In Situ and Context-Aware Target Apps Selection for Unified Mobile Search

Mohammad Aliannejadi
Università della Svizzera italiana (USI)
mohammad.alian.nejadi@usi.ch

Fabio Crestani
Università della Svizzera italiana (USI)
fabio.crestani@usi.ch

Hamed Zamani
University of Massachusetts Amherst
zamani@cs.umass.edu

W. Bruce Croft
University of Massachusetts Amherst
croft@cs.umass.edu

ABSTRACT

With the recent growth in the use of conversational systems and intelligent assistants such as Google Assistant and Microsoft Cortana, mobile devices are becoming even more pervasive in our lives. As a consequence, users are getting engaged with mobile apps and frequently search for an information need using different apps. Recent work has stated the need for a unified mobile search system that would act as a meta search on users’ mobile devices: it would identify the target apps for the user’s query, submit the query to the apps, and present the results to the user. Moreover, mobile devices provide rich contextual information about users and their whereabouts. In this paper, we introduce the task of context-aware target apps selection as part of a unified mobile search framework. To this aim, we designed an in situ study to collect thousands of mobile queries enriched with mobile sensor data from 255 users during a three month period. With the aid of this dataset, we were able to study user behavior as they performed cross-app search. We finally study the performance of state-of-the-art retrieval models for this task and propose a simple yet effective neural model that significantly outperforms the baselines. Our neural approach is based on learning high-dimensional representations for mobile apps and contextual information. Furthermore, we show that incorporating context improves the performance by 20% in terms of nDCG@5, enabling the model to perform better for 57% of users. Our data is publicly available for research purposes.

ACM Reference Format:

1 INTRODUCTION

In recent years, mobile devices have become the main means of connecting to the Internet for many people. This has resulted in a tremendous number of apps that are now available on mobile app markets. In particular, Google Play Store now features more than 3.5 million apps and an average user installs only around 35 of them,1 many of which provide services such as music and location search. In addition, the emergence of intelligent assistants, such as Google Assistant, has made mobile devices even more pervasive, providing users with a universal voice-based search interface. However, as users spend most of their time working with apps (rather than a browser),2 these systems still have a long way to go to provide a unified interface with the wide variety of the apps. For these reasons, we have recently discussed the need for a universal mobile search system that would act as a meta search engine, to which users would submit all their queries. The system should identify the target apps, route the query to them, and display the returned results in an integrated interface. Thus, the first step towards designing a unified mobile search framework is identifying the target apps for a given query, called the target apps selection task [2].

As mobile devices provide rich contextual information about users, previous studies [1, 22, 43] have tried to incorporate query context in various domains. In particular, query context is often defined as the information from the previous queries and their corresponding clickthrough data [40, 41], or situational context such as location and time [6, 20, 43]. However, as user interactions on mobile devices are mostly with apps, exploring apps usage patterns reveals important information about the users contexts, information needs, and behavior. For instance, a user who starts spending more intensely as they do mobile search.

However, our previous attempt to study unified mobile search through crowdsourcing failed to capture users’ contexts while collecting data [2]. In addition, there are some other limitations. For example, we asked the workers to complete a set of given search tasks, which obviously were not their actual information needs, and thus the queries may differ from real search queries. In addition, the workers did not complete their work on mobile devices, which affects their behavior. Furthermore, the user behavior and queries could not be studied in a day-long or week-long period.

The aforementioned limitations have motivated us to conduct the first in situ study on target apps selection for unified mobile

1https://www.thinkwithgoogle.com/advertising-channels/apps/app-marketing-trends-mobile-landscape/
search. This enables us to obtain more clear insights into the task. In particular, we are interested in studying the users’ behavior as they search for their real-life information needs using their own mobile devices. Moreover, we study the impact of contextual information on the apps they use 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. Through an open call, we recruited 255 participants who installed uSearch and used it to report their queries as well as the target apps, right after they did a search on their smartphones. With participants’ consents, uSearch also ran in the background collecting useful contextual data. We have released the code of uSearch to facilitate research on mobile information retrieval. In fact, uSearch is extendable and can be used for collecting data to study various search tasks on mobile devices. 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 conduct an extensive data analysis, aiming to understand how users’ behavior vary across different apps while they search for their information needs. The key findings 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, and leave 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 CNTAS. In our model, we deal with the problem as a ranking task estimating a relevance score for a given context-query-app triple. Our experiments demonstrate that our model significantly outperforms state-of-the-art retrieval models in this task. Also, we show that incorporating context improves nDCG@5 by an average of 20% on all models and improves the performance with respect to 57% of the users.

