Context-aware Target Apps Selection and Recommendation for Enhancing Personal Mobile Assistants

MOHAMMAD ALIANNEJADI*, University of Amsterdam, The Netherlands
HAMED ZAMANI, University of Massachusetts Amherst, USA
FABIO CRESTANI, Università della Svizzera italiana (USI), Switzerland
W. BRUCE CROFT, University of Massachusetts Amherst, USA

Users install many apps on their smartphones, raising issues related to information overload for users and resource management for devices. Moreover, the recent increase in the use of personal assistants has made mobile devices even more pervasive in users' lives. This article addresses two research problems that are vital for developing effective personal mobile assistants: *target apps selection* and *recommendation*. The former is the key component of a unified mobile search system: a system that addresses the users' information needs for all the apps installed on their devices with a unified mode of access. The latter, instead, predicts the next apps that the users would want to launch. Here we focus on context-aware models to leverage the rich contextual information available to mobile devices. We design an *in situ* study to collect thousands of mobile queries enriched with mobile sensor data (now publicly available for research purposes). With the aid of this dataset, we study the user behavior in the context of these tasks and propose a family of context-aware neural models that take into account the sequential, temporal, and personal behavior of users. We study several state-of-the-art models and show that the proposed models significantly outperform the baselines.

CCS Concepts: • **Information systems** → **Query intent**; **Query log analysis**; *Personalization*; **Retrieval on mobile devices**; *Recommender systems*; • **Human-centered computing** → *Field studies*; **Personal digital assistants**;

Additional Key Words and Phrases: Mobile information retrieval, app recommendation, mobile usage understanding, query analysis, user behavior analysis, neural networks

ACM Reference format:
Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W. Bruce Croft. 2021. Context-aware Target Apps Selection and Recommendation for Enhancing Personal Mobile Assistants. *ACM Trans. Inf. Syst.* 39, 3, Article 29 (May 2021), 30 pages.
https://doi.org/10.1145/3447678

*Part of the work reported in this paper was done while Mohammad Aliannejadi was affiliated with the Università della Svizzera italiana (USI).

This work was supported in part by the RelMobIR project of the Swiss National Science Foundation (SNSF), and in part by the Center for Intelligent Information Retrieval. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

Authors' addresses: M. Aliannejadi, IRLab, Faculty of Science, University of Amsterdam, 904 Science Park, Amsterdam, The Netherlands; email: m.aliannejadi@uva.nl; H. Zamani and W. B. Croft, CIIR, College of Information and Computer Sciences, University of Massachusetts Amherst, 140 Governors Dr., Amherst, MA; emails: {zamani, croft}@cs.umass.edu; F. Crestani, Faculty of Informatics, Università della Svizzera italiana (USI), Via Buffi 13, Lugano, Switzerland; email: fabio.crestani@usi.ch.

1046-8188/2021/05-ART29 $15.00
https://doi.org/10.1145/3447678

1 INTRODUCTION

In recent years, the number of available apps on the mobile app market has been growing due to high demand from users, leading to over 3.5 million apps on Google Play Store, for example.[1] As a consequence, users now spend an average of over five hours a day using their smartphones, accessing a variety of applications.[2] An average user, for example, installs over 96 different apps on their smartphones [10]. In addition, the emergence of intelligent assistants, such as Google Assistant, Microsoft Cortana, and Apple Siri, has made mobile devices even more pervasive. These assistants aim to enhance the capability and productivity of users by answering questions, performing actions in mobile apps, and improving the user experience while interacting with their mobile devices. Another goal is to provide users with a universal voice-based search interface; however, they still have a long way to go to provide a unified interface with the wide variety of apps installed on users' mobile phones. The diversity of mobile apps makes it challenging to design a unified voice-based interface. However, given that users spend most of their time working within apps (rather than a browser), it is crucial to improve their cross-app information access experience.

In this article, we aim to address two research problems that are crucial for effective development of a personal mobile assistant: *target apps selection* and *recommendation in mobile devices*. Target apps selection is the key component toward achieving a unified mobile search system—a system that can address the users' information needs not only from the Web but also from all the apps installed on their devices. We argued the need for a universal mobile search system in our previous work [6], where our experiments suggested that the existence of such a system would improve the users' experience. Target apps recommendation, instead, predicts the next apps that the users would want to launch and interact with, which is equivalent to target apps selection with no query.

A unified mobile search framework is depicted in Figure 1. As we see in the figure, with such a framework, the user could submit a query through the system that would then identify the best target app(s) for the issued query. The system then would route the query to the identified target apps and display the results in an integrated interface. Thus, the first step toward designing a unified mobile search framework is identifying the target apps for a given query, which is the target apps selection task [6].

Target apps recommendation is also crucial in a mobile environment. It has attracted a great deal of attention in multiple research communities [12, 28, 49]. Among various benefits and use cases discussed in the literature, we find the following two cases the most important ones: (i) to assist users in finding the right app for a given task the user wishes to perform and (ii) to help the operating system manage its resources more efficiently. It is worth noting that both use cases potentially play essential roles in improving end users' experience. The former reduces the users' effort to find the right app among various apps installed on their phone. However, the latter can affect the users' experience through smart resource management. For instance, a system could remove many background processes of apps that are not going to be used in the near future to avoid excessive battery usage. It can also be used to allocate the required resources to an app that is going to be launched by the user in the immediate future, providing faster and smoother user experience. The use of a target apps recommendation system and a target apps selection system brings even more benefits. While app usage data can help a target apps selection model provide more accurate predictions, the submitted cross-app queries could also improve a recommendation system's performance. For example, in cases when a user is traveling, they would use travel and navigation apps more often. This could be considered as an indication of the current user's information to the system. Also,

[1] https://www.thinkwithgoogle.com/advertising-channels/apps/app-marketing-trends-mobile-landscape/.
[2] http://flurrymobile.tumblr.com/post/157921590345/us-consumers-time-spent-on-mobile-crosses-5.

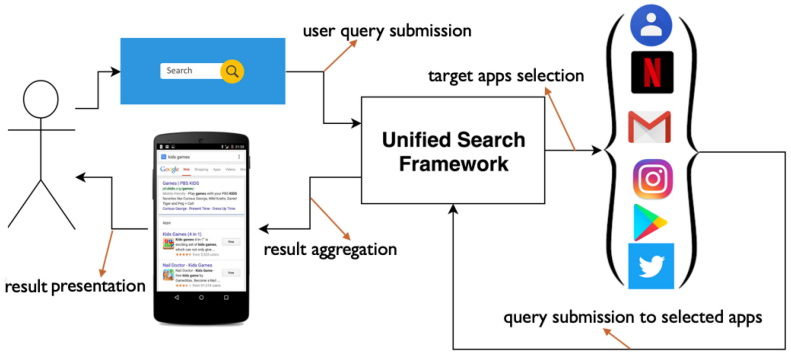


Fig. 1. A typical workflow of a unified mobile search framework.

assume a user submits the query "Katy Perry hits" to Google. The recommendation system could use this information in its prediction and recommend music apps.

As mobile devices provide rich contextual information about users, previous studies [2, 32, 60] have tried to incorporate query context in various domains. In particular, query context is often defined as information provided by previous queries and their corresponding clickthrough data [57, 58], or situational context such as location and time [14, 29, 60]. However, as user interactions on a mobile device are mostly with apps, exploring apps usage patterns reveals important information about the user contexts, information needs, and behavior. For instance, a user who starts spending time on travel-related apps, e.g., TripAdvisor, is likely to be planning a trip in the near future. Carrascal and Church [18] verified this claim by showing that people use certain categories of apps more intensely as they do mobile search. Modeling the latent relations between apps is of great importance, because while people use few apps on a regular basis, they tend to switch between apps multiple times [18]. In fact, previous studies have tried to address app usage prediction by modeling personal and contextual features [10], exploiting context-dependency of app usage patterns [35], sequential order of apps [59], and collaborative models [56].

However, our previous attempt to study unified mobile search through crowdsourcing did not capture users' contexts in the data collection phase [6], because it was done on the phone's browser, failing to record any contextual and sensor data related to the user location and activities. In addition, there are some other limitations. For example, we asked workers to complete a set of given search tasks, which obviously were not generated by their actual information needs, and thus the queries were likely different from their real search queries. In addition, not all of workers completed their tasks on actual mobile devices, which affected their behavior. Furthermore, the user behavior and queries could not be studied over a day-long or week-long continuous period.

These limitations have motivated us to conduct the first *in situ* study of target apps selection for unified mobile search. This enables us to obtain clearer insights into the task. In particular, we are interested in studying the users' behavior as they search for real-life information needs using their own mobile devices. Moreover, we studied the impact of contextual information on the apps they used for search. To this aim, we developed a simple open source app, called *uSearch*, and used it to build an *in situ* collection of cross-app queries. Over a period of 12 weeks, we collected thousands of queries, which enables us to investigate various aspects of user behavior as they search for information in a cross-app search environment.

Using the collected data, we conducted an extensive data analysis, aiming to understand how users' behavior vary across different apps while they search for their information needs. A key finding of our analysis include the fact that users conduct the majority of their daily search tasks

using specific apps, rather than Google. Among various available contextual information, we focus on the users' apps usage statistics as their *apps usage context*, leaving others for future work. This is motivated by the results of our analysis in which we show that users often search on the apps that they use more frequently. Based on the insights we got from our data analysis, we propose a context-aware neural target apps selection model, called Context-aware Neural Target Apps Selection (CNTAS). In addition, as we aimed to model the sequential app usage patterns while incorporating personal and temporal information, we proposed a neural target apps recommendation model, called Neural Sequential target App recommendation (NeuSA), which is able to predict the next apps that a user would launch at a certain time. The model learns complex behavioral patterns of users at different times of day by learning high-dimensional app representations, taking into account the sequence of previously used apps.

In summary, the main contributions of this article are as follows:

- An *in situ* mobile search study for collecting thousands of real-life cross-app queries. We make the app,[3] the collected search query data,[4] and the annotated app usage data[5] publicly available for research purposes.
- The first *in situ* analysis of cross-app queries and users' behavior as they search with different apps. More specifically, we study different attributes of cross-app mobile queries with respect to their target apps, sessions, and contexts.
- A context-aware neural model for target apps selection.
- A personalized sequence-aware neural model for target apps recommendation.
- Outperforming baselines for both target apps selection and recommendation tasks.

Our analyses and experiments lead to new findings compared to previous studies, opening specific future directions in this research area.

This article extends our previous work on *in situ* and context-aware target apps selection for unified mobile search [5]. We previously stressed the importance of incorporating contextual information in a unified mobile search and studied the app usage statistics data to identify the user's intent of submitting a query more accurately. We showed that considering what applications a person has used mostly in the past 24 hours is useful to improve the effectiveness of target apps selection. In this article, we further explore the effect of sequential app usage behavior of users for target apps recommendation. This is an ideal complement to our context-aware target apps selection model as these two components constitute an important part of context-aware mobile computing [23]. In summary, this article extends our previous work as follows:

- It presents a novel personalized time-aware target apps recommendation, called NeuSA.
- It compares the performance of NeuSA to state-of-the-art target apps recommendation.
- It describes the new dataset that we have collected and annotated for target apps recommendation, which we will make publicly available for research purposes.
- It includes more analysis of the collected data and the experimental results.
- It provides more details on our proposed context-aware target apps selection model CNTAS.

This article demonstrates that both our proposed models are able to outperform the state-of-the-art. Also, it provides new analysis and insights into the effect of context in both target apps selection and recommendation tasks. Finally, the joint analysis of context allows the reader to

[3] https://github.com/aliannejadi/uSearch.
[4] https://github.com/aliannejadi/istas.
[5] https://github.com/aliannejadi/LSApp.

observe and compare the effectiveness of analyzing and incorporating user behavior data into the prediction.

The remainder of the article is organized as follows. Section 2 provides a brief overview of the relevant studies in the literature. Section 3 elaborates on our effort for collecting the data, followed by Section 4 where we analyze the collected data in depth. Then, in Sections 5 and 6 we describe both our proposed models for context-aware target apps selection and recommendation, respectively. Section 7 then includes details on the experimental setup, followed by Section 8 discussing and analyzing the results. Finally, Section 9 concludes the article and discusses possible future directions that stem from this work.

