Adding Intelligence to Your Mobile Device via On-Device Sequential Pattern Mining

Abhishek Mukherji  
Samsung Research America  
1732 North 1st Street, San Jose, CA 95112.  
a.mukherji@samsung.com  
Vijay Srinivasan  
Samsung Research America  
1732 North 1st Street, San Jose, CA 95112.  
v.srinivasan@samsung.com  
Evan Welbourne  
Samsung Research America  
1732 North 1st Street, San Jose, CA 95112.  
e.welbourne@samsung.com  

Abstract  
The next revolution in mobile user experience is predicted to be a smart device that can adapt to its user’s lifestyle and surroundings to become a proactive personal assistant. We introduce the idea of Mobile Sequence Mining (MSM) engine that automatically learns phone usage sequential patterns over the rich context data captured within the device. The learned patterns can then enable variety of applications including proactive assistance for a variety of use cases. Unlike existing cloud-based intelligence services (e.g., GoogleNow) that rely on internet access and may compromise privacy, MSM provides device intelligence by leveraging mined longitudinal patterns while preserving privacy via on-device mining. MSM is generic and can provide sequential patterns and predictions over multiple data streams, also allowing individual mobile applications to stream their own private data to mine sequential patterns. In our preliminary tests by deploying MSM on 3 user devices, it mines frequent sequential patterns within 8 minutes over 7-53 days of longitudinal user context data including location, app usage and call logs spanning 137-312 unique contexts. We conclude the paper by enumerating future research challenges for mobile sequence mining.
Introduction

Motivation. Modern mobile devices are equipped to capture and infer rich contextual data [3, 8, 11]. By using sensing infrastructures such as funf [3] and Sensor Manager [8], raw contextual data about the mobile user such as location, app usage, call, SMS, etc. can be easily logged. While instantaneous information such as current GPS co-ordinates and heart-rate can be useful information for a user’s immediate needs; to transform the mobile device into a smart personal assistant that understands the user’s context-specific behavior, longitudinal patterns learnt over the collected context data must be leveraged.

![Figure 1: Example user behavioral sequential patterns.](image)

Figure 1 depicts a few potential sequential patterns of humans. A mobile device that can understand its user’s context-specific needs is predicted to be the next big thing for mobile technology [6]. For example, if the phone can learn that the user stops daily for coffee on his way to work in the morning, it can display coffee discount coupons as the user heads out of home rather than at arbitrary times of the day. Similarly, learned knowledge such as regular shopping habits, outdoor activities, and movie preferences can be leveraged to provide context-specific intelligent personal assistance including preloading relevant apps, providing directions and recommendations.

The state-of-the-art. Existing cloud-based services [6] (e.g., GoogleNow) leverage instantaneous knowledge including current location, recent browsing history and calendar to present relevant weather information, directions, recommendations and appointment reminders. While these services simplify user’s life by preemptively populating the mobile device with potentially useful information at times also using the browsing history on other personal devices (e.g., laptop); they in turn compromise user privacy. All data is uploaded to and processed in the cloud. Apart from the noticeable privacy concerns, such cloud-based services are fundamentally dependent on access to internet as uploading raw logs may consume significant portions of mobile user’s data plan (often limited to 1 or 2 GB per month) and may also suffer from high turnaround times. Further such services still lack support for learning and leveraging longitudinal patterns over combination of context information.

While data mining techniques, such as frequent itemset and sequence mining, are well established as powerful
pattern discovery tools in domains such as science and retail [2, 1, 10]; these techniques have been considered impracticable for mining mobile context logs due to limited processing power of mobile devices. Research efforts towards learning frequent mobile user behavior have been primarily limited to offloading computation to the cloud as done in [9]. These efforts are interesting first steps towards learning mobile usage behavior, and our work is similar in flavor.