In summary, the main contributions of this paper include:

- Designing and conducting an in situ mobile search study for collecting thousands of real-life cross-app queries. Both the app³ and the collected data⁴ are publicly available for research purposes.
- Presenting 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.
- Proposing a context-aware neural model for target apps selection.
- Evaluating the performance of state-of-the-art retrieval models for this task and comparing them against our proposed model.

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

2 RELATED WORK

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

Mobile IR. A mobile IR system aims at enabling users to carry out all the classical IR operations on a mobile device [15], as the conventional Web-based approaches fail to satisfy users’ information needs on mobile devices [12]. In fact, Song et al. [37] found significant differences in search patterns done using iPhone, iPad, and desktop. Research on mobile IR started by Kamvar and Baluja [21] where they did a large-scale mobile search query analysis, finding mobile search topics were less diverse. Guy [19] and Crestani and Du [14] conducted comparative studies on mobile spoken and typed-in queries showing that spoken queries are longer and closer to natural language. Montaze et al. [25] studied search across multiple devices including smartphones. Park et al. [28] represented apps using online reviews for improving the app retrieval performance. Park et al. [27] inferred users implicit intentions from social media for the task of app recommendation. This work is closely related to our previous work [2] where we introduced the need for a unified mobile search framework as we collected cross-app queries through crowdsourcing. In contrast, in this work, we collect real-life cross-app queries over a longer period of time with an in situ study design.

Context-aware IR. Most of the previous work in context-aware search is based on the user’s search history [33, 40, 41]. Shen et al. [33] presented context-sensitive language models based on users’ short-term search history. White et al. [40] investigated ways to optimally combine the query and its context by learning a model that predicts the context weight for each query. Bennett et al. [6] estimated the location preference of a document and used it to improve Web search. Most recently, Zamani et al. [43] explored the effect of situational context for personal search.

Mobile HCI. A large body of research has been done on mobile information need analysis. Sohn et al. [36] conducted a diary study in which they found that contextual features such as activity and time influence 72% of mobile information needs. Church and Oliver [11] did a diary and interview study with the aim of understanding users’ mobile Web behavior. Pielot et al. [30] conducted an in situ study of mobile phone notifications. They found that depending on the type of notifications, different strategies should be employed for delivering them. Carrascal and Church [10] studied user interactions with respect to mobile apps and mobile search, finding that users’ interactions with apps have impact on search. In contrast to this prior research, we conduct a large-scale in situ study, enabling us to collect enough cross-app queries to build a more reliable data collection.

Proactive IR. The aim of proactive IR systems is to anticipate users’ information needs and proactively present information cards to them. Shokouhi and Guo [34] analyzed user interactions with information cards and found that the usage patterns of the cards

---

¹https://github.com/aliannejadi/uSearch
²http://aliannejadi.com/istas.html
We first describe the characteristics of uSearch. Then, we provide 

Federated and aggregated search. Research on unified mobile search has a considerable overlap with federated and aggregated search. While federated search systems assume the environment to be uncooperative and data to be homogeneous, aggregated search systems blend heterogeneous content from cooperative resources [4]. Target apps selection, on the other hand, assumes an uncooperative environment and heterogeneous content. Callan and Connell [7] proposed a query-based sampling approach to probe uncooperative resources. Diaz [17] proposed modeling the query dynamics to detect news queries for integrating the news vertical in SERP.

Query classification. Different strategies are used to assign a query to predefined categories. Kang and Kim [23] defined three types of queries, each of which requiring the search engine to handle differently. Shen et al. [32] introduced an intermediate taxonomy used to classify queries to specified target categories. Cao et al. [8] leveraged conditional random fields to incorporate users’ neighboring queries in a session as context. More recently, Zamani and Croft [44] studied word embedding vectors for the query classification task and proposed a formal model for query embedding estimation.

Neural IR. The successful development of deep neural networks for various tasks has also impacted IR applications. In particular, neural ranking models have recently shown significant improvements in a wide range of IR tasks, such as ad-hoc retrieval [18], question answering [42], and context-aware retrieval [43]. These approaches often rely on learning high-dimensional dense representations that carry semantic information. They can be particularly useful to match queries and documents where minimal term overlap exists. We also take advantage of such latent high-dimensional representations in our model for representing mobile apps.