2 RELATED WORK

Our work is related to the areas of mobile IR, context-aware search, target apps recommendation, human interaction with mobile devices (mobile HCI), and proactive IR. Moreover, relevant related research has been carried out in the areas of federated search and aggregated search and query classification. In the following, we briefly summarize the related research in each of these areas.

A mobile IR system aims at enabling users to carry out all the classical IR operations on a mobile device [23], as the conventional Web-based approaches fail to satisfy users' information needs on mobile devices [20]. Many researchers have tried to characterize the main differences in user behavior on different devices throughout the years. In fact, Song et al. [53] found significant differences in search patterns done using iPhone, iPad, and desktop. Studying search queries is one of the main research topics in this area, as queries are one of the main elements of a search session. Kamvar and Baluja [31] conducted a large-scale mobile search query analysis, finding that mobile search topics were less diverse compared to desktop search queries. Analogously, Guy [26] and Crestani and Du [22] conducted comparative studies on mobile spoken and typed queries showing that spoken queries are longer and closer to natural language. All these studies show that significant changes in user behavior are obvious. Change of the interaction mode, as well as the purpose and change of the information need, are among the reasons for this change [6].

Moreover, there has been studies on how mobile search sessions compare with desktop search sessions [12, 27, 28, 55]. van Berkel et al. [55] did a comprehensive analysis on how various inactivity gaps can be used to define an app usage session on mobile devices where they concluded that "researchers should ignore brief gaps in interaction." Carrascal and Church [18] studied user interactions with respect to mobile apps and mobile search, finding that users' interactions with apps impact search. Also, they found that mobile search session and app usage session have significant differences.

Given that mobile devices provide rich contextual information about users' whereabouts, a large body of research has tried to study the effect of such information on users' behavior. Church and Oliver [19] did a diary and interview study to understand users' mobile Web behavior. Aliannejadi et al. [3] conducted a field study where the recruited participants completed various search tasks in predefined time slots. They found that the temporal context, as well as the user's current activity mode (e.g., walking vs. sitting), influenced their perception of task difficulty and their overall search performance.

Also, Sohn et al. [52] conducted a diary study in which they found that contextual features such as activity and time influence 72% of mobile information needs. This is a very important finding, as it implies that using such information can greatly impact system performance and user satisfaction. In fact, research on proactive IR mainly focuses on this fact [13, 49]. Shokouhi and Guo [49] analyzed user interactions with information cards and found that the usage patterns of the proactive information cards depend on time, location, and the user's reactive search history. Proactive IR is very useful in a mobile context, where the user has a limited attention span for

the mobile device and the applications running on it. Similarly, Benetka et al. [13] studied how various types of activities affect users' information needs. They showed that not only information needs vary across activities, but they also change during an activity. Our work follows a similar line leveraging the changing context to determine the target apps for a given query.

Other works focused on a more comprehensive comparison of user behavior where they found using information from user search sessions among different platforms can be used to improve performance [40]. It has also been shown that using external information such as online reviews can be used to improve the performance of search on mobile devices [43]. Park et al. [42] inferred users' implicit intentions from social media for the task of app recommendation. This last work is closely related to our previous work [6] where we introduced the need for a unified mobile search framework as we collected cross-app queries through crowdsourcing. In contrast, we collect real-life cross-app queries over a longer period with an *in situ* study design in this work.

Research on unified mobile search has considerable overlap with federated, aggregated search, and query classification. While federated search systems assume the environment to be uncooperative and data to be homogeneous, aggregated search systems blend heterogeneous content from cooperative resources [9]. Target apps selection, however, assumes an uncooperative environment with heterogeneous content. Federated search has a long history in IR for Web search. In the case of uncooperative resources Callan and Connell [15] proposed a query-based sampling approach to *probe* the resources. Markov and Crestani [39] carried out an extensive theoretical, qualitative, and quantitative analysis of different resource selection approaches for uncooperative resources. One could study probing for unified mobile search; however, we argue that apps could potentially communicate more cooperatively, depending on how the operating system would facilitate that. More recently, research on aggregated search has gained more attention. Aggregated search share certain similarities with target apps selection in dealing with heterogeneous data [50]. However, research on aggregated search often enjoys fully cooperative resources as the resources are usually different components of the bigger search engine. For example, Diaz [25] proposed modeling the query dynamics to detect news queries for integrating the news *vertical* in SERP. Research on query classification has also been of interest for a long time in the field of IR. Different strategies are used to assign a query to predefined categories. As mobile users are constantly being distracted by external sources, the queries often vary a lot, and it is not easy to determine if a query is related to the same information need that originated the previous query. Kang and Kim [33] defined three types of queries, each of which requiring the search engine to handle differently. Shen et al. [48] introduced an intermediate taxonomy used to classify queries to specified target categories. Cao et al. [16] leveraged conditional random fields to incorporate users' neighboring queries in a session as context. More recently, Zamani and Croft [62] studied word embedding vectors for the query classification task and proposed a formal model for query embedding estimation.

Predicting app usage has been studied for a long time in the field. Among the first works that tried to model app usage, Liao et al. [37] proposed an app widget where users would see a list of recommended apps. Their model predicted the list of apps based on temporal usage profiles of users. Also, Huang et al. [30] studied different prediction models on this problem, including linear and Bayesian, where they found that contextual information, as well as sequential usage data, play important roles for accurate prediction of app usage. As smartphones kept evolving throughout these years, more data about various apps and users' context became available. As a result, more research focused on studying the effect of such information, as well as incorporating them into prediction models. For instance, Lu et al. [38] studied the effect of location data and proposed a model that takes into account GPS data together with other contextual information. Baeza-Yates et al. [10] studied next app recommendation for improved home screen usage experience, extracting a set of personal and contextual features in a more commercial setting. Lee et al. [35] found

that the usage probabilities of apps follow the Zipf's law, as opposed to "inter-running" and running times that follow log-normal distributions. Wang et al. [56] modeled the apps following the idea of collaborative filtering, proposing a context-aware collaborative filtering model to unload and pre-load apps. Xu et al. [59] modeled the sequential app usage using recurrent networks. Zhao et al. [63] proposed the AppUsage2Vec model, inspired by doc2vec. Their proposed architecture includes an app-attention mechanism and a dual-DNN layer.

As indicated in the literature, contextual and personal information have great impact in predicting user behavior on mobile devices. Also, researchers in the areas of federated and aggregated search have shown that contextual information play an important role in improved performance. In this work, we explore various sources of contextual information for both tasks. We also explore the use of recent app usage data as an implicit source of contextual information for target apps selection and show that it indeed provide useful contextual information to the model. Moreover, we study the collected data for both tasks, aiming to shed more light on the task of target apps selection and recommendation.

3 DATA COLLECTION

In this section, we describe how we collected In Situ collection of cross-App mobile Search (ISTAS), which is, to the best of our knowledge, the first *in situ* dataset on cross-app mobile search queries. We collected the data in 2018 by recruiting 255 participants. The participants installed a simple Android app, called uSearch, for at least 24 hours on their smartphones. We asked them to use uSearch to report their real-life cross-app queries as well as the corresponding target apps. We first describe the characteristics of uSearch. Then, we provide details on how we recruited participants as well as the details on how we instructed them to report queries through the app. Finally, we give details on how we checked the quality of the collected data.

3.1 uSearch

To facilitate the query report procedure, we developed uSearch, an Android app shown in Figure 2. We chose the Android platform, because, in comparison with iOS, it imposes less restrictions in terms of sensor data collection and background app activity.

User interface. As shown in Figure 2, uSearch consists of three sections. The upper part lists all the apps that are installed on the phone, with the most used apps ranked higher. The participants were supposed to select the app in which they had carried out their real-life search (e.g., `Facebook`). In the second section, the participants were supposed to enter exactly the same query that they had entered in the target app (e.g., `Facebook`). Finally, the lower part of the app, provided them easy access to a unique ID of their device and an online survey on their demographics and backgrounds.

Collected data. Apart from the participants' input data, we also collected their interactions within uSearch (i.e., taps and scrolling). Moreover, a background service collected the phone's sensors data. We collected data from the following sensors: (i) GPS, (ii) accelerometer, (iii) gyroscope, (iv) ambient light, (v) WiFi, and (vi) cellular. Also, we collected other available phone data that can be used to better understand a user's context. The additional collected data are as follows: (i) battery level, (ii) screen on/off events, (iii) apps usage statistics, and (iv) apps usage events. Note that apps usage statistics indicate how often each app has been used in the past 24 hours, whereas apps usage events provides more detailed app events.[6] Apps usage events record user interactions

[6] https://developer.android.com/reference/android/app/usage/package-summary.

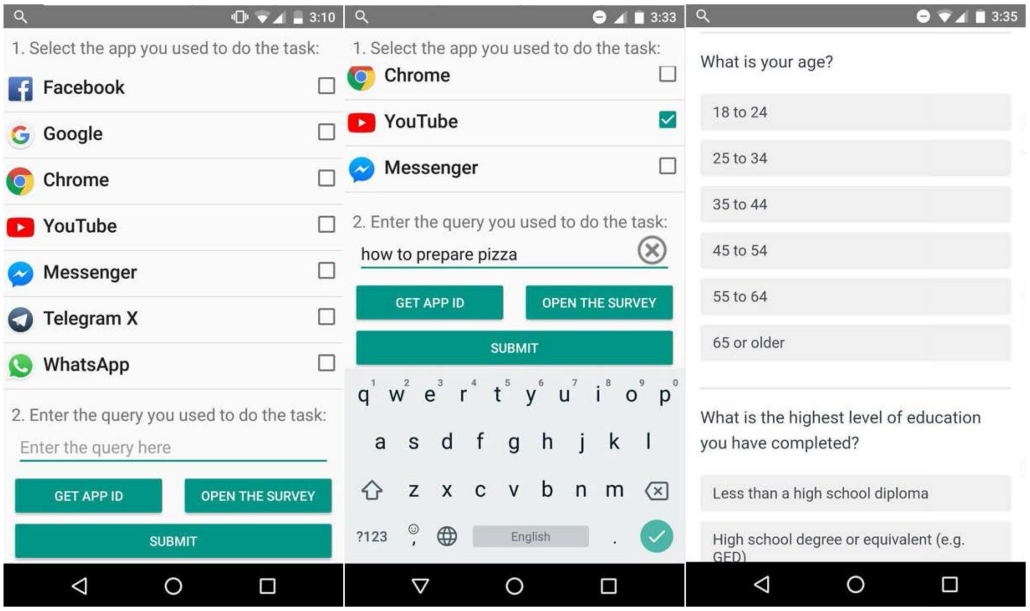


Fig. 2. uSearch interface on LG Google Nexus 5 as well as the survey. Checkboxes are used to indicate the target app for a query.

in terms of (i) launching a specific app, (ii) interacting with a launched app, (iii) closing a launched app, (iv) installing an app, and (v) uninstalling an app. The background service collected the data at a predefined time interval. The data was securely transferred to a cloud service.

3.2 Collection Procedure

We recruited participants through an open call on Amazon Mechanical Turk.[7] The study received the approval of the ethics committee of the university. We provided a clear statement to the participants about the kind of data that we were collecting and the purpose of the study. Furthermore, we used secure encrypted servers to store users' data. We asked the participants to complete a survey inside uSearch. Moreover, we mentioned all the steps required to be done by the participants to report a query. In short, we asked the participants to open uSearch after every search they did using any installed app on their phones. Then, we asked them to report the app as well as the query they used to perform their search task. We encouraged the participants to report their search as soon as it occurs, as it was very crucial to capture their context at the right moment.

After running several pilot studies, over a period of 12 weeks we recruited 255 participants, asking them to let the app running on their smartphones for at least 24 hours and report at least five queries. Since some people may not submit five search queries during the period of 24 hours, we asked them to keep the app running on their phones after the first 24 hours until they report five queries. Also, we encouraged them to continue reporting more than five queries for an additional reward. As incentive, we paid the participants $0.2 per query. We recruited participants only from English-speaking countries.

