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

SIGIR '20, July 25–30, 2020, Virtual Event, China

ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401441

ACM Reference Format:
Yuan Tian, Ke Zhou, Mounia Lalmas, and Dan Pelleg. 2020. Identifying Tasks from Mobile App Usage Patterns. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. 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 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

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

Timestamp	App	SessionID	TaskID	Task Description
18 Jan. 2014 17:49:45	WhatsApp	1	**1**	**Dinning 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**	

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 of task boundary identification. Then an unsupervised learning method is applied to analyze various multi-app tasks for uncovering common patterns among them. The large-scale app usage log data we used are sampled from Verizon Media's Flurry, with more than 17 million logs and 14K worldwide users collected within a seven-day period in March 2017. Last, we discussed the implications of our proposed automatic task segmentation, especially on improving mobile search. Our contributions include:

- A detailed study (§3) of the frequency and patterns of real user app usage logs forming mobile tasks manually labeled by annotators;
- We propose a list of features (§4.1) that move beyond timeouts and demonstrate that they can be used effectively to identify mobile tasks. These features include similarity features for extracting common characteristics and log sequence 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.

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., Levenstein 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 2: Summary of the task annotation procedure

Ref.	Search Task [24]	Search Task [37]	Search Task [28]	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

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 §4.4, 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.

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 $L_n = \{(a_1^n, t_1^n), (a_2^n, t_2^n)..., (a_M^n, t_M^n)\}$ from user u_n, where t_i^n is the corresponding timestamp when using app a_i^n.

Definition 3.1. (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 [6, 9] or 45 seconds [35]. We adopt in the rest of our paper the threshold τ_{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_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 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:

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

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 [35].

3.2.2 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

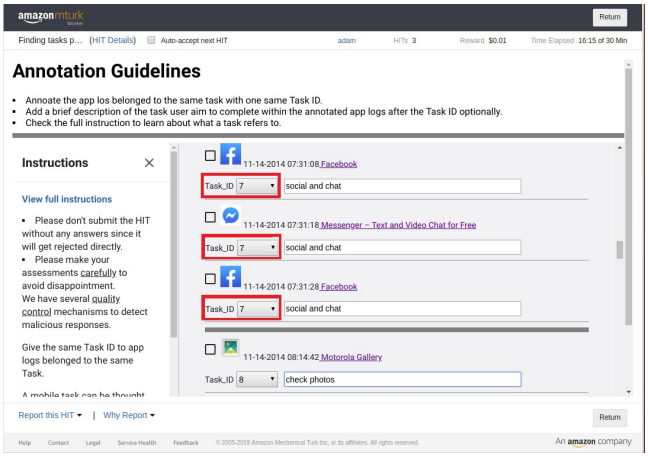

Figure 1: Screenshot of the annotation page on MTurk

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.

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 dataset[1] [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].

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

Table 3: Statistics of the dataset for annotation

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

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

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

Table 4: Statistics of the annotated mobile tasks

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

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[4] (apps); potentially indicating that these two log entries originate from the same task.

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

Table 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 (§ 4.1.2) 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.

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_coeff* 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 $\langle a_1^n, a_2^n \rangle$, 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 $\langle a_1^n, a_2^n \rangle$, 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 [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 $\langle a_1^n, a_2^n \rangle$, *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 [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 *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 [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 $commonw + prisma + time$ 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 [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 ($p \leq 0.05$) using two-tailed T-test compared to the F-measure of best baseline.

