Identifying Tasks from Mobile App Usage Patterns

Yuan Tian
University of Nottingham
Nottingham, UK
yuan.tian@nottingham.ac.uk

Ke Zhou
University of Nottingham
Nottingham, UK
ke.zhou@nottingham.ac.uk

Mounia Lalmas
Spotify
London, UK
mounia@acm.org

Dan Pelleg
Yahoo Research
Haifa, Israel
pellegd@acm.org

ABSTRACT

Mobile devices have become an increasingly ubiquitous part of our everyday life. We use mobile services to perform a broad range of tasks (e.g., booking travel or office work), leading to often lengthy interactions within distinct apps and services. 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. In this work, we first 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. Our classifiers improve on traditional timeout segmentation, achieving over 89% performance for both problems. Secondly, we use our best task classifier on a large-scale data set of commercial mobile app usage logs to identify common tasks. We observe that users’ performed common tasks ranging from regular information checking to entertainment and booking dinner. Our proposed task identification approach provides the means to evaluate mobile services and applications with respect to task completion.

CCS CONCEPTS

• Information systems → Mobile information processing systems; • Human-centered computing → Ubiquitous and mobile computing systems and tools.

KEYWORDS

mobile tasks, apps, usage logs, user behavior modeling

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SIGIR ’20, July 25–30, 2020, Virtual Event, China
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8016-4/20/07...
https://doi.org/10.1145/3397271.3401441

1 INTRODUCTION

Mobile devices are increasingly used to simplify the accomplishment of various everyday activities [4, 5]. Users’ mobile needs span a broad spectrum: simple needs, such as weather information checking, can mostly be satisfied via a single app; but users may also access a series of apps, collect, filter, and synthesize information from multiple sources to solve a complex task, e.g., planning a vacation. Helping users complete tasks [18] 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 [35]. 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 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 [35]. 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 text more than once. If we could understand users’ tasks 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...
### 2 RELATED WORKS

**App Usage Analysis.** Recent studies have focused on understanding users’ behaviours on smartphones. These range from how app usage varies depending on context [3, 15] to deriving groups of mobile users based on their app usage behaviours [43]. These studies only look at within single app usage, and there is less research focusing on user behaviours across apps and sessions, and the types of tasks requiring these behaviors.

A smartphone usage session is commonly defined based on a threshold value of potential idle or standby time between application usage. Carrascal et al. [6, 9] define 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 [21]. Van Berkel et al. [35] conducted a systematic assessment of smartphone usage gaps, and suggested using 45 seconds as a threshold to segment app usage streams into sessions. Rather than sessions, we study segmenting app usage into tasks, organizing logs based on high-level user intentions. The apps may not be consecutively used and a task may span several app usage sessions.

**Task Identification.** Tasks, which are defined as pieces of work, ranging in scope from specific (e.g., sending an email) to broad (e.g., planning a wedding), are central to all aspects of information access and use [18]. The mobile “task” we define in our work is more similar to so-called “task” in search [19, 24, 26], which consist of a set of queries (apps) corresponding to a particular high-level information need, and the queries (apps) are not necessarily the same or even similar. In the context of web search, there have been many attempts to segment and define tasks, relying on a notion of timeout, lexical characteristics [37], and topic [22]. Many of them used the idea of a “timeout” cutoff between queries to bound tasks, i.e. 30 minutes [8, 10, 40]. 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 [7, 14, 24] consider lexical cues and treat queries, titles, and snippets of clicked URLs as bag-of-words, and use some string similarity metrics (e.g., Levenshtein edit distance, n-gram Jaccard) to measure the similarity between queries. However, Huang et al. [23] 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 [22] and [28] to improve the accuracy of task identification.

As we discussed above, many works have been done for task identification in search [22, 24, 26, 37]; 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 (§3.3), we follow the annotation procedure of search tasks shown in Table 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. 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