[7] http://mturk.com.

3.3 Quality Check

During the course of data collection, we performed daily quality checks on the collected data. The checks were done manually with the help of some data visualization tools that we developed. We visualized the use of selected apps in the participant's app-usage history in a timeline to validate a user's claim when they report using a specific app for their search. As we were paying participants a reward per query, we carefully studied the submitted queries as well as user interactions to prevent participants from reporting false queries. For each query, we checked the apps usage statistics and events for the same day. If a participant reported a query in a specific app (e.g., `Facebook`) but we could not find any recent usage events regarding that app, then we assumed that the query was falsely reported. Moreover, if a participant reported more than 10 queries per day, we took some extra quality measures into account. Finally, we approved 6,877 queries of 7,750 reported queries.

3.4 Data Transfer

To prevent unwanted career charges, we limited the data transfer to WiFi only. For this reason, we provided a very flexible implementation to manage the data in our app. In our app design, the data is stored locally as long as the device is not connected to a WiFi network. As soon as a WiFi connection is available, the app uploads the data to the cloud server. We made this point very clear in the instructions and asked the participants to take part in the study only if they had a strong WiFi connection at home or office.

3.5 Privacy Concerns

Before asking for required app permissions, we made clear statement about our intentions on how we were going to use the participants' collected data as well as what was collected from their devices. We ensured them that their data were stored on secure cloud servers and that they could opt out of the study at any time. In that case we would remove all their data from the servers. While granting apps usage access was mandatory, granting location access was optional. We asked participants to allow uSearch access their locations only if they felt comfortable with that. Note that, through the background service, we did not collect any other data that could be used to identify participants.

4 DATA ANALYSIS

In this section, we describe the basic characteristics of ISTAS, and present a thorough analysis of target apps, queries, sessions, and context.

4.1 Basic Statistics

ISTAS. During the period of 86 days, with the help of 255 participants, we collected 6,877 search queries and their target apps as well as sensor and usage data. The collected raw data was over 300 gigabytes. Here, we summarize the main characteristics of the participants based on the submitted surveys. Over 59% of the participants were female. Nearly 50% of them were aged between 25 and 34, followed by 22% between 35 and 44, and 15% 18 and 24 years. Participants were from all kinds of educational backgrounds ranging from high school diploma to PhD. In particular, 32% of them had a college degree, followed by 30% with a bachelor's degree. Smartphone was the main device used for connecting to the Internet for 53% of the participants, followed by laptop (25%). Among the participants, 67% used their smartphones more often for personal reasons rather than for work. Finally, half of the participants stated that they use their smartphones 4 hours a day or more. Table 1 lists basic characteristics of ISTAS. In the table we see that we received 26.97 ± 50.21 queries per user. Notice that the high value of standard deviation (=50.21) is due to existence of

Table 1. Statistics of ISTAS

# queries	6,877
# unique queries	6,262
# users	255
# unique apps	192
# search sessions	3,796
# days data collected	86
Mean queries per user	26.97 ± 50.21
Mean queries per session	1.81 ± 2.88
Mean queries per day	79.96 ± 101.27
Mean days of report per user	7.38 ± 15.95
Mean unique apps per user	5.14 ± 14.06
Mean query terms	3.00 ± 1.96
Mean query characters	17.53 ± 10.46

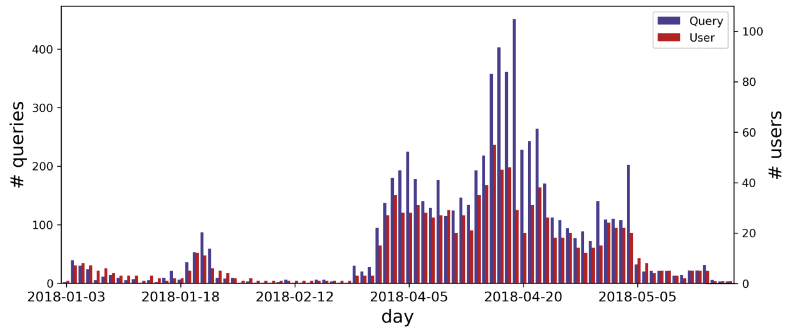


Fig. 3. Number of queries and active participants per day, during the course of data collection (best viewed in color).

some users who submitted numerous queries. Moreover, Figure 3 shows the number of queries and active participants per day during the data collection period. Note that, as shown in Figure 3, in the first half collection period, we were mostly developing the visualization tools and did not recruit many participants.

LSApp. We collected Large dataset of Sequential mobile App usage (LSApp) using the uSearch[8] data collection tool during an eight-month period involving 292 users. Notice that 255 of the users were the same people that were involved in collecting ISTAS. The extra 37 participants were the ones that either did not submit any queries during this period, or submitted low-quality queries and were removed in the quality check phase. Table 2 summarizes the statistics of LSApp. Since we observed many repeated app usage records with very short differences in time (<10 s), we considered all repeated app usage records with less than one minute time difference as one record. Also, as the app usage data includes various system apps, we filtered out background system packages and kept only the most popular apps in the data. We identify the most popular apps based on the data we collected in this dataset.

[8]https://github.com/aliannejadi/uSearch.

Table 2. Statistics of LSApp

# app usage records	599,635
# sessions	76,247
# unique apps	87
# users	292
Mean duration/user	15 days
Mean session time length	5:26
Median session time length	1:46
Mean unique apps in each session	2.18
Median unique apps in each session	2
Mean app switches within a session	5.46
Median app switches within a session	2
# train instances	464,903
# validation instances	66,300
# test instances	132,751

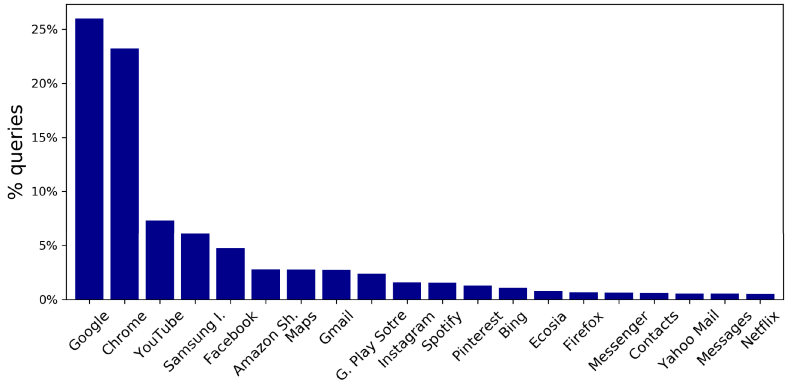
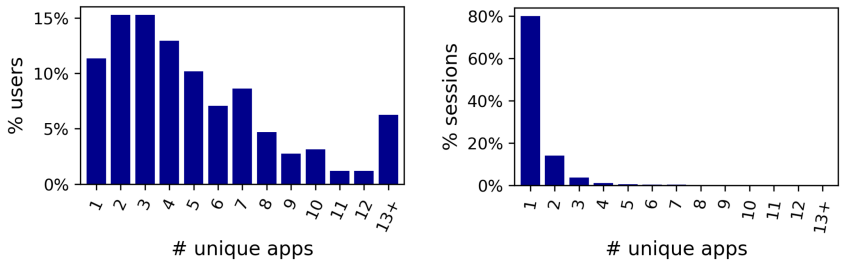


Fig. 4. Number of queries per app for top 20 apps in ISTAS.

4.2 Apps

How apps are distributed. Figure 4 shows how queries are distributed with respect to the top 20 apps. We see that the top 20 apps account for 88% of the searches in ISTAS, showing that the app distribution follows a power law. While `Google` and `Chrome` queries respectively attract 26% and 23% of the target apps, users conduct half (51%) of their search tasks using other apps. This finding is inline with what was shown in a previous work [6], even though we observe a higher percentage of searches done using `Google` and `Chrome` apps. In Reference [6], we collected a dataset cross-app queries called UniMobile under a different experimental setup where we asked the participants to submit cross-app queries for given search tasks. Therefore, the differences in the collected data can be due to two reasons: (i) ISTAS is collected *in situ* and on mobile devices, thus being more realistic than UniMobile; (ii) ISTAS queries reflect real-life information needs rather than a set of given search tasks, hence the information need topics are more diverse than UniMobile. Moreover, we observe a notable variety of apps among the top 20 apps, such as `Spotify` and `Contacts`. We also see `Google Play Store` among the top target apps. This suggests that people use their smartphones to search for a wide variety of information, most of which were done



(a) apps per user (b) apps per session

Fig. 5. Distribution of unique apps per user and per session in ISTAS.

with apps other than `Google` or `Chrome`. It should also be noted that users seek the majority of their information needs on various apps, even though there exists no unified mobile search system on their smartphones, suggesting that they might even do a smaller portion of their searches using `Google` or `Chrome`, if a unified mobile search system was available on their smartphones.

How apps are selected. Here, we analyze the behavior of the participants in ISTAS, as they searched for real-life information needs, in terms of the apps they chose for performing the search. Figure 5(a) shows the distribution of unique apps per user. We can see how many users selected a certain number of unique apps, with an average of 5.14 unique apps per user. Again, this indicates that users seek information in a set of diverse apps. It is worth noting that in Figure 5(a), we observe a totally different distribution compared to Reference [6], where the average number of unique apps per user was much lower. We believe this difference is due to the fact that the participants in our work reported their real-life queries, as opposed to the crowdsourcing setup of Reference [6].

However, Figure 5(b) plots the distribution of unique apps with respect to sessions, which is how many unique apps were selected during a single search session. We see an entirely different distribution where the average number of unique apps per task is 1.36. This shows that while users seek information using multiple apps, they are less open to switching between apps in a single session. This can partly be due to the fact that switching between apps is not very convenient. However, this behavior requires more investigation to be fully understood, that we leave for future work.

4.3 Queries

To understand the differences in user behavior while formulating their information needs using different apps, we conducted an analysis on the attributes of the queries with respect to their target apps. First, we start by studying the number of query terms in each app for the top nine apps in ISTAS.

How query length differs among apps. The upper part of Table 3 lists the distribution of the number of query terms in the whole dataset (denoted by *All*) as well as in each app. It also lists the average query terms per app. As we can see, the average query length is 3.00, which is slightly lower than previous studies on mobile query analysis [26, 31]. However, the average query length for apps that deal with general web search such as `Google` is higher (=3.49). This indicates that users submit shorter queries to other apps. For instance, we see that `Contacts` has the lowest average query length (=1.31), as its queries are mainly contact names. Also, `Gmail` and `Google Play Store` have an average query length lower than 2 as most searches are keyword based (e.g., part of an email subject or an app name). This difference shows a clear behavioral difference in formulating queries using different apps. Moreover, we can see that the distribution of the

Table 3. Cross-app Query Attributes for Nine Apps

	All	Google	YouTube	Facebook	Amazon Sh.	Maps	Gmail	G. Play Store	Spotify	Contacts
# terms	**Query term distribution**									
1	22%	13%	11%	22%	12%	25%	57%	49%	29%	81%
2	28%	26%	29%	48%	45%	27%	30%	33%	35%	10%
3	20%	21%	24%	16%	25%	18%	9%	12%	24%	7%
4	12%	13%	18%	10%	10%	13%	3%	4%	7%	2%
>4	17%	26%	17%	4%	10%	17%	1%	1%	6%	0%
Mean	3.00	3.49	3.19	2.34	2.74	3.07	1.61	1.75	2.31	1.31
τ	**Query overlap**									
>0.25	56%	39%	41%	28%	27%	26%	27%	25%	8%	14%
>0.50	19%	11%	15%	13%	7%	11%	12%	12%	4%	10%
>0.75	13%	5%	8%	11%	5%	9%	12%	10%	2%	10%

The upper part of the table lists the distribution of number of query terms as well as mean query terms per app. The lower part lists the query overlap at different similarity thresholds (denoted by τ) per app. *All* shows query distributions across all apps.

number of query terms varies among different apps; take `Contacts` as an example, whose single-term queries constitute 81% of its query distributions, which are often names of user's personal contacts. This indicates that the structure of queries vary across the target apps. Studying the most frequent query unigrams of each app also confirms this finding. For example, `Google`'s most popular unigrams are mostly stopwords (i.e., "to," "the," "of," "how"), whereas `Facebook`'s most popular unigrams are not (i.e., "art," "eye," "wicked," "candy").

