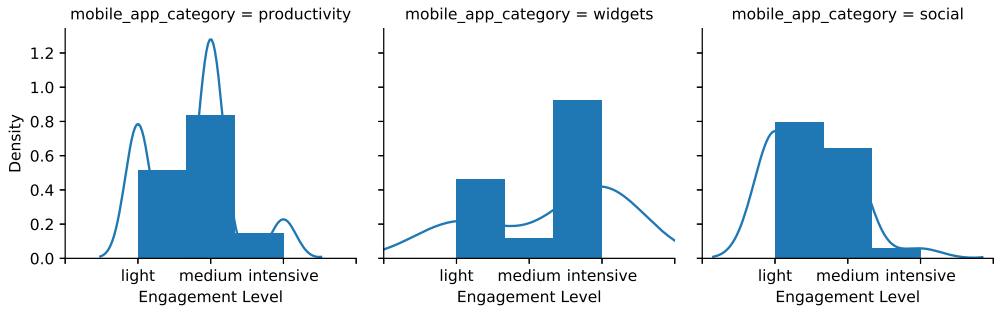
(a) Weather Apps

(b) Shopping Apps

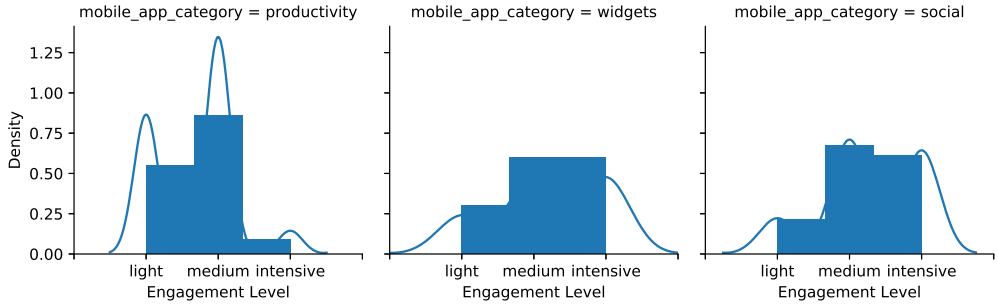
Figure 4.10: Two typical interval time distribution with different engagement levels (we limit the x-axis to 50 hours since over 98% intervals are shorter than 50 hours): (a) General trend illustrated by weather apps: the shorter interval between two accesses, the less time is spent with the next usage; (b) Specific daily periodic pattern illustrated by shopping apps: similar length of time is spent on the same shopping app after a specific interval (i.e., 24-hour).

app category. Two typical patterns are found among all the app categories, which are shown in Figure 4.10. Figure 4.10a denotes the interval time length distribution of weather apps, which represents the general trend that a set of most other app categories will obey, i.e., the shorter interval between two accesses of the same app, the less time (engagement level: light) user will spend within the next usage of this app.

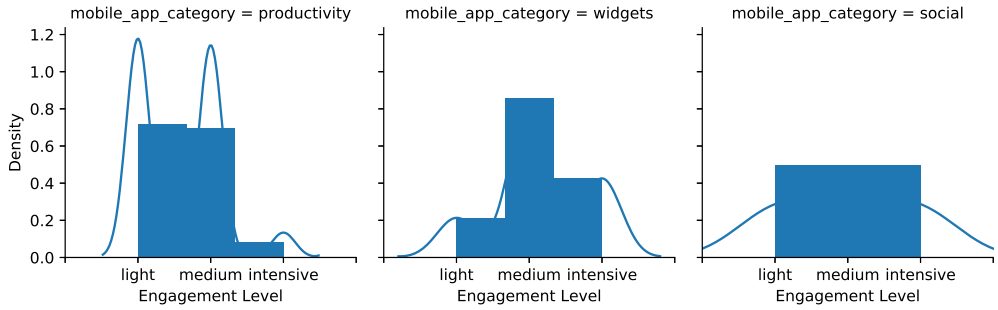
However, as shown in Figure 4.10b (i.e., the shopping apps), the other typical pattern demonstrates that the less time between two accesses, the more likely the second app access will also result in a longer time stay (engagement level: intensive) on the app. This may be because when users are going to buy something, they will browse/revisit the shopping apps multiple times with short breaks (break for checking other information or chat with friends for asking advice) and finally place the order. Additionally, we find that there is a peak around the interval of 24 hours for the engagement level of light/medium for shopping apps. This states that users prefer to



(a) User A



(b) User B



(c) User C

Figure 4.11: Historical pattern with app engagement levels.

spend a similar length of time to regularly check the shopping apps every day (i.e., after every 24 hours), probably for checking the updated discount or product information. This pattern could also be observed with books apps.

4.3.4.2 Historical Interest Pattern

Users' historical interests may be indicative of future intent. For example, users who love sports might potentially access sports apps and consume

more sports-related content than others. Therefore it is important to examine the long-term interest patterns of different users. To clarify that if users also have the historical habit for the app dwell time on different app categories, we select three users whose top three preferred (based on usage frequencies) app categories are all the same, which are productivity, widgets, and social apps. We then illustrate the histogram of their historical engagement levels within these three app categories in Figure 4.11. We can find that *User A* prefers to stay longer (engagement level: intensive) within the widgets apps but spend less time (engagement level: light) with social apps than others. Differently, *User B* has a longer app usage duration (engagement level: intensive) with social apps. Therefore, even for the users with the same historical app preferences, their engagement habits would be different. In a summary, different users may have their own specific historical engagement patterns on app dwell time for various apps.

4.4 App Usage and Engagement Prediction

Based on the statistical analysis in Section 4.3 of app usage duration, we further focus on answering the important question: could we predict which app user will use and how long the user will stay on this app simultaneously? In this section, we start by formulating the next app usage and app dwell time prediction problem, followed by presenting several joint learning strategies for solving these two prediction problems simultaneously.

4.4.1 Problem Formulation

For the next app prediction, many previous researchers have extracted the predictive features and proposed methods to solve it [72, 157, 218, 196].

Similar to Ricardo et al. [8], we model the prediction of the next app as a supervised classification problem. For the novel prediction problem proposed in our work, which is to infer users' app dwell time, we also model it as a multi-class classification problem and the continuous dwell time (app usage duration) is represented as the discrete engagement levels illustrated in Section 4.2.2. More specifically, given the current context c and a user u, we need to predict which app a the user u will use next and the engagement level e (light/medium/intensive) for measuring how long the user will stay with this app. We formulate our prediction problem as quaternion, i.e., $\{c, u, a, e\}$. In contrast to a traditional next app usage prediction formulation as $a^* = \text{argmax}_a \mathcal{F}(a|c, u)$, we formulate the joint learning task for predicting next app and engagement level as follows:

$$(a, e)^* = \underset{a,e}{\text{argmax}} \mathcal{F}((a, e)|c, u). \tag{4.1}$$

Note that the a is in the condition for e since the engagement level is defined based on their corresponding app a (Section 4.2.2).

4.4.2 Joint Learning Prediction Model

We propose three methods to solve the prediction problem formulated in Section 4.4.1: sequential based model (Section 4.4.2.1), stacking based model (Section 4.4.2.2) and boosting based model (Section 4.4.2.3).

4.4.2.1 Sequential based Joint Prediction

The most straightforward method for solving this joint learning problem is to perform these two prediction problems (next app and engagement level prediction) sequentially. It is undoubted that the next app usage

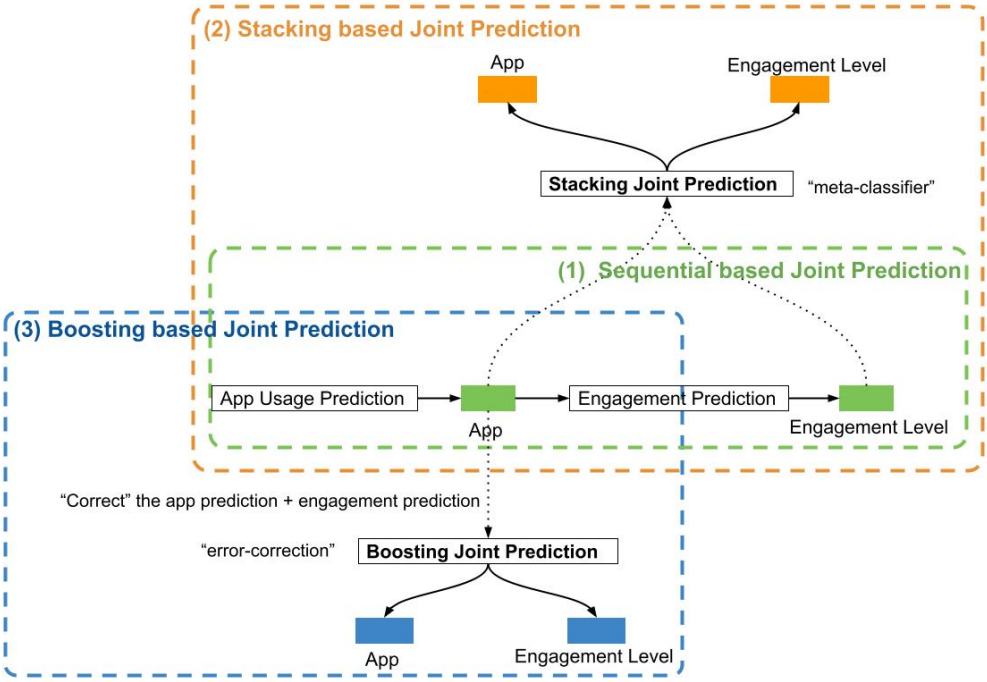


Figure 4.12: Overview of three joint learning strategies: (1) Sequential based joint prediction; (2) Stacking based joint prediction adds a "meta"-classifier to the final stage after we have the prediction results of next app and engagement level sequentially; (3) Boosting based joint prediction has an "error-correction" of next app prediction in the second step for learning of engagement level prediction.

prediction is a key issue to the joint learning task. If a predicted app is not the one user will use next, the engagement level prediction might become meaningless. Therefore, we first apply a supervised next app prediction model to measure which app user will use next:

$$a^* = \underset{a}{\operatorname{argmax}} P(a|c, u). \tag{4.2}$$

Since the engagement level is dependent on which app the user will use next, therefore, another supervised learning model is applied to infer which engagement level the user will have given the predicted app user will use:

$$e^* = \underset{e}{\operatorname{argmax}} P(e|a, c, u). \tag{4.3}$$

Therefore, the sequential based joint prediction could be formulated as follows:

$$(a, e)^* = \underset{a,e}{\operatorname{argmax}} P(a|c, u)P(e|a, c, u). \tag{4.4}$$

As shown in Figure 4.12 (in the green circle), in this sequential-based joint prediction strategy, a next app prediction model is needed to find the proper app a^* user will use and then an engagement level prediction model is needed to infer the engagement level e^* based on a^*. However, the predicted app a^* is not guaranteed to be the right app; moreover, if the app is not the user will use next, the inferred engagement level based on this app would become meaningless. To this end, we propose two other joint learning methods for predicting these two problems more effectively.

4.4.2.2 Stacking based Joint Prediction

Stacking [162] is an ensemble learning technique that combines multiple classification or regression models via a meta-classifier or a meta-regressor. The base-level models are trained based on a complete training set, then the meta-model is trained on the outputs of the base level models as features. So the main idea for our stacking based joint prediction model is to add a "meta"-classifier to the final stage after we got the prediction results on app and engagement level respectively from the sequential based model (Section 4.4.2.1). Then we may improve the performance by adding this "meta-correction" step at the end of the prediction. In our scenario, the stacking consists of two levels which are base learner as level-0 and stacking model leaner as level-1, as shown in Figure 4.12 (in the orange circle). So the base learners (level-0) are composed of the next app usage prediction model (Eq. (4.2)) and engagement level prediction model (Eq. (4.3)), which are the same as in sequential based joint prediction shown in Figure 4.12. The outputs of each of the sequential classifiers (a' and e') are collected to create a new dataset. Then the new dataset is used for stacking model learner (level-1) to provide the final output (a^* and e^*). In this way, the predicted classifications from the two base classifiers at level-0 can be used as input variables into a meta-classifier as a stacking model learner, which will attempt to learn from the data on how to combine the predictions from the base models to achieve the best classification accuracy. The stacking based joint prediction in our scenario could be formulated as:

$$(a, e)' = \underset{a,e}{\operatorname{argmax}} P(a|c, u)P(e|a, c, u), \tag{4.5}$$

$$(a, e)^* = \underset{a,e}{\operatorname{argmax}} P((a, e)|(a, e)'). \tag{4.6}$$

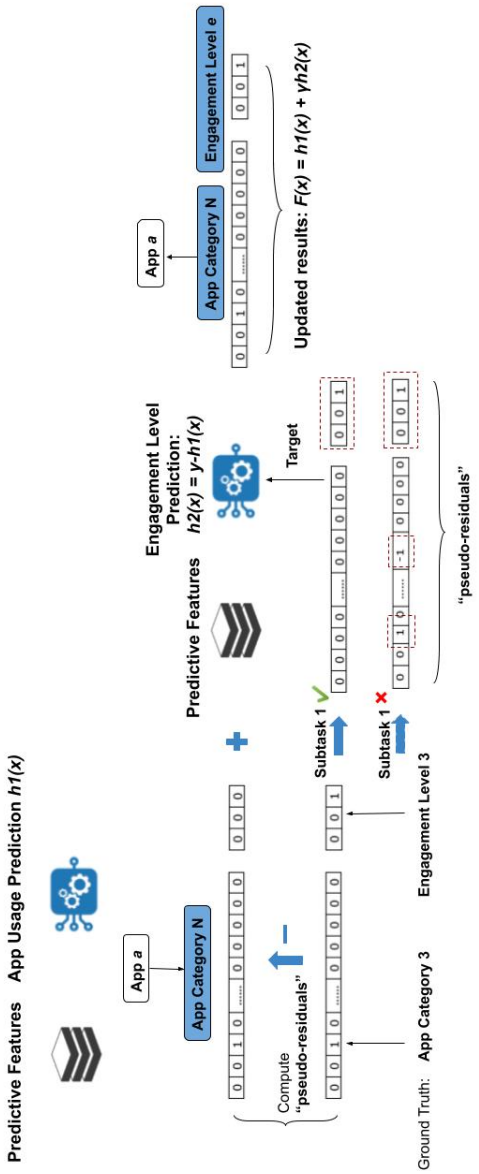


Figure 4.13: Boosting based Joint Prediction

4.4.2.3 Boosting based Joint Prediction Model

Boosting [42] is another ensemble method for improving the prediction model of any given learning algorithms. The idea of boosting is to train weak learners sequentially, each trying to correct its predecessor. According to this idea, we aim to fit our two learners (next app and engagement level) iteratively such that the training of the model at a given step depends on the models fitted at the previous steps. Then we can improve the predictions from our first learner of app prediction by adding the "error-correction"

prediction in the second learner of engagement level prediction. This is shown in Figure 4.12 (in the blue circle).

We firstly introduce how boosting method works in the supervised learning problem. For a given data set with n samples and m features $D = \{(x, y)\}(|D| = n, x \in R^m)$, the goal is to find an approximation $\hat{\mathcal{F}}(x)$ to a function $\mathcal{F}(x)$ that minimizes the expected value of some specified loss function $L(y, \mathcal{F}(x))$. The boosting method assumes a real-valued y and seeks an approximation in the form of a weighted sum of K additive functions (called base/weak learners) $h_k(x)$:

$$\hat{\mathcal{F}}(x) = \sum_{k=1}^{K} \gamma_k h_k(x). \tag{4.7}$$

So the model tries to find an approximation $\mathcal{F}(x)$ that minimizes the loss function L, and the model is updated as follows.

$$\gamma_k = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \mathcal{F}_{k-1}(x_i) + \gamma h_k(x_i)). \tag{4.8}$$

To be specific, in our scenario, the dataset D include features $x = (c, u)$ and label $y = (a, e)$. We have two ($K = 2$) base learners in our prediction, where $h_1(x)$ is the next app usage learner and $h_2(x)$ is the engagement level leaner. The steps for applying the boosting method in our scenario are as shown in Figure 4.13 in details:

1. Fit a model to the data, $h_1(x_i) = P(a_i|c_i, u_i)$ for predicting the next app a_i.

2. Compute the pseudo-residuals between the current predicted app a_i' and the ground truth of finalized results: app with the corresponding engagement level together (a_i, e_i). Since the engagement level is

Algorithm 1: Boosting based Joint Training

SectionetAlgoLined **Input:** Training data
$D = \{x_i = (c_i, u_i), y_i = (a_i, e_i)\}_{i=1}^{n}$.
Output: Boosting based joint classifier
$\mathcal{F}(x) = \mathcal{F}((a, e)|c, u) = \sum_{k=1}^{K} \gamma_k h_k(x)$.
Step 1: *Initialize the model with next app prediction*
Learn $h_1(x_i) = P(a_i|c_i, u_i)$ based on D.
Step 2: *Improve the predictions from next app prediction by adding "error-correction" prediction in the second learner of engagement level prediction.*
1. Compute so-called pseudo-residuals: $r_i = y_i - h_1(x_i)$.
2. Fit the second learner $h_2(x_i)$ to pseudo-residuals. Train it using the training set $\{(x_i, r_i)\}_{i=1}^{n}$.
3. Compute multiplier γ by solving the following one-dimensional optimization problem: $\gamma = \text{argmin} \sum_{i=1}^{n} L(y_i, \mathcal{F}_1(x_i) + \gamma h_2(x_i))$.
Step 3: Update the model:
$\mathcal{F}(x_i) = \mathcal{F}_1(x_i) + \gamma h_2(x_i)$.

defined based on different app categories, additionally, to avoid the sparsity issue caused by using a specific app in comparison, we calculate the "residual" of app prediction results within the app category level. The app category and engagement level are all represented by one-hot vectors and concatenated together during the "residual" calculation as shown in Figure 4.13.

3. Then we fit a model to the residuals, $h_2(x_i) = y_i - h_1(x_i) = r_i$. As the boosting based joint prediction process shown in Figure 4.13, if the first learner output the right predicted app, then the "residual" for the second learner to learn is still only the engagement level ($r_i = e_i$). On the other side, if the first learner output a wrong predicted app, then the residual for the second learner to learn is not only about the engagement level, it also needs to correct the previous results on app prediction.

4. Update the new model $\mathcal{F}(x_i) = \mathcal{F}_1(x_i) + \gamma h_2(x_i)$. Since the "residual" is calculated based on app category level, the finalized output of app

prediction results is also on the category level. We will introduce how to infer the specific app from the predicted app category based on our next app prediction model in Section 4.4.3.

The boosting joint prediction algorithm is shown in Alg. 1. Finally, the boosting based joint prediction could be formulated as:

$$\begin{aligned}(a,e)^* &= \gamma_1 h_1(x) + \gamma_2 h_2(x) \\ &= a' + r \\ &= a' + [a_r, e] \\ &= \underset{a}{\operatorname{argmax}}\, P(a|c,u) + \underset{r}{\operatorname{argmax}}\, P([a_r,e]|a',c,u),\end{aligned} \tag{4.9}$$

where r is the "pseudo-residuals" when comparing the current predicted app with the ground truth (next app and engagement level), and a_r is the difference between the first time predicted app a' and the target app a.

4.4.3 Estimating Components within Joint Prediction Strategies

After introducing the three strategies for building the joint prediction model, we can find that the next app prediction and engagement level prediction are two main components for all three strategies. Furthermore, some of these components could be reusable across different strategies, e.g., the next app prediction of sequential-based strategy could also be used by stacking and boosting based strategies at the first stage of app prediction as shown in Figure 4.12. In the following sections, we discuss how to construct these two components, i.e., the next app prediction (Eq. (4.2)) and

engagement level prediction (Eq. (4.3)), adaptively for each strategy with our proposed predictive features.

4.4.3.1 Next App Usage Prediction

Many previous researchers [218, 72, 8] have proposed different methodologies to solve the next app usage prediction problem based on personalized mechanism. Additionally, some researchers [41, 207] indicated that the generic (user-independent) model can improve the predictive performance of personalized models when the data is not sufficient. A generic model is trained using data from all available users. Inspired by the work of [41], which achieved the best performance by combining the generic model with the personalized model together, we also propose a hybrid next app prediction model with generic and personalized models combined. When building a generic model, the main challenge lies in the fact that the dimensionality of context and output varies depending on all of the users. To learn a generic model and apply it for a given user, generic features and output are needed. In our work, the engagement level is defined based on different app categories. Additionally, the "residual" calculating in boosting based joint prediction strategy (Figure. 4.13) is also based on app category. Therefore, to ensure the generalization of the generic model and to benefit the further prediction of engagement level, the output of the generic model corresponds to the app category. We infer the specific app user will use based on the user's personalized logs given the predicted app category. Specifically, our proposed hybrid next app prediction model could be formulated as the

following function:

$$\begin{aligned} a^* &= \underset{a}{\operatorname{argmax}} P(a|c,u) \\ &= \underset{a}{\operatorname{argmax}} \{P_g(a_c|c,u)P_p(a|c,u), a \in a_c\}, \end{aligned} \tag{4.10}$$

where $P_g(a_c|c,u)$ is the probability that user u will use app category a_c based on our generic app category prediction model; and $P_p(a|c,u)$ is the probability that user u will use the app a based on their own personalized app prediction model and we will limit the a to the apps belonged to the predicted app category a_c. It means we firstly get the app category a_c user will use based on all users' logs and then further rank the apps belonging to this app category based on the personalized logs of user u. Through this way, we can have the intermediate output of the app category, which could be used for benefiting the later engagement level prediction. What is more, this generic category-level prediction model could alleviate the cold-start problems with new apps/users resulted from user-specific prediction models, which may be trained on a limited quantity of logs.

For the generic app category prediction model, we extracted all the features that have correlation with app usage patterns, which have been validated by previous works [41, 207] and also available in our dataset (Section 4.2): User characteristics (age, gender, country, device type, operation system) [209, 85, 93], temporal features (hour of day, day of week) [72, 157, 98], historical preferences [41], last one/two apps used [218] and periodic features [163, 98, 100]. To enrich our predictive features, we further expand users' characteristics by adding their total app usage duration, total app usage frequency, and unique app amount as user characteristic features. To ensure the generalization of the generic model, users' historical preferences are represented based on the total access frequency of each app

category. Moreover, we also add users' app preferences on the last day, the last hour and the last session to expand the context features. In summary, we extracted 14 features related to app usage patterns from 4 aspects: user characteristics, temporal features, short-term context features, and long-term context features. All the features used for next app prediction are described in Table 4.5. Then the personalized next app prediction model is generated based on the predictive features proposed in previous works [157, 218, 8]: hour of day, day of weekday, most recently used apps (the last one/two apps), and periodic feature.

Table 4.5: All the features used in our next app prediction model related to users characteristics and context

Feature Type	**Feature**	**Description**
User	Age*	Users' age group: 13-17, 18-24, 25-34, 35-54 and 55+
	Gender*	Users' gender: male and female
	Device Type*	Users' device type: phone and tablet
	Total App Usage Duration*	Users' total app usage duration for all apps
	Total App Usage Frequency*	Users' total access frequency for all apps
	Total Unique App Amount*	The amount of unique apps the user has accessed
Context	Temporal Features	
	Hour of the Day*	Different hours of a day: 0 - 23
	Day of the Week*	Different days of a week: Monday to Sunday
	Short-term Context Features	
	App Preference in the Last Day	Access frequency of each app category in the last day
	App Preference in the Last Hour	Access frequency of each app category in the last hour
	App Preference in the Last Session	Access frequency of each app category in the last session
	Last Used App*	The last used app in the same session
	Last Used Two Apps	The last used two apps in the same session
	Long-term context Features	
	Periodic Feature	Time intervals between the current time and the last usage of each app category
	Historical App Preference	Total access frequency for each app category

*: The features marked with * are the common predictive features also used in app engagement level prediction.

4.4.3.2 App Engagement Level Prediction

In this subsection, we further explore the app engagement level prediction models that could fit in the different joint prediction strategies. As we mentioned before, we model the novel prediction problem of app engagement level as a multi-class classification problem, where the engagement

level is defined based on different app categories. In the sequential based joint prediction strategy (Section 4.4.2.1) and the level-0 classifiers of the stacking based joint prediction strategy (Section 4.4.2.2), the engagement level prediction model could be trained for each app category respectively (Figure 4.12). For example, if we have predicted that the user will use the news app a^* next, we just select the classifier which has been trained specifically based on the logs of using news apps. Then we use this classifier to predict the engagement level. So the engagement level prediction model for sequential-based and stacking-based joint learning strategies could be formulated as:

$$\begin{aligned} e^* &= \underset{e}{\operatorname{argmax}}\, P(e|a, c, u) \\ &= \underset{e}{\operatorname{argmax}}\{P_{a_c}(e|c, u), a^* \in a_c\}, \end{aligned} \tag{4.11}$$

where a_c is the corresponding app category of the predicted app a^* based on the next app prediction model. It states that we select the engagement level classifier exactly based on the prediction results coming from the next app prediction results. For each app category, we extract the predictive features based on the analysis in Section 4.3: demographic features, device features, hour of day, day of week, last used app, last engagement level, last engagement level of the same app category, periodic feature and historical engagement level feature. Besides the common user characteristic features and some of the context features that have been listed in Table 4.5, we describe the additional features for the app engagement level prediction model in Table 4.6. These features are all established to have impacts on users' app dwell time in Section 4.3, and we will further analyze the feature importance in the following experimental results section Section 4.5.4.1.

Different from the engagement level prediction model $P(e|a, c, u)$ in sequential (Eq. (4.4)) and stacking (Eq. (4.5)) based strategies, the boosting

strategy need to infer the "residual" with $P(r|a, c, u)$ (Eq. (4.11)) and then sum it to the first app prediction result for getting the finalized app and engagement level together. During this process, the next app user will use would be re-inferred during the engagement level prediction, we cannot select a specific classifier based on the first predicted app. Therefore the engagement level prediction model of the boosting-based strategy can only be constructed as a generic model which could be applied to all apps instead of a specific app category. Specifically, the first predicted app a' from the next app prediction model would be treated as an input feature in the following engagement level prediction, which is represented as a one-hot vector as shown in Figure 4.13. Additionally, to ensure the generalization of the input in the engagement level prediction model within boosting-based joint prediction, all the predictive features will also need to be expanded for representing the behaviour pattern coming from all the different app categories (e.g., the feature of historical engagement level for predicted app category need to be extended to historical engagement level for all different app categories). In this model, no matter which app category is predicted to be used next, the same engagement level prediction model will be applied. Since the finalized app and engagement level would be inferred together within the boosting strategy, the formulation of engagement level prediction could not be decomposed, which has been shown in Section 4.4.2.3.

4.5 Experimental Results

In this section, we measure the performance of our proposed prediction models (Section 4.4). Experiments are conducted on a real-world app usage log data (Section 4.2). We use 70% of the data for each user as training

Table 4.6: Additional features used in our app engagement level prediction models related to user characteristics and context

Feature Type	Feature	Description
Context	Short-term Context Features	
	Last Engagement Level	The last engagement level of all app categories
	Last Engagement Level of Predicted App Category	The last engagement level of the usage on predicted app category
	Long-term Context Features	
	Periodic Feature	Time intervals since the last use of all app categories
	Periodic Feature of Predicted App Category	Time intervals since the last use of predicted app category
	Historical Engagement Levels	Historical sum of engagement levels for all app categories
	Historical Engagement Levels of Predicted App Category	Historical count of each engagement level for predicted app categories

data and the remaining 30% as test data. We first evaluate the performance of our proposed prediction model on the classic prediction problem, predicting the next app (Section 4.4.3.1). Then the three joint learning strategies for predicting the next app and engagement level simultaneously (Section 4.4.2) are thoroughly evaluated from different perspectives.

4.5.1 Evaluation

In our proposed prediction models, we measure the performance of all prediction problems based on four metrics: accuracy, precision, recall and F1 score: The accuracy in our problem is defined as the fraction of correctly classified samples; The precision is defined as the ratio:

$$\frac{tp}{(tp + fp)}, \tag{4.12}$$

where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative. The recall is the ratio:

$$\frac{tp}{tp + fn}, \tag{4.13}$$

where fn is the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples; The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score is equal. The formula for the F1 score is:

$$F1 = 2 * \frac{precision * accuracy}{precision + accuracy}. \tag{4.14}$$

Please note that in our multi-class case, all the precision, recall, and F1 scores are the weighted averages of scores for each class. In order to account for label imbalance, metrics are calculated for each label, and their averages are weighted by support (the number of true instances for each label).

For the next app prediction results, we will count the item as correctly predicted when the predicted app is exactly the same as the ground truth app. While evaluating the joint prediction problem (both next app and engagement level), a correctly classified sample indicates that both the predicted app and predicted engagement level are correct. If either of the prediction is wrong, it will not be counted as a correct classification.

4.5.2 Baselines

4.5.2.1 Next App Usage Prediction

In order to comprehensively measure the performance of our proposed hybrid next app prediction model (Section 4.4.3.1), we compare it with state-of-the-art counterparts. Based on the available sources of evidence in our dataset, we first select the two common baseline methodologies: MFU

(Most Frequently Used) and MRU (Most Recently Used) [157, 194, 218]. We then tested additional three methodologies from previous works which conducted the next app prediction by combining app usage history and contexts in a unified manner: SVM+Context [157], CPD [163] and BN [218]. Recently, the neural approaches are popular in solving the time sequence problems, we also investigated the performance of app usage prediction based on LSTM [193] as the baseline:

- *MFU*: the predicted app is the most frequently used app.
- *MRU*: the predicted app is the most recently used app.
- *SVM+Context*: Shin et al. [157] used a SVM classifier [13] and context information (i.e., day of week, hour of day, last used app and time since last app usage) to predict the next app user will use.
- *CPD*: Tan, et al. [163] proposed a prediction framework: Prediction Algorithm with Fixed Cycle Length (PAFCL). They hypothesized that each application has different usage probabilities in the different time slots of a cycle. CPD (Cumulative Probability Distribution) is the method used to choose applications with higher probability as the candidates based on computing the probabilities of each used app in the specific time slot.
- *BN*: Zou et al. [218] proposed a Bayes Network (BN) model which is a linear combination of $p(a_n = A|a_{n-1} = A_{n-1})$ (based on last used app A_{n-1}) and $p(a_n = A|a_{n-2} = A_{n-2})$ (second last used app A_{n-2}).
- *LSTM*: Xu et al. [193] proposed a generic prediction model based on Long Short-term Memory (LSTM), which is an enhancement of the recurrent neural network (RNN) model. The proposed model con-

verts the temporal-sequence dependency and contextual information into a unified feature representation for the next app prediction.

4.5.2.2 Next App and Engagement Level Joint Prediction

Since we are the first work proposed to predict which app user will use and how long the user will stay on that app simultaneously, where the time spent has a high dependency with which app user is engaging, no previous related work could be identified as our baseline methodologies. Therefore, we propose the baselines from two kinds of approaches. Firstly, by following the classic baselines (MFU and MRU) in the next app prediction problem, we present two naive prediction methods as the baselines for our proposed joint prediction problem. Secondly, two recent works on dwell time prediction of online services (e.g., video [188] and media streaming [175]) are selected as baselines. Similar to our work, how long a user stays with an item in these services originally have associations with the content category, and could also be affected by user characteristics and contextual features. The only difference is that they solely predict how long a user will stay based on the specific item, whereas no prediction on which item the user will engage is required. In order to make them comparable baselines, we assumed the ground truth of the next app is known and leveraged those two prior works [175, 188] for predicting engagement (dwell time) of the oracle app. This demonstrates the upper bound of those approaches (oracle performance) within the joint prediction task.

- *MFU*: In our joint prediction problem, we discuss the usage frequency of the two fields together: the next app and engagement level. Then the MFU baseline states that every time we recommend the tuple (a, e) as the app a and engagement level e based on the popularity of

the tuples.

- *MRU*: Similar to the MRU in next app prediction, we hypothesize that a correlation between two sequential tuples of app and engagement level may exist. For example, a user may always like to access the weather app for 10 seconds and then access the news app for 5 minutes. Therefore we generate a correlation-based baseline approach for our joint prediction problem, which aims to predict the next app and engagement level based on the correlation between two sequential tuples (a', e') and (a, e). We first calculate all the transition probabilities from one tuple to another. When we know the last usage tuple is (a', e'), we could recommend the tuple (a, e) that has the highest probability to be used next.

- *CSP*: Wu et al. [188] conducted a large-scale measurement study of engagement on videos. They predicted engagement from video context, topics, and channel reputation, etc. They used linear regression with L2-regularization to predict engagement metrics and state that the Channel Specific Predictor (CSP) performs best, which is to train a separate predictor for each channel instead of using the shared predictor. To fit into our scenario, we generate the same CSP for each app category with their proposed features available in our dataset (e.g., the one-hot encoding of category, mean number of daily usage, mean, std and five points summary of past engagement (dwell time), etc.). The performance of the joint prediction problem is reported based on the oracle app and the predicted dwell time (which would be transformed to engagement levels according to our definition in Sec. 4.2.2.)

- *Aggregated*: Vasiloudis et al. [175] presented the first analysis of session length in a mobile-focused online service (i.e. music streaming

service). They showed that the time length of sessions can differ significantly between users. They used gradient boosted trees with appropriate objectives to predict the length of a session using contextual and user-based features. Their experiment results showed that the aggregated model trained with all the data performed better than the personalized models trained on each user's data. To fit into our prediction problem, we also trained the aggregated model based on all data with their proposed features available in our dataset (e.g., gender, age, device, duration of the user's last session and time elapsed since the last session, etc.). As we mentioned before, the performance of the joint prediction problem is reported based on the oracle app and the predicted dwell time (which would be transformed to engagement levels according to our definition in Sec. 4.2.2).

4.5.3 Hybrid Next App Prediction Model

Our analysis starts with the hybrid next app prediction results given it is the unified component leveraged by all the three different joint learning strategies. It is used at the first step for inferring the next app user will use before predicting its engagement level (Figure 4.12). Table 4.7 shows the performance of baseline methodologies and our proposed hybrid next app prediction model. We test our model with a set of state-of-the-art classification models, including Random Forests [20], L2-regularized Logistic Regression [67], K Nearest Neighbours [33] and Support Vector Machines [21]. These models construct different prediction functions for the data from different aspects, and they can provide more robust results for our prediction. Since our proposed hybrid next app prediction model (Section 4.4.3.1) involves the prediction results of two predictive components

Table 4.7: Performance comparison of next app prediction models (∗ indicates statistically significant (p≤0.01) using two-tailed T-test when compared our hybrid next app prediction model to the best baseline model (LSTM). Bold scores state the best performance in different measurements.

Method	**Measurement**			
	Accuracy	**Precision**	**Recall**	**F1**
MFU	0.486	0.489	0.486	0.486
MRU	0.518	0.521	0.518	0.518
CPD [163]	0.424	0.419	0.424	0.369
BN [218]	0.453	0.483	0.453	0.467
SVM+Context [157]	0.606	0.563	0.606	0.572
LSTM [193]	0.525	0.660	0.525	0.576
Our Hybrid Model	**0.640***	**0.607***	**0.640***	**0.613***

(Eq. (4.13)), we select the best classifiers for both of them within the hybrid model, which combines the results of generic app category prediction with Random Forest classifier and the results of personalized next app prediction with SVM classifier.

From Table 4.7, we can find that our proposed hybrid next app prediction model could significantly improve the performance of the best baseline model (LSTM) by 6.4% on the F1 measure. The worse performance of LSTM is expected since it could not incorporate any user characteristics and contexts (e.g., access time, device type, etc.) during the prediction. While our proposed hybrid next app prediction model not only takes the user characteristics and contextual information into consideration but also learn from users' common patterns to overcome important sources of prediction errors resulting from insufficient training data. It also states that through the use of community similarity and common usage patterns learned based on different app categories, we can improve the next app usage prediction by identifying the generic usage patterns present in similar users - rather than relying solely on the specific app usage patterns. This is also consistent with the findings from previous studies [41, 207] which claim that the generic model can improve the predictive performance of

models solely based on individual's logs. Therefore, our proposed hybrid next app prediction model is adopted to apply to further engagement level prediction for different joint learning strategies.

