Enabling Global Access to Crowdworking

Abstract: We present a system for completing tasks on Mechanical Turk, a web-based crowdsourcing platform, in an Android application. This addresses the gap between earned hourly wage, which does not take into account time spent navigating poorly-designed interfaces or searching for completable tasks for which the worker is qualified, and the effective hourly wage: what a worker actually earns per hour. Our proof-of-concept system demonstrates that certain types of tasks are ideally completed on mobile, while others are merely possible or entirely infeasible. We also draw conclusions about the scalability of our system and its relationship to API or internal access to crowdsourcing platforms like Mechanical Turk and Crowdflower.

Why Android? Workers who depend upon crowdsourcing as a primary source of income are predominantly located in developing regions, where access to smartphones is more common than access to computers. Though the wage for completing Human Intelligence Tasks (HITs) can be as low as one penny, in aggregate, this is sufficient to support individuals, as some workers can earn up to $1,000 each month. In targeted areas, the Android OS holds 79% of market share.

Results

• Proved that a mobile crowdworking platform is viable
• Collaboration with crowdworking company must exist for the mobile app to be successful (Need a worker-side API for accepting and submitting HITs)
• Categorized HITs for mobile: optimized, possible, and infeasible

Optimized

25%

• Mobile data collection and verification of in-person information (temperature, wind, lines at a food truck, photos)
• Audio recording and multimedia tasks

Possible

40%

• Receipt transcription (one of the most common tasks on Mechanical Turk)
• Internet research (e.g., looking up a phone number)

Infeasible

35%

• Extensive, text-heavy survey completion
• Computationally intense tasks (running a program locally)

Implementation

We built a Selenium web-crawler in a headless browser to collect HIT information from Amazon’s Mechanical Turk and store it in a Parse Database. The front-end Android application pulls in this information and dynamically displays it to the worker. On accepting the HIT on the app, another Selenium script is triggered in the Python Flask Server on an AWS EC2 instance allowing us to interact with the site without being tagged as a robot. This allows MobileTurk to display the wide array of jobs on Mechanical Turk in a more accessible way, providing a better, simpler interface for workers.

Next steps

• Ask users to rate their mobile experience after each HIT
• Sort HITs in order of mobile-friendliness
• Continue adding support for more HIT types
• Test HIT completion speed more extensively and with wider audience