### Table 1: An example of mobile task: planning dinner with friends.

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>App</th>
<th>SessionID</th>
<th>TaskID</th>
<th>Task Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>18 Jan. 2014 17:50:59</td>
<td>Yelp</td>
<td>1</td>
<td>1</td>
<td>Dining with friends</td>
</tr>
<tr>
<td>18 Jan. 2014 17:52:11</td>
<td>WhatsApp</td>
<td>2</td>
<td>1</td>
<td>listen to music</td>
</tr>
<tr>
<td>18 Jan. 2014 17:50:59</td>
<td>Music</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>18 Jan. 2014 18:04:22</td>
<td>Facebook</td>
<td>4</td>
<td>3</td>
<td>Social</td>
</tr>
<tr>
<td>18 Jan. 2014 18:03:01</td>
<td>Instagram</td>
<td>5</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>18 Jan. 2014 18:01:59</td>
<td>Yelp</td>
<td>6</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>18 Jan. 2014 18:12:10</td>
<td>Google Maps</td>
<td>7</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>18 Jan. 2014 18:11:58</td>
<td>Uber</td>
<td>8</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>18 Jan. 2014 18:15:34</td>
<td>WhatsApp</td>
<td>9</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

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 first 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. Given that understanding of mobile tasks of a small-scale lab study is limited, and manual labeling of a large-scale data set is infeasible, we further explore complex mobile tasks of a large-scale data set in an automatic fashion. We first segment app usage sequences into tasks based on our proposed best task classifier to identify mobile tasks. These features include similarity features for capturing semantic relatedness between apps; we perform an extensive evaluation on a variety of classification frameworks and empirically report the performance of different classifiers and feature sets (§4.4). We find that our task identification classifiers can achieve an F-measure of 0.89 while the log sequence features are the most useful.

- By running our best task identification classifier on a large-scale commercial mobile app usage logs (§5), we cluster the multi-app tasks based on their characteristics and provide evidence that there actually exist 16 groups of tasks, which could be identified solely from their salient properties, such as regular information checking and meal booking.
Table 2: Summary of the task annotation procedure

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Labeled item</td>
<td>Search query log</td>
<td>Search query log</td>
<td>Search query log</td>
<td>Mobile app usage log</td>
</tr>
<tr>
<td>Time span</td>
<td>7 days</td>
<td>7 days</td>
<td>1 week</td>
<td>1 year</td>
</tr>
<tr>
<td>Task definition</td>
<td>This task defines whether the same app starts a &quot;success&quot; in terms of satisfying the user’s information need.</td>
<td>Group the queries into tasks according to annotators’ understanding of users’ information need.</td>
<td>No</td>
<td>The app usage should be grouped into one task when they all work for the same time.</td>
</tr>
<tr>
<td>Annotator Info.</td>
<td>Search the clicked URLs and browse the clicked URLs</td>
<td>No</td>
<td>No</td>
<td>App description: task-identifying app function, content, and users’ comments.</td>
</tr>
<tr>
<td>No. of Annotators</td>
<td>3</td>
<td>from their laboratory but not directly involved in this work</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Labeled Text</td>
<td>Task ID and optionally task description</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Validation</td>
<td>Cohen’s kappa</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

3 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.

3.1 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 logs $L_n = ((a_{n1}^i, t_{n1}^i), (a_{n2}^j, t_{n2}^j), ..., (a_{nM}^k, t_{nM}^k))$ from user $u_n$, where $t_{ni}^n$ is the corresponding timestamp when using app $a_{ni}^n$.

Definition 3.1. (SESSION $S^n_{T}$) Given user $u_n$’s app usage logs $L_n$ and a fixed time-out threshold $\tau$, a session $S^n_{T}$ is a set of consecutive app usage from $L_n$, such that $\forall (a_{ni}^n, t_{ni}^n) \in S^n_{T}, (a_{n'i'}^n, t_{n'i'}^n) \in S^n_{T}, (a_{ni}^n, t_{ni}^n) \not\in S^n_{T}, |t_{ni}^n - t_{n'i'}^n| \leq \tau_{cut} \text{ and } |t_{n'i'}^n - t_{n'i'}^n| > \tau_{cut}$.

The definition of session implies that $\{S^n_{T}\}_{T=1}^T$ is a set of disjoint partitions of app usage logs $L_n$, such that $\forall i \neq j, S^n_{Ti} \cap S^n_{Tj} = \emptyset$ and $L_n = \bigcup_i S^n_{Ti}$. Typical time-out threshold $\tau_{cut}$ in the context of mobile apps is set to be a short period, i.e., 30 [6, 9] or 45 seconds [35]. We adopt in the rest of our paper the threshold $\tau_{cut} = 45$ seconds to segment app sequences into sessions, following the recommendation from the systematic analysis conducted in [35]. A session, for us, is just a slice of user time. Other definitions (which conflict themselves) involve an absence of periods of inactivity [6, 9], or app used between unlocking and locking the phone [25]; ours does not, since we want to account for tasks, defined below, and use inactivity as a predictor, rather than as a definition.