4.5.4 Next App Usage and App Engagement Level Joint Prediction Strategies

We first evaluate how different models perform for solving our proposed joint prediction problem: which app user will use next and how long the user will stay with this app? Then we further investigate the predictive ability of features in the two prediction problems respectively and the prediction effectiveness of our proposed different joint prediction strategies.

As mentioned, our hybrid next app prediction model is applied to the first stage prediction on the next app for all the joint learning strategies (Figure 4.12). For the prediction of engagement level, the predicted app will be represented in different ways based on different joint prediction strategies. For the sequential and stacking based joint model, the predicted app will be used to select the specific engagement level classifier with the corresponding app category (Eq. (4.14)). For the boosting based joint model, the predicted app coming from the first stage of next app prediction could only be treated as the input feature for the next step prediction of engagement level (Eq. (4.12)). Table 4.8 shows the ultimate performance of different joint prediction strategies and all baselines.

We can find that all our proposed joint prediction models are better than the two classic baselines: MFU and MRU. For another two baselines, we can observe that even we assumed the ground truth of the next app is known, the upper bound performance of these approaches could not beat

Table 4.8: Performance comparison of joint learning prediction models (∗ indicates statistically significant (p≤0.01) using two-tailed T-test compared to the best baseline (CSP)). Bold scores state the best performance in different measurements.

Model	Measurement			
	Accuracy	**Precision**	**Recall**	**F1**
MFU	0.286	0.286	0.286	0.286
MRU	0.308	0.307	0.308	0.308
Aggregated [175] ▷	0.347	0.742	0.347	0.448
CSP [188] ▷	0.339	0.729	0.339	0.467
Sequential	0.375*	0.382*	0.375*	0.369*
Stacking	0.374*	0.381*	0.374*	0.365*
Boosting	**0.485***	**0.483***	**0.485***	**0.483***

▷: We assumed the ground truth of the next app is known and leveraged these baselines for predicting engagement (dwell time) of the oracle app. The performance reported in table demonstrates the upper bound of those approaches (oracle performance) regarding the joint prediction problem.

our proposed best joint prediction model based on boosting strategy. To be specific, it states that we assume the predicted app is exactly the same as the ground truth and only evaluate the engagement level prediction performance, these baselines have performed worse than our boosting-based joint prediction model. It mostly because they didn't model the engagement with comprehensive characteristics as our proposed model, where we take the user characteristics, short/long-term usage patterns all into consideration. Therefore, if the next app prediction task is added, the worse performance of the joint prediction problem should also be expected for these baselines. Among all the three proposed joint prediction strategies, stacking and boosting based strategies are two advanced approaches originally proposed to improve the performance with the most straightforward strategy, sequential based strategy. However, we find that the stacking based strategy does not improve the performance when compared with the benchmark sequential based strategy. This might be due to that, in our scenario, the base learners within stacking are not trained for the same target (we have two different prediction tasks: app and engagement level)

where the stacking principles do not apply. On the other hand, the boosting strategy works best compared to all other joint models. It respectively outperforms the baseline models over 56% and the sequential/stacking models about 31% on F1 measure. We mainly focus on investigating the boosting and sequential strategies in the later sections about how boosting helps in the joint prediction. Firstly, besides the overall performance reported in Table 4.8, we look into the detailed prediction results. It demonstrates that the accuracy of next app prediction and engagement level prediction respectively are: 64% (app) and 58.6% (engagement level) for sequential strategy, 85.6% (app) and 56.7% (engagement level) for boosting strategy, where the accuracy of engagement level is calculated only based on the data with right predicted apps. So we can find that the improvement of overall performance for boosting strategy is mainly because of the "error-correction" step which corrects the app prediction results.

In the following sections, we will first analyze the feature importance within the two prediction problems respectively, especially focus on exploring the predictive ability of the newly proposed features. Then we will dig into how is the effectiveness of boosting based strategy compared to the sequential based strategy.

4.5.4.1 Feature Analysis

The sequential based strategy is implemented by conducting the app prediction and the engagement level prediction sequentially, where the app prediction is absolutely independent with the further engagement level prediction. Therefore, we discuss the analysis of the most impactful features for those two tasks respectively. The next app prediction within the sequential strategy is implemented as same as our proposed hybrid next app

Table 4.9: Feature importance (MDI) of app category prediction.

Feature	**Feature Type**	**Importance (MDI)**
Total_App_Usage_Frequency	User	0.023
Hour	Temporal	0.011
Periodic_Feature	Long-term	0.011
Total_App_Usage_Duration	User	0.010
Historical_App_Preference	Long-term	0.005
Total_Unique_App_Amount	User	0.005
Age	User	0.004
App_Preference_Last_Day	Short-term	0.003
Weekday	Temporal	0.002
Gender	User	0.002
App_Preference_Last_Hour	Short-term	0.002
Device_Type	User	0.001
App_Preference_Last_Session	Short-term	0.001

prediction model, which would infer what app category the user will use and then select the specific app given this predicted app category. To make the comparison more intuitively, we conduct the analysis within the app category level. As we mentioned above, the random forest is selected as the best classifier for the app category prediction problem. The random forest can be used to rank features by their importance in the classifier, which provides useful insights about the discriminative power of the features in the considered problem setting. *Mean Impurity Decrease (MDI)* is the most common way to obtain feature importance from random trees [20]. It is computed by averaging across all the trees in the forest the amount of impurity removed by each feature while traversing down the tree, weighted by the proportion of samples that reached that node during training. Using this method, we obtained the feature importance for all features in the app category prediction, as listed in Table 4.9.

We find that besides the critical temporal feature *hour*, the most important categories of features are mostly user and long-term context features. Compared to user characteristics features such as demographics and devices, the total usage (frequency, duration, and unique app amount) maintain higher impacts on the next app category prediction. The hour of day

Table 4.10: Top feature weights (standardized coefficients ≥ 0.001) for logistic regression model of engagement level prediction. $*$ indicates p-value ≤ 0.001 using Chi-Squared test.

Feature	**Feature Type**	**Weight**
Historical_Level_Light*	Long-term	0.571
Historical_Level_Medium*	Long-term	0.197
Total_App_Usage_Duration*	User	0.175
Historical_Level_Intensive*	Long-term	0.112
Periodic_Feature*	Long-term	0.070
Age*	User	0.032
Last_Used_App*	Short-term	0.019
Hour*	Temporal	0.008
Weekday	Temporal	0.002
Last_Engagement_Level*	Short-term	0.001

feature has been used to identify the salient pattern within app usage behaviour in many previous works [72, 157, 98], e.g. the user usually set an alarm at around 23:00. The popularity of app usage (historical app preference) is also a famous feature that has been established by previous works [100, 41]. Additionally, our finding of the periodic pattern is consistent with the previous works on next app prediction, where the periodic patterns have been identified as the effective feature for inferring the app usage pattern [163, 98, 100]. Liao et al. [100] also stated that the periodical usage feature is the most difficult one to be substituted by other features.

The engagement level prediction is the novel problem proposed in our work and we extracted many different predictive features from user&device characteristics, temporal pattern, and short/long-term context (Section 4.3) to infer how long the user will stay with an app. Similarly, since the engagement level prediction within the sequential strategy can be studied separately, we discuss the impacts brought by different features for engagement level prediction within the sequential based strategy. We opt to build a classifier for predicting users' engagement with three levels: *light*, *Medium* and *Intensive*. Hence, the problem of modelling user engagement turns into a multi-class classification problem. Similar to the next app prediction, we

test and empirically compare the performance of a wide range of classification techniques, including Random Forests (RF), L2-regularized Logistic Regression (LR), K Nearest Neighbours (KNN), and Support Vector Machines (SVM), for predicting the app engagement level, where the Logistic Regression classifier performs best.

Then we examine the contribution of each feature based on the feature coefficients in the logistic regression model. To compare the importance of different features, we divide each numeric variable by two times its standard deviation [53]. Through this way, the resulting coefficients are directly comparable for both binary variables (e.g., categorical dummy variable) and numerical features. Table 4.10 reports the top feature weights (standardized coefficients ≥ 0.001) of the Logistic Regression model for the engagement level prediction. We can find that most of the top features are originated from the long-term context features: historical engagement level preference and periodic pattern. It is not surprising that the historical pattern has more influence on how long a user will stay with an app. If the user always prefers to play games for a longer time, then he may still spend more time in the game app this time. The periodic feature has more impacts on users' engagement level than the temporal context, hour, and weekday. This demonstrates that no matter when the user uses this app, the time since the last use of this app is more important for inferring how long the user will engage with this app. Additionally, the last used app has more impacts on predicting the engagement level compared to all short-term context features. This could provide more insights for the app developers to recommend the contents of different time lengths according to the last used app to improve the user experience. Another finding is that we observe age is the most important signal among all demographics and device characteristics when inferring the app usage duration.

4.5.4.2 How Effective are Boosting-based vs. Sequential-based Strategies?

We have shown the boosting based strategy outperforms the benchmark sequential based strategy by the performance margin of 31% on the F1 measure. Therefore we further conduct some error analysis to understand the underlying reasons.

Firstly, for the novel prediction problem on engagement level, the sequential and boosting based strategies achieve similar performance, which are 58.6% and 56.7% respectively on accuracy. It demonstrates that even the boosting based strategy gets better overall performance in our joint prediction problem, it cannot improve the engagement level prediction results. We further analyze the prediction ability on app engagement level prediction with our proposed sequential strategy. We select only the engagement level prediction results when the next app is correctly predicted. Figure 4.14 shows the confusion matrix of the app engagement level prediction results. For the wrongly predicted results of all engagement levels, we can find that they have higher probabilities to be predicted into the adjacent level. For example, for the intensive engagement level, 19% of them are misclassified as a medium level, which covers about 95% of the wrongly predicted results. Similarly, for the light engagement level, 25% of them are misclassified as a medium level, which is higher than those that are misclassified as intensive. The engagement level prediction results of boosting based strategy also get a similar confusion matrix as shown in Figure 4.14. We now focus on exploring how boosting is more effective in the next app prediction as follows.

For the next app prediction, we have reported that the boosting based strategy improves the accuracy from 64% to 85.6%, which is resulted from

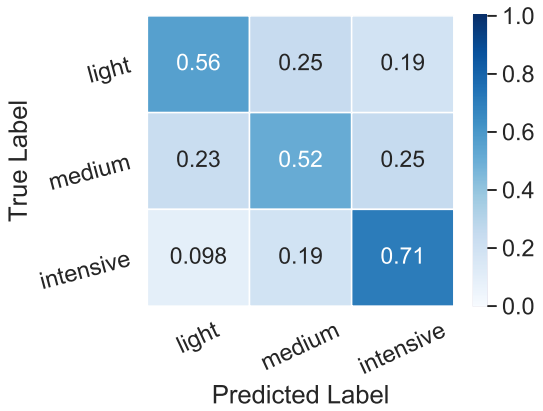


Figure 4.14: The confusion matrix of our prediction model on the engagement level. Darker color means higher probability.

the "error-correction" step. We then compare the app prediction results of sequential based and boosting based strategies. For the sequential based strategy, we found that the incorrect results are mainly generated from three reasons:

(a) ***Global popularity***. We find that for 66.7% (30/45) of the app categories, the top category they were misclassified as is *productivity*. For example, 37.4% of widgets apps are wrongly predicted as productivity, 31,7% of medical apps are wrongly predicted as productivity, and 27.2% of transportation apps are also wrongly predicted as productivity. This could be easily explained since the productivity apps have the highest global popularity among all the app categories. It results in our generic app category classifier inferring that the user has a higher probability to use the productivity apps given all users' data. By analysing these cases, we find that the boosting strategy resolves most of the issues resulted from global popularity. For example, 25.4% of widgets apps and 12.8% of transportation apps are corrected by boosting from being misclassified into the productivity apps. This is because that the prediction model in the boosting step does not involve the global popularity of app categories. Since the "pseudo-residuals" (Figure 4.13) are applied as the ground truth, this makes the

Table 4.11: Illustration of Case Studies in Next App Prediction

Error Reason Type	**Predicted Result**	**Ground Truth**	**Correction while Boosting**	**Correction/Uncorrection Reason**
Global Popularity	productivity	widgets, medical, transportation	widgets, medical, transportation	The "pesudo-residuals" are applied as ground truth in boosting makes the model less impacted by the global popularity of app categories within the dataset.
Short Usage Interval	card	casino	casino	Additional information are provided by the engagement features added in the boosting strategy. Last engagement of casino is longer than the card app, so these cases are corrected from card to casino.
	arcade	adventure	No	No additional information could be provided based on engagement.
Similar App Usage Frequency	lifestyle	food-and-drinks	food-and-drinks	Boosting helps when the engagement pattern provides additional information. Specifically, the results are corrected by boosting since the historical engagement pattern of food-and-drink apps usage of this user is longer than lifestyle apps.
	health-and-fitness	medical	No	None of them are corrected by boosting since no difference exists in the historical engagement pattern of these two apps of the user.

model more sensitive for predicting users' app usage based on the context and user characteristics, rather than the global popularity of app categories within the dataset.

(b) ***Short usage interval***. Another important factor that accounts for the misclassification within the sequential based strategy is the time interval (periodic feature) between app usage. We observe that for those users whose historical app preference deviates from an average user (e.g., productivity is not the top preferred app), the periodic feature (time since the last usage) would have stronger impacts. For example, 30.8% of the adventure apps are wrongly predicted as arcade apps given the time from the last usage of arcade app is short. Since the model learned that the shorter the

time interval, the higher probability the same app would be used again. In these cases, even the adventure has a higher usage frequency in the user's historical app usage pattern, it would be wrongly predicted as arcade since arcade has a higher probability to be re-accessed in a short period of time.

(c) ***Similar app usage frequency***. The third major reason for the misclassified cases is that when two app categories are all recently used with a similar frequency, the one with the higher personalized historical usage frequency would be selected. For example, 11.4% of medical apps are wrongly predicted as health-and-fitness apps because that the medical app and health-and-fitness app are always used in the same session, the health-and-fitness app is predicted as the result since it has higher popularity within the user's historical app usage pattern.

For the latter two error analysis in the sequential based strategy (described above as (b) and (c)), we find that the boosting strategy only helps some of the cases when the engagement information could provide additional insights. For example, among the wrongly predicted 30.8% adventure apps, only 5% of adventure apps are corrected from arcade by boosting since the last engagement level of adventure is longer than arcade. For those cases without this additional information from engagement, they are still wrongly predicted. However, for some specific cases, e.g., 7.8% of casino apps are predicted as card apps for the same reason as adventure&arcade apps, but all of these 7.8% cases are corrected to casino with boosting. This is because for all the cases the last engagement of casino is longer than the card app. Similarly, for the cases which are wrongly predicted because they have similar recent usage patterns, the boosting only helps when the engagement pattern provides additional info. For example, 3.7% of food-and-drink apps are corrected by boosting from lifestyle apps. This is because the historical engagement pattern of food-and-drink apps usage

of this user is longer than lifestyle apps. However, for the 11.4% wrongly predicted medical apps, none of them is corrected by boosting since no difference exists in the historical engagement level pattern of these two apps.

To make the benefits brought by boosting strategy clearer, we also added a summary table of the case studies as Table 4.11, where we can find the corresponding reasons about why some of the cases are corrected by boosting but some others are not.

4.6 Discussion

In this chapter, we explored the factors that affect users' app dwell time from users' app usage logs and show evidence that the next app and how long the user will stay on this app could be predicted simultaneously. First, to answer the research question **RQ 1.3**, we take a systematic approach to uncover the dependency of users' app usage duration on user characteristics and context features based on a large-scale dataset. We then showed that the features related to users' historical engagement pattern and periodic usage pattern are good predictors of how long users will stay with an app. To solve another research question **RQ 1.4**, we propose three different joint prediction strategies and demonstrate that the boosting based strategy performs the best. We further conduct the error analysis on the boosting strategy compared to the benchmark sequential based strategy. Based on the analysis, we find that besides the benefits brought by the "pseudo-residuals" within the boosting principle, the engagement features also provide additional insights to help infer which app user will use. These findings inspire us that we should think of adding more engagement related

features when predicting the next app. As below, We discuss how our findings can be applied to future mobile systems and applications, the limitations in our study and the differences between personalized and general engagement prediction.

4.6.1 Implications

Firstly, the model proposed in our work can be applied for more tailored engagement-aware recommendations on mobile phones. As the operating system has access to all the features used for training the next app and app engagement models in this work, it is uniquely suited to predict a user's likely next app usage and engagement level under the current context. By doing so, the operating system can manage the delivery of content and services to the end-user by matching their engagement demands with the predicted engagement levels of the user. For instance, an app that shows mobile advertisements require high engagement from the end-users and can ask the app provider to push its content to the user when he/she is likely to be highly engaged. The media apps, like video and news apps, could recommend more satisfactory content based on the predicted engagement level of users to improve user experience.

Recently, there is an active area of research for the timely delivery of notifications on mobile devices. Researchers have primarily focused on understanding the receptivity of mobile notifications [117] and predicting opportune moments to deliver notifications in order to optimize metrics such as response time [137]. While response time is indeed a useful metric to optimize for, they do not capture how much engagement the user will show towards the notification. The primary purpose of a notification is to attract user attention and increase the possibility of user engagement with

the notification content. As such, we believe that the models and features we explored in this work can also be incorporated in designing an effective notification delivery mechanism.

4.6.2 Limitations

Although we have explored several meaningful features that could be applied in benefiting users' app engagement (dwell time) prediction, we must acknowledge the limitations of the dataset used in our study. Firstly, similar to all other real-world app usage datasets, the app popularity distribution follows Zipf's law [94, 139], which indicates that only a few apps have high installation/usage whereas many apps have low installation/usage. But doing a balanced prediction would not fit the realistic evaluation settings, which would bring the bias of the users/contexts to be selected. To handle this imbalance issue within app categories while evaluation, we measure the performance of all prediction problems based on four metrics: accuracy, precision, recall and f1. All the metrics are calculated for each label, and their averages are weighted by support (the number of true instances for each label). We also conducted case studies to explore the prediction results based on different app categories. Additionally, our dataset might not be representative of the entire population of mobile users, as the users and apps are only coming from the apps registered in this library. This means that not all the apps usage behaviour of users could be tracked and there may be a selection bias in the subset of users being studied. While this might occur to some extent, given the scale of our dataset (with over 1.3 million logs), we believe our data would still provide useful insights, and our predictive models are the best models so far, effective for most users.

Lastly, the predictive features we extracted can be further enriched. Other

features that precisely characterize users' app engagement behaviours could be further explored, such as the location, wifi access status, battery, device mode setting (e.g., silent), illumination, screen, and blue-tooth. In this chapter, our main aim is to validate that how long a user will stay with an app could be modelled based on the user characteristics and context features. We would like to further explore additional features and models for improvements in our future work.

Despite these limitations, we believe our proposed framework is the first-of-its-kind study to examine, and model the mobile app engagement purely based on features derived from users' app usage logs. We also hope that the framework, models, and insights developed in this chapter can bring clarity and guidance to aid future mobile system developers in designing better, and engagement-aware user experience.

4.6.3 Personalized V.S. General Engagement Prediction

We know that the performance of the personalized model could be highly impacted by whether there is sufficient data for training or not. In this work, we are interested in relatively short-term user app engagement patterns. Therefore, we collect app usage data from all users for a week. Due to the nature of the data, it might be more difficult to acquire sufficient per-user patterns, which is also validated by our experiment results regarding the next app prediction (i.e. personalized model could not outperform our proposed hybrid model). When the long-term data is available, the personalized approach might be more suitable and achieve better performance. We leave the exploration of the personalized model for app usage prediction in our future work. In terms of the app engagement level (dwell

time) prediction, we introduce the engagement levels according to different app categories for handling users' different consumption behaviour on different contents. Hence our engagement level prediction results do not directly translate to user-specific engagement. Additionally, we believe that defining and measuring aggregate engagement is also useful for content producers, e.g., video producers on Youtube. The content providers often do not target a specific user, but a large number of audience. Other than predicting how long user will stay on the app, we also want to provide content producers with a new set of tools to create engaging content and forecast user behaviour. For future work, we would measure users' engagement at the personalized level as complementary to the aggregated engagement study, especially when we have more sufficient training data for individuals. It would help the mobile apps to provide more fine-grained services to a specific user based on the more accurate expected time length.

4.7 Conclusion

In this chapter, we propose to predict both users' next app and the engagement level at the same time. For the first time - to the best of our knowledge - a comprehensive analysis of users' app dwell time is conducted based on the large-scale commercial mobile logs, especially focusing on inferring the correlations between different predictive features and users' app usage duration. We find that the users' historical engagement pattern, periodic behaviour pattern, and the recent usage pattern have more impacts when inferring users' app dwell time. To solve our joint prediction problem on the next app and app engagement level, we propose three strategies, where the boosting based joint prediction model works best. Our experimental results demonstrate that users' next app and engagement level could be ef-

fectively predicted at the same time, and our proposed prediction method outperforms all baseline experiments by a large margin. This work can help for providing more satisfying services to users for improving users' experience on mobile devices.

Here we summarize the app usage behaviour understanding part of this thesis (Part I). By proposing different approaches for modelling mobile users based on cohorts and conducting various experiments, we enhance the app usage prediction especially for alleviating the *cold-start* issue for new users. Furthermore, besides only inferring "what" app the user prefers to use next, we also explore "how long" user will stay with an app. We conduct the first empirical analysis of mobile app engagement based on dwell time and investigate a novel problem on simultaneously predicting which app user will use and how long the user will stay on that app. We conclude from above studies, we now have a better understanding of mobile users' behaviour on specific apps, especially when compared to the existing approaches and models. However, nowadays, users' mobile needs could not be simply satisfied only via a single app. To provide more satisfying services for supporting users' high-level tasks, e.g., dining out with friends, we aim to move on to the next part aims at understanding the mobile tasks based on users' app usage interactions (Part II).

Part II

Extraction and Characterization of Mobile Tasks

Chapter 5

Identifying Tasks from Mobile App Usage Patterns

Part I of this thesis we presented above aims to better understand users' behaviour within single apps. However, users' mobile needs could span a broad spectrum, not only include the simple needs, such as weather information checking, which can mostly be satisfied via a single app; but also the needs to access a series of apps, collect, filter, and synthesize information from multiple sources for solving a complex task, e.g., planning a vacation. Therefore, we now turn to another part of this thesis, which is to identify and characterise tasks based on users' app usage behaviour. In this chapter, we start our investigation by a detailed study of mobile tasks manually labeled by annotators based on the real user app usage logs. This chapter addresses our research questions **RQ 2.1** and **RQ 2.2**, as specified in Section 1.1.

RQ 2.1: What kind of features can be used effectively to identify mobile tasks from app usage logs?

Table 5.1: An example of mobile task: plan to dine out with friends.

Timestamp	**App**	**SessionID**	**TaskID**	**Task Description**
18 Jan. 2014 17:49:45	WhatsApp	1	**1**	**Dining with friends**
18 Jan. 2014 17:50:10	Yelp	1	**1**	
18 Jan. 2014 17:52:15	WhatsApp	2	**1**	
18 Jan. 2014 17:57:10	Music	3	2	listen to music
18 Jan. 2014 18:04:22	Facebook	4	3	Social
18 Jan. 2014 18:05:01	Instagram	4	3	
18 Jan. 2014 18:10:50	Yelp	5	**1**	
18 Jan. 2014 18:11:10	Google Maps	5	**1**	
18 Jan. 2014 18:11:43	Uber	5	**1**	
18 Jan. 2014 18:13:54	WhatsApp	6	**1**	

RQ 2.2: Can we formulate the task identification as a supervised learning problem, which could predict the app usage belong to the same task automatically?

Helping users complete tasks [60] is crucial for a number of applications, such as search systems, digital assistants, and productivity applications. However, little research has explored methods to understand and identify mobile tasks, let alone to support users in task continuation and task completion. A primary mechanism for segmenting logged app usage streams is *session*-based, where short inactivity timeouts (30 or 45 seconds) between user actions are applied as a means to demarcate session boundaries [171]. However, tasks with users' high-level intentions may span multiple sessions and involve different apps, where the empirically-set short timeout threshold may not be a valid criterion.

Consider a hypothetical example of a mobile task of a single user shown in Table 5.1. The logs are automatically segmented into sessions (defined as a series of consecutive app usage without standby over a time threshold) using the 45-second inactivity threshold [171]. They are then manually annotated into tasks with the corresponding task ID. We can observe that Task 1 crosses four sessions and it is interleaved with Task 2 and Task 3. To plan dinner with friends, the user first chats with friends on WhatsApp,

and then access Yelp to look for restaurants and book a table. The user may switch between Yelp and WhatsApp to get confirmation with friends about which restaurant they prefer to go to. Finally, the user copies the restaurant address from Yelp to Google Maps to check where the restaurant is, and then book a ride on Uber. This series of log activities suggest that these interactions belong to the same task, spanning across multiple sessions and that not all apps are used consecutively (interleaved with other tasks). During these types of complex mobile tasks, users always need to access and switch between different apps frequently, as well as searching and editing a similar text more than once. If we could understand users' tasks in advance, these redundant operations could be optimized. Furthermore, if we were able to accurately identify sets of app usage with the same intent, we will be in a better position to evaluate the performance of mobile services from the user's point of view.

To this end, we annotated week-long app usage logs of 20 users into tasks. We report on the properties of these annotated mobile tasks, learning that 22.6% of all the tasks are interleaved and 19.7% of tasks contain multiple different apps. This suggests that mobile task extraction is not a trivial problem. We then built classifiers to identify task boundaries between each sequential pair of app usage, as well as arbitrary pairs of logs that correspond to the same task, despite being interleaved with apps usage from other tasks. Last, we discussed the implications of our proposed automatic task segmentation approach.

The remainder of this chapter is organized as follows. Related work is reviewed in Section 5.1. In Section 5.2, we formally define mobile tasks and present the way we manually annotated the mobile tasks. In Section 5.3, we propose a set of predictive features and evaluate the performance of two set of supervised classification models on both task boundary detection and

same-task identification. We conclude this chapter in Section 5.4.

5.1 Related Work

In the context of web search, there have been many attempts to segment and define tasks, relying on a notion of timeout, lexical characteristics [178], and topic [69]. Many of them used the idea of a "timeout" cutoff between queries to bound tasks, i.e. 30 minutes [28, 24, 189]. As the timeout features only make sense between consecutive queries, these approaches cannot detect interleaved tasks, which are prevalent in real-life query logs. Some approaches [17, 36, 78] consider lexical cues and treat queries, titles, and snippets of clicked URLs as bag-of-words, and use some string similarity metrics (e.g., Levenstein edit distance, n-gram Jaccard) to measure the similarity between queries. However, Huang et al. [70] later pointed out that many queries relating to the same task are dissimilar in their surface form but instead are related at the topic level (e.g., queries expressing car interests: "honda", "nissan", and "ford"). Features that aim to capture topical relatedness have been proposed by [69] and [109] to improve the accuracy of task identification.

As we discussed above, many works have been done for task identification in search [78, 86, 178, 69]; however, how to identify tasks within mobile app usage and what features are effective have not been studied. To extract a ground-truth of mobile tasks (Section 5.2.3), we follow the annotation procedure of search tasks shown in Table 5.2. The biggest challenge in identifying mobile tasks is that the apps do not include abundant information as for search queries. Additionally, most of the app usage logs do not provide detailed behavior information within the apps due to privacy issues.

Table 5.2: Summary of the task annotation procedure

Ref.	Search Task [78]	Search Task [178]	Search Task [109]	Mobile Task (Our work)
Labeled item	each query log	each query log	each query log	each app usage log
Time span	3 days	5 days	1 week	5 days
Task Guidance	"*...they have the same criteria for "success", in terms of satisfying the user's information need...*"	"*...group the queries into tasks according to annotators' understanding of users' information needs...*"	"...claimed to be task-related within each time-gap session..."	"*...The app usages should be grouped into one task when they all work for the same aim...*"(a number of examples for explaining mobile tasks are shown in the annotation page).
Auxiliary Info.	clicked URLs, page titles, relevant snippets, etc.	search for logged queries and browse the clicked URLs.	No	App description for introducing app function, content, and users' comments, etc.
No. of Annotators	a group	3	from their laboratory but not directly involved in this work	3
Label Output	Task ID and description	No	tag and optionally a longer description	Task ID and optionally task description
Validation	No	Cohen's kappa	No	Cohen's kappa

Nonetheless, an essential characteristic of mobile apps is that most of them are created for satisfying the specific needs of users, e.g., weather apps for displaying weather information and map apps for helping users in navigation. Therefore, even if we cannot audit internal interactions within apps, the app description information includes information about the function of the app. We extract the app description information to help annotators in judging mobile tasks.

Another challenge of mobile task identifying is that we have less support information for verifying if two operations are serving one task. In the search tasks, even if the queries have fewer words, some common information can be found in the searched results (clicked URLs); this can help in determining whether two queries related to the same information needs. Going back to our cases, we also measure other supportive operations to help identify mobile tasks, e.g., whether two apps are frequently switched back and forth within a short period of time. In Section 5.3.2, we show that the traditional temporal features combined with our proposed novel app-log features, e.g. similarity features extracting lexical characteristics and log sequence features capturing topic relatedness, can be used to classify app streams into task structure.

5.2 Mobile Tasks

We formally define mobile tasks and formulate the automatic identification of mobile tasks as two supervised machine learning tasks. We then present the way we manually annotated the mobile tasks, which generate the ground-truth of our supervised learning. Lastly, we perform an analysis on the annotated tasks.

5.2.1 Task Definition

App usage log records mobile app interaction behaviours from a set of different users $U = \{u_1, u_2, ..., u_N\}$. It stores a sequence of app usage $L_n = \{(a_1^n, t_1^n), (a_2^n, t_2^n)..., (a_M^n, t_M^n)\}$ from user u_n, where t_i^n is the corresponding timestamp when using app a_i^n.

Definition 5.1 *(SESSION S_t^n) Given user u_n's app usage logs L_n and a fixed time-out threshold τ, a session S_t^n is a set of consecutive app usage from L_n, such that* $\forall (a_i^n, t_i^n) \in S_t^n,\ (a_j^n, t_j^n) \in S_t^n,\ (a_l^n, t_l^n) \notin S_t^n,\ |t_i^n - t_j^n| \leq \tau_{cut}$ *and* $|t_i^n - t_l^n| > \tau_{cut}$.

The definition of *session* implies that $\{S_t^n\}_{t=1}^T$ is a set of disjoint partitions of app usage logs L_n, such that $\forall i \neq j,\ S_i^n \cap S_j^n = \emptyset$ and $L_n = \bigcup_i S_i^n$. Typical time-out threshold τ_{cut} in the context of mobile apps is set to be a short period, i.e., 30 [16, 26] or 45 seconds [171]. We adopt in the rest of the thesis the threshold $\tau_{cut} = 45$ seconds to segment app sequences into sessions, following the recommendation from the systematic analysis conducted in [171]. A session, for us, is just a slice of user time. Other definitions (which conflict themselves) involve an absence of periods of inactivity [16, 26], or app used between unlocking and locking the phone [79]; ours does not, since we want to account for *tasks*, defined below, and use inactivity as a predictor, rather than as a definition.

Definition 5.2 *(TASK T_k^n) Given user u_n's app usage logs L_n, a mobile task T_k^n is a maximum subset* $\max_{T_k^n \in L_n} |T_k^n|$ *of logs in L_n, such that all the app usage in T_k^n correspond to a particular need.*

This definition of mobile task indicates that $\{T_k^n\}_{k=1}^K$ is also a set of disjoint partitions of app usage sequence L_n : $\forall j \neq k, T_j^n \cap T_k^n = \emptyset$ and $L_n = \bigcup_k T_k^n$.

However, each T_k^n is not confined to a particular session S_t^n segmented only based on time threshold; instead, one mobile task can contain multiple sessions, even if they are not consecutive. A mobile task can be thought of as a group of related apps to accomplish a single discrete task. As the example shown in Table 5.1, all the app usages of Whatsapp, Yelp, Google Maps and Uber may span across multiple sessions that are not consecutive. However, they belong to the same task of "planning to have dinner with friends".

5.2.2 Formulation for Supervised Task Learning

5.2.2.1 Task Boundary Detection

If tasks are not interleaved, as assumed in previous work [171], it suffices to find a boundary between one task and the next. To do this we can look at each sequential pair of app usage and ask whether this pair straddles a boundary. Thus we look at *task boundary* detection. Each pair of sequential app usage from a user's log is a possible boundary between tasks. We seek to take each such pair and decide whether the pair crosses a boundary between tasks. Formally we consider the task:

$$\{\langle(a_i^n, t_i^n), (a_j^n, t_j^n)\rangle : (t_i^n < t_j^n) \bigwedge (a_k^n : t_i^n < t_k^n < t_j^n)\} \rightarrow \{0, 1\}$$

where t_i^n is timestamp of app usage a_i^n; $\langle(a_i^n, t_i^n), (a_j^n, t_j^n)\rangle$ represents any *consecutive* app usage pair; $\{0, 1\}$ represents a binary variable whereas 0 and 1, respectively, indicate non-boundary and boundary. This task boundary detection was traditionally addressed using timeouts [171].

5.2.2.2 Same-task Identification

No previous work has addressed interleaved tasks when measuring users' mobile app usage behaviours. Therefore another supervised learning problem is proposed to cover all kinds of tasks identification, no matter whether they are interleaved or not. In this scenario, we must consider all possible pairs of apps usage, and consider whether the pair of apps usage come from the same task. Correctly performing this task will allow interleaved tasks to be identified. We call this *same-task* identification. We seek to learn a classifier to take a pair of app usage logs and map it to 1 if they are from the same task, and 0 if they are from different tasks. We consider all pairs of app usage logs $\langle (a_i^n, t_i^n), (a_j^n, t_j^n) \rangle$ such that a_i^n was accessed before a_j^n:

$$\{\langle (a_i^n, t_i^n), (a_j^n, t_j^n) \rangle : t_i^n < t_j^n\} \rightarrow \{0, 1\}$$

where t_i^n is the timestamp of app usage log a_i^n; here $\langle (a_i^n, t_i^n), (a_j^n, t_j^n) \rangle$ represents any possible app usage pairs.