Potential of on-device mining. Computational power of mobile devices has evolved to modern quad-core Snapdragon and Exynos processors [11]. In our recent work [12], we explored on-device mining of mobile co-occurrence patterns. It demonstrated the feasibility of on-device mining as well as applicability of co-occurrence pattern mining via interesting use cases such as context-aware next-app-to-use and next-contact-to-call predictions. Feasibility of on-device mining is attributed to both the enhanced computational power of devices [11] and the optimized WeMiT mining algorithm [12]. However, since co-occurrence patterns do not capture the ordering among the context items, we explore the mining of sequential patterns in this work. Certain interesting patterns such as listening to music while jogging or using GMaps while driving are captured by co-occurrence mining. However, certain other interesting ordered patterns that cannot be captured by co-occurrence mining can be mined by sequence mining, such as receiving a phone call from Alice followed by exchanging messages with Bob or going shopping followed by a movie and then dinner at a restaurant on weekend evenings (Figure 1). Some applications that can leverage our proposed MSM service are activity recognition [5], and mobile advertising [7]. For example, user’s learnt patterns can be leveraged for preloading content on phone such as apps, news and contacts or for predicting next store visit for effective advertisement.

Contributions. In this paper, we present two main contributions.

Our first key contribution is the design and implementation of the Mobile Sequence Miner (MSM) service that runs entirely on the phone and mines sequential patterns over mobile user’s contextual data. MSM currently works on data from three streams, namely, location, app usage, and call logs. However, MSM is extensible and other streams can be added as well. The mined sequences can be leveraged in future work for context-based recommendations and predictions.

Our second contribution is the evaluation of the MSM engine using phone context logs of 3 users. For our evaluation we deployed the data collection service on 106 real users cellphones as described in [12]. For this paper we pick a sample of 3 such users out of those 106 users for our preliminary experimentation for sequence mining. We collected 7-53 days of data for each of these 3 users for experimentation. In this paper, we explore unimodal sequences, i.e., sequences mined from data from a single stream (e.g., location logs). We conclude the paper by enumerating several open questions with sequence mining, such as mining patterns over multimodal sequences, mobile performance improvements for sequence mining, sequence-based context prediction, and automatic parameter selection for sequence mining.
Figure 2: Mobile sequence miner architecture.
Mobile Sequence Miner Design

Overview

Figure 2 depicts the overall architecture of the Mobile Sequence Mining (MSM) engine together with a running example using app usage logs. The entire MSM service mining sequential patterns from the user context logs was developed in Android [4]. User behavior logs are collected using an infrastructure based on funf [3]. The Intervaled Event Extractor and Sequence DB Generator modules take the context data logs (such as location, app usage and call) as input and generate the basic sequences. Given a set of sequences collected over a period (say, week or month), the Frequent Sequence Miner mines the sequences that qualify a given minimum support. The mined frequent sequences can then be leveraged for predictions and recommendations. The mining is scheduled as a background task at idle times when the phone is on charging and not being used. Thus, avoiding interference with user experience. The Sequence-based Context Prediction and User Profiling component is to be added as future work for utilizing the mined frequent sequences for context-aware predictions and recommendations.

Mobile Sequence Mining Approach

Below we present a detailed description of the MSM engine and describe the various steps, namely, (a.) preprocessing the log data to create intervaled context items; (b.) generating the sequence database of intervaled context items; and (c.) generating frequent sequences.

Extracting intervaled events. As a pre-processing step, we first generate intervaled events from the respective logs. A typical log such as call consists of the fields Contact.Address and Type together with the timestamp. Using the timestamp value we generate time features such as HourOfDay, SegmentOfDay, DayOfWeek and DayType. Each log undergoes specific preprocessing to generate relevant features. For example, the app usage log initially consists of the name of the foreground app sampled at a rate (say, every 1 minute). Further, the consecutive readings that indicate no change in the foreground app are compressed together to achieve intervaled app usage events as depicted in Figure 2 (I). Similarly, the location log consists of the latitude and longitude values sampled at a rate (say, every 30 seconds). We utilized a mapping of latitude and longitude values to the respective place ids as specified in place mappings information. For example, ⟨latitude_work, longitude_work⟩ maps to a certain user’s Work. The place mappings can either be achieved explicitly by the user tagging his/her locations as Home, Work, Mall, etc. or via automated mining. Determination of place ids is outside the scope of this paper. Other logs such as call and SMS undergo similar preprocessing.

