Ubiquitous and Mobile Computing CS 528
MobileMiner: Mining Your Frequent Patterns on Your Phone

Xiaoyan Sun
Mei Yang

Computer Science Dept.
Worcester Polytechnic Institute (WPI)
Motivation:

- Long term Goal:
  - Develop novel middleware and algorithms
  - Mine user behavior patterns entirely on the phone
  - Utilize idle processor cycles

- Accomplished:
  - A novel general-purpose service: MobileMiner
    - http://kingsbsd.github.io/MobileMiner/
  - Discover frequent co-occurrence patterns on the phone
  - The patterns can be used by developers to improve UI
Overview of MobileMiner

- Currently it logs:
  - IDs of the GSM cells you visit.
  - Mobile networks that provide mobile data.
  - Names and BSSIDs of wifi hot-spots.
  - Processes that open network sockets.
  - Socket IP addresses and ports.
  - When 'net-enabled apps send notificationss.

- Export data:
  - Download directory of the device's flash memory, or SD card if it has one
Introduction:

Why Mine Co-occurrence Patterns:

- Preload news ahead
- Intuitive UI
- Send a smart reminder to charge the phone

Main Idea:

- Log raw contextual data
- Mining algorithms can run during idle time (sleeping)
- Charging
- Preload news ahead
- At least 80% battery
- Intuitive UI
- Better privacy guarantees
- Send a smart reminder to charge the phone
- Higher level user context to behavior patterns

Why Mine Patterns on the Mobile Device:

- Mining algorithms can run during idle time (sleeping)
- Charging
- At least 80% battery
- Better privacy guarantees
- Benefits to users with lower-end phones
Contributions:

- **System Design:**
  - Use limited phone resources
  - Provide a prediction engine

- **Performance:**
  - Feasibility of experimental running by using context logs
    - 106 users over 1-3 months
    - Faster than widely used Apriori algorithm
    - Low consumes (0.01-3% battery)

- **Pattern mining:**
  - Analyze patterns individually as well as groups

- **UI improvement:**
  - Predict next outgoing call or app, and provide shortcut icons for them
System Design:

- Data collection:
  - User activity logs (few permissions )

- Base pattern miner:
  - Base Basket extractor
  - Base rule miner

- App usage miner:
  - Filter use threshold
  - App rule miner
  - App pattern retriever: prediction

- Minner scheduler
Basket Extraction and Filtering

**Basket Extraction**
- Continuous context to a set of small discrete values.
  - Eg: locations, battery
- Timestamped context baskets based on temporal overlap
- Compress duplicate baskets

**Basket Filtering**
- Boolean expression
- Utility functions
- Benefits:
  - More accurate
  - Faster
  - Free of noise
Weighted Rule Mining

- **Input**: weighted context baskets
- **Output**:
  - Association rules: \( A \rightarrow B \) \{AtHome, UsingWiFi, 10 – 11pm\} \rightarrow \{ChargingPhone\}
  - **Threshold**:
    - Support \( P(A,B) \)
      - Long duration activities: high
      - Short duration activities: low
    - Confidence \( P(B|A) \)
- **Generate frequent itemsets**:
  - Occur many times as support threshold
  - Confidence exceeds confidence threshold
  - Help predict

**Challenge**:
- Lower support value
- Increasing running time
Apriori VS WeMiT

- **Optimized Apriori**
  - Bottom up
    - If $F_n$ is frequent, generate $F_{n+1}$
    - Downward closer of support
    - Onepass through
  - Optimized
    - If subset of $F_{n+1}$ of size $n$ is not frequent, pruned $F_{n+1}$
    - Avoid single pass through

- **WeMiT**
  (Weighted Mining of Temporal Patterns)
  - Compressed weighted baskets
    - 92.5% decrease compare to uncompressed
    - Modified definition of support
      \[ B^* = \{ b_1^{w_1}, b_2^{w_2}, \ldots, b_n^{w_n} \} \sum_{i=1}^{n} \text{contain}(b_i^{w_i}, X).w_i \]
    - Running time reduced 15 times on average allow several passed through

http://magpiehall.com/apriori-algorithm/

---

Figure 5. Distribution of basket frequencies for a sample user.
Context Prediction

- Search association rules generated by MobileMiner
  - Input: co-occurrence patterns
    - Current context: {Morning, Atwork}
    - Target context: {UseGmail}, {UseOutlook}
  - Prediction based on confidence with decrease order
    - Return max

- Example:
  - \{Morning\} → \{UseGmail\} with confidence 0.9 but \{UseOutlook\} with 0.8
    - Return Gmail
  - \{AtWork, Morning\} → \{UseOutlook\} with confidence 0.9
    - Return both Gmail and Outlook

Figure 6. Context prediction using co-occurrence patterns.
Evaluation—Context Data Collection

- 106 participants
  - At least 40 Users collected more than 40 days
  - Around 25 Users collected 21-40 days
Evaluation—Context Data Collection

- 440 unique context events
  - Call events
    - Type
    - Duration
    - Number
  - SMS events
    - Type
    - Number
  - Inferred place identifiers
    - Home
    - Work
    - Outside
  - Location cluster label
  - Phone charging status
  - Battery levels
  - Foreground app usage events
  - Wi-Fi or cell connective
  - Cell id of current location
  - Binary movement status
Evaluation—System Performance

- Is it feasible to run MobileMiner components on the phone?