Definition 3.2. (TASK $T^n_k$) Given user $u_n$’s app usage logs $L_n$, a mobile app $T^n_k$ is a maximum subset $\max_{T^n_k \in L_n} \{|T^n_k|\}$ of logs in $L_n$, such that all the app usage in $T^n_k$ corresponds to a particular task.

This definition of mobile task indicates that $\{T^n_k\}_{k=1}^K$ is also a set of disjoint partitions of app usage sequence $L_n : \forall j \neq k, T^n_k \cap T^n_j = \emptyset$ and $L_n = \bigcup_k T^n_k$. However, each $T^n_k$ is not confined to a particular session $S^n_k$ 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 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”.

3.2 Formulation for Supervised Task Learning

3.2.1 Task Boundary Detection. If tasks are not interleaved, as assumed in previous work [35], 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:

\[
\{((a_{ni}^n, t_{ni}^n), (a_{n'i'}^n, t_{n'i'}^n)) : (t_{ni}^n < t_{n'i'}^n) \cup (2\tau_{cut} : t_{ni}^n < t_{n'i'}^n)\} \rightarrow \{0, 1\}
\]

where $t_{ni}^n$ is the corresponding timestamp when using app $a_{ni}^n$.
3.3 Mobile Task Annotation

To acquire a ground-truth of mobile task labels for the supervised learning tasks (§3.2), we conduct a mobile task annotation crowdsourcing study. We sample the app usage logs for such annotation from the publicly available UbiqLog dataset1 [33, 34], 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 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 3.

Crowdsourced assessments have been commonly used to obtain labeled data [30, 38]. To identify mobile tasks, three annotators were recruited from Amazon Mechanical Turk [2], 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 the prior work on search task annotation [24, 26].

1Ubiqlog: https://archive.ics.uci.edu/ml/datasets/UbiqLog+(smartphone+lifelogging)
2We 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.

A detailed guideline was presented to the assessors, describing in general what a mobile task is (reformulated from §3.1) 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 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 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 [16] as previous studies that focused on the search task annotation [26, 30]. The same set of logs was annotated by three assessors and we measured the inter-rater agreement. Following [37], 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 Landis and Koch [27], represent the substantial agreement. This partially demonstrates the internal validity of our annotation method.

3.4 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.3 The statistics of those annotated tasks are shown in Table 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

Table 3: Statistics of the dataset for annotation

<table>
<thead>
<tr>
<th>UUser</th>
<th>Log</th>
<th>App/User</th>
<th>Log/User</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>5000</td>
<td>223 ± 83</td>
<td>177 ± 43</td>
</tr>
</tbody>
</table>

3All the annotated data and annotation guideline can be accessed at https://www.cs.nott.ac.uk/~pszkz/mobile-task.html.
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 §3.3.

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.

4 TASK IDENTIFICATION

We evaluate the performance of a set of predictive features (§4.1) using two set of supervised classification models (§4.2) on both task boundary detection and same-task identification.

4.1 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 provides an overview of the features.

4.1.1 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 [24], 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\(^*\) (apps); potentially indicating that these two log entries originate from the same task.

\(^*\)This sequential_log feature does not apply to task boundary detection given all the log entry pairs are sequential in this setting.

4.1.2 Similarity Features. A previous study [41] 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 [39]. 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, first, we use the feature same_cate to 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_coef and word_embed_sim, as defined in Table 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 [1]. The sentence representation is simply an average of the word embedding representations of all the words in the sentence.

4.1.3 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 \((a_1^n, a_2^n)\), we introduce six features based on leveraging historical app usage data:

- PMI: Pointwise Mutual Information (PMI) [12] 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 $(a_i, a_j)$, the app probabilities $P(a_i)$ and $P(a_j)$ are estimated by counting the number of observations of $a_i$ and $a_j$ across all the app usage sessions, and normalizing by $N$, which is the number of sessions. The joint probability, $P(a_i, a_j)$, is estimated by counting the number of times that $a_i$ and $a_j$ co-occurred in the same session, normalizing by $N$.

- **pa12**: $\frac{p(a_i \rightarrow a_j)}{\max_{a_j} p(a_i \rightarrow a_j)}$ is the normalized probability that $a_i$ is used right after $a_j$ within the same session [24]. It is used to measure whether two apps are always used successively.