3 DATA COLLECTION

In this section, we describe how we collected ISTAS, which is, to the best of our knowledge, the first in situ dataset on cross-app mobile search queries. We collected the data by recruiting 255 participants through an open call on the Web. 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 the app as well as the query they used to perform the steps required to be done by the participants in order to report a query. In short, we asked them to open uSearch after every search query to predefined categories. Kang and Kim [23] defined three types of queries, each of which requiring the search engine to handle differently. Shen et al. [32] introduced an intermediate taxonomy used to classify queries to specified target categories. Cao et al. [8] leveraged conditional random fields to incorporate users’ neighboring queries in a session as context. More recently, Zamani and Croft [44] studied word embedding vectors for the query classification task and proposed a formal model for query embedding estimation.

Neural IR. The successful development of deep neural networks for various tasks has also impacted IR applications. In particular, neural ranking models have recently shown significant improvements in a wide range of IR tasks, such as ad-hoc retrieval [18], question answering [42], and context-aware retrieval [43]. These approaches often rely on learning high-dimensional dense representations that carry semantic information. They can be particularly useful to match queries and documents where minimal term overlap exists. We also take advantage of such latent high-dimensional representations in our model for representing mobile apps.

3.1 uSearch

In order to facilitate the query report procedure, we developed uSearch, an Android app shown in Figure 1. 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 1, 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 done 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.3 Apps usage events record user interactions 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 online platform. In the announcement, we provided all the details about the intention of the study as well as the data we were collecting. First, we asked them to complete a survey inside uSearch. Moreover, we mentioned all the steps required to be done by the participants in order to report a query. In short, we asked them 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 is very crucial to capture their context at the right moment.

After running several pilot studies, over the 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 5 queries. Since some people may not submit 5 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 5 queries. Also, we encouraged them to continue reporting more than 5 queries for an additional reward. As incentive, we paid the participants $0.2 per query. We recruited participants only from English-speaking countries.

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. As we were paying the 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, 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 out of 7,750 reported queries.

3.4 Privacy Concerns

Before asking for required app permissions, we made clear statement about our intentions on how we are going to use the participants’ collected data as well as what was collected from their devices. We ensured them that their data was stored on secure cloud servers and that they could opt out at any point of the study. 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 the identity of participants.

3.5 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 limitations are due to lack of an existing unified mobile search app. Hence, building such app is essential for building a more realistic collection.

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

During the period of 86 days, with the help of 255 participants, we were able to collect 6,877 search queries and their target apps as well as sensor and usage data. The collected raw data is over 300 gigabytes. Here, we summarize the main characteristics of the participants based on the submitted surveys. 59% of the participants were female and 50% aged between 25 and 34. 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 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. Moreover, Figure 2 shows the number of queries and active participants per day during the data collection period. Note that as shown in Figure 2, in the first half collection period, we were mostly developing the visualization tools and did not recruit many participants.

4.2 Apps

How apps are distributed. Figure 3 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 in line with what was shown in a previous work [2]. However, we observe a higher percentage of searches done using Google and Chrome apps. This can be due to two reasons: (i) ISTAS is collected in situ and on mobile devices, thus being more realistic; (ii) ISTAS queries reflect real-life information needs rather than a set of given search tasks, hence the information need topics are diverse. 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 search tasks, most of which were done by 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 as they searched for real-life information needs, in terms of the apps they chose for performing the search. Figure 4a 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 4a, we observe a totally different distribution compared to [2], 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 [2].

On the other hand, Figure 4b plots the distribution of unique apps with respect to the sessions, that 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 that we leave for future work.

4.3 Queries

In order to understand the differences in user behavior while formulating their information needs using different apps, we conduct 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 9 apps in ISTAS.

How query length differs among apps. The upper part of Table 2 lists the distribution of the 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 \( r \)) per app. All shows query distributions across all apps.

<table>
<thead>
<tr>
<th># terms</th>
<th>Query term distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>22% 13% 11% 22% 12% 25% 57% 49% 29% 81%</td>
</tr>
<tr>
<td>2</td>
<td>28% 26% 29% 48% 45% 27% 30% 33% 35% 10%</td>
</tr>
<tr>
<td>3</td>
<td>20% 21% 24% 16% 25% 18% 9% 12% 24% 7%</td>
</tr>
<tr>
<td>4</td>
<td>12% 13% 18% 10% 10% 13% 3% 4% 7% 2%</td>
</tr>
<tr>
<td>&gt; 4</td>
<td>17% 26% 17% 4% 10% 17% 1% 1% 6% 0%</td>
</tr>
<tr>
<td>Mean</td>
<td>3.00 3.49 3.19 2.34 2.74 3.07 1.61 1.75 2.31 1.31</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>r</th>
<th>Query overlap</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; 0.25</td>
<td>56% 39% 41% 28% 27% 26% 25% 8% 14%</td>
</tr>
<tr>
<td>&gt; 0.50</td>
<td>19% 11% 15% 13% 7% 11% 12% 12% 4% 10%</td>
</tr>
<tr>
<td>&gt; 0.75</td>
<td>13% 3% 8% 11% 5% 9% 12% 10% 2% 10%</td>
</tr>
</tbody>
</table>