<table>
<thead>
<tr>
<th>Performance Metric</th>
<th>Base Basket Extraction</th>
<th>Base Rule Mining</th>
<th>App Usage Filtering</th>
<th>App Usage Rule Mining</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution time</td>
<td>1.7 seconds</td>
<td>16.5 minutes</td>
<td>1.4 seconds</td>
<td>21.2 seconds</td>
</tr>
<tr>
<td>Memory</td>
<td>9.9 MB</td>
<td>44.2 MB</td>
<td>11.6 MB</td>
<td>1.0 MB</td>
</tr>
<tr>
<td>CPU Utilization</td>
<td>22.9 %</td>
<td>24.3 %</td>
<td>20.8 %</td>
<td>21.9 %</td>
</tr>
<tr>
<td>Number of baskets or rules</td>
<td>114275 baskets 8559 compressed</td>
<td>46675 rules</td>
<td>752 baskets 327 compressed</td>
<td>1062 rules</td>
</tr>
<tr>
<td>Energy per day as % of full battery</td>
<td>&lt;0.01 %</td>
<td>0.45 %</td>
<td>&lt;0.01 %</td>
<td>0.01 %</td>
</tr>
</tbody>
</table>
Evaluation—System Performance

- How does WeMiT compare to Apriori
Evaluation——Patterns Generated

- What are some sample patterns and how do they use them?
  - Analyze the patterns generated by MobileMiner.
  - Get the confidence of each rule in the matrix visualization.
Evaluation—Patterns Generated

- What are some common patterns across multiple users?
  - Rather than the confidence, they show the percentage of users the pattern occurs in, either among all users (left) or among smaller group with very similar co-occurrence patterns (right)
Example Use Cases—App and Call Prediction

- App recommendation service with short cut icons
  - For each app, they show the reason why is was displayed based on the matching co-occurrence pattern

- Two evaluation metrics:
  - **Recall** is the proportion of app launches or outgoing calls for which **they show recommendations to the users.**
  - **Precision** is the proportion of **times the user uses one of shortcut icons** to complete his task.
Example Use Cases—App and Call Prediction

- What is the Recall-precision tradeoff of predictions?

- Recall-precision tradeoff for 1, 3, 5 and 7 shortcut shown.
- Typically, higher recall results in a lower precision, vice versa.
Example Use Cases—App and Call Prediction

- How do they choose the support value for mining patterns?

- 4-5% improvement in precision as the support values decrease from 20 to 5.

- Developer should choose a appropriate support threshold to achieve reasonable prediction accuracy without incurring too much phone resources.
User survey

- How often will users use our app recommendation service?
  - Use the service regularly: 57%
  - Use sometimes: 42%

- Where should the shortcut icons be placed?
  - Phone’s lock screen: 40%
  - Phone’s quick panel: 26%
  - Main tool bar: 33%

- How many shortcut icons should be displayed?
  - More than 6 icons: Very few
  - 4-6 icons: 71%
  - 1-3 icons: 26%
User survey

- Will users prefer a recall less than 100% for improved precision?
  - Higher precision: 54%
  - Always receive recommendations: 9%
  - Either case: 35%

- What is recall-precision tradeoff preferred by users?

<table>
<thead>
<tr>
<th>Precision</th>
<th>No. of recommendations</th>
<th>Recall</th>
<th>Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>90%</td>
<td>3</td>
<td>35%</td>
<td>30.95%</td>
</tr>
<tr>
<td>80%</td>
<td>3</td>
<td>51%</td>
<td>16.67%</td>
</tr>
<tr>
<td>80%</td>
<td>5</td>
<td>68%</td>
<td>23.81%</td>
</tr>
<tr>
<td>80%</td>
<td>7</td>
<td>80%</td>
<td>11.90%</td>
</tr>
<tr>
<td>75%</td>
<td>3</td>
<td>66%</td>
<td>4.76%</td>
</tr>
<tr>
<td>75%</td>
<td>5</td>
<td>87%</td>
<td>11.9%</td>
</tr>
<tr>
<td>75%</td>
<td>7</td>
<td>100%</td>
<td>19.05%</td>
</tr>
</tbody>
</table>
Related Work

- Focus on on-device mining of co-occurrence patterns over users’ mobile context data
  - Compare with other context-aware computation on mobile devices using longitude context data
    - Deal with privacy, data cost, and latency concerns
  - Compare with approaches use specialized predictive classifiers
    - More generalizable
    - Provides more configurability
    - Make predictions with lower accuracy even with missing context events
    - Co-occurrence patterns are more readable and directly usable by end users

- A preliminary version of the work has been presented
Conclusions

- The novel MobileMiner system efficiently generates patterns using limited phone resources
  - 15 times performance improvement over Apriori
  - Generate overall frequent patterns in 16 minutes
  - Detail app usage pattern in 21 seconds
- Found interesting behavior patterns
- Improve the phone UI for launching app or calling contacts
- Future work
  - Explore co-occurrence patterns of events over long time durations
  - Systematically determine the correct frequency of running the mining algorithm
  - Perform a comparison of the context prediction approach
  - Extend to other types of patterns
References


References


References