5.2.3 Mobile Task Annotation

To acquire a ground-truth of mobile task labels for the supervised learning tasks (Section 5.2.2), we conduct a mobile task annotation crowd-sourcing study. We sample the app usage logs for such annotation from the publicly available UbiqLog dataset[1] [147, 146], where participants were required to install the lifelogging app UbiqLog on their phones from November 2013 to January 2014. We select 20 users randomly, for which five-day of app usage logs are collected, including anonymized user ID, app package ID and corresponding timestamps. Furthermore, to help the annotators obtain a

[1]UbiqLog: https://archive.ics.uci.edu/ml/datasets/UbiqLog+(smartphone+lifelogging)

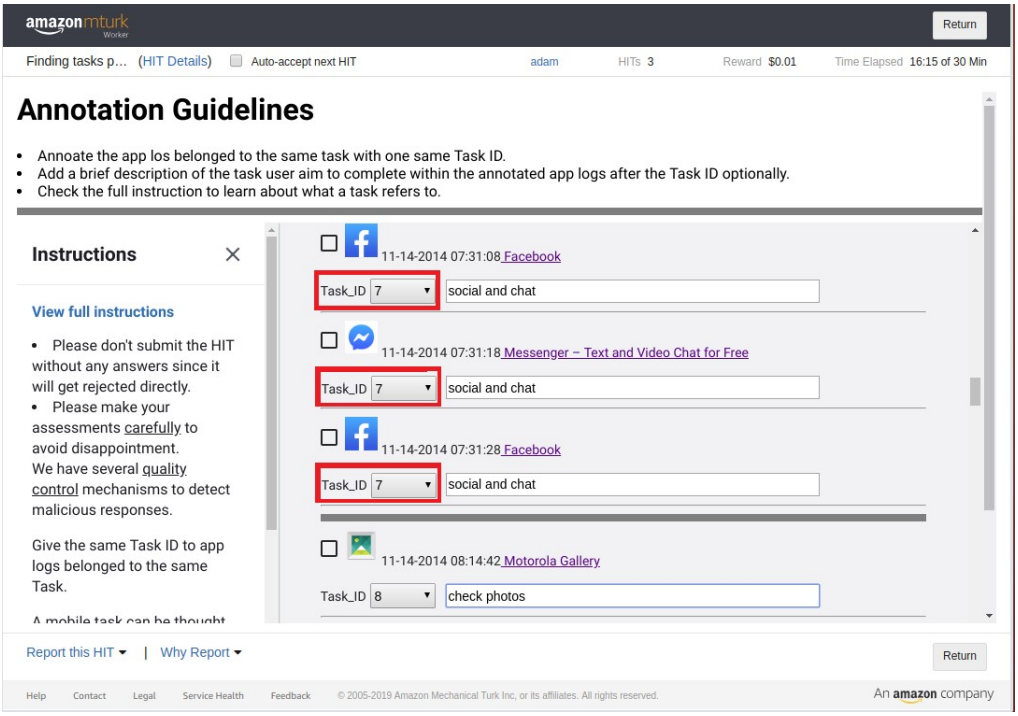


Figure 5.1: Screenshot of the annotation page on MTurk (same task id "7" should be assigned to the three app usage logs in the red rectangle if they belong to the same task).

good understanding of each app, we crawled additional information, such as app title, genre, description, icon and the URL of the app on Google Play.[2] Some statistics of this dataset are shown in Table 5.3.

Crowdsourced assessments have been commonly used to obtain labeled data [184, 119]. To identify mobile tasks, three annotators were recruited from Amazon Mechanical Turk [5], which is a crowdsourcing website for businesses (known as Requesters) to hire remotely located "crowd workers" for performing discrete on-demand tasks that computers are currently unable to do. No personally identifiable information was collected. Only an anonymized ID is used to identify the different annotators.

Since no research has been done for exploring the methods of understanding and identifying mobile tasks, most of our annotation procedures follow

[2]We can trace back an app on Google Play by using the app package ID - the unique identifier, e.g., *com.yahoo.mobile.client.android.weather* is the Yahoo Weather app.

the prior work on search task annotation [78, 86]. A detailed guideline was presented to the assessors, describing in general what a mobile task is (reformulated from Section 5.2.1) and showing several examples demonstrating what constitutes a mobile task. A sequence of app usage is considered as part of a coherent mobile task if they collectively try to achieve a certain goal. One such example of a mobile task is: a user may check the date with the Calendar app when replying/sending an email via the Email app. In this task, the interval between the access of Calendar and that of Email app is very short (e.g., 10 seconds). In addition, the user switches back and forth between these two apps. Therefore, these app usages should be grouped into one task since they work for the same aim – finishing writing the same email (with time information). To ensure the quality of the assessment results, we apply a series of quality control mechanisms. We create a set of "trap hits" to detect whether the assessors made assessments that are consistent with those we know the answer. All the assessments from assessors who fail a number of "trap hits" were removed.

The annotation page presented to the assessors is shown in Figure 5.1, including the app usage information such as timestamp, app icon, and app name. To provide relevant information to the assessors, we provided the URLs directed to the corresponding app info on Google Play, where the assessor could browse the detailed description of app functionalities, screenshots and user comments of this app. Each assessor was asked to select a Task ID number from the drop-down menu to label an app usage, and each app usage log belonged to the same task was labeled with the unique same task ID as shown in Figure 5.1. The annotators were also asked to optionally write a short description for each task. We measured the inter-annotator agreement using Cohen's Kappa [48] as previous studies that focused on the search task annotation [86, 119]. The same set of

Table 5.3: Statistics of the dataset for annotation

#User	#Log	App/User	Log/User
20	3558	22.3 ± 8.3	177.9± 63.7

Table 5.4: Statistics of the annotated mobile tasks

All Tasks	
#Task	**#Task/User**
1414	46.8±38.6
#Single-log Task	**#Multi-log Task**
717 (50.7%)	697 (49.3%)
Multi-log Task	
#Logs/Task	**#Apps/Task**
4.1±3.7	1.7±1.3
#Same-app Task	**#Multi-app Task**
418 (29.6%)	279 (19.7%)
#Interleaved Task	**#Interleaved Task/User**
320 (22.6%)	12.1±11
#Multi-app Task/User	**Task Duration**
9.1±7.7	22±84.3 (min)

logs was annotated by three assessors and we measured the inter-rater agreement. Following [178], we randomly select all the logs from a subset of users (2 out of 20) and instructed three annotators to assess all those logs. Through this strategy, we can exploit the assessments on this subset of user logs to calculate the Kappa among annotators. A Kappa value of 0 implies that any annotator agreement is due to chance, whereas a kappa value of 1 implies perfect agreement. In our data, the Kappa values for the three pairs of annotators were 0.69, 0.65, and 0.71, which, according to [87], represent the *substantial* agreement. This partially demonstrates the internal validity of our annotation method.

5.2.4 Patterns of Mobile Tasks

After aggregating the three assessors' annotations, we ultimately obtain a collection of 1414 tasks annotated out of 20 users' app usage logs. The statistics of those annotated tasks are shown in Table 5.4. We observe that firstly, 49.3% of the mobile tasks require users to visit apps for more than once (*multi-log task*, 49.3%), which consists of both *same-app task* (users

visit the same app more than once, 29.6%) and *multi-app task* (users visit more than one app, 19.7%). An example of such *same-app task* is: a user accessed the Clock app for four times at 6:00, 6:05, 6:35 and 7:05 in the morning. The four alarm clocks wake the user up and provide small nap intervals. Additionally, an example of *multi-app task* is: a user switched between Calendar and Email app as mentioned in Section 5.2.3.

We can also observe that among all these annotated tasks, 22.6% of them are *interleaved*, which means that not all apps within one task are consecutively accessed. For example, the user may perform the task of playing games, interleaved by the task of chatting with friends in between. All the above task patterns demonstrate the need for moving beyond single once app usage and extracting high-level mobile tasks. In particular, when we look into those multi-log tasks, they on average span across 4.1 logs, with 1.7 apps accessed and last 22 minutes. These indicate that cross-log and cross-app mobile task extraction are not trivial problems.

5.3 Task Identification

We evaluate the performance of a set of predictive features (Section 5.3.1) using two set of supervised classification models (Section 5.3.2) on both task boundary detection and same-task identification.

5.3.1 Predictive Features

We describe the features we use in our experiments to classify tasks. We experimented with 14 features related to app usage patterns covering three aspects: temporal, similarity and log sequence. Table 5.5 provides an

overview of the features.

5.3.1.1 Temporal Features

While timeouts alone have been commonly used as predictors of session boundaries, they may help in identifying task boundaries and especially when used with other features. To measure the temporal characteristics of two apps usage, inspired by the study on search task identification [78], we generate three forms of temporal features: *binary_interval*, *time_diff* and *sequential_status*. For a given log pair, both *binary_interval* and *time_diff* features capture the duration of the interval between this log pair, where the longer the duration, the less likely this log pair would be part of the same task. Specifically, the *binary_interval* represents whether the inter-log interval exceeds a threshold (e.g. 10s, 1min, 5min, etc.) while *time_diff* concentrates on the exact inter-log time in seconds. Thirdly, the *sequential_log* feature is used to represent if two app usage logs are sequential in time, with no instance of other log entries[3] (apps); potentially indicating that these two log entries originate from the same task.

5.3.1.2 Similarity Features

A previous study [191] found that apps in the same or similar genre are more likely to be used together. Furthermore, users were found to often browse multiple similar apps to compare and obtain complementary information to accomplish their mobile tasks [187]. To quantify the similarity between any pairs of apps, the broad category or the detailed description of the app can be used. As shown in Table 5.5, first, we use the feature *same_cate* to

[3]This *sequential_log* feature does not apply to task boundary detection given all the log entry pairs are sequential in this setting.

Table 5.5: Overview of features for identifying whether a pair of two sequential or arbitrary app usage logs relate to the same task.

Temporal Features	
binary_interval	Inter-log time threshold as a binary feature (10s, 1min, 5min, 10min, 30min, 60min, 120min), e.g. if inter-log time > 10s, binary_interval (10s) = 1; otherwise binary_interval (10s) = 0.
time_diff	Inter-log time in seconds; We may be able to learn good thresholds of inactivity for identifying task boundaries.
sequential_logs	Binary feature which is positive if the pair of logs are sequential in time, with no intervening actions between the pair of app usage. We expect this feature to be useful for the same-task identification.
Similarity Features	
same_cate	Binary feature for identifying if pairs of apps belong to the same app category.
common_w	Number of words in common of the app descriptions.
tfidf_cosine	Cosine similarity between the term sets of app description while each term is weighted by the tf-idf weighting scheme.
jaccard_coeff	Jaccard coefficient between the term sets of app description.
word_embed_sim	Cosine similarity between the word embeddings (Section 5.3.1.2) based on the term sets of app description.
Log Sequence Features	
PMI	Point mutual information for finding collocations and associations between apps frequently used together within the same session.
pa12	The probability for measuring if two apps are always successively used.
switch_state	Binary feature indicates whether there exists switch (i.e. $a_2^n \rightarrow a_1^n \rightarrow a_2^n$) between a_2^n and a_1^n.
switch_prob	The probability of switch interaction happened with a_1^n and a_2^n.
peos_a2	The probability that app a_2^n is users' last app usage of the day.
no_dist	Number of apps usage logs in between a_1^n and a_2^n. This feature is for same-task identification.

identify if two apps originate from the same broad app category (e.g., shopping and music). To capture a more nuanced similarity between apps, we exploit the app descriptions and leverage four textual similarity measures: *common_w*, *tfidf_cosine*, *jaccard_coeff* and *word_embed_sim*, as defined in Table 5.5. These four similarity measures of any log entry pairs are calculated based on their corresponding app description crawled from Google Play, with all stop words filtered out. The former three measures are based on traditional lexical similarity whereas the *word_embed_sim* approach utilizes the semantic similarity based on the word embedding representations of the app descriptions. The word embedding vectors are based on GloVe vectors trained on Common Crawl [4]. The sentence representation is simply an average of the word embedding representations of all the words in the sentence.

5.3.1.3 Log Sequence Features

Sometimes tasks may contain pairs of apps that are logistically related but do not share common terms in their description. For example, "Yelp" and "Google Maps" may be used to carry one task, planning a dinner with friends; but both apps have no common functions and are not from the same app category. To capture such relationship between pairs of apps $\langle a_1^n, a_2^n \rangle$, we introduce six features based on leveraging historical app usage data:

- *PMI*: Pointwise Mutual Information(PMI) [32] is a measure of correlation defined as:

$$I(x, y) = log \frac{P(x, y)}{p(x)p(y)}$$

  The numerator is the probability of co-occurrence of the events x and y; the denominator is the probability of each event occurring

independently. In our scenario, the higher the PMI between two apps, the higher the possibility that these two apps will co-occur in the same session. To calculate the PMI of any app pair $\langle a_1^n, a_2^n \rangle$, the app probabilities $P(a_1^n)$ and $P(a_2^n)$ are estimated by counting the number of observations of a_1^n and a_2^n across all the app usage sessions, and normalizing by N, which is the number of sessions. The joint probability, $P(a_1^n, a_2^n)$, is estimated by counting the number of times that a_1^n and a_2^n co-occurred in the same session, normalizing by N.

- *pa12*: $\frac{p(a_1^n \to a_2^n)}{\max_{a_j^n} p(a_1^n \to a_j^n)}$ is the normalized probability that a_2^n is used right after a_1^n within the same session [78]. It is used to measure whether two apps are always used successively.

- *switch_state*: captures users' switching between two apps. For any log pair $\langle a_1^n, a_2^n \rangle$, *switch_state* is 1 if a_2^n was used right before $\langle a_1^n, a_2^n \rangle$ in the same session. This represents an in-session user interaction of $a_2^n \to a_1^n \to a_2^n$, which indicates that the user u_n switches back and forth on a_2^n within the same session. Otherwise, *switch_state* is 0.

- *switch_prob*: $\frac{f(a_1^n \to a_2^n \to a_1^n)}{f(a_1^n \to a_2^n)}$ is the probability that a switch between a_1^n and a_2^n happens when accessed sequentially.

- *peos_a2*: since often people finish a task before turning off for the day [78], "last app usage of the day" might be a useful indicator of the last app used in a task. Following from [78], we generate the feature *peos_a2* to capture the probability that a_2^n is the last used app based on aggregating app usage logs of all users before midnight.

- *no_dist*: represent the number of app usage logs (distance) between a_1^n and a_2^n [178]. This feature only applies to those arbitrary app usage log pairs for the same-task identification.

5.3.2 Predictive Models

Given our predictive features, we introduce a set of state-of-the-art algorithms to build models for the two classification problems: task boundary detection and same-task identification. We compare four widely used classifiers: (1) L2-regularized Logistic Regression (LR) [62], as an example of linear classifier; (2) K Nearest Neighbours (KNN) [33], as an example of a non-parametric method for classification; (3) Support Vector Machines (SVM) with Radial Basis Functions kernel [174] as an example of a non-linear classifier; and (4) XGBoost [29], as an example of a state-of-the-art ensemble learning. These models construct different prediction functions for the data from different aspects. Rather than training a personalized classifier for each user, we make our classifiers generic so that they can be applied across all users.

5.3.3 Metrics and Baselines

Four metrics are used to measure the performance of our proposed classification models: accuracy (Acc.), precision (Pre.), recall (Rec.) and F-measure (F-mea.). We measure the performance of each method by splitting the data based on users with 5-fold cross validation (80% users' logs for training and 20% users' logs for testing).

Next, we construct a set of baselines to compare against our proposed approach. Since no prior research was conducted on mobile task identification, we adopt models used in search task identification [144, 86, 78] as our baselines. These models are based on either timeout or similarity between queries. To compare with methodologies using a timeout, we use both a thirty-minute threshold [144], as well as time thresholds learned using

cross-validation. To adapt the similarity-based approaches, we follow [86], which utilizes logistic regression to learn a model using only Levenshtein edit distance between the current (given) query and all previous queries. This is a reasonable baseline under the assumption that an intelligently-chosen threshold applied to the dissimilarity between two queries could provide an accurate prediction of whether two queries related to the same task. In our task identification problem, the Levenshtein distance is calculated based on app descriptions instead of queries. Additionally, Jones et al. [78] use a strong baseline $commonw + prisma + time$ to identify the search tasks, where the *commonw* identifies the number of words in common and *prisma* is the cosine distance between vectors derived from the first 50 search results for the query terms. Since we are using apps instead of queries, *commonw* is replaced by identifying if two apps belong to the same category, whereas *prisma* is replaced by calculating the cosine distance between vectors derived from the app descriptions. Lastly, we also use the mobile session segmentation method with a 45-second threshold [171] as a benchmark, where a session is considered as a task.

5.3.4 Experimental Results

We evaluate the classifiers for task boundary detection as well as identifying whether arbitrary pairs of logs belonging to the same task. Table 5.6 reports the performances of these models with different baselines and feature sets. Only results for the logistic regression classifier are reported in Table 5.6 since it outperforms other classifiers, as we show in Table 5.7. The comparative rankings of models that utilize different feature sets are similar across the different predictive models (Section 5.3.2). From Table 5.6, we can observe that in general, when we combine all three types of

Table 5.6: Performance comparison of different feature sets based on Logistic Regression (5-fold cross validation) for task boundary detection and same-task identification. $*$ indicates statistically significant ($p \leq 0.05$) using two-tailed T-test compared to the F-measure of best baseline. Bold scores state the best performance in each prediction task.

Task Boundary Detection	**Measurements**			
	Acc	Pre	Rec	F-mea
Baselines				
Search Threshold (30min) [144]	0.56	0.33	0.81	0.47
Learned Time Threshold	0.52	0.55	0.83	0.62
Trained Levenshtein distance [86]	0.63	0.61	0.74	0.66
commonw+prisma+time [78]	0.80	0.74	0.94	0.82
Mobile Session Threshold (45s) [171]	0.44	0.37	0.80	0.51
Proposed Features				
Temporal (T)	0.63	0.66	0.65	0.64
Similarity (S)	0.80	0.75	0.92	0.82
Temporal + Similarity (T+S)	0.80	0.75	0.92	0.83
Sequence (LS)	0.87*	0.86*	0.88*	0.87*
Temporal+Similarity+Sequence (T+S+LS)	**0.89***	**0.88***	0.91*	**0.89***
Same-task Identification	**Measurements**			
	Acc	Pre	Rec	F-mea
Baselines				
Search Threshold (30min) [144]	0.77	0.85	0.85	0.85
Learned Time Threshold	0.78	0.78	1.00	0.87
Trained Levenshtein distance [86]	0.74	0.74	1.00	0.84
commonw+prisma+time [78]	0.78	0.78	1.00	0.87
Mobile Session Threshold (45s) [171]	0.73	0.74	0.99	0.84
Proposed Features				
Temporal (T)	0.74	0.74	1.00	0.84
Similarity (S)	0.74	0.74	1.00	0.84
Temporal+Similarity (T+S)	0.78	0.78	0.98	0.86
Sequence (LS)	0.80*	0.78*	1.00*	0.88*
Temporal+Similarity+Sequence (T+S+LS)	**0.82***	**0.82***	0.97*	**0.89***

Table 5.7: Overview of the performance for different classifiers with the best performing feature sets (5-fold cross validation).

Classifiers	**Measurements**			
	Acc	Pre	Rec	F-mea
Task Boundary Detection				
KNN: All Feature Sets (T+S+LS)	0.75	0.69	0.92	0.78
SVM: All Feature Sets (T+S+LS)	0.63	0.60	0.88	0.70
XGBoost: All Feature Sets (T+S+LS)	0.89	0.87	0.90	0.88
LR: All Feature Sets (T+S+LS)	0.89	0.88	0.91	0.89
Same-task Identification				
KNN: All Feature Sets (T+S+LS)	0.75	0.76	0.90	0.82
SVM: All Feature Sets (T+S+LS)	0.76	0.76	0.99	0.85
XGBoost: All Feature Sets (T+S+LS)	0.80	0.78	1.00	0.88
LR: All Feature Sets (T+S+LS)	0.82	0.82	0.97	0.89

features we achieve the best results for both task boundary detection and same-task identification, outperforming all baselines.

When examining solely on task boundary detection, our model exploiting all feature sets achieves the highest F-measure score of 0.89. This is when temporal features are used in conjunction with similarity and log sequence features. This means that time interval, similarity and sequential relationships between apps are complementary, and should all be taken into consideration to detect task boundary. When a set of features is used on its own, log sequence features work best, whereas temporal features perform poorly. Comparing against the baseline approaches, despite its relatively poor performance, our proposed temporal features still outperform the best time-based baseline (Learned Time Threshold with F-mea = 0.62). These results demonstrate that solely using the time interval between two app usage (e.g., Mobile Session Threshold [171]) is not sufficient to indicate that a task has been completed, as assumed in prior studies [171]. We find similar trends for same-task identification. Note that due to the nature of this problem, our training and test data are more biased: the majority of app usage pairs do not belong to the same task. This is the reason why most of the baseline models achieve relatively high performance (with F-measure at around 0.8). Compared to those adapted baselines, models that incorporate log sequence features perform significantly better. When comparing different classifiers, as shown in Table 5.7, we find that the differences are relatively small while the LR classifier performs the best, followed by XGBoost.

Table 5.8: Feature weights (absolute value of standardized coefficients) for logistic regression model to identify the task boundary and pair of logs within the same-task. * indicates p-value ≤ 0.01 using Chi-Squared test.

Task Boundary Detection			**Same-task Identification**		
Feature	**Feature Type**	**Weight**	**Feature**	**Feature Type**	**Weight**
pa12*	Log Sequence	-1.440	time_diff*	Temporal	-0.946
time_diff*	Temporal	0.832	no_dist*	Log Sequence	0.578
common_word*	Similarity	-0.521	PMI*	Log Sequence	0.232
jaccard_cofficient*	Similarity	-0.507	word_embed_sim*	Similarity	0.352
tfidf_cosine*	Similarity	-0.500	binary_interval(120 min)*	Temporal	-0.185
word_embed_sim*	Similarity	-0.460	pa12*	Log Sequence	0.168
PMI*	Log Sequence	-0.338	switch_prob*	log Sequence	0.127
same_cate	Similarity	-0.310	binary_interval(60 min)*	Temporal	0.103
switch_prob*	log Sequence	-0.275	same_cate*	Similarity	0.071
switch_logs*	Log Sequence	-0.097	jaccard_coefficient*	Similarity	-0.069
binary_interval(1min)	Log Sequence	0.088	tfidf_cosine*	Similarity	-0.050
binary_interval(60min)	Temporal	0.075	binary_interval(30 min)*	Temporal	0.042
binary_interval(5min)	Temporal	0.070	common_word*	Similarity	0.041
binary_interval(10min)	Temporal	0.064	sequential_logs*	Temporal	0.037
binary_interval(10s)	Temporal	0.053	binary_interval(5 min)*	Temporal	-0.036
binary_interval(30min)	Temporal	0.040	binary_interval(10 min)*	Temporal	0.013
peos	Log Sequence	0.039	binary_interval(1 min)	Temporal	-0.013
binary_interval(120min)	Temporal	0.036	binary_interval(10 s)	Temporal	0.008
			peos	Log Sequence	0.001

5.3.5 Feature Importance

We showed above that by using a combination of three types of features, we could get the best performance for both task boundary detection and same-task identification based on the logistic regression classifier. In this section, we examine the contribution of each individual feature based on the feature coefficients in their corresponding logistic regression models.

To compare the importance of different features, we divide each numeric variable by two times its standard deviation [53]. This way, the resulting coefficients are directly comparable for both binary variables (e.g., categorical dummy variable) and numerical features. Table 5.8 summarized the feature weights for task boundary detection and same-task identification problems, respectively. It is not surprising that *time_diff* (inter-log time in seconds) is among the strongest signals for both problems. The longer the time interval, the less likely the app usage pair relates to the same task. For the task boundary detection, most of the similarity features receive higher importance weights. This indicates that similar apps that are sequentially used are likely to relate to the same task. This is especially true given the large percentage of *same-app tasks* (29.6%), i.e., users access the same app multiple times sequentially. By contrast, for the same-task identification problem, the log sequence features, especially *no_dist* and *PMI*, receive higher weights. This indicates that, for any arbitrary pair of app usage, co-occurrence based features are more predictive, compared to temporal and similarity-based features. Not surprisingly, if the two app usage log entries are proximate in time (*time_diff*) and more semantically similar to each other (*word_embed_sim*), they are more likely to belong to the same task. Interestingly, when the app pair is both temporally and semantically similar, these two log entries are more likely to form a task

if they are more "distant" (i.e., there are more apps in between, captured by *no_dist*). This implies that those tasks are commonly interleaved with other tasks. Furthermore, we find that the *binary_interval* (120 min) has more influence on same-task identification than task boundary detection. For any arbitrary pair of app usage, if the interval time is longer than two hours, this pair is less likely to belong to the same task. For the similarity features, the semantic-based feature *word_embed_sim* contributes more to the same-task identification than task boundary detection.

5.4 Conclusion

No previous study has analyzed or addressed the automatic identification of mobile tasks. In this chapter, we present a method that accurately determines mobile tasks from users' app usage logs. We showed that a set of temporal, similarity and log sequence features used in combination can effectively predict mobile tasks. When used independently, log sequence features, which capture the hidden relationship between apps perform best. Our proposed method to identify tasks outperform all baselines, even when they are interleaved. We also showed that matching any pairs of logs into one same task is a harder problem, compared to determining task boundaries. This suggests that it may be better to first identify task boundaries, and then extract app usage of specific tasks from the identified task segments.

The model proposed in this chapter sets the stage for evaluating mobile apps and services, not on a per-app basis, but the basis of user tasks. To improve user experience by providing more satisfying services in supporting task continuation and task completion, we need to have more understand-

ing of mobile tasks at the population level, especially for those complex tasks users aimed to complete with more than one app. Therefore, we further follow up on the work presented in this chapter by focusing on the characterizing of complex mobile tasks in the next chapter.

Chapter 6

Characterization and Clustering of Complex Mobile Tasks

In the previous chapter, by manually annotating mobile tasks from a small dataset (Section 5.2.3), we shed lights on some important characteristics of mobile tasks. However, the laboratory study of mobile tasks is still limited. To gain further understanding, mapping large-scale app usage logs to tasks is required. Specifically, learning more about the complex mobile tasks [1] users aimed to complete with cross-app and multi-topic usage patterns on smartphones could help us improve the current mobile systems and applications in supporting task continuation and task completion more effectively. However, there still lacks an accurate picture of how users engage with multiple apps during a task, let alone an in-depth understanding of the underlying high-level user intentions. Therefore, this chapter addresses our research questions **RQ 2.3** and **RQ 2.4**, as specified in Section 1.1.

[1] Referring to [103], which stated that a search task involving searching one kind of information as low complexity, two or more information as moderate and high complexity, we define the *complex mobile tasks* as the tasks that have more than two apps involved. We use complex task interchangeably with *multi/cross-app tasks* for the whole thesis.

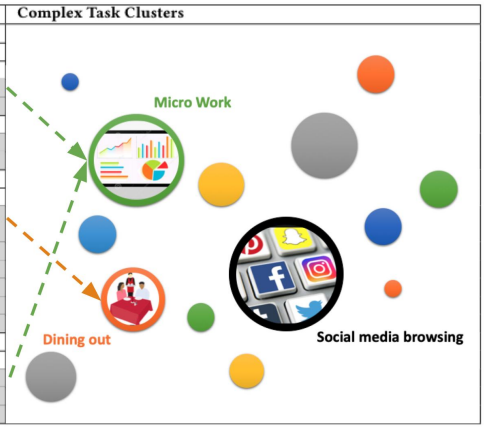
Time	App	SessionID	TaskID & DESC.	Complex Task Clusters
08:30:10	Clock	1	1 Wakeup alarm	
08:35:03	Clock	2	1	
09:45:55	Google	3	2 Search info	
10:54:32	Adobe Reader	4	**3 Scanning PDF**	
10:55:04	CamScanner	4	**3**	
11:58:19	WhatsApp	5	4 Communication	
11:02:05	File Manager	6	**3**	
11:04:51	CamScanner	7	**3**	
12:02:12	Gmail	8	5 Check mail	
12:04:56	Facebook	9	6 Social	
12:49:45	WhatsApp	10	**7 Booking Restaurant for Lunch with Friends**	
12:50:10	Yelp	10	**7**	
12:55:15	WhatsApp	11	**7**	
13:05:10	Google Maps	12	**7**	
13:05:43	Uber	12	**7**	
13:06:17	WhatsApp	12	**7**	
15:48:01	Spotify	13	8 Listen to music	
16:02:12	Spotify	14	8	
16:04:25	Gmail	15	**9 Replying to Email**	
16:05:10	Calendar	15	**9**	
16:05:49	Gmail	15	**9**	

Figure 6.1: An example of mobile tasks: the cross-app tasks 3, 7 and 9 are the complex tasks we aim to understand in this chapter.

RQ 2.3: How to characterize complex mobile tasks based on different attributes?

RQ 2.4: Could we uncover the common patterns exist in complex mobile tasks by dividing them into natural groups that reflect salient patterns?

The Flurry dataset introduced in Section 3.2 is used for conducting the in-breadth understanding of complex mobile tasks. Given the best task boundary detection approach (Table 5.6) proposed in Chapter 5, the app usage logs could first be separated based on the detected boundaries; we then only keep those tasks with at least two different apps (i.e., complex tasks) for our further analysis. We first conduct a comprehensive quantitative study on characterizing the complex tasks based on the Flurry dataset. A generic mobile app navigation model is proposed to present an accurate picture for the micro-level interactions within this analysis, including how users revisit and switch between different apps.

Afterwards, we further investigate if there are common patterns that exist among the complex mobile tasks. As shown in figure 6.1, we can observe that the complex tasks are ever-changing corresponding to the different

apps involved. However, if we look at the higher-level common intentions behind these complex tasks, we can find that Task 3 and Task 9 are all related to doing micro-works on smartphones. Knowing and understanding the common usage patterns/types and recurrent complex tasks could help us to understand the users' needs, better interpret their feedback, and capture the requirements. So besides identifying the specific complex tasks users conducted, we also want to infer the common types behind those complex tasks. An unsupervised learning framework is proposed to cluster all the complex tasks into different groups based on their extracted characteristics. By clustering the complex tasks based on the extracted characteristics, we provide evidence that there actually exist 17 common tasks with 47 sub-tasks, which could be identified solely from their salient properties. Those tasks range from information check, micro documentation work and family entertainments. Our proposed unsupervised learning framework is rigorously validated through a wide set of metrics and statistical measures.

The remainder of this chapter is organized as follows. In Section 6.1, we characterize the mobile complex tasks from three aspects: task context, task complexity, and task content based on the proposed app-stream navigation model. In section 6.2, we employ the unsupervised learning approach to derive generic profiles of these complex mobile tasks. We conclude this chapter in Section 6.3.

6.1 Characterization Approach: App-stream Navigation Model

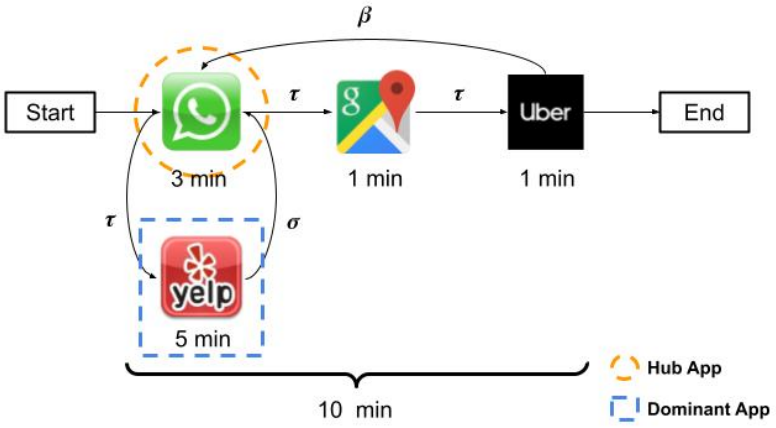
Given the available features we could extract from our app usage logs, we propose to characterize the mobile complex tasks from three aspects:

task context, task complexity, and task content. For the task context, the hour of day and day of week would be used to indicate when the task happens. In the search tasks, the number of queries, amount of websites visited and completion time are always used to measure the task complexity [103, 104]. Similarly, we also propose to use the number of apps/app categories involved, the number of app transitions and task completion time to measure our mobile task complexity. For the task content, it should include what tasks are about and how tasks are completed [97]. To characterize the task content, besides the basic topics of this task, e.g. app categories involved, we propose to capture the relationships among the apps accessed sequentially for measuring how tasks are completed. For example, such important characteristics include the "switch" interaction, which captures how users switch between two apps frequently to accomplish some complex tasks [26]. The different roles played by each app in the task is also worthwhile to pursue. For example, the "hub app", which users always revisit after interacting with other apps in a task, may provide more hints on the user intents. For most app usage logs, user interactions are ordered by timestamps and the accessed apps as shown in Figure 6.2a. To capture these types of navigation and provide additional insights on the metrics that measure task content, we introduce an app-stream navigation model in this chapter.

Similar to the Labelled Transition System (LTS) for modeling user navigation in visiting the websites [176], which uses the click-stream model to capture user actions, the *app-stream* is understood to consist of a series of accesses of apps in a complex task. The user simply opens an app, then leaves that app after some interactions and then visits another app. The transition will be labelled a $\phi - transition$: we interpret the user to undertake a transition $p \xrightarrow{\phi} q$ from app p to app q. We are now able to define an

Time	App	App Category
12:49:45	WhatsApp	Communication
12:50:10	Yelp	Food & Drink
12:55:15	WhatsApp	Communication
13:05:10	Google Maps	Navigation
13:05:43	Uber	Transportation
13:06:17	WhatsApp	Communication

(a) App Usage Logs



(b) App-stream Navigation Model

Figure 6.2: App-stream Navigation Model (Hub App: Whatsapp; Dominant App: Yelp; App Switch: Whatsapp -> Yelp -> Whatsapp; Task Completion Time: 10 min)

LTS describing the app-stream. Let $AS = (\Sigma, \Lambda)$ be a labelled transition system describing app-stream with Σ: The set of states comprising all apps in the task, along with two start ($\top$) and ending ($\bot$) states; Λ: This is the set of actions ϕ the user can take.

6.1.1 Transition Types

To capture the micro-interaction, such as switching apps back and forth within the task, we differentiate some transitions in our *LTS*. These transition labels are:

Opening a new app (τ): A user can choose to open a new app (an app that has not been visited by the user in this task) from the current state.

Going back to the last app (σ): A user accesses (revisits) the last app a that she/he just visited prior to the current app b ($a \neq b$).

Going back to in-task visited app (β): A user accesses the app that has been visited previously in this task, but it is not the last app (revisitations that exclude the *App Switch*).

All the transition types are illustrated in Figure 6.2b with the Task 7 for booking lunch in Figure 6.1. We define the transition between WhatsApp and Yelp, which could be tracked by σ transitions, as *App Switch* behaviours.

6.1.2 Hub Apps

In each complex task, we also want to extract the apps, "hub app", which user always go back to in a task. This could help us identify whether there are specific apps - or even specific publishers - that are particularly effective at attracting and engaging switching between other apps. For instance, users may check the calendar app while replying email, where the email app is the "central" app users are interacting with. Consumers typically generate a greater amount of β or σ transitions to the "hub" app, continually revisiting the "hub" after briefly accessing other apps. We use H_s score to extract the "hub app" from each task, where H_s is the summation on frequency of σ and β transitions in the same task of each app. And the "hub app" H_A will be extracted to be the app that has highest H_s in the task, which is defined as follows:

$$H_s \ = \ |\{ \ a_\phi \mid a_\phi \ = \ \sigma \text{ or } \beta\}| \tag{6.1}$$

$$H\_A = \{\ a \mid H_s(a) \ = \ max\left(H_s(a_i), ..., H_s(a_n)\right),\ a_i \in \Sigma\} \qquad (6.2)$$

Since the tasks with no revisiting transitions will be regarded as tasks in which no "hub app" exists, $H_s(a)$ must be greater than 0. If there are multiple "hub apps" with the same highest H_s, we extract all of them as the "hub apps" of this task. In the example illustrated in Figure 6.2b, the "hub app" is WhatsApp, since the WhatsApp has the highest H_s score, which is 2.

6.1.3 Dominant Apps

Besides measuring the revisited frequency, the time spent on each app within a task should also be taken into consideration. We focus on distinguishing the time spent patterns by identifying if one app usage always occupied most of the time or multiple apps are uniformly used within a complex task. We then extract the app that occupied more than 50% of the total duration of a complex task as the "dominant app", e.g., the "Yelp" app in the lunch booking task (Figure 6.2b).

In summary, we characterize all the complex mobile tasks from three facets as shown in Table 6.1: context, complexity, and content. Each facet is measured based on different features, which could be extracted from the app usage logs given our proposed app-stream navigation model. In the last column, we illustrate the values of all the features by the task in Figure 6.2b. In the following sections, we uncover the various characteristics of complex tasks based on our proposed app-stream navigation model and the large-scale Flurry dataset (Section 3.2).

Table 6.1: Characteristics of complex mobile tasks captured in this study.