How query similarity differs across apps. The lower part of Table 3 lists the query similarity or query overlap using a simple function used in previous studies [6, 21]. We measure the query overlap at various degrees and use the similarity function $\text{sim}(q_1, q_2) = |q_1 \cap q_2|/|q_1 \cup q_2|$, simply measuring the overlap of query terms. We see that among all queries, 18% of them are similar to no other queries. We see a different level of query overlap in queries belonging to different apps. The highest overlap is among queries from Web search apps such as `Chrome` and `Google`. Lower query similarity is observed for personal apps such as `Facebook` and for more focused apps such as `Amazon Shopping`. Note that the query overlap is higher when all app queries are taken into account (All), as opposed to individual apps. This shows that users tend to use the same query or a very similar query when they switch between different apps, suggesting that switching between apps is part of the information seeking or query reformulation procedure on mobile devices.

4.4 Sessions

ISTAS. A session is a "series of queries by a single user made within a small range of time" [51]. Similar to previous work [18, 31, 51], we consider a five-minute range of inactivity as closing a session. ISTAS consists of 3,796 sessions, with 1.81 average queries per session. The majority of sessions have only one query (=66%). Similarly, as shown in Figure 5(b), participants use only one app in the majority of sessions (=80%). We also studied how similar queries were distributed among single-app sessions as compared to multiple-app sessions. We found that queries are similar to each other in multiple-app sessions. More specifically, query overlap at the threshold of >0.25

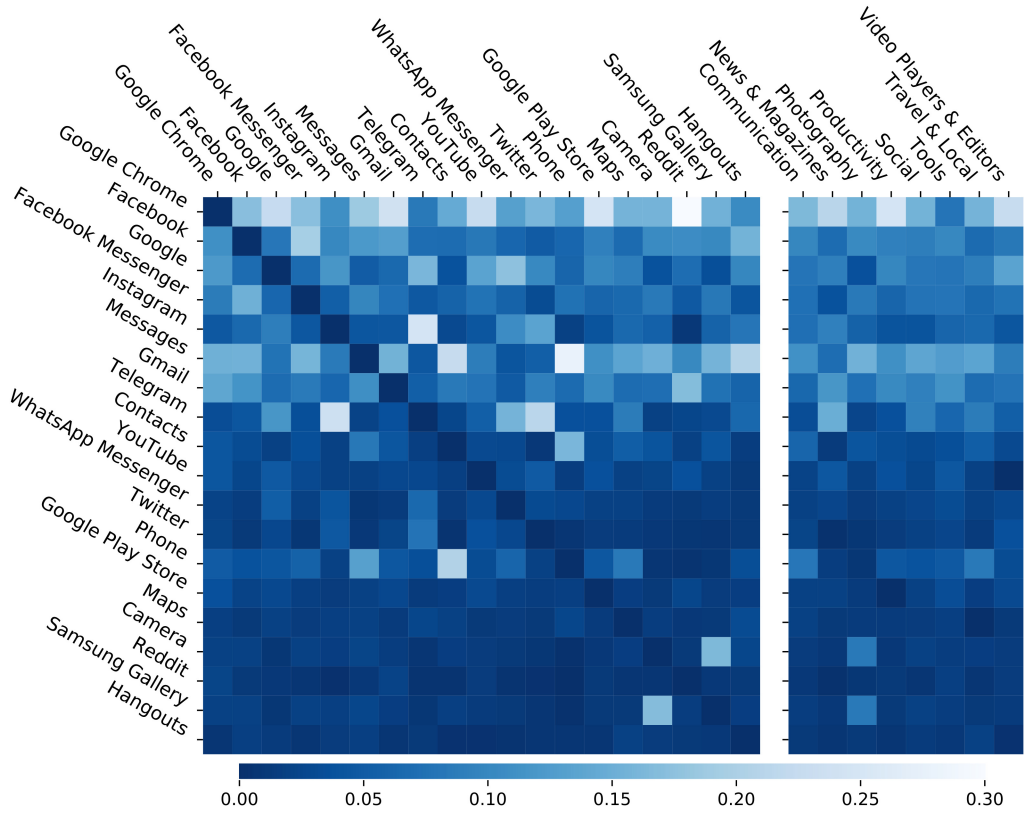
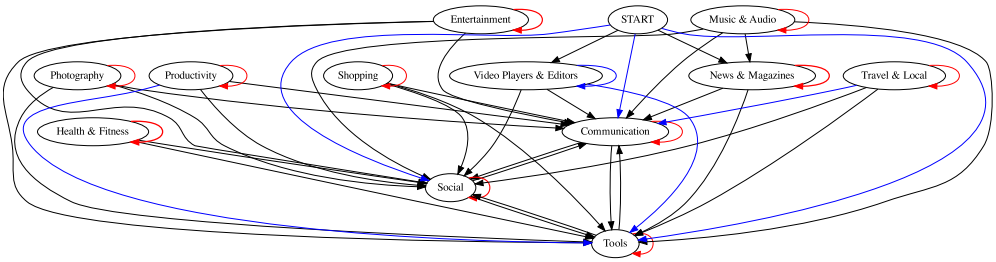


Fig. 6. Heat map depicting co-occurrence of apps in same sessions with other apps in LSApp. The graph on the left shows the co-occurrence at app level, whereas the one on the right shows it at category level. Popular apps such as `Google Chrome` dominantly co-occur with most of other apps in various categories.

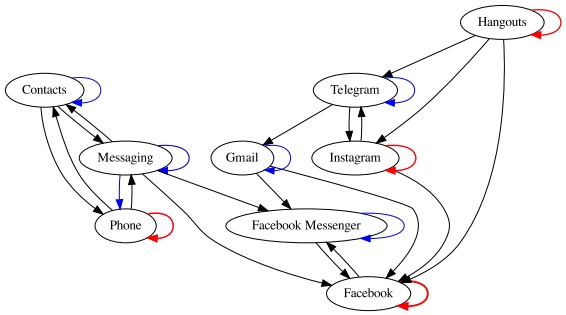
is 49% and 56% in single-app and multiple-app sessions, respectively. This suggests that users tend to switch between apps to search for the same information need as they reformulate their queries.

LSApp. For consistency with the search sessions, we consider a five-minute range of inactivity also for LSApp. It is worth noting that even though the relevant work suggests smaller inactivity periods [18, 55], we assume that a session ends after five minutes of inactivity to tackle the noisy app usage data and appearance of background services while the user is continuously using the same app. The collection contains a total number of 61,632 app usage sessions. Table 2 reports the mean and median length of sessions in terms of time, number of switches between apps. Also, we report the mean and median number of unique apps that users launch in a session. Comparing the number of app switches with unique apps, we see that in many sessions, users tend to work with two apps and do multiple switches between them. To gain more insight into the nature of app switches, we perform the two analyses shown in Figures 6 and 7.

Our first goal here is to show how top-used apps in LSApp are used in the same session by users. To this end, we count the number of co-occurrences in sessions and normalize the numbers by summing over all co-occurrence values. Note that we describe the definition of an app usage session in Section 4.4. Figure 6 illustrates the co-occurrence values in the form of a heat map with other apps displayed based on individual apps on the left, as well as categories on the right. We have used the official taxonomy of apps from Google Play Store. Since every app always co-occurs



(a) Switch pattern among app categories.



(b) Switch pattern among apps belonging to Social and Communication categories.

Fig. 7. App switch pattern in sessions with a Markov model on (a) app category level and (b) apps belonging to two categories. Edges represent a transition probability of over 0.05. Edges are directed and weighted by transition probability, with blue and red edges indicating over 0.2 and 0.4 transition probabilities, respectively.

with itself (hence having the maximum value of each row), we have set the diagonal values to zero for a better quality of the figure. We see from the first column that `Google Chrome` has the highest share of usage compared to other apps, because it has the highest value of most rows. It is interesting to see that users employ more popular apps such as `Google` together with the other apps in most of the sessions. As argued in Reference [6], users tend to use multiple apps to complete a single search task. Switching between popular search apps in our data suggests the same behavioral pattern is observed here. On the right side of the figure, we see how each app co-occurs with other apps based on their categories. It is interesting to observe that some app features could affect what type of apps co-occur. For example, observing the co-occurrences of the "Photography" app category, we see that social networking apps such as `Instagram` and `Telegram` exhibit some of the lowest co-occurrence values. This could be because of the photography features that already exist in such apps. Conversely, we see that apps such as `Messages` and `Gmail` co-occur more frequently. Also, we see that other apps belonging to the same or related categories are, in some cases, used in a session. For example, we see that `Phone` co-occurs with `Messaging` and `Contacts`. It is also interesting to observe the lowest row of the figure, showing the co-occurrence of `Hangouts`. We see that while `Hangouts` exhibits high co-occurrence with social media apps like `Facebook` and `Instagram`, it is not highly used in the same sessions with instant messaging apps such as `WhatsApp Messenger`, `Facebook Messenger`, and `Messages`. This suggests that apps that

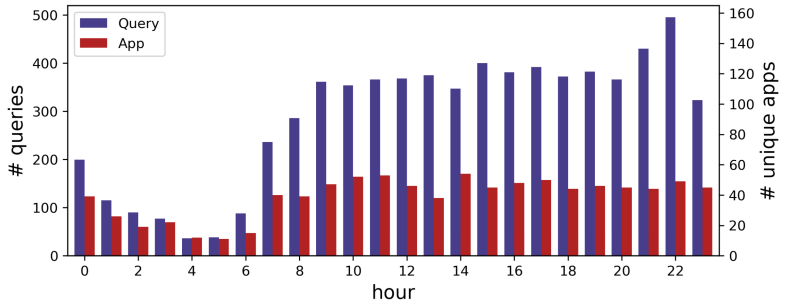


Fig. 8. Time-of-the-day distribution of queries and unique apps (best viewed in color).

fall into the same high-level category (i.e., social networking) tend to co-occur in a session, as users achieve different goals. However, users tend to use only one of the apps that fulfill very similar needs (i.e., instant messaging).

We illustrate the transition probabilities between app categories in Figure 7(a). The figure shows a Markov model of how users switch between apps that belong to different categories in a session. We see that the majority of sessions start with apps of Tools, Social, and Communication categories. Although users switch between various categories of apps, we see that they mostly tend to use apps of the same categories in a single session. This suggests that perhaps the types of tasks they complete in a single session can be done using a single or a set of apps with similar proposes (i.e., belong to the same category). To explore the transition probabilities between apps, we show in Figure 7(b) a Markov model of app transitions in sessions for Social and Communication apps. Here, we also see that even though users switch often among different apps, there is a higher tendency to switch to the same app (i.e., blue- and red-colored edges indicate higher probabilities). This suggests that while users are trying to perform a task, they might be interrupted by environmental distractions or notifications on their phones, closing the current app and opening it later. In particular we see a self-transition probability of over 0.4 on Phone, `Instagram`, `Hangouts`, and `Facebook`. This is perhaps related to the users' tendency to engage with these apps for longer, leading to a higher probability of interruption. Interestingly, we observe that native Communication apps (i.e., `Contacts`, and Phone, `Messaging`) form a cluster on the left side of the figure, with users switching mainly among the three apps while switching to other apps only through `Messaging`.

4.5 Context

Temporal behavior. We analyze the behavior of users as they search with respect to day-of-week and time-of-day. We see that the distribution of queries on different days of week slightly peaks on Fridays. Notice that in this analysis, we only include the users that participated in our study for more than six days. Moreover, Figure 8 shows the distribution of queries and unique target apps across time-of-day for all participants. Our findings agree with similar studies in the field [12, 28]. As we can see, more queries are submitted in the evenings, however we do not see a notable difference in the number of unique target apps.