Baselines	Measurements				Proposed Features	Measurements			
	Acc	Pre	Rec	F-mea		Acc	Pre	Rec	F-mea
Task Boundary Detection									
Search Threshold (30min) [32]	0.56	0.33	0.81	0.47	Temporal (T)	0.63	0.66	0.65	0.64
Learned Time Threshold	0.52	0.55	0.83	0.62	Similarity (S)	0.80	0.75	0.92	0.82
Trained Levenshtein distance [26]	0.63	0.61	0.74	0.66	Temporal + Similarity (T+S)	0.80	0.75	0.92	0.83
commonw+prisma+time [24]	0.80	0.74	0.94	0.82	Sequence (LS)*	0.87	0.86	0.88	0.87
Mobile Session Threshold (45s) [35]	0.44	0.37	0.80	0.51	Temporal+Similarity+Sequence (T+S+LS)*	**0.89**	**0.88**	0.91	**0.89**
Same-task Identification									
Search Threshold (30min) [32]	0.77	0.85	0.85	0.85	Temporal (T)	0.74	0.74	1.00	0.84
Learned Time Threshold	0.78	0.78	1.00	0.87	Similarity (S)	0.74	0.74	1.00	0.84
Trained Levenshtein distance [26]	0.74	0.74	1.00	0.84	Temporal+Similarity (T+S)	0.78	0.78	0.98	0.86
commonw+prisma+time [24]	0.78	0.78	1.00	0.87	Sequence (LS)*	0.80	0.78	1.00	0.88
Mobile Session Threshold (45s) [35]	0.73	0.74	0.99	0.84	Temporal+Similarity+Sequence (T+S+LS)*	**0.82**	**0.82**	0.97	**0.89**

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

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

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 7, we find that the differences are relatively small while the LR classifier performs the best, followed by XGBoost.

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 $\leq$ 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

Table 9: Features used for Clustering

Feature	Dimension	Description
N_a	1	Number of distinct apps
N_{ac}	1	Number of distinct app categories
D	1	Task duration (min)
H	1	Hour
W	1	Day of the week
A	45	"Bag-of-app" vector of app categories
T_p	1	Percentage of visiting the most popular app within that task (the most visited in-task app)
D_p	1	Duration proportion of dominant app within that task (the most time-consuming in-task app)

app category, and usage duration. The dataset was anonymized and encrypted. Given the best task boundary detection approach (Table 6), the app usage logs are first separated based on the detected boundaries; we then only keep those tasks with at least two usage logs (multi-log tasks) for our further analysis.

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

Table 10: Characteristics of Clusters

Cluster	Duration(min)	App Categories	Popular Time Range	Popular App Usage Sequence	Task Label
C1(15.9%)	1.8	transportation, navigation, food-and-drink, communication	afternoon:15:00-21:00	transportation->communication->transportation	meal booking
C2(14.7%)	1.5	navigation,finance, weather, productivity	morning:7:00-11:00	weather->finance->productivity(mail)	regular info checking
C3(13.0%)	1.4	productivity, communication, tools, business	afternoon: 15:00-21:00	productivity->productivity->business	mobile working
C4(11.6%)	0.8	widgets, productivity, communication, photography	afternoon:16:00-20:00	communication->productivity->communication	search while chatting
C5(10.2%)	3.7	finance, business, transportation, news	morning:9:00-13:00	transportation->news	info checking while commuting
C6(6.7%)	6.8	family, strategy, racing, social	night:20:00-22:00	family->social->social	social and games
C7(5.6%)	1.4	weather, widgets, tools, productivity	early morning:4:00-8:00	widgets->weather->productivity	widgets info checking
C8(5.6%)	16.1	adventure, action, racing,comics	afternoon:15:00-18:00	action->action	gaming in the afternoon
C9(5.0%)	3.1	food-and-drink, weather, transportation, shopping	morning:9:00-12:00	weather->transportation	shopping trip to the mall
C10(4.2%)	18	games, puzzle, stimulation, board	morning:9:00 -12:00	games->games->games	multiple games in the morning
C11(2.7%)	32.3	comics, board, puzzle, role-playing	evening:18:00-22:00	puzzle->puzzle	one game at night
C12(2.2%)	43.4	word, role-playing, strategy, food-and-drinks	late night:00:00-10:00	role-playing->food-and-drinks	gaming and booking for late-night food delivery
C13(1.2%)	68.2	music, strategy, comics, adventure	afternoon:15:00-21:00	music->strategy->music	listening to music while playing games
C14(0.8%)	84.0	education,video,health-and-fitness	night:19:00-03:00	video->health-and-fitness	exercising time
C15(0.5%)	130.4	strategy, video, entertainment puzzle	night:19:00-21:00	video->video	video and entertainment
C16(0.1%)	236.8	tools, entertainment, books,personalization	night:20:00-02:00	books->tools->books	media reading

(around 30-60 mins) on one single game at night (C11 - C13), doing exercises (C14), watching videos (C15), and media reading (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 implications of our findings as below.