Facets	**Features**	**Values** (illustrated by the task in Figure 6.2b)
Context	Hour of day	12
	Day of week	Saturday
Complexity	Number of unique apps	4
	Number of unique app categories	4
	Number of app transitions	5
	Task completion time	10 min
Content*	Apps involved	Communication, Food-and-Drink, Navigation, Transportation
	Hub app	Communication
	Dominant app	Food-and-Drink
	First app	Communication
	App Switch	Communication -> Food-and-Drink -> Communication

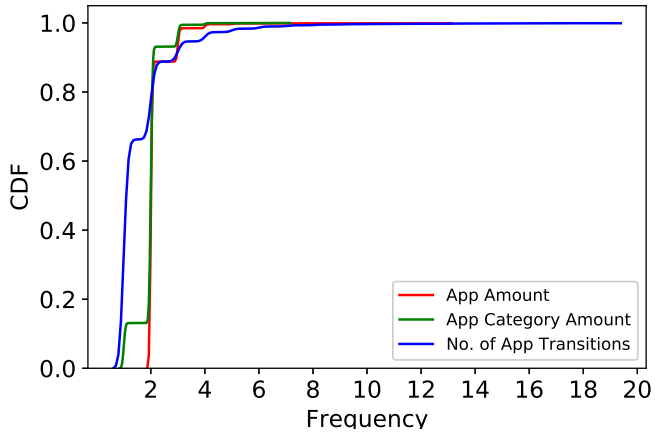
*: the task content is measured based on the app category level to avoid the sparsity issue.

6.1.4 Characteristics of Complex Mobile Tasks

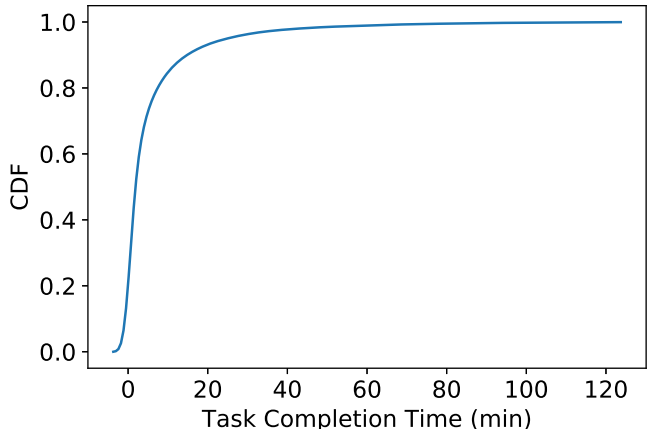
Since the temporal distribution (task context) of complex tasks follow the general temporal pattern of app usage, which grew from 6 am and reached its first peak around 11 am, and were most active during the evening (7 pm to 9 pm) [191, 94], we mainly display the characteristics of task complexity and task content in this section. More specific temporal patterns corresponding to the different complex tasks are explored in Section 6.2.

6.1.5 Task Complexity

We extract four features for capturing the complexity of mobile tasks: app amount, app category amount, number of app transitions and task completion time. The CDF (Cumulative Distribution Function) of them are plotted in Figure 6.3, where we can find that mostly there are fewer than four different apps/app categories involved in the complex tasks with fewer than eight times of app transitions. Additionally, 90% of the complex tasks could be completed in 20 minutes.



(a)



(b)

Figure 6.3: CDF of app amount, app category amount, number of app transition, and task completion time in complex tasks.

App Transitions. Other than the number of apps/app categories involved and total time spent, which have been used as the traditional statistics to characterize an app usage session previously [16], we introduce a new feature of mobile tasks to capture the characteristics of apps accessed, *app transition*, based on our proposed app-stream navigation model. Given a task, the *app-stream* is a time-ordered list of apps accessed by the user within this task; *app transition* is defined as the transition between any two sequential different apps in this *app-stream*. To further explore the correlation between the number of app transitions and the two other gen-

Table 6.2: Distribution of app transition types.

Transition Type	**% Transitions**
Open a new app (τ)	71%
Switch to the last app (σ)	26.7%
Access the previous app (β)	2.3%

eral metrics (app amount involved and task completion time), we calculate the Pearson coefficient between them. We find that no correlation exists between the number of app transitions and task completion time ($\rho = 0.041$), which means that more times of transitions between apps does not indicate longer time spent; a weak positive correlation exists between the number of distinct apps used and the number of app transitions ($\rho = 0.297$), which indicates that more apps involved could lead more potential app transitions among them.

Table 6.2 shows the distribution of transition types in all complex tasks. We can find that over 30% of the transitions in complex tasks are about accessing the apps visited before in the same task. Interestingly, the transition type σ, *switch* between two apps, occupies most of the revisitation behaviour. Thus we may hypothesize that most of the time, users revisit the previous apps to resume the previous unfinished tasks. We will further analyze the *app switch* in the followings.

6.1.6 Task Content.

Hub Apps: Do hub apps always exist in complex tasks? What kinds of apps have higher probabilities to become hub apps? With the definition of "hub app", we extract all the hub apps for each complex task. We find that the "hub app" exists in about 35.3% of complex tasks. Table 6.3 shows popular hub app categories which are the ones with higher

probabilities P_h to become "hub app" when compared to the others:

$$P_h(a) = \frac{F_h(a)}{F(a)}, \tag{6.3}$$

where $F(a)$ means the frequency of complex tasks that have app category a involved, $F_h(a)$ indicates that frequency of the complex tasks that have app category a served as the *hub* app. We can find that the app categories, like productivity, personalization, social and games are relatively popular *hub* apps. Therefore, the mobile operating system may improve the app recommendation services by promoting these *hub* apps on the top of the suggested ranking list or in the quick tab bar when users want to revisit previous apps (double press the home button in iOS). This can facilitate users' access to their interested apps more efficiently when completing complex mobile tasks.

Table 6.3: Popular hub/dominant/first app categories in complex tasks.

Hub	**Prob.** P_h	**Dominant**	**Prob.** P_d	**First**	**Prob.** P_f
productivity	0.14	casino	0.61	navigation	0.50
personalization	0.13	social	0.59	comics	0.50
comics	0.12	card	0.57	books	0.45
music	0.12	puzzle	0.57	utilities	0.44
social	0.11	games	0.56	games	0.44
games	0.11	word	0.55	news	0.42
transportation	0.10	family	0.54	lifestyle	0.42
tools	0.10	arcade	0.52	music	0.42
books	0.09	video	0.52	food-and-drinks	0.41
utilities	0.08	board	0.52	weather	0.41

Dominant Apps: Is there an app always occupied most of the time in each complex task? What kinds of apps have higher probabilities to be the dominant apps? To identify if the dominant app exists, we first extract all the app categories which have the longest time spent in each complex task. We find that 95% of the complex tasks are conducted based on one dominant app category (which occupies more than 50% of the time in a task). This suggests that even users are engaging with multiple

apps together, most of the time they will only engage with a specific app category rather than spending time evenly on different app categories. To find the app categories that are more likely to become the dominant app in each complex task, we extract the top app categories with the highest probability P_d to become the dominant app category in Table 6.3. The P_d is calculated in the similar way of P_h:

$$P_d(a) = \frac{F_d(a)}{F(a)}, \tag{6.4}$$

where $F(a)$ means the frequency of complex tasks that have app category a involved, $F_d(a)$ indicates that frequency of the complex tasks that have app category a works as the *dominant* app. We find that the social, game and video apps are much more likely to become the dominant app category, e.g., casino, social, card, puzzle, games and video, etc. Additionally, the less overlap between these popular dominant apps and hub apps in Table 6.3 also inspires us that the apps users always visited in a complex task may not be the one with most of time spent.

First Apps: What kinds of apps are always the first app in complex tasks? Within the search tasks, researchers [125] found that the first query plays an important role in inferring the task types. Therefore, under our mobile task scenarios, we also want to explore the apps that have a higher possibility to act as the first app and if it could bring any insights for the whole task. We first calculate the probability that if an app category occurs in a complex task, how likely it is the first app user visits in this task by:

$$P_f(a) = \frac{F_f(a)}{F(a)}, \tag{6.5}$$

where $F_f(a)$ indicates the frequency of complex tasks that have app category a works as the first app. The results are shown in Table 6.3. We can

Table 6.4: Popular app categories involved in switch (A->B->A).

Popular A in App Switch	**Probability** P_{sa}	**Popular B in App Switch**	**Probability** p_{sb}
health-and-fitness	0.31	widgets	0.32
personalization	0.27	health-and-fitness	0.32
productivity	0.23	transportation	0.24
tools	0.19	personalization	0.21
music	0.18	communication	0.21
social	0.18	tools	0.18
transportation	0.17	catalogs	0.18
communication	0.15	music	0.16
word	0.13	productivity	0.15
sports	0.13	sports	0.15

find that the navigation, comics, books, utilities and games apps are more likely than other app categories to trigger a complex task.

How likely does the "first app" become the "hub app" /"dominant app" in a complex task? Furthermore, we want to explore whether the first app accessed by users is also the "hub app" or "dominant app" in each complex task. We then observe that for the tasks that have "hub apps", over 95% of the cases, the first app is the same as the "hub app"; for the tasks that have the "dominant apps", 57% of them have the first app as same as the dominant app. Therefore we may hypothesize that when users have a focus while interacting within the complex mobile tasks, they are more likely to start the task by that focus, the *hub* app. Meanwhile, when predicting which app is the *hub* app within the current complex task, the first app user accessed could be a good initial choice for the real-time recommendation.

App Switch: is "app switch" as same as app co-occurrence? Understanding the switch interaction is important as it has more significant implications. For example, if Facebook Messenger sees that people frequently switch back and forth to launch Uber to hire a car, then Facebook Messenger might want to consider integrating Uber (i.e. a riding service)

Table 6.5: Popular app switches (A->B->A).

Popular App Switch	**Probability** P_{sw}
health-and-fitness -> productivity -> health-and-fitness	0.66
productivity -> health-and-fitness-> productivity	0.66
education->education->education	0.59
productivity -> personalization -> productivity	0.57
personalization -> productivity -> personalization	0.53
transportation -> communication -> transportation	0.51
productivity -> widgets -> productivity	0.50
productivity -> communication -> productivity	0.45
personalization -> tools -> personalization	0.45
music -> social -> music	0.41

into their own app. Therefore, the switch is important to be taken into consideration for task mining, task-multiplicity and co-marketing partnerships between apps. Some previous works [16, 187] have reported the co-occurrence of two apps in an app sequence. To identify the differences between the switch and co-occurrence between two apps, we compare the frequency distributions of switch and co-occurrences of all the app pairs that appeared in the same complex task with *Kolmogorov-Smirnov* test and find that, indeed, there is a significant difference in these two engagement patterns ($p\text{-}value<0.01$). This implies that only simply counting co-occurrences is not sufficient to capture the switch behaviours between apps (as there are directional patterns within the app transitions). Modelling complex tasks and their applications require incorporating more fine-grained interactions.

App Switch: What kinds of apps have higher probabilities to be involved in "app switch"? Table 6.4 displays the apps with higher probabilities to be involved in the app switch when compared with other apps. Since there are two positions in an *app switch* (A->B->A), we calculate the probabilities for apps that would be involved in these two positions A

and B respectively as P_{sa} and P_{sb}:

$$P_{sa} = \frac{F_{sa}(a)}{F(a)}, \tag{6.6}$$

$$P_{sb} = \frac{F_{sb}(a)}{F(a)}, \tag{6.7}$$

where the $F(a)$ is the frequency of complex tasks that have app category a involved. Then $F_{sa}(a)$ and $F_{sb}(a)$ indicate the frequency of complex tasks that have app category a involved in an *app switch* in the position A and B respectively. We can observe that when users access the apps such as the health-and-fitness app, they will be more likely (31%) to switch to other apps and then back to these apps again. It is also interesting to observe that users are more likely to switch to widget apps (e.g. notification comes when engaging with other apps). Additionally, we also find that the popular apps accessed in the two positions (A and B) of the switch (A->B->A) are similar (e.g. health-and-fitness app). This may imply that most of the times, these apps may need more co-operations with other apps to help users finish their tasks on mobile devices. We are also interested in the probability that when two apps are visited sequentially (A and B), how likely a σ transition (back to A) will occur next. We extract the most popular app switches in the complex tasks in Table 6.5 by ranking the probability P_{sw}:

$$P_{sw}(a, b) = \frac{F_{sw}(a, b)}{F(a, b)}, \tag{6.8}$$

where $F(a, b)$ means the frequency of complex tasks that have app category a and app category b sequentially accessed. Then $F_{sw}(a, b)$ represents the frequency of complex tasks that after accessing app category a and b, the user also switch back to a. These popular app switches could provide app developers with more insights into app designs for improving user experiences. For example, from the popular switches between health-and-fitness

and productivity, we find that when users access the health-and-fitness and productivity sequentially, i.e. checking email during doing exercises, users have a higher probability to switch between them. These findings may help the app developers to add additional functions to their apps, like mail bulletin in fitness apps.

6.2 Complex Task Clustering

By analyzing the characteristics of complex tasks from a large-scale dataset, we have gained insights of a large spectrum of complex tasks. In this section, we aim to understand the complex tasks by uncovering whether there are common patterns that exist in them based on our extracted characteristics of the tasks. The main objective of this analysis is to divide the complex tasks into natural groups that reflect salient patterns. It can create the taxonomy for mapping users' complex tasks into different types, which could help to provide more satisfying mobile services to users. To this end, we employed unsupervised learning to derive generic profiles of these complex mobile tasks.

6.2.1 Clustering Approach

6.2.1.1 Complex Task Representation in Clustering.

Based on the characteristics of complex tasks extracted in Section 6.1, we represent each complex task using the features listed in Table 6.6 for a total of 231 dimensions. Therefore, each complex task could be represented with three types of features: context, complexity and content. Specifically, they consist of the hour of the day and day of the week the task happened,

Table 6.6: Features used for complex tasks clustering.

Feature	**Facets**	**Dimension**	**Description**	**Values** (illustrated by the task in Figure 6.2b)
H	Context	1	Hour of the day (0 - 23)	12
W	Context	1	Day of the week (Monday - Sunday)	Saturday
N_{ac}	Complexity	1	Number of distinct app categories	4
N_a	Complexity	1	Number of distinct apps	4
N_{at}	Complexity	1	Number of app transitions	5
D	Complexity	1	Task duration (min)	10 min
A	Content	45	Binary vector of app categories to indicate their presence and absence	Communication, Food-and-Drink, Navigation, Transportation
F_a	Content	45	Binary vector of first app category to indicate its presence and absence	Communication
H_a	Content	45	Binary vector of hub app category to indicate its presence and absence	Communication
S_a	Content	45	Binary vector of app categories to indicate their presence and absence in switch units (a->b->a)	Communication, Food-and-Drink
D_a	Content	45	Binary vector of dominant app category to indicate its presence and absence	Food-and-Drink

distinct app categories amount, distinct apps amount, number of app transitions, task duration, app categories involved, hub app category, first app category, switch app categories, and dominant app category within the task. Take the task in Figure 6.2b as an example, the task representation based on its characteristics could depict this specific complex task as it happened at 12 p.m. on Saturday; it had 4 distinct apps with 4 app categories, which were Communication, Food-and-Drink, Navigation and Transportation; it was started by Communication apps usage; it lasted for 10 minutes and dominated by Food-and-Drink apps for 50% of the total task duration. Besides, users have 5 times of app transitions, whereas mostly transitions are about revisiting the hub app — Communication. The switch between Communication and Food-and-Drink apps also existed within this complex task.

6.2.1.2 Clustering Algorithms

Given that there is no ground truth for how many different kinds of complex tasks exist, the outcome of a clustering process is not (always) deterministic and may result in a different partitioning of the data, depending on the specific criteria used (e.g. the number of clusters and type of clustering algorithm). Furthermore, we do not know *a-priori* which clustering algorithm and configuration settings will perform better for the examined domain, so we aim to opt for several representative options. More specifically, we considered centroid-, hierarchical-, and density-based algorithms. Additionally, features we used for complex tasks clustering comprise both numeric (e.g., number of distinct apps, task duration, etc.) and categorical (e.g., hub app category, switch app categories, etc.) features. However, most of the popular clustering algorithms are developed for pure numeric data (for example K-means [110]) since the default "similarity" measures

for grouping data points into clusters is Euclidean distance, which is only suitable for numeric values.

To avoid the common pitfall of resorting to a single clustering algorithm that may not be able to produce proper partitions and also overcome the issues for mixed data types features, we select the clustering algorithm that could be applied to mixed data types for each of the centroid-, hierarchical- and density-based algorithms [3]. *Centroid-based* clustering algorithms drive the notion of similarity by the closeness of a data point to the centroid of the clusters. These algorithms run iteratively to find the local optima and the number of clusters required in advance. Within the centroid-based clustering algorithms, K-prototypes [74, 73] is an extension of K-means for clustering large data sets with mixed numeric and categorical values, whereas new representations of cluster centres and a new definition of distance between a data point and a cluster centre are proposed for mixed datasets (i.e., cluster centres are represented by mean values for numeric features and mode values for categorical features). *Hierarchical-based* clustering algorithms work by forming an initial pair of clusters and then recursively consider whether it is worth splitting/merging each one further; this type of clustering algorithms can be represented as a binary tree (i.e., dendrogram). To deal with the mixed data types in clustering, Philip and Ottaway [140] use Gower's similarity measure [58] to compute the similarity matrix for features in the hierarchical agglomerative clustering. Gower's similarity measure computes the similarity by dividing features into two subsets: one for categorical features and the other for numeric features. *Density-based* clustering algorithms search the data space for areas of the varied density of data points in the data space. They isolate various density regions and assign the data points within these regions in the same cluster. Liu et al. [106] propose a density-based clustering algorithm for mixed

Table 6.7: Summary of Clustering Algorithm Parameters

Category	**Algorithm**	**Parameters**	**Value**
Centroid	K-prototype[74, 73]	Clusters	[2,30]
Hierarchical	Agglomerative [140]	Clusters	[2,30]
		Linkage Type	ward, complete, average
		Affinity	gower's distance
Density	DBSCAN[106]	Neighborhood size	[1e-3, 2e-3, 3e-3, 4e-3, 5e-3, 1e-2, 2e-2, 3e-2, 4e-2, 5e-2]
		Min samples	[5, 10, 50, 100, 300]

datasets. The authors extend DBSCAN [44] algorithm, where entropy is used to compute the distance measure for mixed datasets. Table 6.7 shows the clustering algorithms and the ranges of values we used to parameterize them. The set of clustering algorithms $\times$ their parameter settings (see Table 6.7) results in 217 possible clustering configurations.

6.2.2 Clustering Evaluation

In order to evaluate the quality of the clustering results from the different configurations used, the internal validation measures often reflect the compactness, the connectedness and the separation of the cluster partitions [107], especially when there is no ground truth of the clustering results. Zhao et al. [209] stated that although metrics for measuring clustering performance exist (e.g., Dunn's index [114]), they can not penalize both for complexity and non-uniform distribution of data points across clusters. Therefore, Shannon's entropy [142] is used to reward the clustering performance in their work. They defined a clustering performance (cp) score by weighing four factors. The first and second factors are used to reward the clustering performance using two well-known metrics: Shannon's entropy (E) and Dunn's index. The third and the fourth factors are for penalizing clustering results: since they used the k-means-MeanShift hy-

brid clustering method, so they penalize the results that do not improve over k-means results where the number of final clusters from MeanShift is close to the number of clusters found by K-means; they also penalize the results with non-uniform distribution of items across clusters: particularly those in which the biggest cluster contains most of the items in the dataset. Since our features for clustering is mixed data types and we opt to use multiple clustering algorithms for meta-evaluation, we are not able to use the hybrid clustering method they proposed. Therefore, we removed the third factor in their *cp* score equation, which is specifically defined for their k-means-MeanShift hybrid method. We use the following *cp* score for measuring the clustering performance of different clustering algorithms:

$$cp = 0.3E + 0.23D + 0.23\frac{N-n}{N}. \tag{6.9}$$

The probability used for calculating Shannon's entropy (E) score is the normalized number of tasks in each cluster. Thus, entropy assigns a high value to clustering results that have a uniform distribution of tasks across clusters. Dunn's index (D), on the other hand, is the internal validity that measures the compactness and separation of the clusters obtained. The last factor is for penalizing clustering results that the biggest cluster contains most of the tasks in the dataset, where N is the number of items input to clustering, n is the number of items in the biggest cluster after clustering. By combining these three factors, we guarantee that the resulting clusters are compact, well separated and the number of tasks in the biggest clusters is well distributed (i.e., a single cluster does not contain most of the tasks).

By using the *cp* score in Eq. 6.9, and trying all the configurations of clustering algorithms in Table 6.7, we obtained the best clustering algorithm with the highest *cp* score, which is the hierarchical agglomerative cluster-

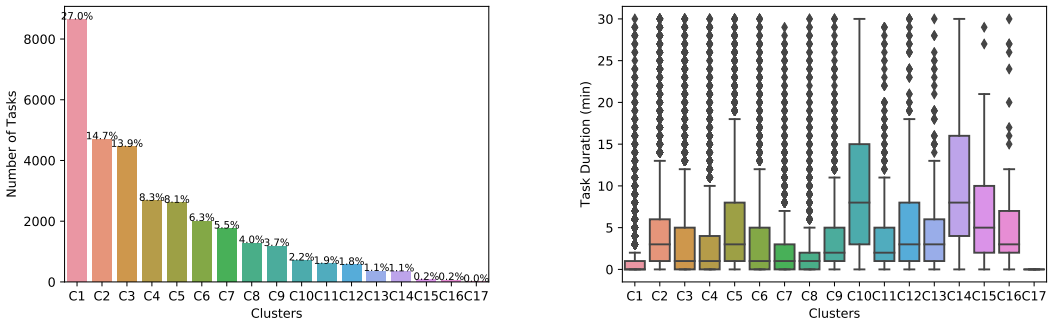
Table 6.8: Meta-evaluating Results of Clustering Algorithms

Category	**Algorithm**	**Config.**	*cp* **score**
Centroid	K-prototype	K = 30	0.403
		K = 26	0.386
		K = 23	0.299
Hierarchical	Agglomerative	K = 17, Linkage = Complete	**0.474**
		K = 30, Linkage = Complete	0.448
		K = 16, Linkage = Complete	0.444
Density	DBSCAN	Neighborhood size = 3e-2, min samples = 5	0.265
		Neighborhood size = 3e-2, min samples = 10	0.265
		Neighborhood size = 1e-2, min samples = 300	0.118

ing algorithm result in 17 clusters with *complete* linkage type. We need to note that, since the three factors E, D and $\frac{N-n}{N}$ have different scales based on our clustering results (e.g., $E \in (0, 3.8)$ and $D \in (0.05, 0.3)$), we scale each of them by the Z-score method to obtain the normalized results. We also list the top-ranked results of each category of algorithms with their corresponding configurations respectively in Table 6.8. We can find that the hierarchical- and centroid-based clustering algorithms perform better than the density-based algorithm for our task.

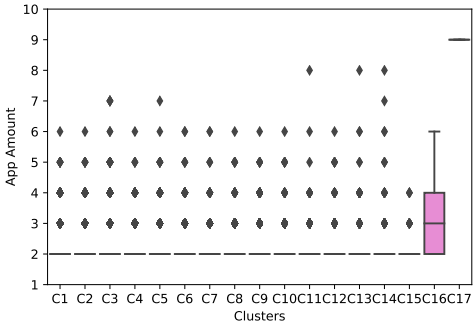
6.2.3 Cluster Analysis

Ultimately, we obtained 17 clusters (types of complex tasks) based on the best performing Hierarchical Agglomerative clustering algorithm (Table 6.8). Figure 6.4a shows the distribution of clusters with respect to its number of data points (i.e., tasks). As we can see, there are 7 clusters consisting of more than 5% of data points. The biggest cluster constitutes 27% of data points. For most of the features related to task complexity (numeric), we display the average value of the corresponding feature for the data points of each cluster in Figure 6.4b - Figure 6.4d. We can find

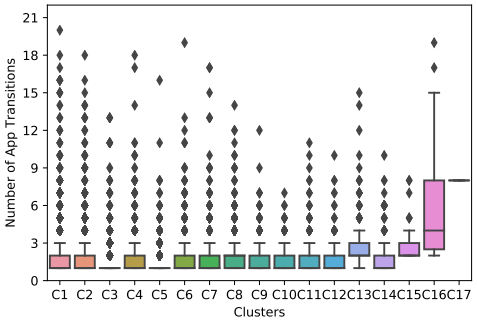


(a) Task Amount

(b) Task Completion Time



(c) App Amount



(d) Number of App Transitions

Figure 6.4: Basic statistics of 17 common clusters (color shown in the figures is only used to distinguish different clusters).

that the values of most complexity features are strikingly similar across all the clusters (e.g., the number of apps/transitions), which makes it hard to visually distinguish them. More differences exist in the task completion time of all clusters, where we can observe that the bigger clusters with more data points are mostly with shorter task completion time. To appropriately describe the differences among the clusters, especially with those categorical features (which cannot be aggregated based on the mean value), we performed a pattern selection step to find the most salient pattern which could distinguish a given cluster from all the others. Then we analyze the clusters based on those selected salient patterns.

6.2.3.1 Salient Pattern Criteria

To better understand the clusters created by our method, we need to select a few salient patterns that could represent the characteristics of each cluster. In general, we want to find the distinctive patterns that could distinguish the cluster from the others. To identify these distinctive characteristics, we use two selection strategies according to the different feature types: (1) *mean*$\pm$*2std* for complexity and content features; (2) *paired t-test p value* for context features.

- *mean*$\pm$*2std* (complexity and content features): The characteristic of a cluster is a special quality or trait that makes it different from others. Thus, one's characteristics usually are not very close to the average of all the clusters. For each complexity and content feature, we assume that the probability distribution of feature values follow the Gaussian distribution. Then for each cluster, we aim to focus on those features with values far away from *mean* of the feature (more than two standard deviations (*std*)), i.e. lying outside the interval of

mean±2std [212], which may be good recognizable characteristics for the cluster.

For a numeric feature F, we firstly normalize all the values by Z-score and then get the interval of *mean±2std* from the normalized feature value of tasks of all clusters directly. Then we compare the mean feature value of each cluster to the general distribution of this feature for identifying if it is a salient feature of this cluster (i.e., falling in or out of the interval of *mean±2std*). For example, if the feature F is task duration (D) and the average task duration of Cluster 1 is 1.6 minutes which is shorter than mean-2std of all clusters, then the task duration of 1.6 minutes will be selected as a salient characteristic for representing Cluster 1. Otherwise, if the task duration of a cluster i falls in the interval of *mean±2std*, it will not be selected to be shown as a salient pattern feature for representing this cluster, which means this cluster has the *normal* task duration, similar to all other clusters.

For a categorical feature F, since we cannot get the distribution of the feature value directly, we model the distribution as the fraction of this categorical feature value appears in each cluster. For example, when the feature F is hub app category (H_a), i.e., Social apps, we will first get the fraction it appears as a hub app in each cluster. E.g., the Social apps have 100 times as the hub apps in all the clusters, which include 12 times (12%) in Cluster 2, 67 times (67%) in Cluster 4, 10 times (10%) in Cluster 5, 8 times (8%) in Cluster 7 and 3 times (3%) in Cluster 11. Then we normalize this fraction distribution across all the clusters and find that the hub app category Social is a salient pattern for Cluster 4 since its normalized fraction value is bigger than *mean+2std* of fraction distribution for all the clusters.

- paired t-test p-value (context features): The temporal (hour and

weekday) feature is a special case which is kind of the mix of numeric and categorical since even they are discrete values, they are also ordered. We firstly get the hour/day frequency distribution (Min-Max scaled) of all tasks as the general pattern of all clusters. Then for each Cluster i, we also get the hour/day frequency distribution (Min-Max scaled) of this cluster. Lastly, we compare the distribution of each Cluster i to the general distribution based on the paired t-test to identify if it could be selected as a salient pattern of Cluster i. When its distribution is significantly different from the general distribution (p-value < 0.001), the hour/weekday temporal pattern of the Cluster i is selected as a salient pattern.

6.2.3.2 Analyzing Clusters based on Selected Salient Characteristics.

In this section, we examine the characteristics of each cluster based on the selected salient patterns. Since the finalized best clustering algorithm for our problem is hierarchical agglomerative clustering, we are able to further split the 17 clusters into 47 sub-clusters (which gets the best *cp* score with the number of clusters under the limits of 50) to dig into the detailed characteristics of the big clusters. We show the salient patterns of each cluster in Table 6.9 - 6.12. Before we analyze the clusters, we need to note that, firstly, the clusters are organized according to how many data points (tasks) are in them, with Cluster 1 (C1) representing the biggest cluster. Secondly, due to the limited space, we only list the top one or two values of each salient pattern in Table 6.9 - 6.12, which are selected based on the criteria defined in Section 6.2.3.1; the symbol "-" means there is no salient pattern for this feature of this cluster. Lastly, since the number of app

Table 6.9: Salient characteristics (Distinctive Hub, Dominant, Switch, Sequence, Duration and Temporal Pattern) of clusters: Part I

Cluster #17	**Subcluster #47** Criteria	**Hub** mean±2*std*	**Dominant** mean±2*std*	**Switch** mean±2*std*	**Sequence** mean±2*std*	**Duration** mean±2*std*	**Temporal** p-value<0.001	**Task Description**
C1 (27%) Multiple Updated Info Check	**C1 (27%)** Multiple Updated Info Check	productivity, widgets, No hub, finance, weather	productivity, widgets, communication, personalization	productivity->weather->productivity, wdigets->productivity->widgets	productivity->finance->weather, productivity->communication->productivity	1.6 min ⊖	-	Check the latest information together, e.g. skim the email (productivity), stock (finance), and weather information quickly.
C2 (14.7%) Social Media Browsing	**C2 (11.2%)** Social Media Browsing	No hub	social	productivity->social->productivity	social->social, social->social->social->social	-	-	Browse among multiple social network apps.
	C9 (3.4%) Social Sharing	social	-	social->communication->social, social->photography->social	social->communication->social, social->photography->social	-	-	Sharing/Uploading info while browsing social media.
C3 (13.9%) One App Oriented Tasks with Info Check at Any Time	**C5 (5.2%)** Info Check after Entertainment	-	reference, comics, travel, entertainment	-	reference->weather, comics->communication, entertainment->news	-	-	After the media entertainment (e.g., books, comics, TVs), users step to obtain some related information.
	C13 (1.9%) Weather Oriented Info Check	weather	weather	weather->travel->weather	weather->social, weather->news	2.3 min ⊖	-	Tasks based on the weather info, e.g., check the weather info within future days for deciding the travelling date.
	C18 (1.3%) Info Check for Exercises	health-and-fitness	health-and-fitness	health->tools->health	navigation->health->weather, health->navigation->weather	-	-	Check the navigation and weather information for running outside.
	C21 (1.1%) Games with Info Check	games	games	games->games->games, games->news->games	games->games, games->weather	-	-	Info check while playing games.
	C25 (0.8%) Info Check while Books Reading	books	books	books->communication->books, books->news->books	books->productivity, books->finance	-	-	Related info check while books reading.
	C26 (0.8%) Info Gathering for Dining Out	transportation, navigation, food	food, navigation, transportation	transportation->productivity->transportation	transportation->productivity, food->productivity	-	-	Multiple info needed for dining out. E.g. check the detailed address info of the booked restaurant from order confirmation email (productivity) for navigation/transportation.
	C27 (0.7%) Finance Oriented Info Gathering	finance	finance	finance->news->finance	finance->productivity, finance->news	-	-	Check the stock trend with updated news to get some hints.

*: this cluster has more app amounts/app categories/app transitions.
⊖: this cluster has relatively shorter task completion time (duration), which falls into the range $> mean + 2std$ within the duration distribution of all tasks.
⊕: this cluster has relatively longer task completion time (duration), which falls into the range $< mean - 2std$ within the duration distribution of all tasks.
P: popular time range of this cluster compared with averaged temporal distribution of all tasks.
NP: unpopular time range of this cluster compared with averaged temporal distribution of all tasks.

Table 6.10: Salient characteristics (Distinctive Hub,Dominant,Switch, Sequence, Duration and Temporal Pattern) of clusters: Part II

Cluster #17	**Subcluster #47** Criteria	**Hub** mean±2*std*	**Dominant** mean±2*std*	**Switch** mean±2*std*	**Sequence** mean±2*std*	**Duration** mean±2*std*	**Temporal** p-value<0.001	**Task Description**
C3 (13.9%) One App Oriented Tasks with Info Check at Any Time	**C27 (0.7%)** Finance Oriented Info Gathering	finance	finance	finance->news->finance	finance->productivity, finance->news	-	-	Check the stock trend with updated news to get some hints.
	C29 (0.7%) Action Games with Notifications Check	action	action	action->productivity->action	action->productivity, action->social	-	-	Notifications replying while playing action games.
	C30 (0.5%) Info Requirement while Mobile Meeting	business	business	business->travel->business, business->productivity->business	business->travel->business, business->tools	-	-	Info needed during mobile meeting, e.g., check documentation or book flights.
	C32 (0.4%) Racing Games with Info Check	racing	racing	racing->widgets->racing	racing->widgets->racing, racing->weather	-	P: 6 pm - 9 pm NP: 11 am - 4 pm	Info check while playing racing games.
	C36 (0.3%) Photography with Discussion and Sharing	photography	-	photography->personalization->photography, photography->communication->photography	photography->communication->photography, phogotraphy->personalization->photography	-	P: 9 am, 1 pm and 7 pm	Discuss or share photos with friends while editing.
	C40 (0.1%) Shortly Interrupted by Notifications	widgets	-	widgets->social->widgets, widgets->communication->widgets	widgets->social->widgets, widgets->communication->widgets	2.4 min $^{\ominus}$	P: 3pm	Interrupted by the pop-up notifications and proceed to check the social media or communication apps in a short time.
	C42 (0.1%) Reference Info Supported Tasks	reference	-	-	reference->productivity, reference->health		P: 7 am and 10 pm	Check the dictionary for some uncommonly used information while engaging (learning) with other Apps.
	C44 (0.1%) Travel Related Info Check	travel	-	travel->social->travel, travel->productivity->travel	travel->productivity->travel, travel->social->travel	-	P: 7 am and 13 pm	Info check during travelling or planning for travel.
	C45 (0.0%)	-	-	-	-	19.8 min $^{\oplus}$	-	
C4 (8.3%) Communication Supported Tasks	**C3 (6.1%)** Auxiliaries Needed Communication	communication	communication	communication->weather->communication, communication->tools->communication	communication->tools->communication, communication->social->communication	-	-	Some auxiliaries are needed while communication with friends (e.g. check weather while discussing the plan for an outing).
	C19 (1.2%) Card Games with Communication	card	card	card->communication->card	card->communication->card, card->communication	14.9 min $^{\oplus}$	-	Intense card games playing with friends.
	C22 (1.1%) Shopping Discussion	shopping	shopping	shopping->communication->shopping	shopping->communication->shopping, shopping->communication	-	P: 12 pm and 10 pm - 0 am	Have some discussions with friends while online shopping.

*: this cluster has more app amounts/app categories/app transitions.
$\ominus$: this cluster has relatively shorter task completion time (duration), which falls into the range $> \ mean + 2std$ within the duration distribution of all tasks.
$\oplus$: this cluster has relatively longer task completion time (duration), which falls into the range $< \ mean - 2std$ within the duration distribution of all tasks.
P: popular time range of this cluster compared with averaged temporal distribution of all tasks.
NP: unpopular time range of this cluster compared with averaged temporal distribution of all tasks.