Apps usage context. We define a user's *apps usage context* at a given time t as the apps usage statistics of that specific user during the 24 hours before t. Apps usage statistics contain details about the amount of time users spent on every app installed on their smartphones. This gives valuable information on users' personal app preferences as well as their contexts. For example, a user who has interacted with travel guide apps in the past 24 hours is probably planning a trip in the near future. Therefore, we analyze how users' apps usage context can potentially help a target

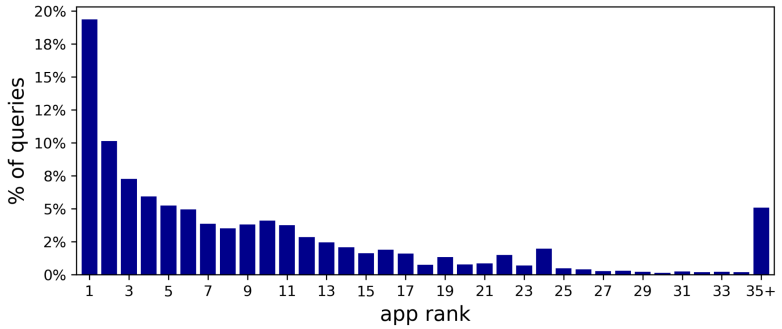


Fig. 9. Apps usage context ranking distribution of relevant target apps. Lower values of x axis mean that the app has been used more often in the past 24 hours.

app selection model. Figure 9 shows the histogram of target app rankings in the users' apps usage contexts. We see that participants often looked for information in the apps that they use more frequently. For instance, 19% of searches were done on the most used app, followed by 10% on the second most used app. We also see that, in most cases, as the ranking increases, the percentage of target apps decreases, suggesting that incorporating users app usage context is critical for target apps selection.

5 CONTEXT-AWARE NEURAL TARGET APPS SELECTION

In this section, we propose a context-aware neural model called CNTAS, which is an extension to our recent neural target apps selection model (i.e., NTAS1) [6]. Our model takes as input a query q, a candidate app a, and the corresponding query context c_q and produces a score indicating the likelihood of the candidate app a being selected by the user as the target app for the query q. In the following, we first describe a *general* framework for context-aware target apps selection and further explain how it is implemented and how context is incorporated into the framework.

Formally, the CNTAS framework estimates the probability $p(S = 1|q, a, c_q; A)$, where S is a binary random variable indicating whether the app a should be selected ($S = 1$) or not ($S = 0$). A denotes the set of candidate apps. This set can be all possible apps, otherwise those that are installed on the user's mobile device, or again a set of candidate apps that is obtained by another model in a cascade setting. The app selection probability in the CNTAS framework is estimated as follows:

$$p(S = 1|q, a, c_q; A) = \psi(\phi_Q(q), \phi_A(a), \phi_C(c_q)), \tag{1}$$

where ϕ_Q, ϕ_A, and ϕ_C, respectively, denote query representation, app representation, and context representation components. ψ is a target apps selection component that takes the mentioned representations and generates an app selection score. These components can be implemented in different ways. In addition, c_q can contain various types of query context, including search time, search location, and the users apps usage.

We implement the component ϕ_Q with two major functions: an embedding function $\mathcal{E} : V \rightarrow \mathbb{R}^d$ that maps each vocabulary term to a d-dimensional embedding space, and a global term weighting function $\mathcal{W} : V \rightarrow \mathbb{R}$ that maps each vocabulary term to a real-valued number showing its global importance. The matrices $\mathcal{E}$ and $\mathcal{W}$ are the network parameters in our model and are learned to provide task-specific representations. The query

representation component ϕ_Q represents a given query $q = \{w_1, w_2, \ldots, w_{|q|}\}$ as follows:

$$\phi_Q(q) = \sum_{i=1}^{|q|} \widehat{\mathcal{W}}(w_i) \cdot \mathcal{E}(w_i),$$

which is the weighted element-wise summation over the terms' embedding vectors. $\widehat{\mathcal{W}}$ is the normalized global weights computed using a softmax function as follows:

$$\widehat{\mathcal{W}}(w_i) = \frac{\exp(\mathcal{W}(w_i))}{\sum_{j=1}^{|q|} \exp(\mathcal{W}(w_j))}.$$

This is a simple yet effective approach for query representation based on the bag of words assumption, which has been proven to be effective for target apps selection [6] and ad-hoc retrieval [24, 47].

To implement the app representation component ϕ_A, we learn a d-dimensional dense representation for each app. More specifically, this component consists of an app representation matrix $\mathcal{A} \in \mathbb{R}^{N \times d}$, where N denotes the total number of apps. Therefore, $\phi_A(a)$ returns a row of the matrix $\mathcal{A}$ that corresponds to the app a.

Various context definitions can be considered to implement the context representation component. General types of context, such as location and time, has been extensively explored in different tasks, such as web search [14], personal search [60], and mobile search [29]. In this article, we refer to the *apps usage time* as context, which is a special type of context for our task. As introduced earlier in Section 4.5, the apps usage context is the time that the user spent on each mobile app in the past 24 hours of the search time. To implement ϕ_C, we first compute a probabilistic distribution based on the apps usage context, as follows:

$$p(a'|c_q) = \frac{\text{time spent on app } a' \text{ in the past 24 hours}}{\sum_{a'' \in A} \text{time spent on app } a'' \text{ in the past 24 hours}},$$

where A is a set of candidate apps. ϕ_C is then computed as:

$$\phi_C(c_q) = \sum_{a' \in A} p(a'|c_q) \cdot \mathcal{A}_C[a'],$$

where $\mathcal{A}_C \in \mathbb{R}^{N \times d}$ denotes an app representation matrix that is different from $\mathcal{A}$ used in the app representation component. This matrix is supposed to learn app representations suitable for representing the apps usage context. $\mathcal{A}_C[a']$ denotes the representation of app a' in the app representation matrix $\mathcal{A}_C$.

In summary, each of the representation learning components ϕ_Q, ϕ_A, and ϕ_C returns a d-dimensional vector. The app selection component is modeled as a fully connected feed-forward network with two hidden layers and the output dimensionality of 1. We use rectified linear unit (ReLU) as the activation function in the hidden layers of the network. Sigmoid is used as the final activation function. To avoid overfitting, the dropout technique [54] is employed. For each query, the following vector is fed to this network:

$$(\phi_Q(q) \circ \phi_A(a)) \cdot |\phi_Q(q) - \phi_A(a)| \cdot (\phi_C(c_q) \circ \phi_A(a)) \cdot |\phi_C(c_q) - \phi_A(a)|,$$

where $\circ$ denotes the Hadamard product, i.e., the element-wise multiplication, and $\cdot$ here means concatenation. In fact, this component computes the similarity of the candidate app with the query content and context, and estimates the app selection score based on the combination of both.

We train our model using pointwise and pairwise settings. In a pointwise setting, we use mean squared error (MSE) as the loss function. MSE for a mini-batch b is defined as follows:

$$\mathcal{L}_{MSE}(b) = \frac{1}{|b|}\sum_{i=1}^{|b|}(y_i - \psi(\phi_Q(q_i), \phi_A(a_i), \phi_C(c_{q_i})))^2,$$

where q_i, c_{q_i}, a_i, and y_i denote the query, the query context, the candidate app, and the label in the ith training instance of the mini-batch. For this training setting, we use a linear activation for the output layer.

CNTAS can be also trained in a pairwise fashion. Therefore, each training instance consists of a query, the query context, a target app, and a non-target app. To this end, we employ hinge loss (max-margin loss function) that has been widely used in the learning to rank literature for pairwise models [36]. Hinge loss is a linear loss function that penalizes examples violating the margin constraint. For a mini-batch b, hinge loss is defined as below:

$$\mathcal{L}_{Hinge}(b) = \frac{1}{|b|}\sum_{i=1}^{|b|}\max\{0, 1 - \text{sign}(y_{i1} - y_{i2})(\widehat{y}_{i1} - \widehat{y}_{i2})\},$$

where $\widehat{y}_{ij} = \psi(\phi_Q(q_i), \phi_A(a_{ij}), \phi_C(c_{q_i}))$.

6 PERSONALIZED TIME-AWARE TARGET APPS RECOMMENDATION

In this section, we propose a neural sequence-aware model called NeuSA, which captures the sequential dependencies of apps as well as users behavior with respect to their usage patterns (i.e., the personal app sequence) and temporal behavior. In the following, we first describe an overview of our target apps recommendation and further explain how it is implemented.

Formally, NeuSA estimates the probability $p(L = 1|u, a, c_u; A)$, where L is a binary random variable indicating whether the app a should be launched ($L = 1$) or not ($L = 0$). A denotes the set of candidate apps. Similarly to CNTAS, this set can be either all apps, those that are installed on the user's mobile device, or a set of candidate apps that is obtained by another model in a cascade setting. The app recommendation probability in the NeuSA framework is estimated as follows:

$$p(L = 1|u, a, c_u; A) = \psi(\phi_U(u), \phi_A(a), \phi_C(c_u)),$$

where ϕ_U, ϕ_A, and $\phi_C(c_u)$ denote user, app, and user context representation components, respectively. ψ is a target apps recommendation component that takes the mentioned representations and generates a recommendation score. Any of these components can be implemented in different ways. In addition, c_u can contain various types of user context, including time, location, and sequence of previously used apps.

We implement the component ϕ_U with an embedding function $\mathcal{E} : \mathcal{U} \rightarrow \mathbb{R}^d$ that maps a user to a d-dimensional embedding space. The matrix $\mathcal{E}$ is the network parameter in our model and is learned to provide task-specific representations.

To implement the app representation component ϕ_A, we learn a d-dimensional dense representation for each app. In more detail, this component consists of an app representation matrix $\mathcal{A} \in \mathbb{R}^{N\times d}$ where N denotes the total number of apps. Therefore, $\phi_A(a)$ returns a row of the matrix $\mathcal{A}$ that corresponds to the app a.

General types of context, such as location and time, has been extensively explored in different tasks, such as web search [14] and mobile search [29]. In this article, we refer to the *k previously used apps* and time as context with $k = 9$. Therefore, we define a window of size k and consider the sequence of used apps just before the time of recommendation as the sequence context. Following Reference [11], we break a full day (i.e., 24 hours) into 8 equal time bins (early morning–late night).

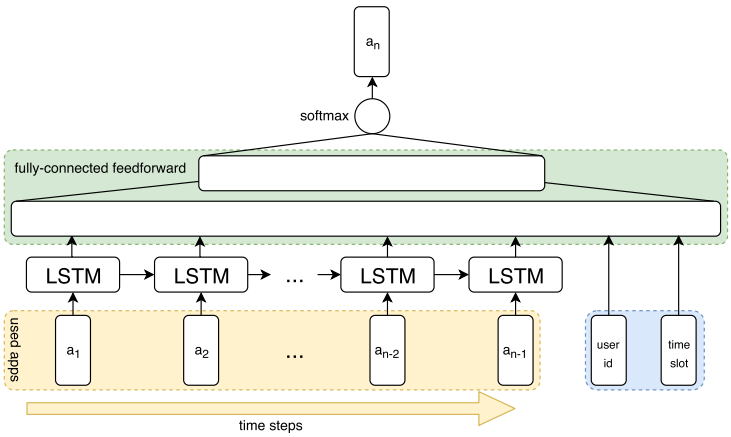


Fig. 10. The architecture of NeuSA.

To implement ϕ_C, we first compute a probabilistic distribution based on the apps usage records, as follows:

$$p(a'|c_u) = \frac{\text{time spent on app } a' \text{ in the current time bin}}{\sum_{a'' \in A} \text{time spent on app } a'' \text{ in the current time bin}},$$

where A is a set of candidate apps. ϕ_C is then computed as:

$$\phi_C(c_u) = \sum_{a' \in A} p(a'|c_u) \cdot \mathcal{A}[a'],$$

where $\mathcal{A} \in \mathbb{R}^{N \times d}$ denotes an app representation matrix. This matrix is supposed to learn app representations suitable for representing sequences of apps. $\mathcal{A}[a']$ denotes the representation of app a' in the app representation matrix $\mathcal{A}$.