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. Smartphone 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' contextual information (e.g., time and location). For example, during a weekend meal time, three apps (communication, food-and-drink, and navigation apps) could be organized into a focused panel to support the meal booking task for dining with friends (C1). Furthermore, app developers should cooperate with other apps to enrich their app experience by adding components of common functionality, to which previously users have to switch back and forth 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.

6 CONCLUSIONS

No previous study has analyzed or addressed the automatic identification 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 optimization 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 infrastructure 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

(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

[1] Explosion AI. 2016. *Spacy similarity*. https://spacy.io/usage/vectors-similarity

[2] Inc Amazon Mechanical Turk. 2005. *Amazon Mechanical Turk*. https://www.mturk.com/

[3] Ricardo Baeza-Yates, Di Jiang, Fabrizio Silvestri, and Beverly Harrison. 2015. Predicting the next app that you are going to use. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. 285–294.

[4] Patrick Baudisch and Christian Holz. 2010. My new PC is a mobile phone. *XRDS: Crossroads, The ACM Magazine for Students* 16, 4 (2010), 36–41.

[5] Genevieve Bell and Paul Dourish. 2007. Yesterday's tomorrows: notes on ubiquitous computing's dominant vision. *Personal and ubiquitous computing* 11, 2 (2007), 133–143.

[6] Matthias Böhmer, Brent Hecht, Johannes Schöning, Antonio Krüger, and Gernot Bauer. 2011. 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*. 47–56.

[7] Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, Aristides Gionis, and Sebastiano Vigna. 2008. The query-flow graph: model and applications. In *Proceedings of the 17th ACM conference on Information and knowledge management*. 609–618.

[8] Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. 2008. 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*. 875–883.

[9] Juan Pablo Carrascal and Karen Church. 2015. An in-situ study of mobile app & mobile search interactions. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 2739–2748.

[10] Lara D Catledge and James E Pitkow. 1995. Characterizing browsing strategies in the World-Wide Web. *Computer Networks and ISDN systems* 27, 6 (1995), 1065–1073.

[11] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[12] Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics* 16, 1 (1990), 22–29.

[13] Thomas Cover and Peter Hart. 1967. Nearest neighbor pattern classification. *IEEE transactions on information theory* 13, 1 (1967), 21–27.

[14] Jianwei Cui, Hongyan Liu, Jun Yan, Lei Ji, Ruoming Jin, Jun He, Yingqin Gu, Zheng Chen, and Xiaoyong Du. 2011. 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*. 135–140.

[15] Trinh Minh Tri Do, Jan Blom, and Daniel Gatica-Perez. 2011. Smartphone usage in the wild: a large-scale analysis of applications and context. In *Proceedings of the 13th international conference on multimodal interfaces*. 353–360.

[16] Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin* 76, 5 (1971), 378.

[17] Andrew Gelman. 2008. Scaling regression inputs by dividing by two standard deviations. *Statistics in medicine* 27, 15 (2008), 2865–2873.

[18] Ahmed Hassan Awadallah, Cathal Gurrin, Mark Sanderson, and Ryen W White. 2019. Task Intelligence Workshop@ WSDM 2019. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 848–849.

[19] Ahmed Hassan Awadallah, Ryen W White, Patrick Pantel, Susan T Dumais, and Yi-Min Wang. 2014. Supporting complex search tasks. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. 829–838.

[20] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media.

[21] Daniel Hintze, Rainhard D Findling, Muhammad Muaaz, Sebastian Scholz, and René Mayrhofer. 2014. Diversity in locked and unlocked mobile device usage. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*. 379–384.

[22] Wen Hua, Yangqiu Song, Haixun Wang, and Xiaofang Zhou. 2013. Identifying users' topical tasks in web search. In *Proceedings of the sixth ACM international conference on Web search and data mining*. 93–102.

[23] Jeff Huang and Efthimis N Efthimiadis. 2009. Analyzing and evaluating query reformulation strategies in web search logs. In *Proceedings of the 18th ACM conference on Information and knowledge management*. 77–86.