Table 6.11: Salient characteristics (Distinctive Hub,Dominant,Switch, Sequence, Duration and Temporal Pattern) of clusters: Part III

Cluster #17	**Subcluster #47** Criteria	**Hub** mean±2*std*	**Dominant** mean±2*std*	**Switch** mean±2*std*	**Sequence** mean±2*std*	**Duration** mean±2*std*	**Temporal** p-value<0.001	**Task Description**
C5 (8.1%) Multiple Games Playing	**C7 (3.7%)** Arcade Games Dominated Entertainment	-	arcade	-	arcade->entertainment, arcade->sports	-	-	Playing Arcade games and other games together.
	C8 (3.5%) Multiple Sports Games	sports	sports	sports->sports->sports, sports->arcade->sports	sports->sports, sports->arcade	-	NP: 6 am - 12 pm	Playing sports and arcade games together.
	C33 (0.3%) Family Entertainment	family	family	family->arcade->family, family->casual->family	family->entertainment, family->arcade->family	-	P: 6pm-8pm NP: 9 pm - 10 am	Family fun time with games and TVs.
	C34 (0.3%) Morning Multiple Games	arcade	-	arcade->simulation- >arcade, arcade->sports->arcade	arcade->arcade->arcade, arcade->entertainment- >arcade	-	P: 8 am - 2 pm NP: afternoon and evening	Playing multiple games during morning time.
	C37 (0.2%) Casual and Arcade Games	casual	casual	casual->arcade->casual, arcade->casual->arcade	casual->arcade, casual->card	-	P: 1 am and 3 pm	Playing casual and arcade games together.
C6 (6.3%) Socializing Involved with Others and Music	**C4 (5.4%)** Others Apps with Socializing	others	others	others->social->others, others->music->others	others->social, social->others->social- >others, others->social->others	-	P:8pm -11 pm NP: 9am - 7pm	Interleaved by socializing or music but mainly engage with others Apps at night.
	C23 (0.8%) Music with Socializing and Others	music	music	music->social->music, music->others->music	music->social->music, music->others->music	-	P: 1 am -3 am NP: 10 am- 3pm	Interleaved by socializing or others but mainly engage with listening to the music at late night.
C7 (5.5%) Complex Tools Usage	**C6 (3.7%)** Tools Cooperation	-	-	-	tools->tools, tools->utilities	-	-	Some tasks need to use multiple tools to complete, e.g. use Calculator and record the results in the Notes.
	C15 (1.8%) Mobile Settings	personalization	personalization	personalization->tools- >personalization, personalization->social- >personalization	personalization->tools- >personalization, personalization->tools	-	P: 11 am and 7-8 pm NP: 12 pm - 6 pm	Manage mobile settings with tools.
C8 (4%) Doc/Photo/Video Editing	**C11 (2.2%)** Micro Documentation Work	-	tools	tools->productivity->tools, productivity->tools- >productivity	communication- >productivity->tools- >productivity, productivity->tools- >productivity->tools- >productivity	2.3 min $^{\ominus}$	-	Reply email or edit documents with communication and auxiliary tools.
	C16 (1.8%) Video/Photo Editing	tools, utilities	-	tools<->photography- >tools	tools->photography- >tools, tools->video->tools	-	-	Edit photos or videos with auxiliary tools.
C9 (3.7%) News Reading	**C10 (2.6%)** Multiple News Source Browsing	news	news	news->news->news, news->communication- >news	news->news, news->communication	-	-	Browse news from multiple platforms.
	C20 (1.1%) Redirect to News	-	-	productivity->news- >productivity	productivity->news, productivity->news- >productivity	-	P: 0 am - 7 am and 7 pm NP: 12 pm and 8 pm-11 pm	Redirect to the news via email links.

*: this cluster has more app amounts/app categories/app transitions.
$\ominus$: this cluster has relatively shorter task completion time (duration), which falls into the range $> mean + 2std$ within the duration distribution of all tasks.
$\oplus$: this cluster has relatively longer task completion time (duration), which falls into the range $< mean - 2std$ within the duration distribution of all tasks.
P: popular time range of this cluster compared with averaged temporal distribution of all tasks.
NP: unpopular time range of this cluster compared with averaged temporal distribution of all tasks.

Table 6.12: Salient characteristics (Distinctive Hub,Dominant,Switch, Sequence, Duration and Temporal Pattern) of clusters: Part IV

Cluster #17	**Subcluster #47** Criteria	**Hub** mean±2*std*	**Dominant** mean±2*std*	**Switch** mean±2*std*	**Sequence** mean±2*std*	**Duration** mean±2*std*	**Temporal** p-value<0.001	**Task Description**
C10 (2.2%) Puzzle Games Time	**C12 (2.0%)** Multiple Puzzle Games	-	puzzle	books->puzzle->books, communication->puzzle->communication	puzzle->puzzle, puzzle->books	-	P: 1 pm, 8 pm, 10 pm and 0 am - 6 am	Puzzle games playing at specific times.
	C39 (0.2%) Intense Puzzle Games with Auxiliaries	puzzle	-	puzzle->productivity->puzzle, puzzle->tools->puzzle	puzzle->productivity->puzzle, puzzle->tools->puzzle	24.2 min $^{\oplus}$	P: 6 pm NP: morning, noon and evening	Intense puzzle games playing with auxiliaries (e.g., notes taking).
C11 (1.9%) Relaxation with Multiple Lifestyle Apps	**C14 (1.8%)** Relaxation with Multiple Lifestyle Apps	lifestyle	lifestyle	lifestyle->lifestyle->lifestyle, lifestyle->communication->lifestyle	lifestyle->lifestyle, lifestyle->lifestyle->lifestyle	-	-	Lifestyle apps used with other apps.
	C46* (0.0%)	-	-	-	-	23.3 min $^{\oplus}$	P: 4 pm	
C12 (1.8%) Video Watching Time	**C17 (1.8%)** Video Watching Time	video	video	video->video->video, video->entertainment->video	video->video->video, video->entertainment	-	-	Video watching time (e.g. switch between multiple video platforms or between videos and TVs).
C13 (1.1%) TV Watching with Socializing and Communication	**C28 (0.7%)** TV Watching with Socializing	-	-	entertainment->social->entertainment	entertainment->social->entertainment, entertainment->social, social->entertainment->social->entertainment	-	P: 6 pm and 9 pm - 2 am NP: 9 am -11 am and 1 pm -3 pm	Check the updated social media notifications while watching TVs.
	C31 (0.5%) TV Watching with Communication	entertainment	-	entertainment->communication->entertainment, communication->entertainment->communication	entertainment->communication->entertainment->communication, entertainment->communication->entertainment	-	P: 9 pm, NP: morning and afternoon	Reply to messages while watching TVs.
C14 (1.1%) Long Time Games Playing at Night	**C24 (0.8%)** Long Time Word Games Playing	word	word	word->tools->word, word->social->word	word->word, word->social	15.3 min $^{\oplus}$	P:9pm-10pm, NP: morning and afternoon	Long time word games playing with socializing or auxiliary tools at night.
	C35 (0.3%) Long Time Board Games Playing	board	board	board->board->board	board->board, board->board->board, board->board->board->board	19.4 min $^{\oplus}$	P: 10pm-0am NP: morning	Long time multiple board games playing at night.
C15 (0.2%) Notes Taking with Other Info Resources	**C38 (0.2%)** Notes Taking with Other Info Resources	utilities	-	utilities->news->utilities, utilities->lifestyle->utilities	utilities->lifestyle->utilities, utilities->news->utilities	-	P:11 am NP: all other times	Take notes to record some important news or items.
C16* (0.2%) Tasks with Complex Topic and Operations	**C41* (0.1%)** Complex Communication Topic Needs Auxiliaries	-	-	personalization->communication->personalization, communication->personalization->communication	personalization->communication->personalization->communication->personalization	-	P: 11 am	Complex communication topic needs auxiliaries (e.g. launcher search bar/browser).
	C43* (0.1%) Complex Discussion Needed during lifestyle	-	-	communication->lifestyle->communication	communication->lifestyle->communication->lifestyle, lifestyle->communication->lifestyle->communication	-	P: 4pm	Frequently switch between lifestyle and communication Apps to discuss some complex topic.
C17* (0.0%)	C47* (0.0%)	-	-	-	-	0 min $^{\ominus}$	-	

*: this cluster has more app amounts/app categories/app transitions.
$\ominus$: this cluster has relatively shorter task completion time (duration), which falls into the range $> mean + 2std$ within the duration distribution of all tasks.
$\oplus$: this cluster has relatively longer task completion time (duration), which falls into the range $< mean - 2std$ within the duration distribution of all tasks.
P: popular time range of this cluster compared with averaged temporal distribution of all tasks.
NP: unpopular time range of this cluster compared with averaged temporal distribution of all tasks.

categories/apps/transitions (feature N_{ac}, N_a, and N_{at}) are similar for most of clusters (as shown in Figure 6.4), we do not show them separately and only mark the clusters with salient values of these features with "$_*$" on the cluster name in Table 6.9-6.12. Based on the distinctive characteristics (i.e., hub app category, dominant app category, switch units and sequences, etc.) of each cluster, we also summarize each cluster with a task label under the cluster number (e.g., we label the biggest Cluster 1[2] as "Multiple Updated Info Check") and a task description in the last column.

From Table 6.9-6.12, we can observe that there exist several common trends of complex mobile tasks from the 47 clusters characteristics:

- *Gathering the real-time information every now and then.* From the top-ranked clusters (C1 and C3), we can find that *gathering the latest information and updates* is a great demand for mobile users to conduct complex tasks. Almost one-third of the complex tasks are performed since users tend to browse multiple information sources together on smartphones to obtain the latest information (C1). These tasks are all very short, lasting only about 1.6 min, during which users may try to skim the latest mails (productivity), stock (finance) and weather information sequentially. Additionally, other than gathering information from multiple sources in one go, this information gathering behaviour also occurs while users are intensively engaging (i.e., the hub and dominant app of the task is the same app) with another app (C3). Examples include: checking news while watching TVs (C3-5), checking the navigation and weather information before going outside for running/cycling (C3-18) and checking flight

[2]Since we have clusters with two kinds of granularities, we use the form of C# to refer the original clusters (17 in total), e.g., C2; we use the form of C#-# to refer to the sub-clusters (47 in total), e.g., C2-9, where the first number is the original cluster number (C2) and the second number is the sub-cluster number (C9).

tickets during online meetings (C3-30). Furthermore, most of these information-gathering complex tasks have no specific temporal pattern, which means users usually perform those types of tasks every now and then.

- *Social or communication dominated complex tasks are popular.* It is widely known that social networking and messaging apps are popular on smartphones [45, 16, 40]. We find that these apps are not only popular for independent usage, they are also engaged in conjunction with multiple other apps together, occupying about 30% of the complex tasks (C2, C4, C6, C13, and C16). We observe that users would intensively browse social networking apps (C2) or communicate with friends (C4). For the social networking dominated complex tasks, most of them are performed because users tend to browse multiple social networking apps together. For the communication dominated complex tasks, other apps are mostly involved as the auxiliaries to facilitate the discussion (e.g., info search). These kinds of social or communication oriented complex tasks could happen all day, which have no specific temporal patterns. Besides, social apps are usually engaged with other apps during specific times, e.g., browsing social media apps while listening to music during the night (C6), checking the social networking apps while watching TV during the evening and night (C13-28). Communication apps are also frequently used within many types of other apps as a supportive channel, e.g., communicating with friends for making shopping decisions (C4-22), playing cards (C4-19) or discussing some complex topics (C16).

- *Different forms of game playing with different game types.* There are three different kinds of game-playing dominated complex tasks (C5, C10, and C14), which account for 11.4% of the complex tasks.

Firstly, for Arcade, Sports and Casual games, users prefer to engage with them in addition to other different games together (C5). For Puzzle games, users mostly only focus on the same category of puzzle games, but auxiliaries are always needed to figure out the puzzle (C10). Similarly, for word and board games, users prefer to only engage with the same kinds of games, but for a longer period of time (C14). Additionally, for all of these game-playing dominated complex tasks, they mostly have specific temporal patterns, e.g., intensive (long-time) word/board game-playing usually happen during night time (C14).

- *Micro-work tasks at any time.* Smartphones are more likely to be used as a "pocket mobile computer" nowadays, whereas diverse tool/utility apps are installed for helping users to complete micro-work tasks. However, many tools and utility apps are designed only with functionalities that focus on a single dimension. Some complex tasks are conducted given cooperation between multiple tool apps are required when completing the micro-work tasks (C7-6), e.g., calculation with note-taking (with Calculator and Notes app). Taking notes on different information sources (C15) are also usually performed. Additionally, users may also use multiple tool apps to complete more advanced personalized mobile settings (e.g., task management) (C7-15). Other than those basic tool cooperation oriented complex tasks, users increasingly prefer to use the smartphone for conducting quick documentation works (C8-11), e.g., replying to email or editing documents (with auxiliary tools) while communicating with colleagues. Editing the videos and photos with auxiliary tools on smartphones (C8-16) is also a growing trend. Most of these micro-work complex tasks have no specific temporal patterns, which means users would

conduct them at any time.

- *Media reading from multiple sources.* The last common pattern for generating complex tasks is media reading. We can observe that, even with one media format, users prefer to engage with multiple sources together, e.g, reading news from multiple different news platforms (C9), engaging with multiple lifestyle apps (C11) and watching videos from different video sites or switch between video and TV sites (C12). Furthermore, it is interesting to find that different from the engagement pattern of video sites, while watching TVs, users would focus on one TV app and did not switch between different TV sources. However, they will be distracted by social media or message notifications (C13).

Our clustering results shed lights on fine-grained patterns of how users perform common complex mobile tasks in the wild.

6.3 Discussion and Conclusion

Users increasingly prefer to complete tasks with mobile devices in their daily life. While modern mobile devices only effectively serve many of the individual apps that correspond to simple mobile needs, users get little or no help when their needs transcend the boundary of a single mobile app. In this chapter, by employing the automatic identification of mobile tasks (chapter 5) in a large-scale dataset, we focus on understanding the complex tasks which are more time-consuming and difficult for users to complete. The complex tasks are characterized from three aspects, including context, content and complexity. Given the extracted characteristics for representing complex tasks, we show evidence that there actually exist several dif-

ferentiable groups of complex tasks, which could be identified solely from their salient properties. It provides us with a fine-grained picture of the common complex mobile tasks users perform in the wild.

6.3.1 Implications

Currently, the cognitive burden of keeping track of complex mobile tasks is placed on the user. The tasks with high-level intentions that span across multiple apps/services should be taken into consideration. Smartphone manufacturers, app developers and anyone who impact the way how apps are engaged on phones, which apps are used and how people select apps to execute, should no longer treat apps independently. Based on the mobile tasks we observed, we elaborate on the implications of our research as below.

Smartphone manufacturers can build smartphones with operating systems that could provide an intelligent switching interface towards improving the user experience. As we have discovered in our work, mobile users have common requirements for engaging with multiple apps together, thereby creating a need for an efficient app switching mechanism. If the switching interface can be improved by absorbing the concept of shared intents in within app switching as well as optimizing the layout design to support navigation between recently used apps, there should be a possibility to enhance the efficiency and user experience for both the workflow and fragmented attention usage on the smartphone. In our work, we have found that some apps have higher possibilities to serve as the "hub" app (has the highest possibility to be switched back to) during a complex task (Section 6.1.2). This feature can be taken into consideration when predicting which app will be switched to next, rather than always popping up the app

most recently used. Additionally, the "hub" app, as the communication app in C4 (Communication Supported Tasks), could be set as the background or positioned within the central view port when designing the layout of the switch panel.

Screen management apps could help users manage their apps in a more efficient way based on the typical complex tasks found in this chapter. Rather than managing apps solely based on their categories, they can be organized based on tasks, taking into account users' contextual information (e.g., time and location). For example, the C10-39 (Intense Puzzle Games with Auxiliaries) always has a higher possibility to happen in the evening. Therefore, the puzzle and notepad, which are originally from two app categories, could be placed closer for ease of access during this time. Furthermore, app developers should also be thinking about improving their services in terms of co-operations with other apps, which is important to be taken into consideration for co-marketing partnerships between apps. For example, we find that users prefer to switch between communication and card apps in C4-19 (Card Games with Communication). App developers may want to add an additional function to their card apps, like a communication bulletin, or messages pop-up. Additionally, the app developers should also improve their services by making it easy to complete any mobile tasks in a single app, which means the design of applications should be self-sufficient. For example, we find that users prefer to access mail, finance, news and weather apps sequentially when doing the morning regular check (C1 - Multiple Updated Info Check). The app developers may need to think about designing an information integration app for helping users obtain the information more efficiently. Some interesting patterns of different information sources could also be aggregated before showing to users, e.g., the relationship between the trend of the stock market and the

latest news.

Here we summarize the task understanding part of this thesis (Part II). In this part, we are the first to propose the study for understanding users' app usage behaviour within the task space and we provide a comprehensive picture of mobile tasks from identifying, analyzing and clustering tasks. Since the extracted tasks from log activity data reveal more detailed intentions and behaviour patterns for mobile users, especially when compared with independent apps, we believe that the mined knowledge of tasks could be leveraged in different scenarios for improving mobile services and user experience. Therefore, we aim to investigate whether and how the task information could be leveraged in different applications in the following part (Part III).

Part III

Leveraging Mobile Task Information

Chapter 7

Inferring Users' Demographics from Mobile Tasks

Part II of this thesis we described above aims to extract and characterize mobile tasks based on users' app usage behaviour. We then aim to investigate how the extracted mobile task information could be leveraged for improving different applications. In the existing research, app interests are traditionally used in constructing mobile user representation, specifically aims at demographics prediction. In this chapter, by leveraging the extracted task information, we aim to validate that if the task-based user representation, which reveals more detailed user intentions and behaviour patterns, could bring additional insights in demographics prediction. We aim to answer the research questions **RQ 3.1** and **RQ 3.2**, as specified in Section 1.1.

RQ 3.1: could we represent users not only based on app interests (used apps), but also on tasks, or task types?

RQ 3.2: would the task-based user representation methods benefit demo-

graphics prediction?

Demographic attributes have been demonstrated as important information in personalization [68], web search [180] and advertisement targeting [96], which can help learning models to infer users' interests and hence improve the performance of applications and services. For example, with the gender information of online users, the advertisers can display dress Ads to female users and shaver Ads to male users. Without the demographic information of users advertisers may show retirement insurance Ads to a teenager user and lipstick Ads to male user, which may be not effective. Additionally, demographic segmentation is the most common type of market segmentation strategy since the basic demographics offer the most common and easy information to interpret statistics that can be used to group entire populations [11]. For example, demographic information is easy to be obtained from many government-maintained census data (available to the public) and free online survey tools, which makes it a great way to monitor societal trends and shifts over time as the data categories and criteria rarely change. Identifying trends can help brands to track, monitor and analyse the customer journey, and can also help make market predictions for the future, for the benefit of both the brand and the consumer.

Predicting demographics from mobile apps have been studied for several years in both data mining and mobile computing fields [19, 199, 130, 213, 95, 133, 210]. However, the techniques used for mobile user representation and modeling are rather straightforward, such as simply using app lists as features [154, 111], without considering more detailed semantic relationship between apps. Tasks have steadily emerged as a more accurate unit to capture users' goal and behavioural insights [60]. Researchers in the web search area have validated that accurate task representations (i.e., "search tasks" – consist of a set of queries corresponding to a particular high-level

information need, and the queries are not necessarily the same or even similar) could be useful in aptly placing the user in the task-subtask space and enable systems to contextually target the user, provide them better query suggestions, personalization, recommendations and help in gauging satisfaction [52, 120, 118].

In this chapter, by leveraging the best-performing task identification model (task boundary detection) proposed in the Chapter 5, we are able to map large-scale app usage logs (Flurry dataset introduced in Section 3.2) to mobile tasks, which consisting of millions of mobile logs from over 10,000 users. Based on the extracted tasks, we observed that even users engaged with similar apps, the different tasks conducted can provide additional insights for distinguishing users' demographics. For instance, within the tasks conducted for taking notes while reading news, where users mostly focus on searching the specific information from reading news, the amount of female users involved is significantly bigger than male users. Differently, within the task for browsing news from multiple resources or users may always be redirected to news reading from other resources, e.g., redirect from email/docs (productivity) or communication apps, the involved male users are significantly more than the female users. Therefore, we believe that tasks play an important role in understanding user's needs and can be leveraged to develop better representations of users' characteristics. Additionally, similar to each app could be assigned to an app category, we could also map each extracted specific task into a task category based on the clustering framework proposed in Chapter 6. Finally, by exploring the different approaches (straightforward, embedding-based and neural-based methods) for generating users' representation based on the traditional app/app category and proposed task/task category units, we validate that the task-based approach performs best for inferring users' representation,

which could effectively improve the performance of gender and age group prediction when compared to all other baseline methods. By demonstrating the task-based user profiling method could work better in demographics prediction, especially when compared with the traditional app-based user representation, we provided the initial steps in shaping future research on investigating whether and how the extracted tasks could be applied for improving mobile user profiling. We believe that task-based applications could be well developed in the future for providing users with more advanced and satisfying services in different scenarios.

The remainder of this chapter is organized as follows. Section 7.1 discusses the prior work. We present the app and task preferences of users in different age/gender group with the category level in Section 7.2. In Section 7.3, we present the different methods to generate user representation aims at demographics prediction. Details of the experiment set-up and results are reported in Section 7.4. In Section 7.5, we discuss the ethical ramifications of our work. We conclude the work in Section 7.6.

7.1 Prior Work

Existing methods for demographic prediction are usually based on popular apps used, number of apps in different app categories, costs in apps, and app description [19, 199, 130, 213, 154, 95, 111, 133, 210], with different machine learning techniques. Table 7.1 presents the summary of these previous works on demographics prediction for mobile users.

We can find that a major portion of existing work has investigated user representation using app or app category units as the fundamental focus of demographics prediction. Only one recent work [210] proposed to infer

Table 7.1: Summary of Demographics Prediction based on Mobile App Interactions

Ref.	Brdar et al. [19]	Ying et al. [199]	Nadeem et al. [130]	Zhong et al. [213]	Seneviratne et al. [154]	Li et al. [95]	Malmi et al. [111]	Neal et al. [133]	Zhao et al. [210]	**Our Work**
Year	2012	2012	2012	2013	2015	2016	2016	2018	2019	2021
#Users	80	185	80	78	218	28,158	3760	189	10,000	11,713
App Related Features	app events count	number of app category, total app usage frequency	number of apps, duration of top app usage	top apps, duration of top apps	costs in apps, app, app category, app description	app, app category	app, app category	top apps	top apps, app category, app sequence	app, app category, **task** (a set of apps usage corresponding to a particular high-level user need), **task category**
Method	KNN,Radial Basis Function Network, and Random Forest (RF)	Multi-Level Classification Model	Bagging Tree Ensemble	Gradient Boosted Decision Tree (GBDT),RF, SVM, and Logistic Regression (LR)	Naive Bayes (NB), SVM	Decision Tree, Perception, SVM, NB, KNN, RF	SVM, LR	SVM, RF, Naive Bayes	GBDT, LR, SVM, Deep Neural Network (DNN)	LR, RF, **CNN**[1], **LSTM**[1], **HAN**[1] and **HURA**[1]
Target	gender, age, marital status, job, children	gender, age, marital status, job, children	gender, age, marital status, job, children	gender, age, marital status, job, children	gender	gender	gender, age, race, marital status, children, income	gender	gender, income	gender, age

[1] CNN: Convolutional Neural Network; LSTM: Long Short Term Memory;
HAN: Hierarchical Attention Networ; HURA: Hierarchical User Representation with Attention.

users' demographics based on Doc2Vec [88] method, where the relationships between apps are considered in generating the user representation. They treated each user as a document and each app as a word. The user representation vector is obtained when the user and app vectors are updated until convergence and then is fed to the classifiers (e.g., GBDT, LR, SVM and DNN) for demographics prediction directly. In our work, we focus on leveraging the semantic information and correlation between app usage instead of treating apps independently. Other than the app and app categories, new units like tasks and task categories would be involved for modelling users. We hypothesize that the mobile tasks which contain more detailed app usage patterns would be more accurate for capturing users' characteristics and benefit the demographics prediction.

In this chapter, we model mobile users based on task representations, which could include both the apps information and the semantic meaning among app usage to provide more detailed user characteristics implied. Wu et al. [186] proposed to apply a hierarchical neural-based network in learning more enhanced user representation and validated that it can effectively improve the performance of demographics prediction. In our work, we adopt such state-of-the-art models for learning user representation in the context of mobile apps. We hypothesized that different tasks have different informativeness for demographics. Instead of combining all tasks from the same user into a long vector for user representation, we experiment with models that automatically extract key features for demographics prediction. Adapting such models for mobile apps are not trivial given we try to model the app, task and user within the model.

7.2 Demographic Differences in App Usage Behaviour

The demographic differences in terms of popularity in apps usage have been explored by previous works, e.g., the top preferred apps for female/male users [211, 111]. In this section, besides exploring the app preferences of users in different age/gender group, we further analyse the demographic differences in complex mobile tasks within the app usage behaviour. The analysis and experiements conducted in this chapter is based on the Flurry dataset introduced in Section 3.2.

7.2.1 App Category Preference

More female users are found in apps that improve their health and lifestyles than male users, while more males are involved in apps with more professional functionalities than female users. In order to discover the gender differences in app category preference, we calculated the distribution of female and male users for 45 app categories respectively based on our dataset, e.g., for the set of users U who used the food-and-drink apps, we calculate the percentage of how many female (U_f) and male (U_m) users exist in them by $P_f = \frac{|U_f|}{|U|}$ and $P_m = \frac{|U_m|}{|U|}$. Then we display the app categories with the highest percentage (P_f or P_m) in Table 7.2, which means they are mostly preferred by the users in specific gender group. We can find that, within the apps to improve health and make users enjoy their lives, like food-and-drinks, medical, health-and-fitness, video, music and lifestyle apps, etc., more female users are found than male users. While within the that have more professional functionalities or skills, e.g., personalization (mobile setting) tools, transportation,

Table 7.2: Top 10 app categories preferred by each gender group.

App Category	% Female Users	App Category	% Male Users
food-and-drink	78.2%	strategy	74.0%
word	77.4%	sports	72.5%
medical	74.5%	personalization	67.0%
health-and-fitness	73.3%	tools	65.3%
casino	71.4%	adventure	65.2%
board	70.9%	transportation	64.9%
video	64.7%	business	63.5%
music	64.6%	action	63.4%
education	64.1%	news	63.1%
lifestyle	59.5%	arcade	60.3%

business and news, etc., male users are relatively more than female users. Additionally, different kinds of games also show different statistics of male and female users, e.g., within the games of word, casino, and board, female users are more than male users, but within the strategy, adventure, action and arcade games, male users are more involved than female users.

Younger (<24) users are found more than other users in games and music. New workers (25-34) are found more than other users in transportation, shopping, lifestyle and communication apps. Middle-age users (35-54) are more involved in functional apps. Order users (55+) are more involved in card, board, and word games. Similarly, to discover the age differences in app category preference, we calculated the percentage of different age group users for 45 app categories respectively based on our dataset. The top app categories preferred by each age group are reported in Table 7.3.

We can find that within the music and game apps, younger (13 - 24) users are significantly more than users in other age groups. Other than that, within the apps for learning (i.e., education apps), teenagers (13 - 18) are found more than other users. For the apps like food-and-drink, video and social apps, more users between 18 and 24 years old are found since these apps might be more helpful when they have just entered the college/univer-

Table 7.3: Top 5 app categories preferred by each age group.

App Category	% 13-17	App Category	% 18-24	App Category	% 25-34
board	20.8%	role-playing	36.2%	comics	54.8%
music	19.7%	food-and-drink	30.5%	transportation	50.5%
games	19.5%	video	28.4%	shopping	44.9%
education	16.5%	music	28.3%	lifestyle	41.0%
word	16.2%	social	26.1%	communication	40.9%
App Category	**% 35-54**	**App Category**	**% 55+**		
adventure	59.1%	casino	22.7%		
tools	50.7%	card	20.8%		
news	50.1%	board	18.8%		
widgets	50.1%	word	16.2%		
family	49.2%	weather	14.3%		

sities. Users between 25 and 34, who are mostly the newcomers just stepped into their career and started to earn money by themselves, are found more than other users in the apps like transportation, shopping, lifestyle and communication apps, which all work for their daily life. The distribution of users in functional apps (e.g., tools, news, widgets) shows that users between 35 and 54 are the most. Within the apps for playing card, board, and word games, older users (55+) are found significantly more than other users, which may be because they have more spare time for leisure life.

7.2.2 Discriminatory Complex Task Categories

We then explore the demographic differences in mobile tasks. We run the best-performing task identification model (task boundary detection proposed in Chapter 5) on the app usage logs to extract all the apps usage belonging to the same task, where a classifier based on temporal, similarity and sequential features is applied to identify task boundaries between each sequential pair of app usage. Therefore, the app usage logs of each user are segmented into different mobile tasks and these extracted tasks could be classified into two kinds of mobile tasks: single-app tasks (e.g., spotify->spotify) and complex (multi-app) tasks (e.g.,

Table 7.4: Complex task categories with their corresponding characteristics (i.e., popularity, popular app sequences, and task duration). Chi-square denote the discriminatory power of the task category for inferring user's gender and age respectively. The bigger the standardized residual (bigger residuals are marked as bold) of the specific gender/age group is, more users in this gender/age group exist in this task category.

Complex Task Types Characteristics				Standardized Residual							Chi-square	
%Popularity	**Task Categories**	**Popular App Sequences**	**Duration**	**female**	**male**	**13 - 17**	**18 - 24**	**25 - 34**	**35 - 54**	**55+**	**Gender**	**Age**
27.0% C1	Multiple Updated Info Check	productivity->finance->weather	1.6 min	-2.7	**2.5**	-6.1	-4.2	1.5	**4.3**	-1.3	13.7	37.3
11.2% C2	Social Media Browsing	social->social	5.2 min	-3.1	**3.0**	-0.8	**2.3**	**3.0**	-2.4	-3.5	18.7	0.7
6.1% C3	Communication Needs Other Auxiliaries	Communication->Tools->Communication	4.9 min	-2.2	**2.1**	-3.9	-4.2	**3.4**	1.8	-1.7	9.6	15.1
5.4% C4	Others Apps with Socializing	Others->Social->Others	6.4 min	-0.8	0.8	1.3	-0.2	-0.2	0.4	-1.5	1.3	1.6
5.2% C5	Related Info Check during Entertainment	Entertainment -> News	5.1 min	1.3	-1.2	-0.0	-0.7	-1.8	1.3	2.0	3.2	0.0
3.7% C6	Tools Cooperation	Tools->utilities	4.3 min	-3.1	**2.9**	-0.5	-0.4	-1.0	1.9	-1.5	18.2	0.2
3.7% C7	Arcade Games Dominated Entertainment	Arcade->Entertainment	7.6 min	-4.3	**4.1**	1.2	0.2	-0.5	1.3	-3.8	35.0	1.4
3.5% C8	Multiple Sports Games	Sports->Sports->Sports	6.9 min	-7.2	**6.8**	-0.9	-1.8	-1.7	**2.9**	0.2	97.3	0.8
3.4% C9	Interrupted Social Media Browsing	Social->Communication->Social	10.9 min	-1.5	1.4	-1.1	**2.3**	**2.7**	-2.6	-2.0	4.2	1.1
2.6% C10	Multiple News Source Browsing	News->News->News	5.8 min	-3.2	**3.0**	-3.3	-3.9	-2.3	**3.7**	**5.4**	19.2	10.7
2.2% C11	Mobile Micro Work (e.g. reply to email)	Communication->Productivity->Tools->Productivity	2.3 min	-1.7	1.6	-1.1	-2.7	-0.2	2.4	-0.3	5.5	1.2
2.0% C12	Multiple Puzzle Games	Puzzle->Puzzle	13 min	**3.4**	-3.3	1.1	-2.0	-1.5	0.4	4.3	22.4	1.3
1.9% C13	Weather Oriented Info Check	Weather->News	2.3 min	-0.1	0.1	-2.7	-4.2	-4.0	**5.0**	**5.9**	0.1	7.4
1.8% C14	Relaxation with Multiple Lifestyle App	Lifestyle->Lifestyle	6 min	**2.1**	-2.0	-2.0	-1.6	**4.7**	-1.6	-2.0	8.2	4.1
1.8% C15	Mobile Settings	Personalization->Tools->Personalization	3.1 min	-2.3	**2.1**	0.3	1.2	-0.5	0.0	-1.2	9.6	0.0
1.8% C16	Video/Photo Editing	tools->Photography->Tools	5.9 min	-1.7	1.6	-1.1	-0.5	-0.6	1.6	-0.9	5.6	1.1
1.8% C17	Switch between Multiple Videos/TVs	Video->Entertainment->Video	8.3 min	**2.9**	-2.7	**3.3**	**4.6**	0.4	-3.3	-3.1	16.0	11.0
1.3% C18	Info Check for Exercises	Navigation->Heath-and-fitness->Weather	2.9 min	**3.6**	-3.4	-0.4	-0.2	0.1	0.5	-0.7	24.9	0.2
1.2% C19	Card Games with Communication	Card->Communication->Card	14.9 min	0.5	-0.5	-3.0	-2.2	-3.4	1.9	**9.9**	0.5	9.1
1.1% C20	Redirect to News Reading	Productivity->News->Productivity	2.9 min	-3.6	**3.4**	-1.9	-2.3	-3.4	**4.7**	1.4	24.4	3.8
1.1% C21	Info Check while Playing Games	Games->Weather	12.4 min	**6.6**	-6.2	**5.3**	1.5	-4.8	-2.8	**10.1**	81.7	27.6
1.1% C22	Shopping Discussion	Shopping->Communication->Shopping	3.9 min	0.2	-0.2	-2.9	-2.5	**6.3**	-2.1	-2.3	0.1	8.5
0.8% C23	Music with Socializing	Music->Social->Music	6.9 min	**2.3**	-2.1	**6.1**	**5.9**	-0.9	-4.3	-2.7	9.8	36.8
0.8% C24	Long Time Word Games Playing	Word->Word	15.3 min	**3.3**	-3.1	2.5	1.1	-3.1	-0.6	4.4	20.6	6.0
0.8% C25	Info Check while Reading Books	Books->Communication->Books	4.6 min	0.5	-0.5	-1.5	-1.7	-1.2	2.0	1.8	0.5	2.2
0.8% C26	Info Check for Dining Out	Transportation/Food->productivity	7.6 min	1.1	-1.1	-1.7	1.3	-1.6	0.6	2.2	2.5	2.8
0.7% C27	Finance Oriented Info Check	Finance->News->Finance	2.8 min	-0.8	0.7	-2.6	-1.3	1.6	0.3	0.5	1.2	6.9
0.7% C28	TV Watching with Socialization	Entertainment->Social->Entertainment	7.5 min	-2.0	**1.9**	0.5	1.8	0.1	-0.8	-1.5	7.9	0.3
0.7% C29	Action Games with Info Check	Action->Productivity->Action	9.7 min	-2.2	**2.1**	1.6	-1.2	-0.8	1.2	-1.1	9.0	2.5
0.5% C30	Info Check Supported for Mobile Meeting	Business->Travel->Business	3.7 min	-1.1	1.2	-1.1	-0.3	0.7	0.7	-1.9	2.2	1.3
0.5% C31	TV Watching with Communication	Entertainment->Communication->Entertainment	7.9 min	-1.0	0.9	-1.0	-1.2	-0.2	0.8	1.2	1.9	1.0
0.4% C32	Racing Games with Info Check	Racing->Widgets->Racing	9.1 min	0.1	-0.1	**2.6**	-1.0	-0.4	0.5	-1.6	0.0	6.8
0.3% C33	Family Entertainment	Family->Entertainment	8.9 min	0.0	-0.0	1.5	-1.1	-0.4	1.1	-1.8	0.0	2.3
0.3% C34	Morning Multiple Games	Arcade->Simulation->Arcade	11 min	-1.5	1.4	-0.3	1.5	-1.0	0.6	-1.3	4.3	0.1
0.3% C35	Long Time Board Games Playing	Board->Board->Board	19.4 min	**2.5**	-2.4	**2.7**	-1.5	-1.1	-1.2	**5.2**	12.0	7.1
0.3% C36	Photography with Info Check	Photo->Communication->Photo	4.1 min	0.2	-0.2	-0.7	0.2	1.7	-0.9	-1.0	0.1	0.5
0.2% C37	Casual and Arcade Games	Casual->Arcade->Casual	11.8 min	0.1	-0.1	0.6	-0.6	-0.2	0.3	-0.1	0.0	0.4
0.2% C38	Notes Taking with Info Resources	Utilites->News->Utilites	9.1 min	**2.8**	-2.6	0.0	-0.7	0.5	-0.6	1.5	14.4	0.0
0.2% C39	Intense Puzzle Games with Auxiliaries	Puzzle->Tools->Puzzle	24.2 min	**2.6**	-2.5	1.0	-2.2	-0.7	0.4	3.1	13.2	1.0
0.1% C40	Shortly Interrupted by Notifications	Widgets->Social->Widgets	2.4 min	-0.1	0.1	-0.6	-1.3	-0.6	1.8	-0.4	0.0	0.3
0.1% C41	Complex Communication Topic Needs Auxiliaries	Personalization->Communication->Personalization	10 min	-1.1	1.0	-0.7	**2.3**	-1.4	0.4	-0.6	2.2	0.4
0.1% C42	Reference Info Supported Tasks	Reference->Productivity	8.6 min	0.8	-0.8	0.5	0.6	-1.4	0.4	0.7	1.3	0.2
0.1% C43	Complex Discussion Needed during Lifestyle	Communication->Lifestyle->Lifestyle	13.1 min	-0.2	0.2	-0.6	1.5	1.1	-1.4	-0.5	0.1	0.4
0.1% C44	Travel Related Info Check	Travel->Productivity->Travel	5.2 min	0.1	-0.1	0.1	-1.3	0.1	0.6	0.3	0.0	0.0

facebook->messenger->facebook), whereas each complex task involve at least two different apps. Since the demographic differences among single-app tasks are actually as same as the app preferences, we only focus on analysing the demographic differences in the complex tasks within this section.