Each of the representation learning components ϕ_U, ϕ_A, and ϕ_C returns a d-dimensional vector. The app recommendation component is modeled as a recurrent neural network consisting of Long Short-Term Memory (LSTM) units. After modeling the sequence of apps in this layer, the parameters, together with user and time features are passed to a fully connected feedforward network with two hidden layers. We use ReLU as the activation function in the hidden layers of the network. Softmax is used as the final activation function. To avoid overfitting, the dropout technique [54] is employed. We train our model using pointwise training setting where we use cross entropy as the loss function. Figure 10 depicts the architecture of our proposed network.

7 EXPERIMENTAL SETUP

In this section, we evaluate the performance of the proposed model in comparison with a set of baseline models.

7.1 Target Apps Selection

Data. We evaluate the performance of our proposed models on the ISTAS dataset. We follow two different strategies to split the data: (i) In *ISTAS-R*, we randomly select 70% of the queries for training, 10% for validation, and 20% for testing; (ii) In *ISTAS-T*, we sort chronologically the queries of each user and keep the first 70% of each user's queries for training, the next 10% for validation, and the last 20% for testing. ISTAS-T is used to evaluate the methods when information about users' search history is available. To minimize random bias, for ISTAS-R we repeated the

experiments 10 times and report the average performance. The hyper-parameters of all models were tuned based on the nDCG@3 value on the validation sets.

Evaluation metrics. Effectiveness is measured by four standard evaluation metrics that were also used in Reference [6]: mean reciprocal rank (MRR), and normalized discounted cumulative gain for the top 1, 3, and 5 retrieved apps (nDCG@1, nDCG@3, nDCG@5). We determine the statistically significant differences using the two-tailed paired t-test with Bonferroni correction at a 95% confidence interval ($p < 0.05$).

Compared methods. We compared the performance of our model with the following methods:

- *Most Frequently Used (MFU):* For every query we rank the apps in the order of their popularity in the training set as a static (query independent) model.
- *QueryLM, BM25, BM25-QE:* For every app we aggregate all the relevant queries from the training set to build a document representing the app. QueryLM is the query likelihood retrieval model [45]. For BM25-QE, we adopt Bo1 [8] for query expansion. We use the Terrier [41] implementation of these methods.
- *k-NN, k-NN-AWE:* To find the nearest neighbors in k nearest neighbors (k-NN), we consider the cosine similarity between the TF-IDF vectors of queries. Then, we take the labels (apps) of the nearest queries and produce the app ranking. As for k-NN-AWE [62], we compute the cosine similarity between the average word embedding (AWE) of the queries obtained from GloVe [44] with 300 dimensions.
- *ListNet, ListNet-CX:* For every query-app pair, we use the scores obtained by BM25-QE, k-NN, k-NN-AWE, and MFU as features to train ListNet [17] implemented in RankLib.[9] For every query, we consider all irrelevant apps as negative samples. ListNet-CX also includes users' apps usage context as an additional feature.
- *NTAS*: A neural model approach that we designed for the target apps selection task in our previous work [6]. We use the NTAS1 model due to its superior performance compared to NTAS2.
- *Contextual baselines:* To carry out a fair comparison between CNTAS and other context-aware baselines, we apply a context filter to all non-contextual baselines. We create the context filter as follows: for every app α in the training samples of user u, we take the time that u has spent on α in the past 24 hours as its score. We then perform a linear interpolation with the scores of all the mentioned baselines. Note that all scores are normalized. All these models are denoted by a *-CR* suffix.

7.2 Target Apps Recommendation

Data. For every user, we take the 70% earliest app usage records as training set, 10% next records as validation, and 20% latest records as test set.

Evaluation metrics. Effectiveness is measured by 6 standard evaluation metrics: MRR, normalized discounted cumulative gain for the top 1, 3, and 5 predicted apps (nDCG@1, nDCG@3, nDCG@5), and recall for the top 3 and 5 predicted apps (Recall@3, Recall@5). Our choice of evaluation metrics was motivated by the two main purposes of app recommendation we discussed in Section 1. The MRR and nDCG@k metrics are intended to evaluate the effectiveness for improved homescreen app ranking user experience, whereas Recall@k mainly evaluates how well a model is able to pre-load the next app among the top k predicted apps.

[9] https://sourceforge.net/p/lemur/wiki/RankLib/.

We determine the statistically significant differences using the two-tailed paired t-test at a 99.9% confidence interval ($p < 0.001$). Note that we apply the Bonferroni correction for the test against the baselines (i.e., * in Table 6).

We compare the performance of our models with the following methods:

- *MFU*: For every test instance we rank the apps in the order of their popularity in the training set as a static recommendation model.
- *Most Recently Used (MRU)*: For every test instance we rank the apps in the order of the their interaction time, so that the most recent apps are ranked higher.
- *Bayesian and Linear [30]*: We implement the two baselines proposed by Huang et al. [30], namely, Bayesian and Linear. Both baselines incorporate various contextual information in modeling app usage. In this work, we only use the contextual information available in our dataset, i.e., time, weekday, user, and previous app.
- *LambdaMART and ListNet*: For a given candidate app and every app in the sequence context, we compute the cosine similarity of their representation and consider it as a feature. The app representations are the AWE of app descriptions on Google Play Store. Other features include the recommendation time and current user. These features were used to train LambdaMART and ListNet as state-of-the-art learning to rank (LTR) methods, implemented in RankLib.[10]
- *k-NN and DecisionTree*: Similar to LTR baselines, we take AWE similarity between app pairs as well as user and time as classification features. We also include the apps that appear in the context sequence as additional features. We train kNN and DecisionTree classifiers implemented in scikit-learn.[11]
- TempoLSTM [59] models the sequence of apps using a two-layer network of LSTM units. The temporal information as well as the application is directly passed to each LSTM node.
- *NeuSA$_{w/o\ user}$, NeuSA$_{w/o\ time}$, and NeuSA$_{w/o\ user,\ w/o\ time}$*: These are three variations of our model. The only difference is in the use of time and user features in the models. NeuSA$_{\text{w/o user}}$ is trained without user data; NeuSA$_{\text{w/o time}}$ without time data; and NeuSA$_{\text{w/o user, w/o time}}$ without neither of them.

8 RESULTS AND DISCUSSION

In the following, we evaluate the performance of CNTAS trained on both data splits and study the impact of context on the performance. We further analyze how the models perform on both data splits.

8.1 Target Apps Selection

Performance comparison. Table 4 lists the performance of our proposed methods versus the compared methods. First, we compare the relative performance drop between the two data splits. We see that almost all non-contextual models perform worse on ISTAS-T compared to ISTAS-R, whereas almost all context-aware models perform better on ISTAS-T. Among the non-contextual methods, ListNet is the most robust model with the lowest performance drop and k-NN-AWE is the only method that performs better on ISTAS-T (apart from MFU). Worse results achieved by MFU suggests that ISTAS-T is less biased toward most popular apps, hence being more challenging. However, QueryLM exhibits the highest performance drop (−27% on average), as opposed to Contextual-k-NN-AWE with the highest performance improvement on ISTAS-T (+10%

[10] https://sourceforge.net/lemur/wiki/RankLib/.
[11] https://scikit-learn.org/.

Table 4. Performance Comparison with Baselines on ISTAS-R and ISTAS-T Datasets

Methods	ISTAS-R Dataset				ISTAS-T Dataset			
	MRR	nDCG@1	nDCG@3	nDCG@5	MRR	nDCG@1	nDCG@3	nDCG@5
MFU	0.4502	0.2597	0.4435	0.4891	0.4786	0.2884	0.4752	0.5173
QueryLM	0.3556	0.2431	0.3534	0.3900	0.2706	0.1486	0.2713	0.3097
BM25	0.4205	0.3134	0.4363	0.4564	0.3573	0.2447	0.3771	0.3948
BM25-QE	0.4319	0.2857	0.4371	0.4727	0.3930	0.2411	0.4053	0.4364
k-NN	0.4433	0.2761	0.4455	0.4811	0.4067	0.2294	0.3982	0.4655
k-NN-AWE	0.4742	0.2937	0.4815	0.5211	0.4859	0.2950	0.4919	0.5392
ListNet	0.5170	0.3330	0.5211	**0.5623**	0.5118	0.3219	**0.5208**	0.5572
NTAS-pointwise	0.5221	0.3427	0.5231	0.5586	0.5162	0.3385	0.5162	0.5550
NTAS-pairwise	**0.5257**	**0.3468**	**0.5236**	0.5618	**0.5214**	**0.3427**	0.5183	**0.5580**
Context-Aware Methods								
MFU-CR	0.4903	0.3015	0.4901	0.5268	0.5289	0.3576	0.5358	0.5573
QueryLM-CR	0.4540	0.2773	0.4426	0.5013	0.4696	0.3023	0.4597	0.5145
BM25-CR	0.5398	0.3653	0.5394	0.5871	0.5249	0.3496	0.5255	0.5723
BM25-QE-CR	0.5215	0.3398	0.5223	0.5693	0.5230	0.3474	0.5260	0.5728
k-NN-CR	0.4978	0.3114	0.4926	0.5431	0.5161	0.3481	0.4956	0.5555
k-NN-AWE-CR	0.5144	0.3233	0.5142	0.5632	0.5577	0.3722	0.5612	**0.6086**
ListNet-CR	0.5391	0.3544	0.5417	0.5845	0.5599	0.3780	0.5657	0.6037
ListNet-CX	0.5349	0.3580	0.5343	0.5784	0.5019	0.3139	0.5153	0.5521
NTAS-pointwise-CR	0.5532	0.3745	0.5580	0.5883	0.5627	0.3865	0.5663	0.5965
NTAS-pairwise-CR	0.5576	0.3779	0.5568	0.5870	0.5683	0.3923	0.5661	0.6047
CNTAS-pointwise	0.5614*	0.3833*	**0.5592**	0.5901	0.5702*	0.4146*	0.5655	0.5938
CNTAS-pairwise	**0.5637***	**0.3861***	0.5586	**0.5924***	**0.5738***	**0.4182***	**0.5679**	0.6071

The superscript * denotes significant differences compared to all the baselines.

on average). This indicates that k-NN-AWE is able to capture similar queries effectively, whereas QueryLM relies heavily on the indexed queries. It should also be noted that MFU performs better on ISTAS-T indicating that it is more biased toward popular apps.

Among the non-contextual baselines, we see that NTAS-pairwise performs best in terms of most evaluation metrics on both data splits, this is because it learns high dimensional app and query representations that help it to perform more effectively. We see that applying the contextual filter improves the performance of all models. These improvements are statistically significant in all cases, so are not shown in the table. Although this filter is very simple, it is still able to incorporate useful information about user context and behavior into the ranking. This also indicates the importance of apps usage context, as mentioned in Section 4.5. Among the context-aware baselines, we see that NTAS-pairwise-CR performs best in terms of MRR and nDCG@1, while k-NN-AWE-CR and ListNet-CR perform better in terms of other evaluation metrics. It should also be noted that ListNet-CR performs better than ListNet-CX. This happens due to the fact that ListNet-CX integrates the apps usage context as an additional feature, whereas ListNet-CR is the result of the combination of ListNet and the contextual filter.

We see that our proposed CNTAS outperforms all the baselines with respect to the majority of evaluation metrics. In particular CNTAS-pairwise exhibits the best performance. The achieved improvements in terms of MRR and nDCG@1 are statistically significant. The reason is that CNTAS is able to learn latent features from the interaction of mobile usage data in the context.