Table 2: Cross-app query attributes for 9 apps. The upper part 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 \( r \)) per app. All shows query distributions across all apps.
4.4 Sessions
A session is a “series of queries by a single user made within a small range of time” [35]. Similar to previous work [10, 21, 35], we consider a 5-minute range of inactivity to close 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 4b, 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 more similar to each other in multiple-app sessions. More specifically, query overlap at the threshold of > 0.25 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.

4.5 Context
Temporal behavior. We analyze the behavior of users as they search with respect to day-of-week and time-of-day. The distribution across day-of-week among the participants who reported their queries for more than 6 days slightly peaks on Fridays. Moreover, Figure 5 shows the distribution of queries and unique target apps across time-of-day for all participants. 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 app selection model. Figure 6 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,
to a $d$-dimensional embedding space, and a global term weighting function $W : V \rightarrow \mathbb{R}$ that maps each vocabulary term to a real-valued number showing its global importance. The matrices $E$ and $W$ are the network parameters in our model and are learned to provide task-specific representations. The query representation component $\phi_Q$ represents a given query $q = \{w_1, w_2, \cdots, w_{|q|}\}$ as follows:

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

which is the weighted element-wise summation over the terms’ embedding vectors. $W$ is the normalized global weights computed using a softmax function as follows:

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

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 [2], ad-hoc retrieval [16], and query performance prediction [45].

To implement the app representation component $\phi_A$, we learn a $d$-dimensional dense representation for each app. In more detail, this component consists of an app representation matrix $A \in \mathbb{R}^{N \times d}$ where $N$ denotes the total number of apps. Therefore, $\phi_A(a)$ returns a row of the matrix $A$ 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 [6], personal search [43], and mobile search [20]. In this paper, 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 $\phi_C$, 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. $\phi_C$ is then computed as:

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

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

In summary, each of the representation learning components $\phi_Q$, $\phi_A$, and $\phi_C$ returns a $d$-dimensional vector. The query 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 [38] is employed. For each query, the following vector is fed to this network:

$$(\phi_Q(q) \circ \phi_A(a)) \cdot \phi_Q(q) \cdot \phi_A(a) \cdot \phi_C(c_q) .$$

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 context 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:

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

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 $i$th 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 [24]. 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:

$$L_{Hinge}(b) = \frac{1}{|b|} \sum_{i=1}^{|b|} \max(0, 1 - \text{sign}(y_{1i} - y_{2i})(\tilde{y}_{1i} - \tilde{y}_{2i})).$$

6 EXPERIMENTS

In this section, we evaluate the performance of the proposed models in comparison with a set of state-of-the-art IR models. We also study the performance of the models with respect to the apps and users.

6.1 Experimental Setup

Dataset. 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 set; (ii) In ISTAS-T, we sort the queries of each user chronologically and keep the first 70% of each user’s queries for training, the next 10% for validation, and the last 20% for testing set. 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 [2]: 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:
Table 3: Performance comparison with baselines on ISTAS-R and ISTAS-T. The superscript * denotes significant differences compared to all the baselines.

<table>
<thead>
<tr>
<th>Methods</th>
<th>ISTAS-R Dataset</th>
<th>ISTAS-T Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MRR</td>
<td>nDCG@1</td>
</tr>
<tr>
<td>StaticRanker</td>
<td>0.4502</td>
<td>0.2597</td>
</tr>
<tr>
<td>QueryLM</td>
<td>0.3556</td>
<td>0.2431</td>
</tr>
<tr>
<td>BM25</td>
<td>0.4205</td>
<td>0.3134</td>
</tr>
<tr>
<td>BM25-QE</td>
<td>0.4319</td>
<td>0.2857</td>
</tr>
<tr>
<td>k-NN</td>
<td>0.4433</td>
<td>0.2761</td>
</tr>
<tr>
<td>k-NN-AWE</td>
<td>0.4742</td>
<td>0.2937</td>
</tr>
<tr>
<td>ListNet</td>
<td>0.5170</td>
<td>0.3330</td>
</tr>
<tr>
<td>NTAS-pointwise</td>
<td>0.5221</td>
<td>0.3427</td>
</tr>
<tr>
<td>NTAS-pairwise</td>
<td>0.5257</td>
<td>0.3468</td>
</tr>
</tbody>
</table>