- **switch_state**: captures users’ switching between two apps. For any log pair $(a_i, a_j)$, **switch_state** is 1 if $a_j$ was used right before $(a_i, a_j)$ within the same session. This represents an in-session user interaction of $a_j \rightarrow a_i$ which indicates that the user switches back and forth on $a_j$ within the same session. Otherwise, **switch_state** is 0.

- **switch_prob**: $\frac{f(a_i \rightarrow a_j)}{f(a_i \rightarrow a_j)}$ is the probability that a switch between $a_i$ and $a_j$ happens when accessed sequentially.

- **peso_a2**: since often people finish a task before turning off for the day [24], “last app usage of the day” might be a useful indicator of the last app used in a task. Following from [24], we generate the feature **peso_a2** to capture the probability that $a_i$ 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 $a_i$ and $a_j$ [37]. This feature only applies to those arbitrary app usage log pairs for the same-task identification.

### 4.2 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) [20], as an example of linear classifier; (2) K Nearest Neighbours (KNN) [13], as an example of a non-parametric method for classification; (3) Support Vector Machines (SVM) with Radial Basis Functions kernel [36] as an example of a non-linear classifier; and (4) XGBoost [11], 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.

### 4.3 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 [24, 26, 32] 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 [32], as well as time thresholds learned using cross-validation. To adapt the similarity-based approaches, we follow [26], 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. [24] use a strong baseline common + prism + time to identify the search tasks, where the common identifies the number of words in common and prism 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, common is replaced by identifying if two apps belong to the same category, whereas prism 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 [35] as a benchmark, where a session is considered as a task.

### 4.4 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 6 reports the performances of these models with different baselines and feature sets. Only results for the logistic regression classifier are reported in Table 6 since it outperforms other classifiers, as we show in Table 7. The comparative rankings of models that utilize different feature sets are similar across the different predictive models (§4.2). From Table 6, we can observe that in general, when we combine all three types of 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 [35]) is not sufficient to indicate that a task has been completed, as assumed in prior studies [35]. 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
Table 6: Performance comparison of different feature sets based on Logistic Regression (5-fold cross validation) for task boundary detection and same-task identification. * indicates statistical significant (p ≤ 0.05) using two-tailed T-test compared to the F-measure of best baseline.

<table>
<thead>
<tr>
<th>Baselines</th>
<th>Measurements</th>
<th>Proposed Features</th>
<th>Measurements</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acc  Pre Rec F-me</td>
<td>Temporal (T)</td>
<td>0.63 0.66 0.65 0.64</td>
</tr>
<tr>
<td>Task Boundary Detection</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search Threshold (30min) [32]</td>
<td>0.56 0.33 0.81 0.47</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Learned Time Threshold</td>
<td>0.52 0.51 0.83 0.62</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trained Levenshtein distance [26]</td>
<td>0.63 0.61 0.74 0.66</td>
<td></td>
<td></td>
</tr>
<tr>
<td>commonw+prisma+time [24]</td>
<td>0.80 0.74 0.94 0.82</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mobile Session Threshold (45s) [35]</td>
<td>0.44 0.37 0.80 0.51</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proposed Features</td>
<td></td>
<td>Temporal+Similarity (T+S)</td>
<td>0.80 0.75 0.92 0.83</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Similarity (S)</td>
<td>0.80 0.75 0.92 0.83</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sequence (LS)</td>
<td>0.87 0.86 0.88 0.87</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Temporal+Similarity+Sequence (T+S+LS)</td>
<td>0.89 0.88 0.91 0.89</td>
</tr>
<tr>
<td>Same-task Identification</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search Threshold (30min) [32]</td>
<td>0.77 0.85 0.85 0.85</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Learned Time Threshold</td>
<td>0.78 0.78 1.00 0.87</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trained Levenshtein distance [26]</td>
<td>0.74 0.74 1.00 0.84</td>
<td></td>
<td></td>
</tr>
<tr>
<td>commonw+prisma+time [24]</td>
<td>0.78 0.78 1.00 0.87</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mobile Session Threshold (45s) [35]</td>
<td>0.73 0.74 0.99 0.84</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proposed Features</td>
<td></td>
<td>Temporal+Similarity+Sequence (T+S+LS)</td>
<td>0.82 0.82 0.97 0.89</td>
</tr>
</tbody>
</table>

Table 7: Overview of the performance for different classifiers with the best performing feature sets (5-fold cross validation). * indicates statistical significant (p ≤ 0.05) using two-tailed T-test compared to the F-measure of the best performing logistic regression (LR) model.