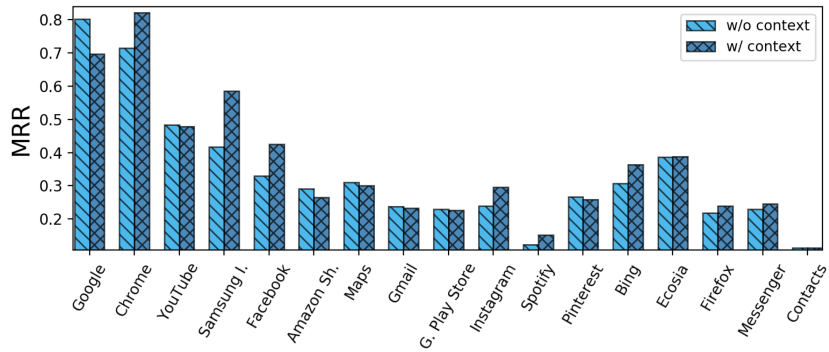
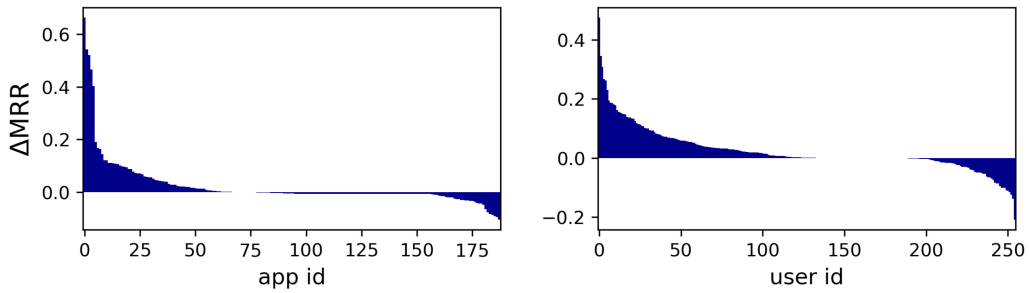


Fig. 11. Performance comparison with respect to certain apps with and without context.



(a) ΔMRR per app (b) ΔMRR per user

Fig. 12. Performance difference per app and per user in terms of ΔMRR.

These interactions can reveal better information for better understanding the user information needs.

Impact of context on performance per app. In this experiment we demonstrate the effect of context on the performance with respect to various apps. Figure 11 shows the performance for queries that are labeled for specific target apps (as listed in the figure). We see that the context-aware model performs better while predicting social media apps such as `Facebook` and `Instagram`. However, we see that the performance for `Google` drops as it improves for `Chrome`. This happens because users do most of their browsing activities on `Chrome`, rather than on `Google`; hence the usage statistics of `Chrome` helps the model to predict it more effectively. Moreover, we study the difference of MRR between the model with and without context for all apps. Our goal is to see how context improves the performance for every target app. We see in Figure 12(a) that the performance is improved for 39% of the apps. As shown in the figure, the improvements are much larger compared with the performance drops. Among the apps with the highest context improvements, we can mention `Quora`, `Periscope`, and `Inbox`.

Impact of context on performance per user. Here we study the difference of MRR between the model with and without context for all users. Our goal is to see how many users are impacted positively by incorporating context in the target apps selection model. Figure 12(b) shows how performance differs per user when we apply context compared with when we do not. As we can see, users' apps usage context is able to improve the effectiveness of target apps selection for the majority of users. In particular, the performance for 57% of the users is improved by incorporating

Table 5. Performance Analysis Based on Query Length, Dividing the Test Queries into Three Evenly Sized Length Buckets

	Short queries	Med. queries	Long queries
	MRR	MRR	MRR
w/o context	0.5302	0.4924	0.4971
w/ context	**0.5733**	**0.5190**	**0.4977**

Table 6. Performance Comparison with Baselines on the LSApp Dataset

Method	MRR	nDCG@1	nDCG@3	nDCG@5	Recall@3	Recall@5
MFU	0.2630	0.1009	0.2800	0.3413	0.3229	0.4554
MRU	0.3652	0.0323	0.3232	0.4106	0.2928	0.4898
Bayesian [30]	0.1461	0.0679	0.1506	0.1687	0.1664	0.2047
Linear [30]	0.1363	0.0592	0.1433	0.1626	0.1636	0.2043
LambdaMART	0.3809	0.2257	0.4532	0.4739	0.4734	0.4958
ListNet	0.4992	0.3908	0.5477	0.5683	0.5632	0.6069
k-NN	0.4824	0.3699	0.5158	0.5519	0.5396	0.6165
DecisionTree	0.5025	0.4422	0.5315	0.5375	0.5343	0.5471
TempoLSTM [59]	0.6869	0.5715	0.7424	0.7730	0.7668	0.8316
NeuSA$_{\text{w/o user, w/o time}}$	0.6924*	0.5677	0.7539*	0.7817*	0.7817*	0.8547*
NeuSA$_{\text{w/o user}}$	0.6971**	0.5721*	0.7592**	0.7940**	0.7873**	0.8608**
NeuSA$_{\text{w/o time}}$	0.7036**†	0.5745*	0.7672**†	0.8053**†	0.7964**†	0.8770**†
NeuSA	0.7049**†‡	0.5763**†	0.7680**†	0.8062**†	0.7969**†	0.8779**†

The superscripts $*$, $\star$, $\dagger$, and $\ddagger$ denote significant improvements compared to all the baselines, NeuSA$_{\text{w/o user, w/o time}}$, NeuSA$_{\text{w/o user}}$, and NeuSA$_{\text{w/o time}}$ respectively ($p < 0.001$).

the apps usage context. In fact, we observed that users with the highest impact from context use less popular apps.

Impact of context on performance per query length. We create three buckets of test queries based on query length uniformly. Therefore, the buckets will have approximately equal number of queries. The first bucket, called Short queries, contains the shortest queries, the second one, called Med. queries, constitutes of medium-length queries and the last bucket, called Long queries, obviously includes the longest queries of our test set. Table 5 lists the performance of the model with and without context in terms of MRR. As we can see, the average MRR for all three buckets is improved as we apply context. However, we observe that as the queries become shorter, the improvement increases. The reason is that shorter queries tend to be more general or ambiguous, and thus query context can have higher impact on improving search for these queries.

8.2 Target Apps Recommendation

In the following, we evaluate the performance of NeuSA trained on LSApp and study the impact of time and user features as well as of the learned app representations.

Performance comparison. Table 6 lists the performance of our proposed method as well as its variations and baselines. As we can see, ListNet exhibits the best performance among LTR baselines and DecisionTree among classification baselines. Moreover, all models outperform MFU in terms of all evaluation metrics. In particular, we see that Recall@5 is improved for all methods, indicating that allowing most used apps to run in the background is not effective. Also, we see

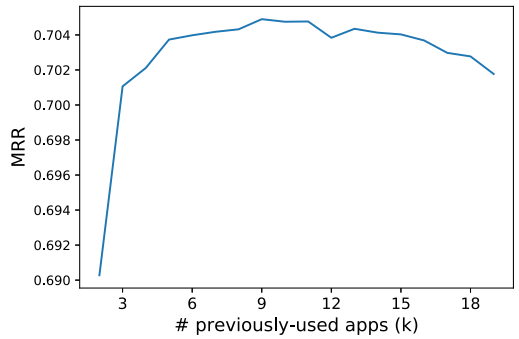


Fig. 13. Performance of NeuSA in terms of MRR for different number of previously used apps as context (k).

that while ListNet consistently outperforms LambdaMART, k-NN exhibits a better performance than DecisionTree in terms on Recall@3 and Recall@5. We see that all models, including MFU and MRU, outperform the statistical baselines, namely, Bayesian and Linear. The large margin in the performance of simple models such as k-NN with these two models indicates the effectiveness of representation-based features (i.e., AWE similarity) for this task. Furthermore, we see that NeuSA outperforms all the baselines by a large margin in terms of all evaluation metrics. For instance, we see a 39% relative improvement over DecisionTree in terms of MRR and a 40% relative improvement over k-NN in terms of Recall@5. This suggests that learning high dimensional sequence-aware representation of apps enables the model to capture users behavioral patterns in using their smartphones. It is worth noting that NeuSA achieves a high value of Recall@5, suggesting that a mobile operating system is able to pre-load 5 apps with an 87% recall value.

Impact of time and user features. To evaluate the impact of time and user features we compare the performance of NeuSA with three variations called $\text{NeuSA}_{\text{w/o user}}$, $\text{NeuSA}_{\text{w/o time}}$, and $\text{NeuSA}_{\text{w/o user, w/o time}}$. As we described earlier, these three models are trained after removing user and time features from the data. We see that in all cases, the performance consistently drops. In particular, we see that when both user and time features are removed, $\text{NeuSA}_{\text{w/o user, w/o time}}$ exhibits the largest performance loss, while still outperforming all the baseline models for the majority of metrics. As we add the user feature to the model, we see that the performance improves, showing that a personalized app recommendation model is effective. In particular, we see that $\text{NeuSA}_{\text{w/o time}}$ outperforms $\text{NeuSA}_{\text{w/o user, w/o time}}$ significantly in terms of all evaluation metrics. Also, we see a large drop of performance when we remove the user data from NeuSA, confirming again that personal app usage patterns should be taken into consideration for this problem. Therefore, a practical system can be trained on a large dataset of app usage from various users and be fine-tuned on every user's phone according to their personal usage behavior. Furthermore, although we see that adding time to $\text{NeuSA}_{\text{w/o user, w/o time}}$ model results in significant improvements (i.e., $\text{NeuSA}_{\text{w/o user}}$), we do not observe the same impact after adding the user data to the model (comparing NeuSA against $\text{NeuSA}_{\text{w/o time}}$). This suggests that while temporal information contain important information revealing time-dependent app usage patterns, it does not add useful information to the personal model. This can be due to the fact that the personal information already conveys the temporal app usage behavior of the user (i.e., each user temporal behavior is unique).

Impact of number of the context length. Here, we evaluate the effect of the number of previously used apps that we consider in our NeuSA model. To do so, we keep all the model parameters the same and change the number of apps in the context (k). We plot the performance of NeuSA for various k values in Figure 13. As we see in the figure, even though the performance somewhat

converges with $k \geq 3$, the best performance is achieved with $k = 9$. This indicates that while the model depends highly on the latest three apps that have been used by the user, it can learn some longer patterns in some rare cases. Moreover, it is worth noting that the model's performance using only one app in the context in terms of MRR is 0.5509, indicating that using only one app is not enough for accurate prediction of next-app usage.

9 CONCLUSIONS AND FUTURE WORK

In this article, we conducted the first *in situ* study on the task of target apps selection, which was motivated by the growing interest in intelligent assistants and conversational search systems where users interact with a universal voice-based search system [1, 4, 7, 34, 61]. To this aim, we developed an app, uSearch, and recruited 255 participants, asking them to report their real-life cross-app mobile queries via uSearch. We observed notable differences in length and structure among queries submitted to different apps. Furthermore, we found that while users search using various apps, a few apps attract most of the search queries. We found that even though `Google` and `Chrome` are the most popular apps, users do only 26% and 23% of their searches in these apps, respectively. The *in situ* data collection enabled us to collect valuable information about users' contexts. For instance, we found that the target app for 29% of the queries were among the top two most used apps of a particular user. Inspired by our data analysis, we proposed a model that learns high-dimensional latent representations for the apps usage context and predicts the target app for a query. The model was trained with an end-to-end setting. Our model produces a score for a given context-query-app triple. We compared the performance of our proposed method with state-of-the-art retrieval baselines splitting data following two different strategies. We observed that our approach outperforms all baselines, significantly.

Furthermore, we proposed a neural sequence-aware model, called NeuSA, for predicting next app usage. NeuSA learns a high-dimensional representation for mobile apps, incorporating the app usage sequence as well as temporal and personal information into the model. We trained the model on the app usage data collected from 292 real users. The results showed that the proposed model is able to capture complex user behavioral patterns while using their phones, outperforming classification and LTR baselines significantly in terms of nDCG@k, Recall@k, and MRR.

Limitations. Like any other study, our study has some limitations. First, the study relies on self-reporting. This could result in specific biases in the collected data. For instance, participants may prefer to report shorter queries simply because it requires less work. Also, in many cases, participants are likely to forget reporting queries or do not report all the queries that belong to the same session. Second, the reported queries are not actually submitted to a unified search system and users may formulate their queries differently is such setting. For example, in a unified system a query may be "videos of Joe Bonamassa" but in `YouTube` it may be "Joe Bonamassa." Both the mentioned limitations are mainly due to lack of an existing unified mobile search app. Hence, building such app would be essential for building a more realistic collection. Also, our study does not consider the users' success or failure in their search. Submitting queries in certain apps could result in different chances of success, and consequently, affect users' behavior in the session to submit other queries in the same app or other apps. Finally, more efficient data collection strategies could be employed based on active learning [46].