Let I = {i_1, i_2, ..., i_n} be a set of all items. For the mobile user context data Location:PlaceID:Home, Call:Contact:X, and App:AppName:GMaps are some example context items. An itemset is a subset of items denoted as I. For example {Call:Contact:X, time:DayType:Weekday, time:SegmentOfDay:Evening} is an itemset. A sequence S is denoted by \{I_1 \rightarrow I_2 \rightarrow \ldots I_l\}, where I_j is an itemset. An item i can occur at most once in an itemset of a sequence, but can occur multiple times in different itemsets of a sequence. The number of itemsets within a sequence defines the length of the sequence such that a sequence with l itemsets is called a l-sequence.
Generating the sequence database. A sequence database \( S_{DB} \) is a set of tuple \( \langle S_{id}, S \rangle \) where \( S_{id} \) is the sequence id and \( S \) is a sequence. For the purpose of generating the sequence database \( S_{DB} \) over mobile context data we use a sequence length parameter that denotes the length of sequence \( S \) in each tuple. We iterate over the intervalled events together with their time features, each time sliding by 1 event to form the mobile context sequence database. An example of generating sequence database with sequence length of 3 and sliding by 1 event at a time is depicted in Figure 2 (II). Sequence length plays an important role in determining the execution time of the sequence mining algorithm as longer the sequences in the database, higher the execution times for processing them. A tuple \( \langle S_{id}, S \rangle \) is said to contain a sequence \( S_i \), if \( S_i \) is a subsequence of \( S \). Further, the support of a sequence \( S_i \) is the number of tuples in the sequence database \( S_{DB} \) containing \( S_i \).

Frequent Sequence Mining. Given a sequence database \( S_{DB} \) and the \( \text{min.support} \) threshold, sequential pattern mining is to find the complete set of sequential patterns in the database whose support qualifies the \( \text{min.support} \) threshold. We employ a well-known sequence mining algorithm called PrefixSpan [10] for mining sequential patterns over the mobile context sequence database that is generated from the context logs. An example of generating frequent sequences is depicted in Figure 2 (III).

Selecting Meaningful Sequences. The sequence mining algorithm generates frequent sequences considering each context item individually with no understanding of the semantic meaning or relationships between the items. Thus, there may be several uninteresting mined sequences such as \( \text{Call:Contact:X} \rightarrow \text{time:SegmentOfDay:Evening} \), where item \( \text{Call:Contact:X} \) may be part of an itemset \( \{ \text{Call:Contact:X}, \text{time:DayType:Weekday}, \text{time:SegmentOfDay:Afternoon} \} \) and item \( \text{time:SegmentOfDay:Evening} \) may be part of an itemset \( \{ \text{Call:Contact:Y}, \text{time:DayType:Weekday}, \text{time:SegmentOfDay:Evening} \} \). Thus, we perform an additional filtering step to eliminate meaningless frequent sequences and output only meaningful sequences of the form \( \{ \text{Call:Contact:X}, \text{time:DayType:Weekday}, \text{time:SegmentOfDay:Afternoon} \} \rightarrow \{ \text{Call:Contact:Y}, \text{time:DayType:Weekday}, \text{time:SegmentOfDay:Evening} \} \). In other words, we output only the sequences where each itemset contains the time features and the event (e.g., app used, location placeID, call, etc.).

Evaluation

Context Data Collection
To evaluate our Mobile Sequence Mining (MSM) service, we collected mobile context data from 3 participants. We developed an Android application called EasyTrack [12] using the Funf sensing library [3] to collect different types of anonymized and encrypted context data. We collected 20 days, 53 days and 7 days of data, respectively for the three users. For evaluation purposes, we included the following timestamped context events: (1) inferred place identifiers of home and work, (2) location cluster label for the current location obtained using a location clustering algorithm, (3) Apps used including time and duration (4) call events including call type (incoming, outgoing, missed, voice mail), time, duration, and number. Overall, the total count of distinct context
Figure 3: Execution times for location logs.