However, most of the complex tasks are only conducted by users for once or twice, it would be too sparse for inferring users' demographic differences if each specific complex task is considered. In chapter 6, we have stated that the complex mobile tasks can be grouped into different categories based on our proposed clustering approach within various task characteristic, e.g., number of distinct apps or app categories, task duration, hour of day, day of week, hub app, dominant app and switch patterns, etc. Therefore, by leveraging our task clustering approach, we are able to map all our extracted complex tasks into different task categories (i.e. 44 clusters (clusters with popularity less than 0.05% are removed)), which can be used for avoiding the sparsity issue while inferring users' demographics based on specific tasks directly. For example, the specific complex tasks could be social->social, social->social->social and social->communication->social, since these tasks have similar task characteristics (e.g., task duration are all around 5 min, the hub and dominant apps within the task are all social apps, etc.), they could all be assigned to the same task category "Social Media Browsing" (C2). Table 7.4 present all the complex tasks ranked by their popularity with the corresponding characteristics, where the popular app sequences are selected based on the relative frequency across all clusters.

To analyze the demographics differences among different complex task categories, we extract the discriminatory power [122] of each complex task category. Indeed, the more discriminatory a task category is, the bigger

differences exist in the user distribution of this task category when compared with the original user distribution in the dataset. It helps differentiating user's age or gender. In [122], the chi-square is used to calculate the discriminatory power since it provides a quantitative measure for determining if the distribution of observations (frequencies) has a relationship with the distribution of expected frequencies. Assuming that we are given a set of task categories conducted by the users within demographic labels (e.g., male v.s. female), we make use of the well-established chi-square test to compute the discriminatory power of a given task category based on:

$$\chi^2 = \sum [\frac{(O-E)^2}{E}]$$

where O = observed frequency (users in this task category); E = expected frequency (users in the whole dataset).

Table 7.4 shows the chi-square of users' age/gender distribution for each task category and the standardized residual of each age/gender group, where we mark the top-ranked (i.e., top 10 for gender and top 5 for age) user group for each task category as blue. The unstandarized residual is the simple difference of the observed and expected values $R = O - E$. The standardized residual [156] is found by dividing the difference of the observed and expected values by the square root of the expected value $R_s = \frac{O-E}{\sqrt{E}}$. We report the standarized residual since it can be interpreted as a standard score for providing information about which user group contributes to a significant Chi-square regardless of the total amount of users in the task category. So the bigger the standardized residual is, the more users in this age/gender group exist in the task category. We can find that:

Given the distribution of users in different tasks, more female users are found in game or entertainment complex tasks while

male users are more than female users in the complex tasks for information gathering. Additionally, the users with adjacent age would be found more than other users in some specific tasks. From the Table 7.4, we can observe that the top preferred complex tasks are mainly about entertainment and relaxation, e.g., Info Check while Playing Games (C21), Long Time Word Games Playing (C24), Relaxation with Multiple Lifestyle App (C14), Music with Socializing (C23) and Switch between Multiple Videos/TV (C17). within the complex tasks like Multiple Updated Info Check (C1), Social Media Browsing (C2), Communication Needs Other Auxiliaries (C3), Multiple News Source Browsing (C8) and Redirect to News Reading (C20), male users involved are more than female users. When referring to the users in different age group, we find that the adjacent age groups would share similar task categories. For example, the news and weather related complex tasks, i.e., Multiple News Source Browsing (C10) and Weather Oriented Info Check (C13), all have the older users (55+) and middle-aged users (35 - 54) as the biggest age groups. The social and communication related complex tasks, i.e., Social Media Browsing (C2) and Interrupted Social Media Browsing (C9), have more 18 - 24 and 25 - 34 users than other user groups. The complex tasks like Switch between Multiple Lifestyle App (C17) and Music with Socializing (C23) have the most youth users (13 - 17 and 18 -24).

Besides the app categories (Table 7.2 and 7.3), the complex task categories could provide additional information for inferring users' demographics. For example, games, utilities, news, tools, and navigation apps are all involved in the top-ranked complex task categories for female users (C12, C18, C21,C38 and C39), but none of them are listed in the top app categories (Table 7.2). Some of them are even the top-ranked app categories for male users, e.g., news and tools. It may result from that

using these apps independently is the usage style more attractive to female users, but some specific complex tasks (i.e., Notes Taking with Info Resources (C38)) are more likely to enable female users engage with apps collaboratively. Similarly, the communication, entertainment and social apps related complex tasks (C2, C3, C7, and C28) which have more male users involved are also not within their top-ranked app categories. The communication apps are not listed in the most popular app categories for users over 55, but appear in their top-ranked complex tasks (C19). These all indicate that tasks with more detailed app interaction patterns could provide additional insights when inferring users' demographics when compared to the traditional app preference.

The complex tasks, even with the same app categories involved, could help us distinguish users from different age or gender groups. For example, news apps are involved in the top complex tasks for both female and male users, but more female users are found in the task to take some notes while reading news or focus on searching specific information from reading news (C38). Differently, male users are found be more than female users in the tasks for browsing news from multiple resources (C10) or be redirected to news reading from other resources (C20), e.g., redirect from email/docs (productivity) or communication. Within the tasks for watching TV/videos, more female users are found in the task to switch between multiple video sites or TVs (C17), but more male users are involved in the task to check social networking while watching TV (C28). These observations all indicate that complex tasks even with similar apps involved can help us infer users' demographics. Similar cases also exist in users of different age groups, e.g, both Communication Needs Auxiliaries (C3) and Complex Communication Topic Needs Auxiliaries (C41) are tasks mainly about communication, but they can still be used to distinguish the users

in the two different age ranges (i.e., 25 - 34 and 18 - 24) since the tasks provide more detailed interaction patterns, i.e. C41 have more switches between apps and longer task completion time than C3.

7.3 User Representation based on App Usage Behaviours

In this section, we present the different methods to generate user representation aims at demographics prediction. The goal of generating the user representation is to classify a user u into a demographic category y (e.g., an age group in age prediction and a gender category in gender prediction) based on the apps usage logs generated by this user, i.e., $[a_1, a_2, ..., a_N]$, where N is the number of apps usage logs and these logs are sorted by their timestamps. Each app a belongs to an app category a_c, such as the app "Instagram" belongs to "Social" category. As mentioned in the section above, each app usage a_N could also be mapped to a task t and the task t could be assigned in one of the task categories t_c. We introduce both the existing approaches for modeling users based on apps/app categories and the proposed approaches that get tasks and task categories involved while inferring users representations.

7.3.1 Benchmark

Many previous works have intuitively exploited the apps used by users to represent each user for predicting user attributes [154, 111, 133, 210]. In detail, they took each app as a dimension and represent each user as an app-based vector (e.g., binary "bag-of-apps" vector [111]). For the app-based

representation vector, if all the apps are used to build the user representation, the user vector will be dramatically long. So they proposed different strategies to reduce the dimensionality of app-based representation vectors: (1) Filter the top apps based on user engagement (apps installed by at least 10% of the users [111]) or apps with the top usage frequency (i.e., top 1000 [133] and top 500 [210]). (2) employs the Truncated Singular Value Decomposition (TSVD). Malmi et al. [111] set the number of dimensions equal to the number of app categories in their dataset and use the SVD components directly as the feature for user representation. (3) aggregates the apps to category level: Malmi et al. [111] and Seneviratne et al. [154] all used the number of apps in each app category to generate the user representation vector. (4) Other than the user representation only based on app or app categories, Zhao et al. [210] introduced the temporal feature for generating user representation. They represented each user using the top 500 apps and their usage percentage in four different time periods (i.e., night (0:01 am to 6:00), morning (6:01 to 12:00), afternoon (12:01 to 18:00), and evening (18:01 to 0:00)).

Currently, techniques used for mobile user representation and modeling are rather straightforward, such as simply using app lists or app usage frequency as features, without considering the correlation or semantic relationship (hidden insights) between apps. As we mentioned above, tasks have been increasingly used as a more accurate unit for representing users' behaviour [166]. The mobile task would consist of a set of apps usage corresponding to a particular high-level intention and the apps are not necessarily the same or even similar. Therefore, in this work, we propose to represent users not only on a per-app basis, but on the basis of tasks. We originally set the benchmark method for generating task-based user representation as same as the app-based method, which is to take each task as a

dimension and represent each user as an task-based vector. However, since the tasks have more serious sparsity issue than apps, the task categories are mainly used for generating the user representation. Furthermore, to avoid the app information loss in coarse-grained task categories, we concatenate best-performing baseline method from app or app category based vector with the task category based vector together for generating the benchmark method of task involved user representation.

7.3.2 Embedding based Representation

We have the assumption that the apps occurring in a similar context tend to have similar characters, as well as words and sentences [124], where a word embedding is a learned representation for text where words that have the same meaning have a similar representation. In our app usage scenario, if we learned the app embedding, the camera app representation should be closer to the gallery app and we should not treat them independently. Therefore, instead of simply concatenating all independent apps/app categories and tasks vectors together, we further propose to use the embedding based app/task vector for generating the user representation.

Zhao et al. [210] proposed to apply Doc2Vec [88] to model the app usage sequence for learning the user representation. Doc2Vec predicts the next word by exploring a paragraph and a word sequence in a given context in the paragraph. More specifically, every word is mapped to a unique vector, as well as each paragraph. Word vectors are averaged, concatenated, or summed as a feature vector that is concatenated with the paragraph vector for predicting the next word. Taking the analogy to word and document modeling, they treated each user as a document and each app as a word, to model the app usage sequence. During the training procedure, user and app

vectors are updated until convergence, where the user representation vector is obtained to feed the classifiers for demographic directly. In our work, we propose to leverage the classic Word2Vec [124] framework directly, and we name it as the "App2Vec" method in our work. We used the Continuous Bag of Word (CBOW) method for training the App2Vec model with different window size. In the CBOW architecture, the model predicts the current app from a window of surrounding context apps based on a two-layer neural networks. Afterwards, the user representation is generated based on the mean of all the embedding vectors of apps he/she has used before.

Neither in Word2Vec nor Doc2Vec, the model predicts the current word from a window of surrounding context apps, where the order of apps does not influence the prediction. We hypothesize that the sequential and contextual information of app usage are all meaningful for user representation. So lastly, we further propose to learn the task embedding vector which can take the sequential information within the app usage into consideration for generating the user representation vector. By leveraging the seq2seq architecture [31], which is originally proposed to generate sequences of words by predicting the next word while considering the entire sequence, we are able to encode the app usage sequence within each app usage sequence within a task based on the context vector C. This vector is suitable for the representation of tasks because it is originally designed to summarize all the encoded information, which represents a semantic summary of the input sequence. Specifically, we learn the task vector based on the context vector C with the same setting as Lee et al.[91], where they proposed to use of a variant of the conventional seq2seq architecture for learning the app sequence representation that receives an app usage sequence as the input of the encoder and generates the same app usage sequence in the decoder. In our scenario, we extract the app sequences based on the task identifica-

tion model as the input and output of the seq2seq model. We name this method as "Task2Vec". Finally, the user representation vector is generated by averaging the vectors of tasks he/she has conducted.

7.3.3 Neural based Representation

The embedding based method has tried to improve the user representation from learning the semantic information within the app usage, but they are still based on the straightforward aggregation method for inferring users' representation, i.e., average/sum of app/task vectors. In this section, we introduce several deep learning methods for inferring users' representation specifically aims for demographics prediction since we hypothesize that they would be more efficient to automatically find out the key features/patterns that are more predictive but less intuitive, including CNN [82], LSTM [65], Hierarchical Attention Network (HAN) [197] and Hierarchical User Representation with Attention (HURA) [186].

Both CNN and LSTM have been applied in solving the text classification problems, e.g., sentiment classification [82, 145]. A document/sentence is usually a combination of words or word sequence, then the CNN and LSTM are trained to learn the representation of the document/sentence aims for sentiment classification based on the input of concatenated word embeddings (Word2Vec). The text analysis method based on CNN can obtain important features of text through pooling but it is difficult to obtain contextual information. The LSTM can obtain context information but the order of words will lead to bias. In our scenario, the app usage sequence of each user could be treated as the word sequence of each document. So we tried to test on both of CNN and LSTM for solving our demographics prediction problem, where both app embeddings (i.e. App2Vec) and task

embeddings (i.e. Task2Vec) are used as the input.

Furthermore, we provide another two hierarchical structure based deep learning methods to infer users' representation with the combination of both apps and tasks: Hierarchical Attention Network (HAN) [197] and Hierarchical User Representation with Attention (HURA) [186]. The HAN is originally designed to obtain a better representation of documents by incorporating knowledge of document structure for text classification than the common deep learning methods of CNN and LSTM. It aimed to capture two basic insights about document structure. Since documents have a hierarchical structure (words form sentences, sentences form a document), they likewise constructed a document representation by first building representations of sentences and then aggregating those sentences into a document representation. Second, to capture the patterns that different words and sentences in a documents are differentially informative, their model included two levels of attention mechanisms [9, 190] - one at the word level and one at the sentence level - that let the model to pay more or less attention to individual words and sentences when constructing the representation of the document. Therefore, by adopting the HAN architecture, we are also able to model the users based on the two-layer hierarchical structure of attention mechanisms, which are apps from tasks and tasks from a user. Specifically, we focus on user demographics prediction in this work. Assume that a user has L tasks t_i and each task contains M_i apps. a_{im} with $m \in [1, M]$ represents the apps in the ith task. The HAN model projects the raw user into a vector representation, on which we build a classifier to perform user demographics prediction. We then present how we build the user level vector progressively from app vectors by using the hierarchical structure as shown in Fig. 7.1.

App Encoder Given a task with apps a_{im}, $m \in [1, M]$, we first embed the

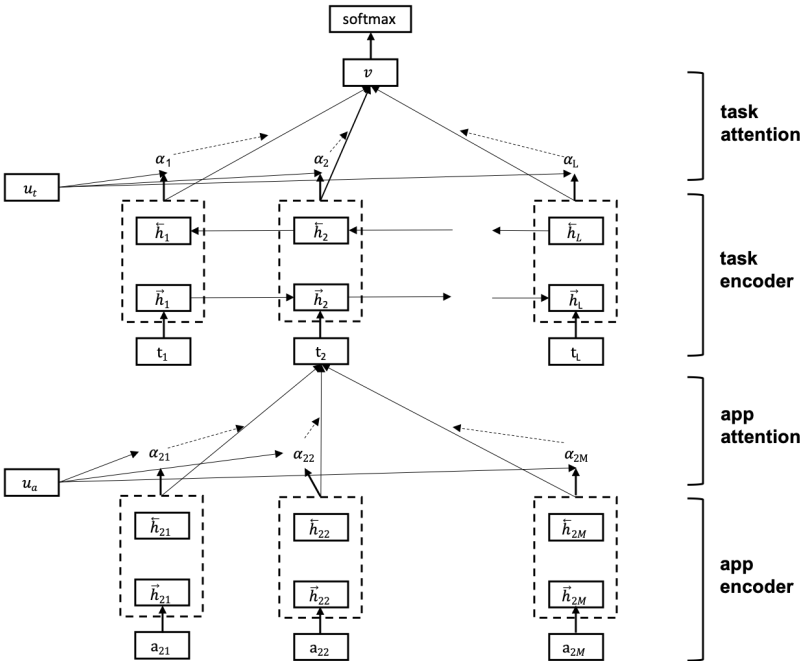


Figure 7.1: Hierarchical Attention Network (HAN) for demographics prediction based on both app and task units.

apps to vectors through an embedding matrix W_e, $x_{ij} = W_e a_{ij}$. Then a bidirectional GRU [9] is used to obtain an annotation for a given app a_{im} by concatenating the forward hidden state $\overrightarrow{h}_{im} = \overrightarrow{GRU}(x_{im})$ and backward hidden state $\overleftarrow{h}_{im}\overleftarrow{GRU}(x_{im})$, i.e., $h_{im} = [\overrightarrow{h}_{im}, \overleftarrow{h}_{im}]$.

App Attention Not all apps contribute equally to the representation of the user characteristics. Hence, attention mechanism is applied to extract such apps that are more important to meaning of the task and aggregate the representation of those informative apps to form a task vector. Specifically,

$$u_{im} = tanh(W_a h_{im} + b_a), \tag{7.1}$$

$$\alpha_{im} = \frac{exp(u^u_{ima})}{\sum_m exp(u^u_{ima})}, \tag{7.2}$$

$$t_i = \sum_m \alpha_{im} h_{im}. \tag{7.3}$$

Task Encoder Given the task vectors t_i, we can get a user vector in a similar way by applying a bidirectional GRU to encode the tasks: $h_i = [\overrightarrow{h}_i, \overleftarrow{h}_i]$, where $\overrightarrow{h}_i = \overrightarrow{GRU}(t_i)$ and $\overleftarrow{h}_i = \overleftarrow{GRU}(t_i)$. h_i summarized the neighbor tasks around a task i but still focus on task i.

Task Attention To reward tasks that are clues to correctly classify a user, attention mechanism is used again and a task level context vector u_t is introduced to measure the importance of tasks. This yields:

$$u_i = tanh(W_t h_i + b_t), \tag{7.4}$$

$$\alpha_i = \frac{exp(u_i^u{}_t)}{\sum_i exp(u_i^u{}_t)}, \tag{7.5}$$

$$v = \sum_i \alpha_i h_i \tag{7.6}$$

where v is the user vector that summarizes all the information of tasks conducted by the user. Then it can be used as features for user classification:

$$p = softmax(W_c v + b_c) \tag{7.7}$$

Lastly, other than HAN, we also applied HURA for building user representation based on the hierarchical structure of apps and tasks. The HURA was inspired by HAN, which is proposed to predict users' gender and age from their search queries [186]. The general hierarchical structure of HURA model is similar to HAN, which also learned a representation vector for each search query from words using a word encoder and a word-level attention network to select important words. Then it learned a representation vector

for each user based on the representations of his/her search queries using a query encoder and a query-level attention network to select informative search queries for demographic prediction. The biggest difference of HURA when compared to HAN is that they use CNN instead of GRU for capturing the local context of words and search queries. They stated that CNN is more appropriate for learning query representation and user representation since search queries are quite different from sentences, where a search query is usually a combination of several keywords, rather than a complete sentence. To test that if our tasks composed of apps usage are more similar to queries composed of several keywords, we also leverage HURA for inferring user representation aims at demographics prediction.

7.4 Experiments

7.4.1 Experimental Settings

In our experiments, the app embeddings were trained based on all the app usage logs in the training data using the word2vec[1] tool. The embedding dimension is 300 for App2Vec, CNN (App2Vec), and LSTM (App2Vec) and the window size is 4. The hidden dimension of the context vector in Task2Vec is 100. So the input embedding dimension for CNN (Task2Vec) and LSTM (Task2Vec) is also 100. For the two hierarchical structure based networks, the window sizes of app-level CNN and task-level CNN in HURA are both 3, and the number of filters in these CNN networks is 300. In HAN, the GRU dimension is 50. In this case, a combination of forward and backward GRU gives us 100 dimensions for app/task annotation. The app/task context vectors also have a dimension of 100. For all the neural-

[1]https://radimrehurek.com/gensim/models/word2vec.html

based methods, Adam [84] was used as the optimizer for model training. The batch size was set to be 128. Dropout technique [158] was used to mitigate overfitting and was applied to CNN layers. The dropout rate was set to 0.2. Early stopping strategy [27] was also used. If the loss of validation data does not decline after 10 epochs, then the training will be terminated. For all the approaches, the training, validation and test data are split based on users, i.e., 70%, 10% and 20% of users are used for training, validation and testing. All the hyperparameters were selected according to the validation data. Each experiment was repeated 10 times independently and average results were reported.

7.4.2 Baselines

The existing approaches for demographics prediction of mobile users [154, 111, 133, 210] are compared as the baselines in our work. We ran all the baselines with their proposed best-performing classifiers and parameter settings:

- *Bag-of-Apps (BoA)* [111]: Binary vector with each app as a dimension.
- *Bag-of-Top-Apps (BoTA)* [111, 133, 210]: Filter the top apps based on user engagement (apps installed by at least 10% of the users [111]) or apps with the top usage frequency (i.e., top 1000 [133] and top 2000 [210]).
- *TSVD of Bag-of-Apps (TSVD BoA)* [111]: Malmi et al. [111] set the number of dimensions equal to the number of app categories in their dataset and use the SVD components directly as the feature for user representation.

- *Number of Apps in each App Category (NAAC)* [111, 154]: Malmi et al. [111] and Seneviratne et al. [154] all used the number of apps in each app category to generate the user representation vector.

- *Temporal App/App Category Usage Frequency (TAUF / TACUF)* [210]: Zhao et al. [210] introduced the temporal feature for generating user representation. They represented each user using the top 500 apps and their usage percentage in four different time periods (i.e., night (0:01 am to 6:00), morning (6:01 to 12:00), afternoon (12:01 to 18:00), and evening (18:01 to 0:00)). Additionally, they further explored app categories to represent each user for demographic prediction. They represented each user's app usage using the categories and usage percentage in different hours.

- *Doc2Vec [210]*: Zhao et al. [210] proposed to apply Doc2Vec [88] to model the app usage sequence for learning the user representation in demographics prediction.

7.4.3 Experimental Results

In this section, we evaluate the performance of demographics prediction based on user representation learned from apps and tasks with baseline methods and our proposed approaches. As reported in Table 3.2, we have two genders (i.e., male and female) and five age groups (i.e. 13 - 17, 18 - 24, 25 - 34, 35 - 54 and 55+) in total, where users do not have an even distribution across all the age groups. We used the accuracy, macro-averaged f1 score as evaluation metrics. The results are summarized in Table 7.5.

For all the baseline methods, they are mainly conducted by classifying users into different groups based on interacted app or app categories with

Table 7.5: Overview of demographics prediction results: F1 are macro-averaged scores (bold ones are the best performing models).

Method	Age		Gender	
	Accuracy	**F1**	**Accuracy**	**F1**
Baselines				
BoA [111]	0.523	**0.461**	0.710	**0.710**
BoTA (2000) [210]	0.523	0.460	0.700	0.700
TSVD BoA [111]	0.512	0.439	0.677	0.677
NAAC [111, 154]	0.433	0.354	0.700	0.700
TAUF [210]	0.485	0.413	0.686	0.686
TACUF [210]	0.426	0.302	0.680	0.680
Doc2Vec [210]	0.427	0.282	0.660	0.659
Our Approaches				
Without Tasks				
App2Vec	0.543	**0.470**	0.714	0.715
CNN (App2Vec)	0.545	0.467	0.722	**0.722**
LSTM (App2Vec)	0.520	0.461	0.708	0.709
With Tasks				
Bag-of-TaskCategory	0.411	0.424	0.686	0.686
Bag-of-App-TaskCategory	0.525	**0.471**	0.722	**0.722**
Task2Vec	0.526	0.467	0.711	0.709
CNN (Task2Vec)	0.526	0.450	0.727	**0.727**
LSTM (Task2Vec)	0.528	0.468	0.693	0.695
HAN (Two Layer: App-Task)	0.552	**0.488**	0.720	0.720
HURA (Two Layer: App-Task)	0.518	0.461	0.742	**0.742**

a machine learning classifier (e.g., SVM, Random Forest, and Logistic Regression as summarized in Table 7.1). When there are different settings with the same baseline approach, e.g., different ways to select top apps for Bag-of-Top-Apps (BoTA), we only report the setting with the best performance. Among all the baselines, we can find that the Bag-of-Apps (BoA) performs best in both age and gender prediction, which is also consistent with the previous results in [111, 210]. For our proposed approaches without tasks information involved, we can find that App2Vec method performs better than all the baselines in both age and gender prediction since it not only keeps the specific app details but also captures the complex semantics and contexts of apps usage. For the CNN and LSTM with App2Vec as input embeddings, only CNN outperforms the original App2Vec in the gender prediction, which may state that the local context within app usage learned by CNN is more important in gender prediction. As mentioned in

Section 7.3.1, the benchmark method with tasks involved is to concatenate best-performing baseline method with the task category based vector together, which is displayed as the Bag-of-App-TaskCategory in Table 7.5. We can find that it outperforms the best baseline method , Bag-of-Apps (BoA), both in age and gender prediction, which indicates that the tasks, even in a coarse-grained task category level representation can bring additional benefits in users' demographics prediction. We further test the performance when only the task category is used for generating the user representation (Bag-of-TaskCategory). We observe that the performance for age prediction is much improved (about 20%) when it is compared with the baseline methods where only app categories are taken into consideration (NAAC[111, 154] and TACUF [210]). It states that task categories could cover more detailed user characteristics which especially benefit the age prediction.

Lastly, the HAN and HURA perform best in age and gender prediction respectively when compared to all other methods. They are both neural-based approaches that leveraged the hierarchical structure of apps from tasks and tasks from a user. Instead of CNN and LSTM, where all the apps/tasks of the same user are concatenated together as a long text for building user representation, the hierarchical neural models could be more effective for capturing the complex relationships between apps, apps to tasks and tasks to users. In addition, they incorporate both app-level and task-level attention networks into the approach to attend differently to different apps and tasks based on their contributions to demographic prediction task. From the prediction results, we can find that HAN performs best in age prediction, and HURA performs best in gender prediction. It may be because that, in the HAN method, bidirectional GRU is used to learn task representation from apps and learn user representation from

tasks where the sequential information of apps and tasks can be more captured. Differently, CNN is used in HURA for learning the task and user representations, where the sequential information would not be learned as in HAN. Therefore, we can also summarize that the sequential information within the apps and tasks would be more important in inferring users' age than gender.

7.4.4 Case Study

In this section, we conducted several case studies to further explore whether tasks could bring more benefits when learning user representations for demographics prediction. We have several observations. First, some male users have apps similar to female users. Thus, predicting users' gender only based on their apps is very challenging. For example, the apps used by one male user are food-and-drink, social, productivity, lifestyle and video, where three of them are all top apps preferred by females (Table 7.2). So it's not surprising that the app-based method (both BoA and App2Vec) all predict this user as female. But the HURA predicted the gender correctly since it captured the male characteristics by learning from the tasks the user usually conducted, i.e., social->social, which is a salient task category (C2) for male users list in Table 7.4. Second, users in adjacent age groups originally have similar app preferences, especially for the users in 13-17 and 18-24. For example, one user used apps like games, video, entertainment, arcade, health-and-fitness, productivity, and music. We can find that games and music apps are preferred by users in 13-17, video and music apps are the top preferred apps for users in 18-24. So it's also hard to infer users' age only based on their used apps. The BoA and App2Vec all predicted this user as 18-24, but he is actually from the age group of

13-17. The HAN predicted the age correctly since the user prefer to conduct the tasks about Info Check while Playing Games (C21) (Table 7.4), e.g., games->productivity, games->social->games. Third, many users only have the general common apps which have no significant preference in any age group, e.g., productivity, utilities, others, sports, personalization and food-and-drinks. For these users, only the task-based approach could predict their age correctly by extracting the additional information from tasks they conducted. Thus, these case studies validate that the motivations of our approach are reasonable and the user representation learned from the task-based hierarchical neural approaches are more effective in predicting demographics, especially when compared to the models that only take the apps into consideration.

7.5 Ethical Discussion

Businesses often tried to collect or infer demographic information about customers to help them understand who is buying their products. But when businesses tried to market their products and services to the groups that are most likely to buy them, advertisements also help perpetuate societal stereotypes, which may have crossed an ethical boundary. For example, advertisements that feature women cleaning in an attempt to sell cleaning products perpetuate the stereotype that women are responsible for housekeeping. Determining how we should use user profiles to create effective advertising without perpetuating stereotypes or creating advertisements that may be offensive to some populations is also an ethical dilemma that businesses must face.

Nowadays, there is a tendency to believe that dealing with a machine rather

than with a human is more objective and rational, and is free of biases [7, 108]. However, many previous studies have found that machine learning algorithms tend to reproduce human biases [18, 23, 161]. For example, "historical" data were always used to train the algorithm so that algorithms can "learn" the patterns. This poses the risk of reproducing and amplifying the biases already present in our society and in our databases. For instance, gender stereotypes in algorithms were trained on Google News [18], where the algorithms tended to associate nouns such as "brilliant", "architect", and "great", more often with the word "he", whereas they frequently associated the words "mom", "housewife", and "princess" with the word "she". Evidence of gender bias has also been found in algorithms of online communities that affect labor markets [115]. It was reported that LinkedIn reflected a gender bias, with an algorithm suggesting male names (e.g. Stephen Williams) when searching for female candidates (e.g., Stephanie Williams) [37]. Potential gender bias was also identified in the automated-job alerts where one user reported finding that changing the gender on the job platform from female to male resulted in better-qualified and better-paid job offers [112].

As we are the first work conducted for demonstrating the benefits brought by mobile tasks in improving user profiling, demographic information was leveraged to validate the improvements as it's more consistent across different systems and more common information to represent the entire population. However, in the decision to create and bring algorithms to market, the ethics of likely outcomes must be considered—especially to see its potential for harm, and where there is a risk of perpetuating existing biases or making protected groups more vulnerable to existing societal inequalities. There have been many researchers who worked on how to reduce consumer harm within algorithmic bias [90]. The procedures that are commonly used

in the social sciences to study human behaviour and cognition (i.e., controlled experiments; correspondence testing procedure) can be successfully applied to audit the fairness of algorithms [112]. To minimise the harm of use of the demographic information under the mobile task context, we will explore how to involve controlled experiments, independent review and other remedies to assure that the task-based recommendation system works fairly in future work. We also strongly suggest the operators of the algorithms continue to play a role in identifying and correcting biased outcomes long after an algorithm is developed, tested, and launched.

7.6 Conclusions

In this chapter, we study an interesting and challenging problem, i.e., predicting the demographics of mobile users based on their tasks extracted from app usage logs. We analyse the additional insights brought by tasks when compared to the independent apps. We propose different approaches for measuring users app usage behaviour with both apps and tasks. Finally, two hierarchical neural models are validated for inferring better user representations aims at gender and age group prediction respectively. They are able to first use an app encoder to learn the representation of tasks from apps, and then use a task encoder to learn user representation from tasks. Additionally, we also incorporated both the app-level and task-level attention networks to select and highlight the important apps and tasks to learn more informative user representation for demographic prediction. Experiments results based on the real-world large-scale dataset show that the mobile task based user modelling approaches could effectively improve the performance of gender and age group prediction when compared to all the baseline methods.

From this study, we validate that task is a more accurate unit to capture users' characteristics and behaviour insights, especially when compared to apps. While thinking back to the traditional next app prediction problem, we believe that if we could know the specific task user aim to conduct instead of only the app user would access in advance, we would work better for providing more satisfying services to users, especially for tracking the task progress and supporting task completion. Therefore, in the next chapter, we further investigate the next task prediction problem.

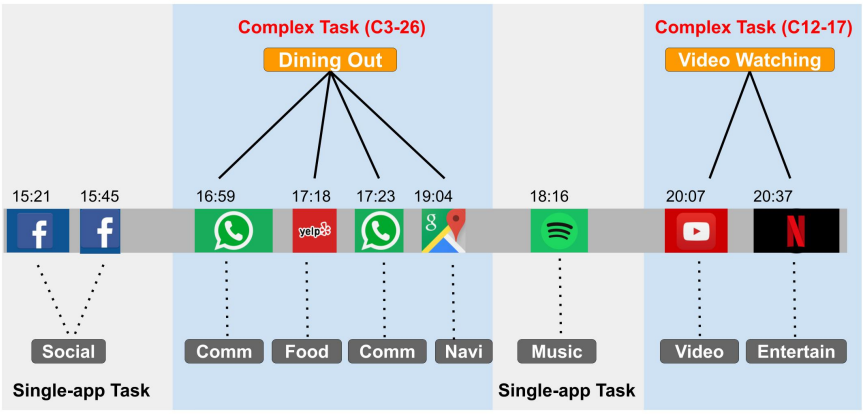
Chapter 8

Complex Task Prediction

In the previous chapter, we focused on validating the benefits brought by tasks while profiling users in demographics prediction, where tasks would capture more users' characteristics and behaviour insights than apps. Nowadays, users are increasingly relying on intelligent assistants (IAs) within mobile devices to complete their tasks. We believe that if we could know not only the single app user would access, but also the specific complex task user would conduct in advance, the IAs would work better for tracking the task progress and supporting task completion. For the example of "Dining Out" task, if we only measure users' app usage behaviour, the IAs may only be able to predict that user would like to use a single food app, but have no ability to make the seamless integration with different related apps (navigation and transportation, etc.). So this chapter addresses our research question **RQ 3.3**, as specified in Section 1.1:

RQ 3.3: Could we predict what the next complex task is by measuring users' app usage behaviour within task space?

It is well known that intelligent systems should help people get tasks done



Measurements	Historical Logs Representation for Prediction	Historical Logs Representation for Prediction
App Category	Social, Social	Social, Social, Communication, Food-and-drinks, Communication, Navigation, Music
Complex Tasks	None	Dining Out (C3-26)
App Category + Complex Tasks	Social, Social	Social, Social, Dining Out (C3-26), Music

Figure 8.1: Illustration of complex task prediction based on different measurements.

- task-aware recommendation and assistance are the critical areas that require task information inferred from user behaviours. By knowing the specific complex task user would engage in, the IAs could be able to track the task progress and task completion. Under the "Dining Out" task context, if the user only booked the table of the restaurant without checking the navigation or traffic status, the IAs could push notifications or suggestions to remind users to call a taxi in advance due to the rush hours for having dinner.

To clarify the complex task prediction problem, we illustrate the experimental process as shown in Figure 8.1. Firstly, based on the task boundary identification model proposed in Chapter 5, we can identify all the tasks based on users' app usage logs. All these extracted tasks could be classified into two kinds of mobile tasks: single-app tasks and multi-app (complex) tasks, whereas each complex task could be labelled with a specific task type (i.e., 47 sub-clusters) based on the clustering results in Chapter 6 given its characteristics, e.g., "Dining Out" (C3-26) and "Video Watching"

(C12-17). We then only focus on proposing models for the new problem of predicting the next complex tasks users will perform, given predicting single-app tasks would be equivalent to the next app prediction problem.