Future work. The next step in this research would be exploring the influence of other types of contextual information, such as location and time, on the target apps selection and recommendation tasks. In addition, it would be interesting to explore result aggregation and presentation in the future, considering two important factors: information gain and user satisfaction. This direction can be studied in both areas of information retrieval and human-computer interaction.

Furthermore, based on our findings in the analyses, we believe that mobile search queries can be leveraged to improve the user experience. For instance, assuming a user searches for a restaurant using a unified search system and finds some relevant information on `Yelp`. In this case, considering the user's personal preference as well as the context, the system could send the user a notification with information about the traffic near the restaurant. This would certainly improve the quality of the user experience. We also plan to investigate if the demographics of the participants are linked to particular queries and behavior. And if such behavioral biases exist, how different models are able to address such issues?

ACKNOWLEDGMENTS

We thank the anonymous reviewers for the valuable feedback.

REFERENCES

[1] Mohammad Aliannejadi, Manajit Chakraborty, Esteban Andrés Ríssola, and Fabio Crestani. 2020. Harnessing evolution of multi-turn conversations for effective answer retrieval. In *CHIIR*. 33–42.
[2] Mohammad Aliannejadi and Fabio Crestani. 2017. Venue appropriateness prediction for personalized context-aware venue suggestion. In *SIGIR*. 1177–1180.
[3] Mohammad Aliannejadi, Morgan Harvey, Luca Costa, Matthew Pointon, and Fabio Crestani. 2019. Understanding mobile search task relevance and user behaviour in context. In *CHIIR*. 143–151.
[4] Mohammad Aliannejadi, Julia Kiseleva, Aleksandr Chuklin, Jeff Dalton, and Mikhail S. Burtsev. 2020. ConvAI3: Generating clarifying questions for open-domain dialogue systems (ClariQ). *CoRR* abs/2009.11352 (2020).
[5] Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W. Bruce Croft. 2018. In situ and context-aware target apps selection for unified mobile search. In *CIKM*. 1383–1392.
[6] Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W. Bruce Croft. 2018. Target apps selection: Towards a unified search framework for mobile devices. In *SIGIR*. 215–224.
[7] Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W. Bruce Croft. 2019. Asking clarifying questions in open-domain information-seeking conversations. In *SIGIR*. 475–484.
[8] Giambattista Amati. 2003. *Probability Models for Information Retrieval Based on Divergence from Randomness*. Ph.D. Dissertation. University of Glasgow, UK.
[9] Jaime Arguello. 2017. Aggregated search. *Found. Trends Inf. Retriev.* 10, 5 (2017), 365–502.
[10] Ricardo A. Baeza-Yates, Di Jiang, Fabrizio Silvestri, and Beverly Harrison. 2015. Predicting the next app that you are going to use. In *WSDM*. 285–294.
[11] Linas Baltrunas, Karen Church, Alexandros Karatzoglou, and Nuria Oliver. 2015. Frappe: Understanding the usage and perception of mobile app recommendations in-the-wild. *CoRR* abs/1505.03014 (2015).
[12] Steven M. Beitzel, Eric C. Jensen, Abdur Chowdhury, David A. Grossman, and Ophir Frieder. 2004. Hourly analysis of a very large topically categorized web query log. In *SIGIR*. 321–328.
[13] Jan R. Benetka, Krisztian Balog, and Kjetil Nørvåg. 2017. Anticipating information needs based on check-in activity. In *WSDM*. 41–50.
[14] Paul N. Bennett, Filip Radlinski, Ryen W. White, and Emine Yilmaz. 2011. Inferring and using location metadata to personalize web search. In *SIGIR*. 135–144.
[15] James P. Callan and Margaret E. Connell. 2001. Query-based sampling of text databases. *ACM Trans. Inf. Syst.* 19, 2 (2001), 97–130.
[16] Huanhuan Cao, Derek Hao Hu, Dou Shen, Daxin Jiang, Jian-Tao Sun, Enhong Chen, and Qiang Yang. 2009. Context-aware query classification. In *SIGIR*. 3–10.
[17] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: From pairwise approach to listwise approach. In *ICML*. 129–136.
[18] Juan Pablo Carrascal and Karen Church. 2015. An in-situ study of mobile app & mobile search interactions. In *CHI*. 2739–2748.
[19] Karen Church and Nuria Oliver. 2011. Understanding mobile web and mobile search use in today's dynamic mobile landscape. In *Mobile HCI*. 67–76.
[20] Karen Church, Barry Smyth, Keith Bradley, and Paul Cotter. 2008. A large scale study of European mobile search behaviour. In *Mobile HCI*. 13–22.
[21] Karen Church, Barry Smyth, Paul Cotter, and Keith Bradley. 2007. Mobile information access: A study of emerging search behavior on the mobile Internet. *Trans. Web* 1, 1 (2007), 4.

[22] Fabio Crestani and Heather Du. 2006. Written versus spoken queries: A qualitative and quantitative comparative analysis. *J. Assoc. Inf. Soc. Technol.* 57, 7 (2006), 881–890.
[23] Fabio Crestani, Stefano Mizzaro, and Ivan Scagnetto. 2017. *Mobile Information Retrieval*. Springer.
[24] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. 2017. Neural ranking models with weak supervision. In *SIGIR*. 65–74.
[25] Fernando Diaz. 2009. Integration of news content into web results. In *WSDM*. 182–191.
[26] Ido Guy. 2016. Searching by talking: Analysis of voice queries on mobile web search. In *SIGIR*. 35–44.
[27] Martin Halvey, Mark T. Keane, and Barry Smyth. 2006. Mobile web surfing is the same as web surfing. *Commun. ACM* 49, 3 (2006), 76–81.
[28] Martin Halvey, Mark T. Keane, and Barry Smyth. 2006. Time based patterns in mobile-internet surfing. In *CHI*. 31–34.
[29] Shun Hattori, Taro Tezuka, and Katsumi Tanaka. 2007. Context-aware query refinement for mobile web search. In *SAINT Workshops*.
[30] Ke Huang, Chunhui Zhang, Xiaoxiao Ma, and Guanling Chen. 2012. Predicting mobile application usage using contextual information. In *Ubicomp*. 1059–1065.
[31] Maryam Kamvar and Shumeet Baluja. 2006. A large scale study of wireless search behavior: Google mobile search. In *CHI*. 701–709.
[32] Maryam Kamvar and Shumeet Baluja. 2007. The role of context in query input: Using contextual signals to complete queries on mobile devices. In *Mobile HCI*. 405–412.
[33] In-Ho Kang and Gil-Chang Kim. 2003. Query type classification for web document retrieval. In *SIGIR*. 64–71.
[34] Antonios Minas Krasakis, Mohammad Aliannejadi, Nikos Voskarides, and Evangelos Kanoulas. 2020. Analysing the effect of clarifying questions on document ranking in conversational search. In *ICTIR*. 129–132.
[35] Joohyun Lee, Kyunghan Lee, Euijin Jeong, Jaemin Jo, and Ness B. Shroff. 2016. Context-aware application scheduling in mobile systems: What will users do and not do next? In *UbiComp*. 1235–1246.
[36] Hang Li. 2011. *Learning to Rank for Information Retrieval and Natural Language Processing*. Morgan & Claypool.
[37] Zhung-Xun Liao, Po-Ruey Lei, Tsu-Jou Shen, Shou-Chung Li, and Wen-Chih Peng. 2012. Mining temporal profiles of mobile applications for usage prediction. In *ICDM*. 890–893.
[38] Eric Hsueh-Chan Lu, Yi-Wei Lin, and Jing-Bin Ciou. 2014. Mining mobile application sequential patterns for usage prediction. In *GrC*. 185–190.
[39] Ilya Markov and Fabio Crestani. 2014. Theoretical, qualitative, and quantitative analyses of small- document approaches to resource selection. *ACM Trans. Inf. Syst.* 32, 2 (2014), 9–37.
[40] George D. Montanez, Ryen W. White, and Xiao Huang. 2014. Cross-device search. In *CIKM*. 1669–1678.
[41] Iadh Ounis, Gianni Amati, Vassilis Plachouras, Ben He, Craig Macdonald, and Douglas Johnson. 2005. Terrier information retrieval platform. In *ECIR*. 517–519.
[42] Dae Hoon Park, Yi Fang, Mengwen Liu, and ChengXiang Zhai. 2016. Mobile app retrieval for social media users via inference of implicit intent in social media text. In *CIKM*. 959–968.
[43] Dae Hoon Park, Mengwen Liu, ChengXiang Zhai, and Haohong Wang. 2015. Leveraging user reviews to improve accuracy for mobile app retrieval. In *SIGIR*. 533–542.
[44] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*. 1532–1543.
[45] Jay M. Ponte and W. Bruce Croft. 1998. A language modeling approach to information retrieval. In *SIGIR*. 275–281.
[46] Md. Mustafizur Rahman, Mucahid Kutlu, and Matthew Lease. 2019. Constructing test collections using multi-armed bandits and active learning. In *WWW*. 3158–3164.
[47] Ivan Sekulic, Amir Soleimani, Mohammad Aliannejadi, and Fabio Crestani. 2020. Longformer for MS MARCO document re-ranking task. *CoRR* abs/2009.09392 (2020).
[48] Dou Shen, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2006. Building bridges for web query classification. In *SIGIR*. 131–138.
[49] Milad Shokouhi and Qi Guo. 2015. From queries to cards: Re-ranking proactive card recommendations based on reactive search history. In *SIGIR*. 695–704.
[50] Milad Shokouhi and Luo Si. 2011. Federated search. *Found. Trends Inf. Retriev.* 5, 1 (2011), 1–102.
[51] Craig Silverstein, Monika Rauch Henzinger, Hannes Marais, and Michael Moricz. 1999. Analysis of a very large web search engine query log. *SIGIR Forum* 33, 1 (1999), 6–12.
[52] Timothy Sohn, Kevin A. Li, William G. Griswold, and James D. Hollan. 2008. A diary study of mobile information needs. In *CHI*. 433–442.
[53] Yang Song, Hao Ma, Hongning Wang, and Kuansan Wang. 2013. Exploring and exploiting user search behavior on mobile and tablet devices to improve search relevance. In *WWW*. 1201–1212.
[54] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15, 56 (2014), 1929–1958.

[55] Niels van Berkel, Chu Luo, Theodoros Anagnostopoulos, Denzil Ferreira, Jorge Gonçalves, Simo Hosio, and Vassilis Kostakos. 2016. A systematic assessment of smartphone usage gaps. In *CHI*. 4711–4721.
[56] Yingzi Wang, Nicholas Jing Yuan, Yu Sun, Fuzheng Zhang, Xing Xie, Qi Liu, and Enhong Chen. 2016. A contextual collaborative approach for app usage forecasting. In *UbiComp*. 1247–1258.
[57] Ryen W. White, Paul N. Bennett, and Susan T. Dumais. 2010. Predicting short-term interests using activity-based search context. In *CIKM*. 1009–1018.
[58] Biao Xiang, Daxin Jiang, Jian Pei, Xiaohui Sun, Enhong Chen, and Hang Li. 2010. Context-aware ranking in web search. In *SIGIR*. 451–458.
[59] Shijian Xu, Wenzhong Li, Xiao Zhang, Songcheng Gao, Tong Zhan, Yongzhu Zhao, Wei-wei Zhu, and Tianzi Sun. 2018. Predicting smartphone app usage with recurrent neural networks. In *WASA*. 532–544.
[60] Hamed Zamani, Michael Bendersky, Xuanhui Wang, and Mingyang Zhang. 2017. Situational context for ranking in personal search. In *WWW*. 1531–1540.
[61] Hamed Zamani and Nick Craswell. 2020. Macaw: An extensible conversational information seeking platform. In *SIGIR*. 2193–2196.
[62] Hamed Zamani and W. Bruce Croft. 2016. Estimating embedding vectors for queries. In *ICTIR*. 123–132.
[63] Sha Zhao, Zhiling Luo, Ziwen Jiang, Haiyan Wang, Feng Xu, Shijian Li, Jianwei Yin, and Gang Pan. 2019. AppUsage2Vec: Modeling smartphone app usage for prediction. In *ICDE*. 1322–1333.

Received April 2020; revised January 2021; accepted January 2021