Quick Question:
Interrupting Users for Microtasks with Reinforcement

Bo-Jhang Ho
Bharathan Balaji
Mehmet Koseoglu
Sandeep Sandha
Siyou Pei
Mani Srivastava
Quick Question: Interrupting Users for Microtasks with Reinforcement

Introduction

- Human attention is a scarce resource in modern computing. Many **microtasks** vie for user attention (e.g. check mail, reminders, nudge behavior, get feedback, label datasets).
- **User interruptibility** has been extensively but mostly addressed using **Supervised Learning (SL)**.
- We model user interruptibility using **Reinforcement Learning (RL)**.
- RL can improve on existing solutions with:
  - Sequential decision making
  - Exploration
  - Online learning
- User study with push notification microtasks

Methods - System Overview

**Agent**

1. **State** (sensor data)
2. **Action** (notify, stay silent)
3. **Reward** (accept, dismiss)

**Environment**

- Smartphone user

**Smartphone sensors as user context**
- Time of day, day of week
- Location
- Motion
- Screen on/off, Ringtone silent/normal
- Time since last notification

**Client-server architecture**

- **User Interface**
- **Microtask Pool**
- **Sensing Module**

**Server**

- **HTTP Server**
- **Database**
- **RL agent**
- **SL agent**

**Client**

Request:
- Task response
- User state

Response:
- Show microtask or silent
Quick Question: Interrupting Users for Microtasks with Reinforcement

<table>
<thead>
<tr>
<th>Reinforcement learning algorithm</th>
<th>Supervised learning algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong># answered notifications</strong></td>
<td><strong># answered notifications</strong></td>
</tr>
<tr>
<td><strong># dismissed notification</strong></td>
<td><strong># dismissed notification</strong></td>
</tr>
<tr>
<td><strong>Answer rate</strong></td>
<td><strong>Answer rate</strong></td>
</tr>
<tr>
<td><strong>Dismiss rate</strong></td>
<td><strong>Dismiss rate</strong></td>
</tr>
<tr>
<td>Week 1: 214.9</td>
<td>Week 1: 88.6</td>
</tr>
<tr>
<td>Week 2: 182.2</td>
<td>Week 2: 108.3</td>
</tr>
<tr>
<td>Week 3: 138.6</td>
<td>Week 3: 128.6</td>
</tr>
<tr>
<td>Avg (SL-train): 153.6</td>
<td>Avg (SL-test): 164.6</td>
</tr>
</tbody>
</table>

**User Study and Agent Setup**

- 5 weeks. 15 participants in SL and RL each.
- **State**: Sensor measurements from phone
- **Action**: Send or do not send notification
- Take decision every 15 minutes
- **SL agent**: Random forest classifier. Randomly send notifications for 3 weeks, use collected data to train model. Use SL model for last 2 weeks. One model per user.
- **RL agent**: Advantage Actor Critic (a2c). Re-train every week. Separate policy for each user. RL agent used the following reward:

  \[
  reward = \begin{cases} 
  1 \times t^{0.9}, & \text{if answered,} \\
  -0.1, & \text{if ignored,} \\
  -5, & \text{if dismissed} 
  \end{cases}
  \]

**Evaluation Summary**

- RL gets more microtasks answered, but SL achieves better answer rate.
- RL can suppress dismissed notifications.
- User experience in RL is initially worse, but improves over time.
- RL can better identify available moments.
- A2C converges in a week; it can adapt to user preference change and capture the weekly pattern.