The remainder of this chapter is organized as follows. In Section 8.1, we present the formulation and method for the next task prediction problem. Experimental results are detailed in Section 8.2. We conclude the work in Section 8.3.

8.1 Method

The goal of the complex task prediction is to infer the detailed type of the task, i.e. 47 sub-clusters labels (Section 6.2.3.2). Therefore, the prediction could be modelled as a multi-class classification problem of 47 types within the complex task space T. The **complex task prediction problem** is formally defined as follows: given the app usage logs $L = \{(a_1, t_1), (a_2, t_2), ..., (a_i, t_i)\}$ (where t_i is the corresponding timestamp when using app a_i) used before the prediction time and temporal context C (hour of day and day of week), the problem of complex task prediction is to find a task $\hat{T}$ that has the highest probability of being used under C. Specifically, we aim to solve:

$$\hat{T} = \underset{T_j \in T}{\operatorname{argmax}} P(T_j | L, C)$$

Since there are no previous works conducted for complex mobile task prediction, we first propose two common baselines used by the previous next app prediction studies [157, 194, 218]: MFU (most frequently used) and MRU (most recently used). Specifically, the MFU indicates that the predicted task type is the most frequently used task. The MRU means the

predicted task type is the most recently used task.

For adapting the other app usage prediction models as the baseline models to our complex task prediction problem, we need to note the sparsity issue encountered by the complex tasks. As shown in Table 5.3, only 20% of users' tasks could be identified as complex tasks, so a user may only conduct a specific type of complex task for once within all his/her historical app usage behaviour (especially when regarding our week-long dataset). Therefore, many of the personalized app usage prediction models could not be applied to our complex task prediction problem, especially for those which leverage the periodic pattern [100, 163] and context patterns [218]. We then select a generic app usage prediction model (by leveraging all users' logs) proposed by Do et al. [41] as another baseline approach. They extracted the predictive feature based on the historical count of app-view-events of different apps that are used at least once a week by the user, and also time-of-day and day-of-week to represent the temporal context. In our problem, since we have a relatively larger scale dataset with more than 9K different apps, to ensure the generalization of the generic prediction model, we replaced the apps with app categories for building the feature vector, while the output of the generic model corresponds to the different types of complex tasks. Finally, given the task identification model and our proposed clustering framework, we are able to map user's app usage logs into different tasks and propose to extract the predictive feature based on the task unit. Specifically, we propose to generate the predictive features by three different aggregation approaches based on users' historically engaged complex tasks:

1. *Binary Feature*: binary vector to indicate the appearance of different complex task types;

2. *Frequency Feature*: vector of bag-of-task to indicate the times users engaged with all the different complex task types;

3. *Distribution Feature*: vector of the probability distribution to indicate the frequency distribution (sum to 1) users have engaged with all the different complex task types.

Other than these features for modelling users' historical app usage behaviours based on complex tasks, we also keep the temporal context features: the hour of day and day of the week while doing the prediction. Besides the features only based on complex tasks modelling, we also propose to extract the predictive features based on app categories with different aggregation methods since they could monitor all the app usage behaviour of users and can be compared to the complex tasks representations. Additionally, the hybrid predictive features that include both app category- and task-based representations are also evaluated.

To summarize, to further compare the performance for measuring users' app usage logs with different units for complex task prediction, we propose to extract the predictive features based on three different units: app category, complex task, and the combination with both of them. As shown in the Table at the bottom of Figure 8.1, where we can observe the differences for measuring user's historical logs with different units. For example, when predicting the complex task "Dining Out", no complex tasks have been performed previously, if we measure user's historical behaviour based on complex task units, no information could be used for the prediction. When it comes to the next prediction for "Video Watching", there is one complex task that has been conducted before. Different from the app category measurement, multiple apps usages (communication, food-and-drinks and navigation) would be merged into one complex task representation ("Dining

Out") for modelling the user's historical behaviour and then be used for the prediction.

8.2 Experimental Results

We conduct the traditional machine learning classification experiments based on our extracted predictive features and the ground truth (i.e., 47 types of complex tasks). We apply the 10-fold cross-validation to evaluate each model. At each time, we split all the users into training and test set: the logs of 9-fold of the users are used as the training set, and the remaining 1-fold users' logs are used as the test set. We use a set of state-of-the-art algorithms to build models for our prediction problem: (1) Random Forest (RF) [20], as an example of ensemble learning method; (2) K Nearest Neighbours (KNN) [33], as an example of the non-parametric method for classification; (3) L2-regularized Logistic Regression (LR) [62], as an example of the linear classifier. We report four metrics with 10-fold cross-validation: accuracy (acc), precision (pre), recall (rec), and F1 score (F1).

Table 8.1 reports the performances of baselines and different measurement unit representations with different feature extraction methods. We mainly report the results for the logistic regression classifier in Table 8.1 since it outperforms other classifiers, as we show in Table 8.2. Firstly we can find that all the representation methods that have task signals involved outperform all the baselines. While for the app category based representation, only the distribution method performs better. When comparing the different measurements between app category and task, we can find that task-based representation improves the F1 score by about 4.2% than the

Table 8.1: Performance comparison for complex task type prediction results with the logistic regression classifier ($*$ indicates statistical significant ($p \leq 0.01$) using two-tailed T-test compared to the best baseline:MFU; † indicates statistical significant ($p \leq 0.01$) using two-tailed T-test compared to the best app category based representation with distribution aggregation approach). Bold scores state the best performance of independent representation and hybrid representation respectively.

Baseline					**App-category-based Representation (A)**					**Task-based Representation (T)**				
Method	**Acc.**	**Pre.**	**Rec.**	**F1**	**Method**	**Acc.**	**Pre.**	**Rec.**	**F1**	**Method**	**Acc.**	**Pre.**	**Rec.**	**F1**
MFU	0.472	0.538	0.472	0.491	Binary	0.371	0.703	0.371	0.469	Binary	0.426	0.718	0.426	0.517
MRU	0.424	0.448	0.424	0.432	Frequency	0.423	0.636	0.423	0.487	Frequency	0.437	0.682	0.437	0.508
Do et al.[41]	0.423	0.636	0.423	0.487	Distribution*	0.408	0.803	0.408	0.525	Distribution*†	0.441	0.795	0.441	**0.547**
Hybrid (App Category + Task) Representation														
Binary (A) +					**Frequency (A) +**					**Distribution (A) +**				
Method	**Acc.**	**Pre.**	**Rec.**	**F1**	**Method**	**Acc.**	**Pre.**	**Rec.**	**F1**	**Method**	**Acc.**	**Pre.**	**Rec.**	**F1**
Binary (T)	0.439	0.691	0.439	0.522	Binary (T)	0.408	0.728	0.408	0.496	Binary (T)	0.447	0.711	0.447	0.535
Freq.(T)	0.452	0.643	0.452	0.517	Freq. (T)	0.453	0.640	0.453	0.512	Freq. (T)	0.450	0.665	0.450	0.519
Dist. (T)	0.478	0.728	0.478	0.566	Dist. (T)	0.405	0.726	0.405	0.493	Dist. (T)*†	0.473	0.772	0.473	**0.572**

Table 8.2: Performance comparison for complex task type prediction results based on different classifiers. We only report the best aggregation approach for each classifier.

Classifier	Acc.	Pre.	Rec.	F1
App-category-based Representation (A)				
RF (Freq.)	0.436	0.627	0.436	0.500
KNN (Freq.)	0.375	0.466	0.375	0.406
LR (Dist.)	0.408	0.803	0.408	**0.525**
Task-based Representation (T)				
RF (Dist.)	0.456	0.578	0.456	0.499
KNN (Freq.)	0.408	0.507	0.408	0.440
LR (Dist.)	0.441	0.795	0.441	**0.547**
Hybrid Representation (A+T)				
RF (Dist.)	0.472	0.654	0.472	0.535
KNN (Dist.)	0.391	0.526	0.391	0.437
LR (Dist.)	0.473	0.772	0.473	**0.572**

app category prediction when singly used. The best overall representation is modelling the historical logs by both complex tasks and app categories, which could improve the best baseline by about 16.5%. Additionally, we can find that the best aggregation approach to generate the predictive feature is the distribution probability of different tasks types/app categories for predicting the complex task.

8.3 Conclusion and Discussion

To summarize, our experiment results validate that the task-based representation of users' app usage behaviour could improve the complex task prediction on smartphones significantly, especially when compared to the traditional measurement based on app category units. It indicates that no matter what kind of task-based applications need to be explored, for knowing which complex task users aims to conduct, both app usage and task signals should be taken into consideration for modelling users' be-

haviour. Otherwise, users' superior intentions of tasks could not be well understood. We need to note that, the improvements of leveraging task representation are not as huge as we expected when comparing the task and app category-based representation. This is mostly because the complex tasks only occupied less than 20% of tasks (Table 5.3) among all the users' app usage tasks, while 80% of users' behaviour could be caught by the app category measurement. Therefore, the more serious cold-start issue would be encountered by task representations when conducting the prediction. As the example shown in Figure 8.1, when predicting the "Dining Out" task, there are no complex tasks that have been performed by the user before, so the predictive features based on task representation could not provide any information. Additionally, this is also partially due to the nature of our data collected. Since we only utilized one-week app usage logs, it might be more difficult to acquire sufficient complex tasks for each user. When the long-term data is available, the sparsity issue of complex tasks could be alleviated, whereas the performance based on task measurement could be further boosted.

By knowing which task the user aims to perform in advance, the intelligent assistants (IAs) could automatically help users organize tasks across domains or apps given a user's request expressed at the level of tasks. To be specific, upon knowing the task "Dining Out", the IA should recommend popular restaurants by browsing Yelp (even with multiple similar food apps together), then book the table and report the navigation according to Google Maps for the one chosen by the user. Optimized transportation method and estimated time spent also need to be recommended to the user based on the real-time traffic status. Finally, IA could also be able to send the detailed information of booked restaurant, table number, time, or car number of the taxi to the user's messages box or display them

on the home. Instead of only recommending which app users would use next, tasks should be understood by IAs, especially for those tasks across multiple apps, which are time-consuming and difficult for users. In other areas, researchers have argued for decades that information retrieval (IR) systems should help people get tasks done. For the mobile search, Carrascal et al. [26] highlighted that tasks are more important than individual mobile app usage. They found that the categories of apps used before and after a mobile search, as well as complex switching between apps and search, point to an overarching theme of *task completion*. So if we could infer which task the user is going to perform, the mobile search could preempt these behaviours and offer more proactive search experiences. Essentially supporting users in task continuation and task completion. For example, when the task of "Multiple Updated Info Check" has been detected, the user aims to collect all the latest information from different sources, not a specific app. Even the user only engaged with a finance app just now, if they searched "USPS", we should know that he/she probably wants to get the latest tracking information under the task context rather than recommendations of post services. We leave the validation on benefits brought by task-based modelling of app usage behaviour in mobile search as our future work when the search data could be collected.

Here we summarize the task-based application part of this thesis (Part III). In part III of this thesis, we firstly demonstrate the task-based user profiling method could work better in demographics prediction, especially when compared with the traditional app-based user representation. Secondly, we further validate that by measuring users' app usage behaviour within the task space, we would be able to predict which task users aim to conduct next. Since we are the first work to propose understanding users' tasks based on their app usage behaviour, within both of these two studies, we

aim to provide the initial steps in shaping future research on investigating whether and how the extracted tasks could be applied for improving mobile services. We believe that task-based applications could be well developed in the future for providing users with more advanced and satisfying services in different scenarios. In the following chapter, we draw conclusions from all research chapters and present our main findings and directions for future work.

Chapter 9

Conclusion & Future Work

In this thesis, we present work towards inferring users' needs and tasks from their app usage interactions, which could help the operating system and app developers deliver more satisfying content and services to the end-user. Firstly, for improving the existing models on inferring users' app usage needs, more comprehensive studies are conducted. We investigate how to alleviate the *cold-start* issue for app usage prediction problem. Specifically, when new users come, who do not have much data to be learned, how could we still infer their app usage needs effectively. Furthermore, other than the only predicting the next app user would use, we also try to model how long the user would stay with this app and propose to predict both which app user would use and app dwell time simultaneously. Secondly, while modern mobile devices only effectively serve the individual apps that correspond to simple needs, e.g., weather checking, users get little or no help when their needs transcend the boundary of a single mobile app. We aim to further understand the high-level intention with more semantic meaning within users' minds while engaging with different apps. Traditionally, the logged app usage streams could be segmented into different sessions based on the

short inactivity timeouts (30 or 45 seconds). However, users' tasks with their high-level intentions may span multiple sessions and involve different apps, where the empirically-set short timeout threshold may not be a valid criterion, e.g., dining out, vacation plan. So we investigate the methods for identifying tasks from users' app usage interactions and characterizing the extracted tasks for better understanding users' high-level intentions. Lastly, given the extracted tasks, we also try to leverage the task information in different applications for validating the benefits brought by tasks in providing more satisfying services and improving the user experience. For now, little research has explored methods to understand and identify mobile tasks, let alone to support users with more satisfying services and content, especially in task continuation and task completion.

The six research chapters in our thesis addressed the challenges of inferring user needs and tasks from app usage interactions as follows by answering three big research questions:

RQ1. How could we improve the methods for inferring users' needs on single apps, especially when compared to the existing models?

RQ2. Other than the traditional approach for modelling users' behaviour based on specific apps, could we measure users' app usage behaviour within task space?

RQ3. How could we leverage the extracted tasks information in different applications for improving mobile services?

Firstly, we aim to infer users' needs that could be satisfied by single apps from Chapter 3 to 4 (Part I), which especially for improving the existing methods on app usage prediction (**RQ 1**). In particular, in Chapter 3, we demonstrate that the *cohort* information could be utilized for next app

category prediction, especially on alleviating the user *cold-start* problem. In Chapter 4, we explore the factors that affect users' app dwell time from user characteristics and various context information. We show evidence that the next app and how long the user will stay on this app could be predicted simultaneously.

Secondly, we aim to investigate how to identify tasks from users' app usage behaviour and characterize the extracted tasks for uncovering the common patterns among tasks in the wild (**RQ 2**) from Chapter 5 to 6 (Part II). In particular, in Chapter 5, we propose to set the stage for evaluating mobile apps usage, not on a per-app basis, but on the basis of user tasks. We first propose a method that accurately determines mobile tasks from users' app usage logs. We showed that a set of temporal, similarity and log sequence features used in combination can effectively predict mobile tasks. In Chapter 6, the automatic identification of mobile tasks proposed in Chapter 5 is employed in a large-scale dataset, where we try to understand the mobile tasks that exist in a wide range of smartphone usage. Given the extracted characteristics for representing complex tasks, we show evidence that there actually exist several differentiable groups of complex tasks, which could be identified solely from their salient properties.

Lastly, we aim to demonstrate that the extracted task information could be leveraged in various applications (**RQ 3**) from Chapter 7 to 8 (Part III). In Chapter 7, we demonstrate that mobile task based user modelling approaches could effectively improve the performance of gender and age group prediction, especially when compared to the traditional user representation based on app interests. In Chapter 8, we further validate that tasks are more accurate units to capture users' goal and behavioural insights for predicting the next complex task user will perform.

Below, we provide a more detailed summary of the contributions and results of our research, and answer the research questions set out at the beginning of this thesis (Section 1.1). In Section 9.2, we discuss the ethical and privacy concerns for machine-learning and algorithmic decision-making systems. We conclude with an outlook on future research directions in Section 9.3.

9.1 Summary of Main Findings

The main objective of this thesis is to infer users' needs and tasks from their app usage interactions and then leverage the learned knowledge to enhance current mobile services for improving user experience. We summarize the main findings as follows.

9.1.1 Understanding App Usage Behaviour

There have been many works conducted on modelling users' app usage behaviour, especially focused on predicting which app users will use next [163, 98, 72, 157, 8, 203]. We start our work by investigating how could we leverage the cohort information for alleviating the *cold-start* problem when new users come. In Chapter 3, we design and conduct experiments to answer the following questions:

RQ 1.1: how could we model users' cohorts based on users' characteristics and logs readily available for mobile app usage?

RQ 1.2: could we employ signals from users who are similar along one or more dimensions, i.e., those in the same *cohort* for improving the prediction performance, especially for alleviating the *cold-start* problem?

Through our study, we demonstrate the value of cohorts, especially for new users. We firstly establish a comprehensive taxonomy to generate cohorts using logs readily available for mobile app usage from three aspects: demographics, psychographics, and behavioural patterns. We demonstrate that modelling user interests within these cohorts can enhance state-of-the-art app category usage prediction methods, leading to significant gains in the prediction performance. Additionally, our proposed cohort modelling method can effectively alleviate the user cold-start issues, especially when compared with the personalized prediction models. For a new user without much interaction data, general cohorts information such as interests or community could be collected, e.g., a user could label themselves as car lovers or young parents. Users' next app category usage could be predicted with relatively high accuracy using this basic cohort information. The cohort labels can also be utilized to explain the prediction model, enabling the recommendation to be more transparent and interpretable.

After answering the questions above, we could be able to predict the app preferred by users no matter they are new coming or not. Then in Chapter 4, we aim to further investigate that besides predicting which app users would use, if we can also model how long users would stay with apps. Intuitively, game apps, in general, have a higher probability to be used for a long period, whereas weather app is, not surprisingly, shorter. So we then state that it's not meaningful to only predict how long a user will stay regardless of which app user is engaging. Therefore we conduct comprehensive analysis and propose several strategies to answer the following questions:

RQ 1.3: What are the factors (user characteristics and contexts) that influence the app dwell time?

RQ 1.4: Can we predict which app users will use next and how long the user will stay on this app simultaneously?

To answer the research question **RQ 1.3**, we take a systematic approach to uncover the dependency of users' app usage duration on user characteristics (e.g., demographics) and context features (e.g., hours of the day, historical habit, last used app, etc.) based on a large-scale dataset. We show that the features related to the users' historical pattern, periodic behaviour pattern, and the recent usage pattern have more impacts when inferring users' app dwell time. For example, for the periodic pattern, a user prefers to spend a similar length of time to regularly check the shopping apps every day (i.e., after every 24 hours), probably for checking the updated discount or product information. This pattern could also be observed with books apps. Given the predictive factors we extracted for modelling app dwell time, we then propose three different joint prediction strategies: sequential, stacking and boosting, to solve another research question **RQ 1.4**, where the experimental results show that the boosting based strategy performs the best. We demonstrate that users' next app and the time spent could be effectively predicted at the same time.

9.1.2 Extracting and Characterizing Mobile Tasks

After conducting more comprehensive study for improving existing research on inferring users' app usage behaviour, we aim to further explore if we could infer users' tasks where users' needs may transcend the boundary of a single mobile app. Existing mobile systems handle mostly simple user needs, where a single app is taken as the unit of interaction. To understand users' expectations and to provide context-aware services, it is important to model users' interactions in the task space. No previous study has analyzed

or addressed the automatic identification of mobile tasks. In Chapter 5, by conducting a small-scale user study based on users' annotated tasks, we aim to answer the following research questions:

RQ 2.1: What kind of features can be used effectively to identify mobile tasks from app usage logs?

RQ 2.2: Can we formulate the task identification as a supervised learning problem, which could predict the app usage belong to the same task automatically?

To answer these questions above, we propose and evaluate a method for the automated segmentation of users' app usage logs into task units. We focus on two problems: (i) given a sequential pair of app usage logs, identify if there exists a task boundary, and (ii) given any pair of two app usage logs, identify if they belong to the same task. We model these as classification problems that use features from three aspects of app usage patterns: temporal, similarity, and log sequence. We show that these features used in combination can effectively predict mobile tasks. When used independently, log sequence features, which capture the hidden semantic relationship between apps perform best. The log sequence features cover the patterns such as if two apps are always used successively within the same session and the probability that a switch between two apps happens when accessed sequentially. Based on our proposed task identification models, we are able to map the large-scale app usage logs into task space automatically, which sets the stage for evaluating mobile apps and services, not on a per-app basis, but on the basis of user tasks.

After establishing the important first step in modelling mobile app usage from the task perspective within a small-scale study, in Chapter 6, we further aim to understand mobile tasks in the wild, especially focus on the

complex mobile tasks which get multiple different apps involved since they are more time-consuming and difficult for users to complete:

RQ 2.3: How to characterize complex mobile tasks based on different attributes?

RQ 2.4: Could we uncover the common patterns that exist in complex mobile tasks by dividing them into natural groups that reflect salient patterns?

For characterizing complex mobile tasks based on the large-scale commercial mobile app usage logs, a generic mobile app navigation model is firstly proposed to present an accurate picture for the micro-level interactions within this analysis, including how users revisit and switch between different apps. Then given the available features we could extract from the logs and the proposed navigation model, we analyze all mobile complex tasks from three aspects: task context (e.g. hours of the day), task complexity (e.g., number of different apps, task duration), and task content (e.g., app categories involved, switch between apps). To further uncover the common patterns that exist in all complex tasks based on the extracted characteristics, we employ the unsupervised learning approach to divide them into natural groups that reflect salient patterns. We observe that users generally perform 17 common tasks with 47 sub-tasks, ranging from “social media browsing" to “dining out" and “family entertainments". It provides us with a more fine-grained picture of the common complex mobile tasks users perform in the wild.

9.1.3 Leveraging Mobile Task Information

Based on our proposed mobile task identification model (Chapter 5) and the complex task clustering framework (Chapter 6), we are able to extract all complex mobile tasks from users' app usage logs, and also assign each task into a task type. We then intend to explore how the extracted task information could be leveraged in providing more satisfying services and improving user experience. Mobile user profiling is traditionally based on app interests, such as simply using app lists as features to predict the users' demographics. In Chapter 7, to apply the benefits brought by tasks, which can cover more detailed behaviour patterns and intentions while engaging with different apps, we aim to answer the following questions:

RQ 3.1: could we represent users not only based on app interests (used apps), but also on tasks, or task types?

RQ 3.2: would the task-based user representation methods benefit demographics prediction?

We firstly analyse the additional insights brought by tasks when compared to apps for distinguishing users in different ages and genders. We find that tasks could better capture the heterogeneity in user information and help us in modelling users. Then we propose different approaches for inferring users' representation both based on apps/app categories and task/task types. We finally demonstrate that the mobile user representation based on tasks with advanced neural models could effectively improve the performance of gender and age group prediction, especially when compared with the app-based user representation methods.

The experiment results of task-based user representation indicate that tasks have become a more accurate unit to capture users' goal and behaviour in-

sights, especially when compared to apps. Besides the traditional next app prediction problem, we believe that predicting which task the user would conduct next should be more important for improving mobile services. For the "Dining Out" task, if we only model users' behaviour based on independent apps, the mobile system may only be able to recommend a single food app to users, but have no ability to make the seamless integration with different related apps (navigation and transportation, etc.) under the task context. Therefore, in Chapter 8, we further aim to answer that:

RQ 3.3: Could we predict what the next complex task is by measuring users' app usage behaviour within task space?

Given the complex task clustering framework (Chapter 6), we are able to extract complex mobile tasks from users' app usage logs and assign a task type label to each complex task. We model the next complex task as a multi-class classification problem of 47 types (i.e., number of clusters) within the complex task space. We generate the predictive features by three different aggregation approaches based on users' historically engaged app categories and complex tasks. We then validate that the task-based representation of users' app usage behaviour could improve the complex task prediction on smartphones significantly.

Both of the task-based applications introduced above all provide preliminary steps in shaping future research on investigating whether and how the task information could be leveraged into different scenarios. More future work could be conducted by leveraging our task extraction and characterization methods.

9.2 Discussion on Ethical and Privacy Concerns for Artificial Intelligence (AI) Systems

Nowadays, more and more sectors increasingly turned to AI systems and machine learning algorithms to automate simple and complex decision-making processes [90]. The availability of massive data sets also made it easy to derive new insights through computers, and the algorithms have become more sophisticated and pervasive tools for automated decision-making [132]. As discussed in Section 7.5, previous research has shown that algorithms can exhibit and even amplify gender bias. While machine-learning algorithms enable companies to realize new efficiencies, in the case of self-learning systems, feeding biased data to self-learning systems can lead to unintended and sometimes dangerous outcomes [116]. For example, a racist machine that switched from tweeting that "humans are super cool" to praising Hitler and spewing out misogynistic remarks was created by Microsoft when they tried to converse with millennials via a chatbot plugged into Twitter famously created [38]. And this scary conclusion to a one-day experiment resulted from a very straightforward rule about machine learning, which learned exactly what the model was taught. Additionally, the harmful results would also be generated when the model was trained based on incomplete data. A machine-learning system[1] that makes recommendations for criminal sentencing, was proving imperfect at predicting which people are likely to re-offend, which is because the training model includes race as an input parameter, but not more extensive data points like past arrests. Nowadays, researchers are becoming more aware of the biases that these applications can contain and are attempting to address them. A tax-

[1]Correctional Offender Management Profiling for Alternative Sanctions (COMPAS)

onomy for fairness definitions has been defined by the machine learning researchers in order to avoid the existing bias in AI systems [116]. In our future research, we will also work on improving our mobile task based AI systems in terms of mitigating the bias and unfairness.

Additionally, reams of data from mobile phones and other online devices expand the volume, variety, and velocity of information about every facet of our lives and put privacy into the spotlight as a global public policy issue. As artificial intelligence evolves, it magnifies the ability to use personal information in ways that can intrude on privacy interests by raising analysis of personal information to new levels of power and speed [81]. Even the use of extensive personal data has been governed by modern data privacy guidelines, such as the EU's General Data Protection Regulation (GDPR) [177], where GDPR sets a specific requirement called data minimization, which means that organizations can collect only data that is necessary, how to address use personal information in artificial intelligence systems is still an open research question. In our study, all the data used were anonymized by removing all personally identifiable data prior to processing. It is a temptation to leverage data without thinking about the human cost, impact and outputs. As data modellers and algorithm designers, we always need to consider privacy, provenance, as well as the impact of each potential use and design. In our future work, more diverse representation in implementation and oversight will be further explored for protecting users' privacy.

9.3 Future Work

Users increasingly prefer to complete tasks with mobile devices in their daily life. In this thesis, we argue that modern mobile devices only effec-

tively serve many of the individual apps that correspond to simple mobile needs, users get little or no help when their needs transcend the boundary of a single mobile app. The tasks with high-level intentions especially for those span across multiple apps/services should be taken into consideration for providing more satisfying services to users. Specifically, this opens up many interesting and important directions for future work. We discuss each direction below.

9.3.1 Task enhanced Mobile Search

Future mobile search experiences should take tasks into account. Some researchers have investigated how to leverage task context information for query suggestion [24, 120] and ranking documents [189] in Web search. They validated that task information could provide better context for information retrieval (IR) systems to learn from. Different from the traditional IR on desktop, where their task context information could be extracted from search logs, Carrascal et al. [26] have stated that mobile search queries are mostly related to the apps used before and after. In this thesis, we have proposed the task identification model and a task clustering framework which could extract tasks from app usage logs automatically and further group them into different categories with the semantic label assigned. Furthermore, we also proposed the task prediction model which could also provide task context for future search. So how to leverage the mobile task context extracted from app usage for improving mobile search should be explored in future work.

Firstly, given the task extracted from apps used before a search, the mobile system could offer more proactive search experiences under the task context and essentially supporting users in task continuation and task completion.

A task context embedding architecture to learn the representation of queries by leveraging the task context information from historical app usage and search logs could be explored. For example, when the task of "Multiple Updated Info Check" has been learned, the user aims to collect all the latest information from different sources, not a specific app. Even the user only engaged with a finance app just now and has no search actions before, if he searched "USPS", we should know that he/she probably wants to get the latest tracking information under the current task context rather than recommendations of the most popular post services.

Additionally, the task-based approach for addressing unseen queries and new search scenarios should also be explored. Personalized search systems always rely on being able to find pertinent information in that user's search history, which can be challenging for unseen queries and new search scenarios. It would be a more serious problem in mobile search since most queries may be resulted from apps usage, not the historical search behaviour [26]. By leveraging the task information extracted from the app usage logs, we can build richer models of users' current and historical context which may help improve the likelihood of finding relevant content and enhance the relevance and coverage of personalization methods. The historical tasks extracted could be mined to find other users performing similar tasks to the current user and leverage their on-task behaviour to identify Web pages to promote in the current searched results ranking. The validation of benefits brought by task-based modelling of app usage behaviour in mobile search could be explored in future work when the mobile search data is also available.

9.3.2 Task Supported Intelligent Assistants (IAs)

Understanding users' behaviours in the task space is an essential part to improve the intelligent agent services (e.g., Siri, Google Now and Microsoft Cortana) on mobile devices in the future. Such systems should make use of a plethora of signals including user's interactions and contextual information to provide assistance by making recommendations and performing actions. However, as stated by [160], current IAs tend to be limited to specific apps. Their experiments showed that understanding high-level tasks allows the agent to actively suggest relevant apps, which helps users pursue particular goals and reduces the cost of users' self-management. Furthermore, with improved speech recognition systems, more and more users are increasingly relying on such digital assistants to fulfil their request given a user's request expressed, in language, at the level of tasks. Upon receiving "can you help me book a table in May Restaurant at 7:00 p.m. and send me the directions to get there?", the IA should open the OpenTable Reservations app for booking the table and then open the GoogleMaps to get the directions to that restaurant, finally display the direction on the home screen or send it to users' messages box. By doing so, we could potentially simplify the process of launching apps one by one, which is time-consuming and difficult for users, especially for elders and ones with (visual) disabilities.

By leveraging the task identification, clustering and prediction methods proposed in this thesis, we are able to extract and predict the high-level tasks users aim to conduct while engaging with the mobile devices. However, how to map the extracted tasks to the language level of tasks and a series of operations/services provided to users interactively should be investigated in the future. The most straightforward way for mapping the

extracted tasks to language expression would be using some string similarity metrics (e.g., Levenstein edit distance, n-gram Jaccard) to measure the similarity between task labels and users' expressions. Additionally, as we also proposed some embedding architecture to represent the semantic meaning within tasks, i.e., Task2Vec (Chapter 7), more natural language processing (NLP) knowledge could be leveraged for learning the similarity between users' expression and tasks extracted. Then for mapping the task to a set of operations that need to be conducted by IAs automatically, we actually have provided many hints from the bottom-up perspective, i.e., by knowing which hub app, dominant app, and switch patterns, have been covered within a task, we could assign a task label to them. So how to map the task to operations within a top-down manner (i.e., knowing the task and then mapping the task to recommended operations) could be further explored in the future.

9.3.3 Implication on Future Data and Technologies

9.3.3.1 Data

In general, training data for machine learning projects has to be representative of the real world. However, data bias can occur in a range of areas, from human reporting and selection bias to algorithmic and interpretation bias. Similar to all other real-world app usage datasets, the dataset used in our study was also limited in representing the entire population of mobile users. As the users and apps were only coming from the apps registered in a specific library, it means that not all the apps usage behaviour of users could be tracked and there may be a selection bias in the subset of users being studied. While this might occur to some extent, given the scale of our dataset (with over 1.3 million logs), we believe our data would still

provide useful insights, and our predictive models are the best models so far, effective for most users.

As more representative and larger datasets would be collected, we generated a webpage for sharing the detailed annotation procedure and the annotated data as reference for helping the researchers validate our work in the future. All the information, including full instructions of annotation guidelines, how labellers will be filtered, the design of the annotation page on Amazon Mechanical Turk and analysed statistics of annotated results, can be accessed at https://mobile-task-annotation.github.io/. A more representative dataset and improved annotation processes not only boost the accuracy of the model, but can also alleviate the issues of ethics, fairness, and inclusion. We strongly suggest that future researchers keep working on handling the dataset to avoid bias where possible.

9.3.3.2 Technologies

In this section, we take the emerging split-screen feature on mobile devices as an example to discuss how the changes in technologies might affect future work in our related research areas. Besides having more space to view the apps, the trend for bigger screens on mobile devices also opens up the possibility to open more than one app in split-screen mode. On Android, the split-screen mode has become a native feature that allows users to view two apps side by side on the phone screen [76]. It not only allows users to open more than one app simultaneously but also enables those apps to run simultaneously in some cases.

The split-screen mode and our proposed complex mobile tasks complement each other. For instance, a user might open up a finance-related app such as a spreadsheet or bank app and then open up a calculator or another

finance app. All of the information that might be needed to set up a budget or check balances can, effectively, be open and on-screen at once. This task has been reflected as the complex task C7-6 Tool Cooperation in Section 6.2.3.2. Within the split-screen mode, users just need to navigate each as they normally would but without alternating between different apps through the Recent Apps view. For another example, a user might open up a YouTube video or some other media app on their smartphone. If a text message or other message comes in, they can then use the split-screen tool to respond in a larger format than the notifications shade allows for, all while continuing to watch or listen. This task has also been reflected in the complex task C13-31 TV Watching with Communication within Section 6.2.3.2.

With the increasing development of split-screen feature and other technologies for helping users engage with more than one app at the same time, how to infer users' needs and tasks, in particular those complex tasks with at least two apps involved, will not only be based on the app usage logs, more touch screen gestures (e.g., scroll down/up and zoom in/out) and more fine-grained interaction patterns should be taken into consideration. Specifically, the switch and revisit patterns among app usage were originally important features for identifying mobile tasks since apps need to be accessed sequentially for completing the task. However, as the interaction method with apps changed according to the split-screen mode, where the information of two apps can be shown to the users at the same time without switching between them, researchers and developers should pay more attention on how to measure users' app usage behaviour in different approaches instead of only based on the sequence of apps usage, additional features for understanding users' app usage behaviour should be further explored under the new interactive mobile environment.

Bibliography

[1] Adomavicius, G., and Tuzhilin, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge & Data Engineering*, 6 (2005), 734–749.

[2] Agichtein, E., Brill, E., and Dumais, S. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval* (2006), ACM, pp. 19–26.

[3] Ahmad, A., and Khan, S. S. Survey of state-of-the-art mixed data clustering algorithms. *IEEE Access 7* (2019), 31883–31902.

[4] AI, E. Spacy similarity, 2016.

[5] Amazon Mechanical Turk, I. Amazon mechanical turk, 2005.

[6] Apple. App store - apple, 2021.

[7] Araujo, T., Helberger, N., Kruikemeier, S., and De Vreese, C. H. In ai we trust? perceptions about automated decision-making by artificial intelligence. *AI & SOCIETY 35*, 3 (2020), 611–623.

[8] Baeza-Yates, R., Jiang, D., Silvestri, F., and Harrison, B. Predicting the next app that you are going to use. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining* (2015), ACM, pp. 285–294.

[9] Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[10] Barnes, S. J. The mobile commerce value chain: analysis and future developments. *International journal of information management 22*, 2 (2002), 91–108.

[11] Beane, T., and Ennis, D. Market segmentation: a review. *European journal of marketing* (1987).

[12] Bilenko, M., and White, R. W. Mining the search trails of surfing crowds: identifying relevant websites from user activity. In *Proceedings of the 17th international conference on World Wide Web* (2008), ACM, pp. 51–60.

[13] Bishop, C. M. *Pattern recognition and machine learning*. springer, 2006.

[14] Blair, I. Mobile app download and usage statistics (2021), 2021.

[15] Bogina, V., and Kuflik, T. Incorporating dwell time in session-based recommendations with recurrent neural networks. In *RecTemp@ RecSys* (2017), pp. 57–59.

[16] Böhmer, M., Hecht, B., Schöning, J., Krüger, A., and Bauer, G. Falling asleep with angry birds, facebook and kindle: a large scale study on mobile application usage. In *Proceedings of the*

13th international conference on Human computer interaction with mobile devices and services (2011), ACM, pp. 47–56.