### Context-Aware Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>ISTAS-R Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MRR</td>
</tr>
<tr>
<td>StaticRanker-CR</td>
<td>0.4903</td>
</tr>
<tr>
<td>QueryLM-CR</td>
<td>0.4540</td>
</tr>
<tr>
<td>BM25-CR</td>
<td>0.5398</td>
</tr>
<tr>
<td>BM25-QE-CR</td>
<td>0.5215</td>
</tr>
<tr>
<td>k-NN-CR</td>
<td>0.4978</td>
</tr>
<tr>
<td>k-NN-AWE-CR</td>
<td>0.5144</td>
</tr>
<tr>
<td>ListNet-CR</td>
<td>0.5391</td>
</tr>
<tr>
<td>ListNet-CX</td>
<td>0.5349</td>
</tr>
<tr>
<td>NTAS-pointwise-CR</td>
<td>0.5532</td>
</tr>
<tr>
<td>NTAS-pairwise-CR</td>
<td>0.5576</td>
</tr>
<tr>
<td>CNTAS-pointwise</td>
<td>0.5614*</td>
</tr>
<tr>
<td>CNTAS-pairwise</td>
<td>0.5637*</td>
</tr>
</tbody>
</table>

### Results and Discussion

6.2 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.

**Performance comparison.** Table 3 lists the performance of our proposed methods as well as 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 least performance drop and k-NN-AWE is the only method that performs better on ISTAS-T (apart from StaticRanker). On the other hand, QueryLM exhibits the most performance drop (−27% on average), as opposed to Contextual-k-NN-AWE with the highest performance improvement on ISTAS-T (+10% on average). This indicates that k-NN-AWE is able to capture similar queries more effectively, whereas QueryLM relies heavily on the indexed queries. It should also be noted that StaticRanker performs better on ISTAS-T indicating that it is more biased towards more popular apps.

Among the non-contextual baselines, we see that NTAS-pairwise performs best in terms of most evaluation metrics on both data.

- **StaticRanker:** 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 [31]. For BM25-QE, we adopt Bo1 [3] for query expansion. We use the Terrier [26] 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 [44], we compute the cosine similarity between the average word embedding (AWE) of the queries obtained from GloVe [29] 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 StaticRanker as features to train ListNet [9] implemented in RankLib⁶. 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 [2]. We use the NTAS1 model due to its superior performance compared to NTAS2.
- **Contextual baselines:** In order 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 q in the training samples of user u, we take the time that u has spent on q 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.

---

⁶https://sourceforge.net/p/lemur/wiki/RankLib/
with respect to the majority of evaluation metrics. In particular,
Table 4: Performance analysis based on query length, dividing
the test queries into three evenly-sized length buckets.

<table>
<thead>
<tr>
<th></th>
<th>Short queries</th>
<th>Med. queries</th>
<th>Long queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o context</td>
<td>MRR</td>
<td>MRR</td>
<td>MRR</td>
</tr>
<tr>
<td>w/ context</td>
<td>0.5302</td>
<td>0.4924</td>
<td>0.4971</td>
</tr>
</tbody>
</table>

Impact of context on performance per app. In this experiment
we demonstrate the effect of context on the performance with re-
spect to various apps. Figure 7 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 predict-
ing 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 ac-
tivities on Chrome, rather than 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 8a that the per-
formance 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. We saw that these
apps were less popular among our participants and their represent-
ations were weaker than the other apps, that is why the contextual
information shows the highest improvement for them.

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
8b 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 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. Following
Zamani et al. [46], we create three buckets of test queries based on
query length uniformly. Therefore, the buckets will have approxi-
ately 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, includes the longest queries of our test set. Ta-
ble 4 lists the performance of the model with and without context
in terms of MRR. As we 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.

7 CONCLUSIONS AND FUTURE WORK
In this paper, 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. To this
aim, we developed an app, called uSearch, and recruited 255 par-
ticipants, 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, 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.

An immediate future work can be exploring the influence of other types of contextual information, such as location and time, on the target apps selection task. 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, 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 push a notification with information about the traffic near the restaurant.

Acknowledgements. 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.

REFERENCES

[9] 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.