[24] Rosie Jones and Kristina Lisa Klinkner. 2008. 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*. 699–708.

[25] Simon L Jones, Denzil Ferreira, Simo Hosio, Jorge Goncalves, and Vassilis Kostakos. 2015. Revisitation analysis of smartphone app use. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. 1197–1208.

[26] Alexander Kotov, Paul N Bennett, Ryen W White, Susan T Dumais, and Jaime Teevan. 2011. Modeling and analysis of cross-session search tasks. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. 5–14.

[27] J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics* (1977), 159–174.

[28] Claudio Lucchese, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, and Gabriele Tolomei. 2011. Identifying task-based sessions in search engine query logs. In *Proceedings of the fourth ACM international conference on Web search and data mining*. 277–286.

[29] Ujjwal Maulik and Sanghamitra Bandyopadhyay. 2002. Performance evaluation of some clustering algorithms and validity indices. *IEEE Transactions on pattern analysis and machine intelligence* 24, 12 (2002), 1650–1654.

[30] Rishabh Mehrotra, Ahmed Hassan Awadallah, Milad Shokouhi, Emine Yilmaz, Imed Zitouni, Ahmed El Kholy, and Madian Khabsa. 2017. Deep sequential models for task satisfaction prediction. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 737–746.

[31] Steven M Pincus. 1991. Approximate entropy as a measure of system complexity. *Proceedings of the National Academy of Sciences* 88, 6 (1991), 2297–2301.

[32] Filip Radlinski and Thorsten Joachims. 2005. Query chains: learning to rank from implicit feedback. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. 239–248.

[33] Reza Rawassizadeh, Elaheh Momeni, Chelsea Dobbins, Pejman Mirza-Babaei, and Ramin Rahnamoun. 2015. Lesson learned from collecting quantified self information via mobile and wearable devices. *Journal of Sensor and Actuator Networks* 4, 4 (2015), 315–335.

[34] Reza Rawassizadeh, Martin Tomitsch, Katarzyna Wac, and A Min Tjoa. 2013. UbiqLog: a generic mobile phone-based life-log framework. *Personal and ubiquitous computing* 17, 4 (2013), 621–637.

[35] Niels van Berkel, Chu Luo, Theodoros Anagnostopoulos, Denzil Ferreira, Jorge Goncalves, Simo Hosio, and Vassilis Kostakos. 2016. A systematic assessment of smartphone usage gaps. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 4711–4721.

[36] Vladimir Vapnik. 2013. *The nature of statistical learning theory*. Springer science & business media.

[37] Hongning Wang, Yang Song, Ming-Wei Chang, Xiaodong He, Ryen W White, and Wei Chu. 2013. Learning to extract cross-session search tasks. In *Proceedings of the 22nd international conference on World Wide Web*. 1353–1364.

[38] Ryen W White, Matthew Richardson, and Wen-tau Yih. 2015. Questions vs. queries in informational search tasks. In *Proceedings of the 24th International Conference on World Wide Web*. 135–136.

[39] Dan Wu, Shaobo Liang, and Yuan Tang. 2017. Towards better understanding of app transitions in mobile search. *iConference 2017 Proceedings* (2017).

[40] Biao Xiang, Daxin Jiang, Jian Pei, Xiaohui Sun, Enhong Chen, and Hang Li. 2010. Context-aware ranking in web search. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. 451–458.

[41] Qiang Xu, Jeffrey Erman, Alexandre Gerber, Zhuoqing Mao, Jeffrey Pang, and Shobha Venkataraman. 2011. Identifying diverse usage behaviors of smartphone apps. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. 329–344.

[42] Aston Zhang, Amit Goyal, Ricardo Baeza-Yates, Yi Chang, Jiawei Han, Carl A Gunter, and Hongbo Deng. 2016. Towards mobile query auto-completion: An efficient mobile application-aware approach. In *Proceedings of the 25th International Conference on World Wide Web*. 579–590.

[43] Sha Zhao, Julian Ramos, Jianrong Tao, Ziwen Jiang, Shijian Li, Zhaohui Wu, Gang Pan, and Anind K Dey. 2016. 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*. 498–509.