<table>
<thead>
<tr>
<th>Classifiers Measurements</th>
<th>Proposed Features Measurements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Boundary Detection</td>
<td></td>
</tr>
<tr>
<td>KNN: All Feature Sets (T+S+LS)</td>
<td>0.75 0.69 0.92 0.78</td>
</tr>
<tr>
<td>SVM: All Feature Sets (T+S+LS)</td>
<td>0.63 0.60 0.89 0.77</td>
</tr>
<tr>
<td>XGBoost: All Feature Sets (T+S+LS)</td>
<td>0.89 0.87 0.90 0.88</td>
</tr>
<tr>
<td>LR: All Feature Sets (T+S+LS)</td>
<td>0.89 0.88 0.91 0.89</td>
</tr>
<tr>
<td>Same-task Identification</td>
<td></td>
</tr>
<tr>
<td>KNN: All Feature Sets (T+S+LS)</td>
<td>0.75 0.76 0.90 0.82</td>
</tr>
<tr>
<td>SVM: All Feature Sets (T+S+LS)</td>
<td>0.76 0.76 0.99 0.85</td>
</tr>
<tr>
<td>XGBoost: All Feature Sets (T+S+LS)</td>
<td>0.80 0.78 1.00 0.88</td>
</tr>
<tr>
<td>LR: All Feature Sets (T+S+LS)</td>
<td>0.82 0.82 0.97 0.89</td>
</tr>
</tbody>
</table>

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 [17]. This way, the resulting coefficients are directly comparable for both binary variables (e.g., categorical dummy variable) and numerical features. Table 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.

4.5 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 [17]. This way, the resulting coefficients are directly comparable for both binary variables (e.g., categorical dummy variable) and numerical features. Table 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 UNDERSTANDING TASKS IN THE WILD

By manually annotating mobile tasks from a small dataset (§3.3), we shed lights on some important characteristics of mobile tasks (§3.4), such as multi-app and interleaved tasks. However, it remains difficult to infer common task patterns given the size of the dataset. To gain further understanding, mapping large-scale app usage logs to tasks is required. However, manual labeling of a large data set is time-consuming and infeasible. In this section, by leveraging our best-performing task identification model (task boundary detection) and a large-scale dataset of app usage logs from the Verizon Media’s Flurry mobile analytics platform, we aim to gain insights of a large spectrum of mobile tasks. The dataset consists of a sample of logs recorded from a week (6th -12th) in March 2017 of 17 million logs from 9K unique apps and 14K users. Each record consists of the user’s anonymized ID, demographics, operating system, timestamp,
Table 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.

<table>
<thead>
<tr>
<th>Task Boundary Detection</th>
<th>Feature</th>
<th>Feature Type</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>time_diff</td>
<td>log</td>
<td>Sequence</td>
<td>-1.080</td>
</tr>
<tr>
<td>common_word</td>
<td>similarity</td>
<td></td>
<td>-0.521</td>
</tr>
<tr>
<td>word_emb_sim</td>
<td>similarity</td>
<td></td>
<td>-0.307</td>
</tr>
<tr>
<td>binary_interval(10min)</td>
<td>temporal</td>
<td></td>
<td>-0.358</td>
</tr>
<tr>
<td>time</td>
<td>log</td>
<td>Sequence</td>
<td>0.650</td>
</tr>
<tr>
<td>time_diff</td>
<td>log</td>
<td>Sequence</td>
<td>0.746</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Same-task Identification</th>
<th>Feature</th>
<th>Feature Type</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>time_diff</td>
<td>temporal</td>
<td></td>
<td>0.446</td>
</tr>
<tr>
<td>common_word</td>
<td>log</td>
<td>Sequence</td>
<td>0.574</td>
</tr>
<tr>
<td>word_emb_sim</td>
<td>log</td>
<td>Sequence</td>
<td>0.232</td>
</tr>
<tr>
<td>binary_interval(10min)</td>
<td>temporal</td>
<td></td>
<td>-0.185</td>
</tr>
</tbody>
</table>