Figure 4: Execution times for call logs.

Figure 5: Execution times for app-usage logs.

Figure 6: Sequence count for location logs.

Figure 7: Sequence count for call logs.

Figure 8: Seq. count for app-usage logs.
items collected for each user were as follows: 137 for user 1, 312 for user 2, and 191 for user 3.

**System Performance**

We measured the performance of our MSM system on Android Samsung Galaxy S3 phones. We vary the support threshold used for mining sequences, and evaluate the running time of MSM in generating sequences over three data probes: location, app usage, and call log. For each support threshold, we also evaluate the number of sequences generated to show the impact of lowering support on generating an increasing number of sequences.

Figures 3, 4, and 5 show the results for execution time of MSM over the probes of location, call, and app usage respectively; from these figures, we observe that decreasing the support threshold increases the running time of MSM. We find that user 2, with the most amount of longitudinal context data, requires the highest execution times for running MSM compared to users 1 and 3. However, the running time is still less than 1 hour in the worst case, and on average consumes 5-15 minutes across the other users, probes and support values. Thus, we show here that sequence mining on mobile devices without cloud support is computationally feasible. In Figures 6, 7, and 8, we show the results for the number of sequences generated for each user and each probe type; from these figures, we observe that by reducing the support value, we are able to generate more frequent sequences that might occur fewer times over the longitudinal context data logged on the user's phone.

**Scheduling MSM on phone.** Modern smartphones have powerful quad-core processors [11] and are also

<table>
<thead>
<tr>
<th>User</th>
<th>Log Type</th>
<th>Frequent Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>App</td>
<td>time:frameweekday_night,App:AppUsed:android,mms=&gt;time:frameweekday_night,App:AppUsed:facebook</td>
</tr>
<tr>
<td>U1</td>
<td>App</td>
<td>time:frameweekday_night,App:AppUsed:facebook=&gt;time:frameweekday_night,App:AppUsed:browser</td>
</tr>
<tr>
<td>U1</td>
<td>Call</td>
<td>time:frameweekday_evening,Call:CONTACT:A=&gt;time:frameweekday_evening,Call:CONTACT:Z</td>
</tr>
<tr>
<td>U1</td>
<td>Call</td>
<td>time:frameweekday_evening,Call:CONTACT:X=&gt;time:frameweekday_morning,Call:CONTACT:A=&gt;time:frameweekday_afternoon,Call:CONTACT:Z</td>
</tr>
<tr>
<td>U2</td>
<td>Call</td>
<td>time:frameweekday_morning,Call:CONTACT:P=&gt;time:frameweekday_morning,Call:CONTACT:Q</td>
</tr>
<tr>
<td>U2</td>
<td>Call</td>
<td>time:frameweekday_morning,Call:CONTACT:P=&gt;time:frameweekday_afternoon,Call:CONTACT:R</td>
</tr>
<tr>
<td>U2</td>
<td>Loc</td>
<td>time:frameweekday_morning,Loc:PlaceId:c10=&gt;time:frameweekday_morning,Loc:PlaceId:Work=&gt;time:frameweekday_morning,Loc:PlaceId:c10=&gt;time:frameweekday_morning,Loc:PlaceId:Work</td>
</tr>
<tr>
<td>U3</td>
<td>Loc</td>
<td>time:frameweekday_morning,Loc:PlaceId:c3=&gt;time:frameweekday_morning,Loc:PlaceId:Home=&gt;time:frameweekday_morning,Loc:PlaceId:c3=&gt;time:frameweekday_morning,Loc:PlaceId:Home</td>
</tr>
<tr>
<td>U3</td>
<td>Call</td>
<td>time:frameweekday_night,Call:CONTACT:A=&gt;time:frameweekday_afternoon,Call:CONTACT:B=&gt;time:frameweekday_afternoon,Call:CONTACT:B</td>
</tr>
<tr>
<td>U3</td>
<td>Call</td>
<td>time:frameweekday_afternoon,Call:CONTACT:C=&gt;time:frameweekday_afternoon,Call:CONTACT:C</td>
</tr>
<tr>
<td>U3</td>
<td>Call</td>
<td>time:frameweekday_night,Call:CONTACT:F=&gt;time:frameweekday_night,Call:CONTACT:G</td>
</tr>
<tr>
<td>U3</td>
<td>Call</td>
<td>time:frameweekday_morning,Call:CONTACT:H=&gt;time:frameweekday_morning,Call:CONTACT:I</td>
</tr>
</tbody>
</table>