[17] Boldi, P., Bonchi, F., Castillo, C., Donato, D., Gionis, A., and Vigna, S. The query-flow graph: model and applications. In *Proceedings of the 17th ACM conference on Information and knowledge management* (2008), pp. 609–618.

[18] Bolukbasi, T., Chang, K.-W., Zou, J. Y., Saligrama, V., and Kalai, A. T. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *Advances in neural information processing systems 29* (2016).

[19] Brdar, S., Culibrk, D., and Crnojevic, V. Demographic attributes prediction on the real-world mobile data. In *Proc. Mobile Data Challenge by Nokia Workshop, in Conjunction with Int. Conf. on Pervasive Computing, Newcastle, UK* (2012).

[20] Breiman, L. Random forests. *Machine learning 45*, 1 (2001), 5–32.

[21] Burges, C. J. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery 2*, 2 (1998), 121–167.

[22] Byström, K., and Hansen, P. Work tasks as units for analysis in information seeking and retrieval studies. *Emerging frameworks and methods* (2002), 239–251.

[23] Caliskan, A., Bryson, J. J., and Narayanan, A. Semantics derived automatically from language corpora contain human-like biases. *Science 356*, 6334 (2017), 183–186.

[24] Cao, H., Jiang, D., Pei, J., He, Q., Liao, Z., Chen, E., and Li, H. Context-aware query suggestion by mining click-through and

session data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (2008), pp. 875–883.

[25] Cao, H., and Lin, M. Mining smartphone data for app usage prediction and recommendations: A survey. *Pervasive and Mobile Computing 37* (2017), 1–22.

[26] Carrascal, J. P., and Church, K. An in-situ study of mobile app & mobile search interactions. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (2015), ACM, pp. 2739–2748.

[27] Caruana, R., Lawrence, S., and Giles, L. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. *Advances in neural information processing systems* (2001), 402–408.

[28] Catledge, L. D., and Pitkow, J. E. Characterizing browsing strategies in the world-wide web. *Computer Networks and ISDN systems 27*, 6 (1995), 1065–1073.

[29] Chen, T., and Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (2016), pp. 785–794.

[30] Chen, T., He, T., Benesty, M., Khotilovich, V., and Tang, Y. Xgboost: extreme gradient boosting. *R package version 0.4-2* (2015), 1–4.

[31] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[32] Church, K. W., and Hanks, P. Word association norms, mutual information, and lexicography. *Computational linguistics 16*, 1 (1990), 22–29.

[33] Cover, T. M., Hart, P. E., et al. Nearest neighbor pattern classification. *IEEE transactions on information theory 13*, 1 (1967), 21–27.

[34] Covington, P., Adams, J., and Sargin, E. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems* (2016), ACM, pp. 191–198.

[35] Cox, D. R. The statistical analysis of series of events. *Monographs on Applied Probability and Statistics* (1966).

[36] Cui, J., Liu, H., Yan, J., Ji, L., Jin, R., He, J., Gu, Y., Chen, Z., and Du, X. Multi-view random walk framework for search task discovery from click-through log. In *Proceedings of the 20th ACM international conference on Information and knowledge management* (2011), pp. 135–140.

[37] Day, M. How linkedin's search engine may reflect a gender bias. *The Seattle Times 31* (2016).

[38] DeBrusk, C. The risk of machine-learning bias (and how to prevent it). *MIT Sloan Management Review* (2018).

[39] Diaz-Aviles, E., Lam, H. T., Pinelli, F., Braghin, S., Gkoufas, Y., Berlingerio, M., and Calabrese, F. Predicting user engagement in twitter with collaborative ranking. In *Proceedings of the 2014 Recommender Systems Challenge* (2014), ACM, p. 41.

[40] Do, T. M. T., Blom, J., and Gatica-Perez, D. Smartphone usage in the wild: a large-scale analysis of applications and context.

In *Proceedings of the 13th international conference on multimodal interfaces* (2011), ACM, pp. 353–360.

[41] Do, T. M. T., and Gatica-Perez, D. Where and what: Using smartphones to predict next locations and applications in daily life. *Pervasive and Mobile Computing 12* (2014), 79–91.

[42] Drucker, H., Cortes, C., Jackel, L. D., LeCun, Y., and Vapnik, V. Boosting and other ensemble methods. *Neural Computation 6*, 6 (1994), 1289–1301.

[43] Ehmke, C., and Wilson, S. Identifying web usability problems from eye-tracking data. In *Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI... but not as we know it-Volume 1* (2007), British Computer Society, pp. 119–128.

[44] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd* (1996), vol. 96, pp. 226–231.

[45] Falaki, H., Mahajan, R., Kandula, S., Lymberopoulos, D., Govindan, R., and Estrin, D. Diversity in smartphone usage. In *Proceedings of the 8th international conference on Mobile systems, applications, and services* (2010), ACM, pp. 179–194.

[46] Ferreira, D., Goncalves, J., Kostakos, V., Barkhuus, L., and Dey, A. K. Contextual experience sampling of mobile application micro-usage. In *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services* (2014), ACM, pp. 91–100.

[47] Fix, E., and Hodges Jr, J. L. Discriminatory analysis-nonparametric discrimination: consistency properties. Tech. rep., California Univ Berkeley, 1951.

[48] Fleiss, J. L. Measuring nominal scale agreement among many raters. *Psychological bulletin 76*, 5 (1971), 378.

[49] Fox, S., Karnawat, K., Mydland, M., Dumais, S., and White, T. Evaluating implicit measures to improve web search. *ACM Transactions on Information Systems (TOIS) 23*, 2 (2005), 147–168.

[50] Friedman, N., Geiger, D., and Goldszmidt, M. Bayesian network classifiers. *Machine learning 29*, 2 (1997), 131–163.

[51] Fu, B., Lin, J., Li, L., Faloutsos, C., Hong, J., and Sadeh, N. Why people hate your app: Making sense of user feedback in a mobile app store. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (2013), pp. 1276–1284.

[52] Garigliotti, D., and Balog, K. Generating query suggestions to support task-based search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2017), pp. 1153–1156.

[53] Gelman, A. Scaling regression inputs by dividing by two standard deviations. *Statistics in medicine 27*, 15 (2008), 2865–2873.

[54] Girardello, A., and Michahelles, F. Appaware: which mobile applications are hot? In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services* (2010), pp. 431–434.

[55] Google. Google play, May 2018.

[56] Google. Select a category for your app or game, May 2018.

[57] Google. Google play app store - google play apps, 2021.

[58] Gower, J. C. A general coefficient of similarity and some of its properties. *Biometrics* (1971), 857–871.

[59] Hamka, F., Bouwman, H., De Reuver, M., and Kroesen, M. Mobile customer segmentation based on smartphone measurement. *Telematics and Informatics 31*, 2 (2014), 220–227.

[60] Hassan Awadallah, A., Gurrin, C., Sanderson, M., and White, R. W. Task intelligence workshop@ wsdm 2019. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining* (2019), pp. 848–849.

[61] Hassan Awadallah, A., White, R. W., Pantel, P., Dumais, S. T., and Wang, Y.-M. Supporting complex search tasks. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management* (2014), pp. 829–838.

[62] Hastie, T., Tibshirani, R., and Friedman, J. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[63] Hidasi, B., and Tikk, D. General factorization framework for context-aware recommendations. *Data Mining and Knowledge Discovery 30*, 2 (2016), 342–371.

[64] Hintze, D., Findling, R. D., Muaaz, M., Scholz, S., and Mayrhofer, R. Diversity in locked and unlocked mobile device

usage. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication* (2014), pp. 379–384.

[65] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

[66] Homma, R., Soejima, K., Yoshida, M., and Umemura, K. Analysis of user dwell time on non-news pages. In *2018 IEEE International Conference on Big Data (Big Data)* (2018), IEEE, pp. 4333–4338.

[67] Hosmer Jr, D. W., Lemeshow, S., and Sturdivant, R. X. *Applied logistic regression*, vol. 398. John Wiley & Sons, 2013.

[68] Hu, J., Zeng, H.-J., Li, H., Niu, C., and Chen, Z. Demographic prediction based on user's browsing behavior. In *Proceedings of the 16th international conference on World Wide Web* (2007), ACM, pp. 151–160.

[69] Hua, W., Song, Y., Wang, H., and Zhou, X. Identifying users' topical tasks in web search. In *Proceedings of the sixth ACM international conference on Web search and data mining* (2013), pp. 93–102.

[70] Huang, J., and Efthimiadis, E. N. Analyzing and evaluating query reformulation strategies in web search logs. In *Proceedings of the 18th ACM conference on Information and knowledge management* (2009), pp. 77–86.

[71] Huang, J., White, R. W., and Dumais, S. No clicks, no problem: using cursor movements to understand and improve search. In *Proceedings of the SIGCHI conference on human factors in computing systems* (2011), ACM, pp. 1225–1234.

[72] Huang, K., Zhang, C., Ma, X., and Chen, G. Predicting mobile application usage using contextual information. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing* (2012), ACM, pp. 1059–1065.

[73] Huang, Z. Clustering large data sets with mixed numeric and categorical values. In *Proceedings of the 1st pacific-asia conference on knowledge discovery and data mining,(PAKDD)* (1997), Citeseer, pp. 21–34.

[74] Huang, Z. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data mining and knowledge discovery 2*, 3 (1998), 283–304.

[75] Hunter, J. S. The exponentially weighted moving average. *Journal of quality technology 18*, 4 (1986), 203–210.

[76] Jackson, W. An introduction to android 7.0 nougat. In *Android Apps for Absolute Beginners*. Springer, 2017, pp. 1–15.

[77] Jiang, J., He, D., and Allan, J. Searching, browsing, and clicking in a search session: Changes in user behavior by task and over time. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval* (2014), pp. 607–616.

[78] Jones, R., and Klinkner, K. L. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *Proceedings of the 17th ACM conference on Information and knowledge management* (2008), pp. 699–708.

[79] Jones, S. L., Ferreira, D., Hosio, S., Goncalves, J., and Kostakos, V. Revisitation analysis of smartphone app use. In *Pro-*

ceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (2015), ACM, pp. 1197–1208.

[80] Kelly, D., and Belkin, N. J. Display time as implicit feedback: understanding task effects. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval* (2004), ACM, pp. 377–384.

[81] Kerry, C., et al. Protecting privacy in an ai-driven world. *https://www. brookings. edu/research/protecting-privacy-in-an-ai-driven-world/* (2020).

[82] Kim, Y. Convolutional neural networks for sentence classification. emnlp, 2014.

[83] Kim, Y. H., Kim, D. J., and Wachter, K. A study of mobile user engagement (moen): Engagement motivations, perceived value, satisfaction, and continued engagement intention. *Decision Support Systems 56* (2013), 361–370.

[84] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[85] Kooti, F., Grbovic, M., Aiello, L. M., Bax, E., and Lerman, K. iphone's digital marketplace: Characterizing the big spenders. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining* (2017), ACM, pp. 13–21.

[86] Kotov, A., Bennett, P. N., White, R. W., Dumais, S. T., and Teevan, J. Modeling and analysis of cross-session search tasks. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval* (2011), pp. 5–14.

[87] Landis, J. R., and Koch, G. G. The measurement of observer agreement for categorical data. *biometrics* (1977), 159–174.

[88] Le, Q., and Mikolov, T. Distributed representations of sentences and documents. In *International conference on machine learning* (2014), PMLR, pp. 1188–1196.

[89] LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature 521*, 7553 (2015), 436–444.

[90] Lee, N. T., Resnick, P., and Barton, G. Algorithmic bias detection and mitigation: Best practices and policies to reduce consumer harms. *Brookings Institute: Washington, DC, USA* (2019).

[91] Lee, Y., Park, I., Cho, S., and Choi, J. Smartphone user segmentation based on app usage sequence with neural networks. *Telematics and Informatics 35*, 2 (2018), 329–339.

[92] Lehmann, J., Lalmas, M., Yom-Tov, E., and Dupret, G. Models of user engagement. In *International Conference on User Modeling, Adaptation, and Personalization* (2012), Springer, pp. 164–175.

[93] Li, H., and Lu, X. Mining device-specific apps usage patterns from large-scale android users. *arXiv preprint arXiv:1707.09252* (2017).

[94] Li, H., Lu, X., Liu, X., Xie, T., Bian, K., Lin, F. X., Mei, Q., and Feng, F. Characterizing smartphone usage patterns from millions of android users. In *Proceedings of the 2015 Internet Measurement Conference* (2015), ACM, pp. 459–472.

[95] Li, H., Xu, Z., Zhu, H., Ma, D., Li, S., and Xing, K. Demographics inference through wi-fi network traffic analysis. In *Computer*

Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on (2016), IEEE, pp. 1–9.

[96] Li, L., Mei, T., Niu, X., and Ngo, C.-W. Pagesense: style-wise web page advertising. In *Proceedings of the 19th international conference on World wide web* (2010), pp. 1273–1276.

[97] Li, Y., and Belkin, N. J. A faceted approach to conceptualizing tasks in information seeking. *Information Processing & Management 44*, 6 (2008), 1822–1837.

[98] Liao, Z.-X., Lei, P.-R., Shen, T.-J., Li, S.-C., and Peng, W.-C. Mining temporal profiles of mobile applications for usage prediction. In *Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on* (2012), IEEE, pp. 890–893.

[99] Liao, Z.-X., Li, S.-C., Peng, W.-C., Philip, S. Y., and Liu, T.-C. On the feature discovery for app usage prediction in smartphones. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on* (2013), IEEE, pp. 1127–1132.

[100] Liao, Z.-X., Pan, Y.-C., Peng, W.-C., and Lei, P.-R. On mining mobile apps usage behavior for predicting apps usage in smartphones. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management* (2013), ACM, pp. 609–618.

[101] Lim, S. L., Bentley, P. J., Kanakam, N., Ishikawa, F., and Honiden, S. Investigating country differences in mobile app user behavior and challenges for software engineering. *IEEE Transactions on Software Engineering 41*, 1 (2014), 40–64.

[102] Lin, J., Sugiyama, K., Kan, M.-Y., and Chua, T.-S. Addressing cold-start in app recommendation: latent user models constructed

from twitter followers. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval* (2013), ACM, pp. 283–292.

[103] Liu, J., Cole, M. J., Liu, C., Bierig, R., Gwizdka, J., Belkin, N. J., Zhang, J., and Zhang, X. Search behaviors in different task types. In *Proceedings of the 10th annual joint conference on Digital libraries* (2010), pp. 69–78.

[104] Liu, J., Gwizdka, J., Liu, C., and Belkin, N. J. Predicting task difficulty for different task types. *Proceedings of the American Society for Information Science and Technology 47*, 1 (2010), 1–10.

[105] Liu, X., Ai, W., Li, H., Tang, J., Huang, G., Feng, F., and Mei, Q. Deriving user preferences of mobile apps from their management activities. *ACM Transactions on Information Systems (TOIS) 35*, 4 (2017), 1–32.

[106] Liu, X., Yang, Q., and He, L. A novel dbscan with entropy and probability for mixed data. *Cluster Computing 20*, 2 (2017), 1313–1323.

[107] Liu, Y., Li, Z., Xiong, H., Gao, X., and Wu, J. Understanding of internal clustering validation measures. In *2010 IEEE International Conference on Data Mining* (2010), IEEE, pp. 911–916.

[108] Logg, J. M., Minson, J. A., and Moore, D. A. Algorithm appreciation: People prefer algorithmic to human judgment. *Organizational Behavior and Human Decision Processes 151* (2019), 90–103.

[109] Lucchese, C., Orlando, S., Perego, R., Silvestri, F., and Tolomei, G. Identifying task-based sessions in search engine query

logs. In *Proceedings of the fourth ACM international conference on Web search and data mining* (2011), pp. 277–286.

[110] MacQueen, J., et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (1967), vol. 1, Oakland, CA, USA, pp. 281–297.

[111] Malmi, E., and Weber, I. You are what apps you use: Demographic prediction based on user's apps. In *ICWSM* (2016), pp. 635–638.

[112] Martínez, N., Vinas, A., and Matute, H. Examining potential gender bias in automated-job alerts in the spanish market. *Plos one 16*, 12 (2021), e0260409.

[113] Mathur, A., Lane, N. D., and Kawsar, F. Engagement-aware computing: Modelling user engagement from mobile contexts. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (2016), ACM, pp. 622–633.

[114] Maulik, U., and Bandyopadhyay, S. Performance evaluation of some clustering algorithms and validity indices. *IEEE Transactions on pattern analysis and machine intelligence 24*, 12 (2002), 1650–1654.

[115] May, A., Wachs, J., and Hannák, A. Gender differences in participation and reward on stack overflow. *Empirical Software Engineering 24*, 4 (2019), 1997–2019.

[116] Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., and Galstyan, A. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR) 54*, 6 (2021), 1–35.

[117] Mehrotra, A., Pejovic, V., Vermeulen, J., Hendley, R., and Musolesi, M. My phone and me: understanding people's receptivity to mobile notifications. In *Proceedings of the 2016 CHI conference on human factors in computing systems* (2016), ACM, pp. 1021–1032.

[118] Mehrotra, R. Inferring user needs & tasks from user interactions. In *ACM SIGIR Forum* (2019), vol. 52, ACM New York, NY, USA, pp. 176–177.

[119] Mehrotra, R., Awadallah, A. H., Shokouhi, M., Yilmaz, E., Zitouni, I., El Kholy, A., and Khabsa, M. Deep sequential models for task satisfaction prediction. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (2017), pp. 737–746.

[120] Mehrotra, R., and Yilmaz, E. Task embeddings: Learning query embeddings using task context. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (2017), pp. 2199–2202.

[121] Mehrotra, R., Yilmaz, E., and Verma, M. Task-based user modelling for personalization via probabilistic matrix factorization. In *RecSys Posters* (2014), Citeseer.

[122] Mehrotra, R., Zitouni, I., Hassan Awadallah, A., Kholy, A. E., and Khabsa, M. User interaction sequences for search satisfaction prediction. In *Proceedings of the 40th International ACM SIGIR conference on research and development in information retrieval* (2017), pp. 165–174.

[123] Microsoft. Windows apps – microsoft store, 2021.

[124] Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[125] Mitsui, M., Liu, J., and Shah, C. How much is too much? whole session vs. first query behaviors in task type prediction. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (2018), pp. 1141–1144.

[126] Mobile, N. Critical mass: The worldwide state of the mobile web. *Nielsen Company* (2008).

[127] Mukherjee, S., Mukherjee, S., Hasegawa, M., Awadallah, A. H., and White, R. Smart to-do: Automatic generation of to-do items from emails. *arXiv preprint arXiv:2005.06282* (2020).

[128] Murnane, E. L., Abdullah, S., Matthews, M., Kay, M., Kientz, J. A., Choudhury, T., Gay, G., and Cosley, D. Mobile manifestations of alertness: Connecting biological rhythms with patterns of smartphone app use. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services* (2016), ACM, pp. 465–477.

[129] Musto, C., Semeraro, G., De Gemmis, M., and Lops, P. Word embedding techniques for content-based recommender systems: An empirical evaluation. In *Recsys posters* (2015).

[130] Nadeem, S., and Weigle, M. C. Demographic prediction of mobile user from phone usage. *Age 1* (2012), 16–21.

[131] Natarajan, N., Shin, D., and Dhillon, I. S. Which app will you use next?: collaborative filtering with interactional context. In *Pro-*

ceedings of the 7th ACM conference on Recommender systems (2013), ACM, pp. 201–208.

[132] Nateghi, R., and Aven, T. Risk analysis in the age of big data: The promises and pitfalls. *Risk Analysis 41*, 10 (2021), 1751–1758.

[133] Neal, T. J., and Woodard, D. L. A gender-specific behavioral analysis of mobile device usage data. In *Identity, Security, and Behavior Analysis (ISBA), 2018 IEEE 4th International Conference on* (2018), IEEE, pp. 1–8.

[134] Nelissen, K., Snoeck, M., Broucke, S. V., and Baesens, B. Swipe and tell: Using implicit feedback to predict user engagement on tablets. *ACM Transactions on Information Systems (TOIS) 36*, 4 (2018), 1–36.

[135] Pan, W., Aharony, N., and Pentland, A. Composite social network for predicting mobile apps installation. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2011), vol. 25.

[136] Parate, A., Böhmer, M., Chu, D., Ganesan, D., and Marlin, B. M. Practical prediction and prefetch for faster access to applications on mobile phones. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing* (2013), pp. 275–284.

[137] Pejovic, V., and Musolesi, M. Interruptme: designing intelligent prompting mechanisms for pervasive applications. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (2014), ACM, pp. 897–908.

[138] Pérez-Torres, R., Torres-Huitzil, C., and Galeana-Zapién, H. Power management techniques in smartphone-based mo-

bility sensing systems: A survey. *Pervasive and Mobile Computing 31* (2016), 1–21.

[139] Petsas, T., Papadogiannakis, A., Polychronakis, M., Markatos, E. P., and Karagiannis, T. Measurement, modeling, and analysis of the mobile app ecosystem. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS) 2*, 2 (2017), 7.

[140] Philip, G., and Ottaway, B. Mixed data cluster analysis: an illustration using cypriot hooked-tang weapons. *Archaeometry 25*, 2 (1983), 119–133.

[141] Pielot, M., Church, K., and De Oliveira, R. An in-situ study of mobile phone notifications. In *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services* (2014), ACM, pp. 233–242.

[142] Pincus, S. M. Approximate entropy as a measure of system complexity. *Proceedings of the National Academy of Sciences 88*, 6 (1991), 2297–2301.

[143] Quinlan, M. L. *Just ask a woman: Cracking the code of what women want and how they buy.* John Wiley & Sons, 2003.

[144] Radlinski, F., and Joachims, T. Query chains: learning to rank from implicit feedback. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining* (2005), pp. 239–248.

[145] Rao, G., Huang, W., Feng, Z., and Cong, Q. Lstm with sentence representations for document-level sentiment classification. *Neurocomputing 308* (2018), 49–57.

[146] Rawassizadeh, R., Momeni, E., Dobbins, C., Mirza-Babaei, P., and Rahnamoun, R. Lesson learned from collecting quantified self information via mobile and wearable devices. *Journal of Sensor and Actuator Networks 4*, 4 (2015), 315–335.

[147] Rawassizadeh, R., Tomitsch, M., Wac, K., and Tjoa, A. M. Ubiqlog: a generic mobile phone-based life-log framework. *Personal and ubiquitous computing 17*, 4 (2013), 621–637.

[148] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work* (1994), ACM, pp. 175–186.

[149] Revels, J., Tojib, D., and Tsarenko, Y. Understanding consumer intention to use mobile services. *Australasian Marketing Journal (AMJ) 18*, 2 (2010), 74–80.

[150] Rivron, V., Khan, M. I., Charneau, S., and Chrisment, I. Exploring smartphone application usage logs with declared sociological information. In *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom)(BDCloud-SocialCom-SustainCom)* (2016), IEEE, pp. 266–273.

[151] Said, A., De Luca, E. W., and Albayrak, S. How social relationships affect user similarities. In *Proceedings of the International Conference on Intelligent User Interfaces Workshop on Social Recommender Systems, Hong Kong* (2010).

[152] Schafer, J. B., Frankowski, D., Herlocker, J., and Sen, S. Collaborative filtering recommender systems. In *The adaptive web*. Springer, 2007, pp. 291–324.

[153] Seki, Y., and Yoshida, M. Analysis of user dwell time by category in news application. In *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)* (2018), IEEE, pp. 732–735.

[154] Seneviratne, S., Seneviratne, A., Mohapatra, P., and Mahanti, A. Your installed apps reveal your gender and more! *ACM SIGMOBILE Mobile Computing and Communications Review 18*, 3 (2015), 55–61.

[155] Shah, C., and White, R. W. Task intelligence for search and recommendation. *Synthesis Lectures onSynthesis Lectures on Information Concepts, Retrieval, and Services 13*, 3 (2021), 1–160.

[156] Sharpe, D. Chi-square test is statistically significant: Now what? *Practical Assessment, Research, and Evaluation 20*, 1 (2015), 8.

[157] Shin, C., Hong, J.-H., and Dey, A. K. Understanding and prediction of mobile application usage for smart phones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing* (2012), ACM, pp. 173–182.

[158] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research 15*, 1 (2014), 1929–1958.

[159] Su, X., and Khoshgoftaar, T. M. A survey of collaborative filtering techniques. *Advances in artificial intelligence 2009* (2009).

[160] Sun, M., Chen, Y.-N., and Rudnicky, A. I. Learning user intentions spanning multiple domains. In *Workshop on Smart Connected and Wearable Things 2016* (2016), p. 31.

[161] Sweeney, L. Discrimination in online ad delivery. *Communications of the ACM 56*, 5 (2013), 44–54.

[162] Syarif, I., Zaluska, E., Prugel-Bennett, A., and Wills, G. Application of bagging, boosting and stacking to intrusion detection. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition* (2012), Springer, pp. 593–602.

[163] Tan, C., Liu, Q., Chen, E., and Xiong, H. Prediction for mobile application usage patterns. In *Nokia MDC Workshop* (2012), vol. 12.

[164] Tavakol, M., and Brefeld, U. Factored mdps for detecting topics of user sessions. In *Proceedings of the 8th ACM Conference on Recommender Systems* (2014), pp. 33–40.

[165] Tian, Y., Zhou, K., Lalmas, M., Liu, Y., and Pelleg, D. Cohort modeling based app category usage prediction. In *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization* (2020), pp. 248–256.

[166] Tian, Y., Zhou, K., Lalmas, M., and Pelleg, D. Identifying tasks from mobile app usage patterns. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (2020), pp. 2357–2366.

[167] Tian, Y., Zhou, K., and Pelleg, D. Characterization and prediction of mobile tasks. Under review by ACM Transactions on Information Systems (TOIS).

[168] Tian, Y., Zhou, K., and Pelleg, D. What and how long: Prediction of mobile app engagement. *arXiv preprint arXiv:2106.01490* (2021).

[169] Trippas, J. R., Spina, D., Scholer, F., Awadallah, A. H., Bailey, P., Bennett, P. N., White, R. W., Liono, J., Ren, Y., Salim, F. D., et al. Learning about work tasks to inform intelligent assistant design. In *Proceedings of the 2019 Conference on Human Information Interaction and Retrieval* (2019), pp. 5–14.

[170] Tu, Z., Li, R., Li, Y., Wang, G., Wu, D., Hui, P., Su, L., and Jin, D. Your apps give you away: Distinguishing mobile users by their app usage fingerprints. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 2*, 3 (2018), 138.

[171] van Berkel, N., Luo, C., Anagnostopoulos, T., Ferreira, D., Goncalves, J., Hosio, S., and Kostakos, V. A systematic assessment of smartphone usage gaps. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (2016), pp. 4711–4721.

[172] Van Canneyt, S., Bron, M., Haines, A., and Lalmas, M. Describing patterns and disruptions in large scale mobile app usage data. In *Proceedings of the 26th International Conference on World Wide Web Companion* (2017), International World Wide Web Conferences Steering Committee, pp. 1579–1584.

[173] van Vugt, H. C., Konijn, E. A., Hoorn, J. F., Keur, I., and Eliéns, A. Realism is not all! user engagement with task-related interface characters. *Interacting with Computers 19*, 2 (2007), 267–280.

[174] Vapnik, V. *The nature of statistical learning theory*. Springer science & business media, 2013.

[175] Vasiloudis, T., Vahabi, H., Kravitz, R., and Rashkov, V. Predicting session length in media streaming. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2017), pp. 977–980.

[176] Viermetz, M., Stolz, C., Gedov, V., Skubacz, M., et al. Relevance and impact of tabbed browsing behavior on web usage mining. In *Web Intelligence* (2006), vol. 2006.

[177] Voigt, P., and Von dem Bussche, A. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing 10*, 3152676 (2017), 10–5555.

[178] Wang, H., Song, Y., Chang, M.-W., He, X., White, R. W., and Chu, W. Learning to extract cross-session search tasks. In *Proceedings of the 22nd international conference on World Wide Web* (2013), pp. 1353–1364.

[179] Wang, T., Chen, J., Zhuang, F., Lin, L., Xia, F., Du, L., and He, Q. Capturing attraction distribution: Sequential attentive network for dwell time prediction.

[180] Weber, I., and Castillo, C. The demographics of web search. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval* (2010), pp. 523–530.

[181] Welke, P., Andone, I., Blaszkiewicz, K., and Markowetz, A. Differentiating smartphone users by app usage. In *Proceedings*

of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing (2016), ACM, pp. 519–523.

[182] White, R. W., Chu, W., Hassan, A., He, X., Song, Y., and Wang, H. Enhancing personalized search by mining and modeling task behavior. In *Proceedings of the 22nd international conference on World Wide Web* (2013), ACM, pp. 1411–1420.

[183] White, R. W., and Hassan Awadallah, A. Task duration estimation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining* (2019), pp. 636–644.

[184] White, R. W., Richardson, M., and Yih, W.-t. Questions vs. queries in informational search tasks. In *Proceedings of the 24th International Conference on World Wide Web* (2015), pp. 135–136.

[185] Wikipedia. Nearest centroid classifier — Wikipedia, the free encyclopedia, 2018. [Online; accessed 4-February-2018].

[186] Wu, C., Wu, F., Liu, J., He, S., Huang, Y., and Xie, X. Neural demographic prediction using search query. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining* (2019), pp. 654–662.

[187] Wu, D., Liang, S., and Tang, Y. Towards better understanding of app transitions in mobile search. *iConference 2017 Proceedings* (2017).

[188] Wu, S., Rizoiu, M.-A., and Xie, L. Beyond views: Measuring and predicting engagement in online videos. In *Proceedings of the International AAAI Conference on Web and Social Media* (2018), vol. 12.

[189] Xiang, B., Jiang, D., Pei, J., Sun, X., Chen, E., and Li, H. Context-aware ranking in web search. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval* (2010), pp. 451–458.

[190] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning* (2015), PMLR, pp. 2048–2057.

[191] Xu, Q., Erman, J., Gerber, A., Mao, Z., Pang, J., and Venkataraman, S. Identifying diverse usage behaviors of smartphone apps. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference* (2011), ACM, pp. 329–344.

[192] Xu, R., Frey, R. M., and Ilic, A. Individual differences and mobile service adoption: An empirical analysis. In *2016 IEEE Second International Conference on Big Data Computing Service and Applications (BigDataService)* (2016), IEEE, pp. 234–243.

[193] Xu, S., Li, W., Zhang, X., Gao, S., Zhan, T., and Lu, S. Predicting and recommending the next smartphone apps based on recurrent neural network. *CCF Transactions on Pervasive Computing and Interaction 2*, 4 (2020), 314–328.

[194] Xu, Y., Lin, M., Lu, H., Cardone, G., Lane, N., Chen, Z., Campbell, A., and Choudhury, T. Preference, context and communities: a multi-faceted approach to predicting smartphone app usage patterns. In *Proceedings of the 2013 International Symposium on Wearable Computers* (2013), ACM, pp. 69–76.

[195] Yan, J., Chu, W., and White, R. W. Cohort modeling for enhanced personalized search. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval* (2014), pp. 505–514.

[196] Yan, T., Chu, D., Ganesan, D., Kansal, A., and Liu, J. Fast app launching for mobile devices using predictive user context. In *Proceedings of the 10th international conference on Mobile systems, applications, and services* (2012), ACM, pp. 113–126.

[197] Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies* (2016), pp. 1480–1489.

[198] Yi, X., Hong, L., Zhong, E., Liu, N. N., and Rajan, S. Beyond clicks: dwell time for personalization. In *Proceedings of the 8th ACM Conference on Recommender systems* (2014), ACM, pp. 113–120.

[199] Ying, J. J.-C., Chang, Y.-J., Huang, C.-M., and Tseng, V. S. Demographic prediction based on users mobile behaviors. *Mobile Data Challenge* (2012), 1–6.

[200] Yom-Tov, E., Lalmas, M., Baeza-Yates, R., Dupret, G., Lehmann, J., and Donmez, P. Measuring inter-site engagement. In *Big Data, 2013 IEEE International Conference on* (2013), IEEE, pp. 228–236.

[201] Yu, D., Li, Y., Xu, F., Zhang, P., and Kostakos, V. Smartphone app usage prediction using points of interest. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 1*, 4 (2018), 1–21.

[202] Yu, K., Zhang, B., Zhu, H., Cao, H., and Tian, J. Towards personalized context-aware recommendation by mining context logs through topic models. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining* (2012), Springer, pp. 431–443.

[203] Zhang, C., Ding, X., Chen, G., Huang, K., Ma, X., and Yan, B. Nihao: A predictive smartphone application launcher. In *International Conference on Mobile Computing, Applications, and Services* (2012), Springer, pp. 294–313.

[204] Zhang, Y., Chen, X., et al. Explainable recommendation: A survey and new perspectives. *Foundations and Trends® in Information Retrieval 14*, 1 (2020), 1–101.

[205] Zhang, Y., Zhang, M., Liu, Y., Tat-Seng, C., Zhang, Y., and Ma, S. Task-based recommendation on a web-scale. In *2015 IEEE International Conference on Big Data (Big Data)* (2015), IEEE, pp. 827–836.

[206] Zhao, S., Li, S., Ramos, J., Luo, Z., Jiang, Z., Dey, A. K., and Pan, G. User profiling from their use of smartphone applications: A survey. *Pervasive and Mobile Computing 59* (2019), 101052.

[207] Zhao, S., Luo, Z., Jiang, Z., Wang, H., Xu, F., Li, S., Yin, J., and Pan, G. Appusage2vec: Modeling smartphone app usage for prediction. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)* (2019), IEEE, pp. 1322–1333.

[208] Zhao, S., Pan, G., Zhao, Y., Tao, J., Chen, J., Li, S., and Wu, Z. Mining user attributes using large-scale app lists of smartphones. *IEEE Systems Journal 11*, 1 (2016), 315–323.

[209] Zhao, S., Ramos, J., Tao, J., Jiang, Z., Li, S., Wu, Z., Pan, G., and Dey, A. K. Discovering different kinds of smartphone users through their application usage behaviors. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (2016), ACM, pp. 498–509.

[210] Zhao, S., Xu, F., Luo, Z., Li, S., and Pan, G. Demographic attributes prediction through app usage behaviors on smartphones. In *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers* (2018), pp. 870–877.

[211] Zhao, S., Xu, Y., Ma, X., Jiang, Z., Luo, Z., Li, S., Yang, L. T., Dey, A., and Pan, G. Gender profiling from a single snapshot of apps installed on a smartphone: An empirical study. *IEEE Transactions on Industrial Informatics 16*, 2 (2019), 1330–1342.

[212] Zhao, S., Zhao, Y., Zhao, Z., Luo, Z., Huang, R., Li, S., and Pan, G. Characterizing a user from large-scale smartphone-sensed data. In *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers* (2017), ACM, pp. 482–487.

[213] Zhong, E., Tan, B., Mo, K., and Yang, Q. User demographics prediction based on mobile data. *Pervasive and mobile computing 9*, 6 (2013), 823–837.

[214] Zhou, K., Yang, S.-H., and Zha, H. Functional matrix factorizations for cold-start recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval* (2011), ACM, pp. 315–324.

[215] Zhou, T., Qian, H., Shen, Z., Zhang, C., Wang, C., Liu, S., and Ou, W. Jump: A joint predictor for user click and dwell time. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence. AAAI Press* (2018), pp. 3704–3710.

[216] Zhu, H., Cao, H., Chen, E., Xiong, H., and Tian, J. Exploiting enriched contextual information for mobile app classification. In *Proceedings of the 21st ACM international conference on Information and knowledge management* (2012), pp. 1617–1621.

[217] Ziff, R. Psychographics for market segmentation. *Journal of Advertising Research 11*, 2 (1971), 3–9.

[218] Zou, X., Zhang, W., Li, S., and Pan, G. Prophet: What app you wish to use next. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication* (2013), ACM, pp. 167–170.