Table 9: Features used for Clustering

<table>
<thead>
<tr>
<th>Feature</th>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_a$</td>
<td>1</td>
<td>Number of distinct apps</td>
</tr>
<tr>
<td>$N_{acc}$</td>
<td>1</td>
<td>Number of distinct app categories</td>
</tr>
<tr>
<td>$D$</td>
<td>1</td>
<td>Task duration (min)</td>
</tr>
<tr>
<td>$H$</td>
<td>1</td>
<td>Hour</td>
</tr>
<tr>
<td>$W$</td>
<td>1</td>
<td>Day of the week</td>
</tr>
<tr>
<td>$A$</td>
<td>45</td>
<td>&quot;Bag-of-app&quot; vector of app categories</td>
</tr>
<tr>
<td>$P_e$</td>
<td>1</td>
<td>Percentage of visiting the most popular app within that task (the most visited-in-task app)</td>
</tr>
<tr>
<td>$P_d$</td>
<td>1</td>
<td>Duration proportion of dominant app within that task (the most time-consuming-in-task app)</td>
</tr>
</tbody>
</table>

5.1 Clustering

The main objective of this analysis is to divide multi-log tasks into natural groups that reflect salient patterns of mobile tasks. We employ unsupervised learning to derive generic profiles of these tasks. We represent each task using the features in Table 9. Each task can be represented by the number of distinct apps, the number of distinct app categories, task duration, hour of the day, day of the week, app categories, the percentage of accessing the most popular app within the task, and duration proportion of the most time-consuming app in the task.

Clustering methods can be applied to automatically find clusters of data points with similar characteristics within an n-dimensional space. Some clustering methods are fast to execute, but they require the number of clusters to be set a priori before the clustering takes place, like k-means. Therefore, to cluster the tasks into different groups, a challenge is to determine the appropriate number of clusters. Given it is difficult to know $k$ in advance, we follow the clustering method from [43]. They make use of k-means with a pre-specified number of clusters and then cluster the centroids found using MeanShift. Specifically, they first select $k$ to be significantly larger than the number of natural clusters suitable to the problem for k-means, and then use MeanShift to merge centroids generated by k-means to match the natural clusters. In this way, the more computationally complex MeanShift clustering step can be performed quickly as its input data is much smaller than the original dataset. Following their method, we test different $k$ from 2 to 100 for both k-means and the k-means-MeanShift hybrid method.

For the clustering results, we exclude those clustering configurations ($k$) that produced clusters with less than 0.1% of the tasks, which is not suitable for our purpose of analyzing common task patterns. In what follows, the clustering results are evaluated by their cp score [43], which are mainly used to reward the uniform distribution of data points across clusters (Shannon’s entropy [31]) and the internal validity (Dunn’s index [29]) within each cluster.

Ultimately, we obtain 16 clusters (types of tasks) based on the highest cp score, and show the characteristics of these tasks in Table 10. To characterise the clusters, we also present an example of the most popular app usage sequence based on the frequency of that app usage sequence within each cluster, normalized by the total frequency of that sequence across all clusters. We observe that these clusters have on average 1.7±0.2 apps in each task; the proportion of visiting the most popular app in the task is 0.67±0.13; and the duration proportion of dominant app is 0.90±0.05. This indicates that most of the time, even multiple apps are engaged, users mainly focus on interacting with one of the apps during the task. Based on the characteristics (i.e., feature distributions and popular app sequences) of those clusters, we label each cluster with a task label (e.g., we label C1 as a "meal booking" task). From Table 10, we can observe there exist several unique groups of mobile tasks, whereas the temporal pattern plays a strong role in distinguishing those tasks:

- For the top ranked clusters, transportation and navigation apps are very popular in those multi-log tasks, which indicates that users tend to interact with different apps while on the move. For example, users might be planning to eat out for dinner (C1), commute (C2, C5) and go shopping (C9).
- Users tend to regularly browse multiple data sources together on mobile devices to obtain the latest information. These tasks are all very short, from 1.4 min to 3.7 min, which include morning regular information check (C2) on weather and finance (e.g., stock), reading latest news during commuting (C5), and consuming personalized information from various widgets (C7).
- Business related tasks may arise as users tend to perform micro-work tasks on smartphones, especially after work (C3) or on the move (C5). Those business apps include document editor/reader, remote desktop and email management apps.
- Communication apps are frequently used within many types of mobile tasks as a supportive channel. For example, users might communicate with colleagues while conducting some work-related tasks (C3). On the other hand, other apps or services are often used to facilitate communication as well. For example, users perform searches to gather information to share with friends while chatting (C4).
- Entertainment related tasks such as gaming occur throughout different times of the day, including: (a) "snack-style" gaming (around 15 mins) with multiple mini games in the morning (C10) and afternoon (C8); (b) "intensive" gaming
which apps are used and how people select apps to execute, should
into considerations.