Table 1: Example Frequent Sequences.
typically unused for a majority of time such as at night when the user is sleeping and the phone is charging. MSM may be compute-intensive and may be executed on the phone periodically during this idle time with little or no impact on the end user. We define the phone to be idle whenever it is charging, there are no foreground applications, and the battery level is at least 80%. In [12] we show that users have between 1-10 hours of idle time each day, which may be aggregated across multiple days for mining algorithms that do not need to refresh their patterns every day. Thus it may be sufficient to schedule sequence mining algorithms need to be executed once a week at the maximum.

Generated Sequential Patterns
Table 1 shows some interesting patterns mined for the 3 users from their respective logs namely, location, call and app usage. For example, user 1 frequently uses android messaging app followed by facebook on weekday nights. User 1 also frequently calls contact Y followed by contact Z on weekday evenings. User 2 also shows similar frequent calling sequences such as calling contact P on weekday mornings followed by calling contact Q on weekday mornings or calling contact R on weekday afternoons. For User 2 we also detect an interesting sequential pattern of locations visited between place c10 and work on weekday mornings. c10 is likely a cafetaria. User 3 frequently moves between place c3 (possibly a nearby grocery store) and his/her home. Several interesting sequential patterns over user 3's call logs are also detected involving upto 3 contacts being called in a sequence.

Future Challenges
In this paper, we present a first prototype of MSM. There are several open questions and challenges that need to be addressed in order to realize our vision of using sequential patterns to improve phone user experience:

Multi-Modal Context Sequences. In this paper, we have analyzed and mined for frequent sequences involving time and only one context type such as location, call, or app context. We are further interested in exploring sequences across multiple contexts. However, two key challenges that need to be addressed are the increased performance penalties of mining multi-modal sequences, and the appropriate segmentation that needs to be used to create discrete itemsets when considering multiple contexts simultaneously.

Sequence-based Context Prediction. In this paper, we have mined the frequent sequences. However, in the future, we are interested in predicting future user contexts based on exact and fuzzy matching with the user's frequent sequences. A key challenge is to weigh multiple matching contexts based on the exactness and the frequency of occurrence of matched sequences.

Automatic Parameter Selection. Frequent sequence mining involves specifying several parameters that impact performance and the granularity of the mined sequences, namely support, the maximum sequence length, and the maximum gap among consecutive sequence itemsets. In the future, we need informed approaches that automatically select the best parameters or explore multiple useful parameters based on application needs.

Mobile Performance improvements. Finally, there is also opportunity for leveraging the specific properties of mobile data to improve the running time of frequent
sequence mining [12]. Improving the running time will enable mining sequences across multiple contexts and mining parameters.

Conclusion
In this work, we presented our Mobile Sequence Mining (MSM) engine that automatically learns phone usage sequential patterns over the rich context data captured within the device. MSM provides device intelligence by leveraging mined longitudinal patterns while preserving privacy via on-device mining. In our preliminary tests by deploying MSM on 3 user devices, MSM mined frequent sequential patterns in 8 minutes over 7-53 days of longitudinal user context data spanning 137-312 unique contexts. We concluded the paper by enumerating future research challenges for mobile sequence mining.

References