Rather than managing apps solely based on their categories, they
improving the user experience. The switch interface can improve
user experience by absorbing the concept of shared intents in as
well as optimizing the layout design to support navigation between
recently used apps. This could enhance the effectiveness of the
workflow and improve users’ fragmented attention usage on the
smartphone. Those apps, such as communications, that users often
switch back and forth could be located in the key position when
designing the layout of that switch panel.

Screen management apps could help users manage their apps in
a more efficient way based on the typical tasks found in this paper.
Rather than managing apps solely based on their categories, they
can be organized based on tasks, taking into account users’ con-
textual information (e.g., time and location). For example, during a
weekend meal time, three apps (communication, food-and-drink,
weather, productivity) could be organized into a focused panel to
support the meal booking task for dining with friends (C1). Fur-
thermore, app developers should cooperate with other apps to
optimize the layout design to support navigation between
different apps to access. For example, we found that
users need to switch between communication and search apps in
C4. App developers may want to add an additional function to
their communication app, like a search bulletin, or search results
pop-up. In summary, smartphone manufacturers, app developers
and anyone who impacts 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. They should take mobile tasks
into considerations.

5.2 Implications
As we have discovered in our analysis, mobile users have common
requirements for engaging with multiple apps together, thereby
creating a need for an efficient app switching mechanism. Smart-
phone manufacturers can build smartphones with the operating
systems that aim to provide intelligent switching interface towards
improving the user experience. The switch interface can improve
user experience by absorbing the concept of shared intents in as
well as optimizing the layout design to support navigation between
recently used apps. This could enhance the effectiveness of the
workflow and improve users’ fragmented attention usage on the
smartphone. Those apps, such as communications, that users often
switch back and forth could be located in the key position when
designing the layout of that switch panel.

Screen management apps could help users manage their apps in
a more efficient way based on the typical tasks found in this paper.
Rather than managing apps solely based on their categories, they
can be organized based on tasks, taking into account users’ con-
textual information (e.g., time and location). For example, during a
weekend meal time, three apps (communication, food-and-drink,
weather, productivity) could be organized into a focused panel to
support the meal booking task for dining with friends (C1). Fur-
thermore, app developers should cooperate with other apps to
optimize the layout design to support navigation between
different apps to access. For example, we found that
users need to switch between communication and search apps in
C4. App developers may want to add an additional function to
their communication app, like a search bulletin, or search results
pop-up. In summary, smartphone manufacturers, app developers
and anyone who impacts 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. They should take mobile tasks
into considerations.

(around 30-60 mins) on one single game at night (C11 - C13),
doing exercises (C14), watching videos (C15), and media read-
ing (C16) with smartphones. All those tasks suggest mobile
devices are increasingly used for recreation.

Based on the mobile tasks we observed, we elaborate on the impli-
cations of our findings as below.

6 CONCLUSIONS
No previous study has analyzed or addressed the automatic iden-
tification of mobile tasks. In this paper, we presented 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. Our research is an important first step
in modeling mobile app usage from the task perspective.

User behaviours are largely determined by their own goals, tasks
and preferences, so mining knowledge about user tasks from log
activity data can reveal different user intentions and behavioural
patterns. These can provide unique signals for user-centric op-
timization and personalization. Based on the task segmentation
and same-task identification models proposed in our work, future
research should investigate whether and how information about
tasks could be leveraged to improve mobile services. For instance,
incorporating models of task prediction into a mobile device infras-
structure could improve the user experience in many ways. If the
mobile devices were able to identify the past apps and interactions
related to the user’s current long-term intent, this past information
could be retained and displayed to help the user re-establish the
context of a long-term mobile task, relieving them from the burden
of recalling past interactions. Furthermore, if we could predict that
a user was going to return to a task that has only been temporarily
suspended, the mobile device could help support future related app
interactions. As a second example, the extracted tasks could provide
more fine-grained predictive contexts to improve services, such as
mobile search, allowing for a more personalized and engaging
experience. For example, the probability of issuing certain queries
might be higher after performing certain tasks [42]. Likewise, the
probability of completing certain tasks can be higher after different
types of mobile search interactions.

Our work sets the stage for evaluating mobile apps and services,
not on a per-app basis, but on the basis of user tasks. In future work,
we would like to incorporate more fine-grained user interactions

Table 10: Characteristics of Clusters

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Duration(min)</th>
<th>App Categories</th>
<th>Popular Time Range</th>
<th>Popular App Usage Sequence</th>
<th>Task Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1(56.9%) 1.8</td>
<td>transportation, navigation, food-and-drink, communication</td>
<td>afternoon:15:00-21:00</td>
<td>transportation—communication—transportation</td>
<td>meal booking</td>
<td></td>
</tr>
<tr>
<td>C2(14.7%) 1.5</td>
<td>navigation, finance, weather, productivity</td>
<td>morning:09:00-11:00</td>
<td>productivity—finances—productivity</td>
<td>info checking</td>
<td></td>
</tr>
<tr>
<td>C3(13.0%) 1.4</td>
<td>productivity, communication, tools, business</td>
<td>afternoon:15:00-21:00</td>
<td>productivity—communication—business</td>
<td>mobile working</td>
<td></td>
</tr>
<tr>
<td>C4(1.5%) 1.3</td>
<td>widgets, productivity, communication, photography</td>
<td>afternoon:16:00-20:00</td>
<td>communication—productivity—communication</td>
<td>search while chatting</td>
<td></td>
</tr>
<tr>
<td>C5(2.9%) 1.1</td>
<td>finance, business, transportation, news</td>
<td>morning:09:00-13:00</td>
<td>transportation—news</td>
<td>info checking while commuting</td>
<td></td>
</tr>
<tr>
<td>C6(7.7%) 6.8</td>
<td>family, strategy, racing, social</td>
<td>night:20:00-22:00</td>
<td>family—social—social</td>
<td>social and games</td>
<td></td>
</tr>
<tr>
<td>C7(6.3%) 1.4</td>
<td>weather, widgets, tools, productivity</td>
<td>early morning:04:00-06:00</td>
<td>widgets—weather—productivity</td>
<td>widgets info checking</td>
<td></td>
</tr>
<tr>
<td>C8(6.5%) 1.6</td>
<td>adventure, action, racing, comics</td>
<td>afternoon:15:00-18:00</td>
<td>action—action</td>
<td>gaming in the afternoon</td>
<td></td>
</tr>
<tr>
<td>C9(3.7%) 1.1</td>
<td>food-and-drink, weather, transportation, shopping</td>
<td>morning:09:00-12:00</td>
<td>weather—transportation</td>
<td>shopping trip to the mall</td>
<td></td>
</tr>
<tr>
<td>C10(2.5%) 18</td>
<td>games, puzzle, stimulation, board</td>
<td>morning:09:00-12:00</td>
<td>games—games—games</td>
<td>multiple games in the morning</td>
<td></td>
</tr>
<tr>
<td>C11(2.7%) 32.3</td>
<td>comics, board, puzzle, role-playing</td>
<td>evening:18:00-22:00</td>
<td>puzzle—puzzle</td>
<td>one game at night</td>
<td></td>
</tr>
<tr>
<td>C12(4.5%) 48.2</td>
<td>word, role-playing, strategy, food-and-drinks</td>
<td>late night:22:00-00:00</td>
<td>role-playing—food-and-drinks</td>
<td>gaming and booking for late-night food delivery</td>
<td></td>
</tr>
<tr>
<td>C13(2.5%) 68.2</td>
<td>music, strategy, comics, adventure</td>
<td>afternoon:15:00-21:00</td>
<td>music—strategy—music</td>
<td>listening to music while playing games</td>
<td></td>
</tr>
<tr>
<td>C14(3.0%) 94.0</td>
<td>education, role, health-and-fitness</td>
<td>night:19:00-03:00</td>
<td>video—healths—healths</td>
<td>exercising time</td>
<td></td>
</tr>
<tr>
<td>C15(3.0%) 130.4</td>
<td>strategy, video, entertainment puzzle</td>
<td>night:19:00-21:00</td>
<td>video—video</td>
<td>video and entertainment</td>
<td></td>
</tr>
<tr>
<td>C16(1.9%) 236.8</td>
<td>tools, entertainment, books, personalization</td>
<td>night:20:00-02:00</td>
<td>books—tools—books</td>
<td>media reading</td>
<td></td>
</tr>
</tbody>
</table>
(e.g., contents browsed within the app) to identify mobile tasks. We also intend to optimize user satisfaction in the context of the mobile task, with the aim to improve mobile search (e.g., query auto-completion and search re-ranking).

REFERENCES