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.

Introduction & Problem Statement

Crowdworking is the process of using groups of people to perform tasks or gather data through tasks called “Human Intelligence Tasks”, or HITs. These tasks are typically tasks that cannot yet be completed by computers, surveys to gather information about something, etc. Amazon’s Mechanical Turk is one such platform for crowdworking; “requesters” put jobs on the website and “workers” complete the task for a small amount of money (typically $0.01 to $0.10).

No mobile platforms for crowdworking exist, so as a group interested in crowdsourcing and its accessibility worldwide, we wanted to create one. More specifically, we were motivated to create a mobile crowdworking platform for Amazon’s Mechanical Turk so that crowdworkers in developing countries can more efficiently complete HITs. A large portion of crowdworkers resides in countries (India, Malaysia, Thailand, etc.) with limited access to desktop computers, but extended access to mobile phones. Therefore in areas with limited computer access, it is much easier for workers to perform tasks on their phones. Unfortunately, crowdsourcing websites render poorly on mobile phones and make it very difficult for users to complete HITs in a way that can earn them a useful wage. The absence of a useful mobile platform motivated us to create one, so that crowdworkers in developing countries can benefit more from crowdworking platforms.

Also, a mobile platform will benefit workers in developed countries such as the United States. Idle periods of time like waiting for the bus, waiting in line, or sitting during long car rides offer workers a great chance to perform HITs and make money. Access to a mobile crowdworking platform will allow workers to utilize these periods of time much more effectively than a mobile browser.

Approach

Knowing that the end result of our project was the Android application, we split into groups and worked on the various supporting technologies that our app would need. This began with the database. We decided to work with a Parse database as we were most familiar with it, it provided a clean UI that would allow us to easily add test data, it didn’t require us to host it and
it was a free service. We then set out to populate the database with the necessary information. As Amazon has withheld a public worker-side api for Mechanical Turk, we were forced to develop our own scripts to scrape MTurk ourselves. We started this using python and Beautiful Soup. This allowed us to get the title, requester, payout, and groupid for each of the available HITs and the Parse api allowed us to add that directly to the database as it was scraped. We had to add in delays on the page requests to avoid Amazon flagging our address as we scraped each HIT for this information.

Once we had this preliminary information, we began work on the Android application itself. We developed a login page and added the users to the Parse database. We then developed a listview to display the basic information about the available HITs and a preview page where the user could get more in-depth information. It was at this point that we moved to allowing the user to accept a given hit on the platform. This was the most challenging aspect of the project because MTurk requires a ‘signature’ that is generated at the time of the user clicking the accept button in order to actually accept a hit. We first attempted to generate this signature ourselves, but found that we needed additional information about the user’s amazon account. We chose to abandon this route as it would require much more work on the user side than we believed any new user would be willing to go through. We instead attempted to scrape this data in a similar way to scraping the original HIT information. This too proved futile as the signature is obviously a dynamically generated aspect of the site and thus was not accessible through Beautiful Soup. After trying a handful of scrapers, we landed on Selenium as our solution. Selenium is meant for testing of mobile apps and sites, but can be used to scrape web data as well. We set to work trying to incorporate Selenium on the app itself, but after several attempts we realized that this aspect of the process should be handled on the server.

So we set out to establish a server for our project. At first we just wanted to be able to trigger the accepting script, so we tried out Amazon’s Lambda product. We hooked the Lambda instance up to the Amazon SNS platform and attempted to post from the app to the SNS topic to trigger the script. We found that the Lambda scripting platform was not truly capable of handling the dependencies that Selenium required so we had to find another solution. We decided to spin up an EC2 instance and run our own server there. We went with a Python Flask server and developed a route to accept a hit. We had to run a headless scraper using selenium on the instance, but we able to successfully interact with the MTurk site to accept a hit without raising Amazon’s anti-robot precautions. On accepting the hit, the script would then grab the HTML of the HIT itself and return that back to the user on the app. This was our next big challenge.

Since there is no standard HIT format of any type, each hit was very unique in style and HTML coding. This meant we had to dynamically parse through the iframe of the hit for the necessary information. We were able to accomplish this by looking at the html hierarchy and converting tags like input, text and buttons into xml that the app understood and could display as android versions of these web elements. We associated the various ids of the original elements with ids for the android elements and on submitting had another Selenium script send the necessary keys to the various inputs and submit the HIT.

With all of these pieces in place, a user of the application could see all of the HITs available to them on the Mechanical Turk platform, could view and accept a HIT to their account, complete it mobily and submit it back to the system. We believe we are the first to open
up crowdworking to the mobile community, and while this is merely a proof of concept currently, we hope to see a great development and utilization of mobile users in the near future.

**Results & Measurements**

A significant amount of learning resulted from the efforts of the MobileTurk journey this past academic year. We highlight three main topics of results in this section: viability of a mobile crowdworking mobile platform, need for collaboration with a larger company for scalable version of MobileTurk, and categorization of Human Intelligence Task.

Firstly, we proved that a mobile platform where crowdworkers connect, accept, complete, and submit HITs is a very viable, and exciting area of crowdworking development. The absence of such a platform was suggestive of the challenge in independently developing one alongside the lack of investment from the primary crowdworking platforms of Mechanical Turk and Crowdflower.

Many of our technical challenges, detailed in discussion, arose from the lack of an internally-supported API accompanying any of these platforms. In a way, we created not only an application, but an entire system of data extraction which could be decoupled from Mechanical Turk and exchanged to Crowdflower, or from our Android front-end to another system (iOS, for instance). Although the system sometimes became needlessly complex to accommodate Mechanical Turk’s policies around robotic crawling, overall the results are quite modular and transferable.

Finally, in analyzing the types of tasks that we wanted to enable and optimize, we gained a nuanced understanding of the types of tasks that are available. Mechanical Turk is well-known for its tedious receipt transcription tasks, and similar image captioning tasks, which represented around 40% of tasks available for beginner workers without Master qualification or any similar credentials. We classified these as possible, though they do not fully utilize the capabilities of a mobile platform.

We identified that surveys and app review tasks represent approximately 35% of the HITs on Mechanical Turk at the time we surveyed. We classified these as infeasible, as their extensive text entry and local computational resource usage would tax the user’s patience with typing on a mobile phone or data networks and download caps in some regions that we targeted.

Then there are many small-batch tasks that vary wildly in purpose, from selecting the most appealing photos of clothes, horses, or kittens, to assessing the quality of a translation from Arabic to English. These tasks vary from day-to-day as they correspond to academic or business researchers who post infrequently, and represent perhaps 10-15% of the tasks available at any one time.

Certain tasks, like audio recording for the purposes of NLP analysis, are ideal for mobile use - the phone equipment facilitates recording more seamlessly than a desktop. Our original estimations put this share of HITs around 25%, though after further exploration, we believe it may be much smaller for unqualified workers; our final analysis of over a thousand HITs placed it at 5%.
Moving forward, we outlined a number of exciting potential extensions of MobileTurk, areas for new crowdsourcing that isn’t currently supported or requested within modern platforms. The applications for disaster recovery, of using mobile location tracking to respond quickly to emergencies, is first and foremost in our minds. Creating template tasks for law enforcement or on-the-ground volunteer organizations would take advantage of a pre-existing workforce (and truthfully, these properties could be built into an operating system in the same manner that Amber Alerts sometimes are). We believe that the existence of a mobile platform like MobileTurk - or a fully-supported system separate from Mechanical Turk - could fundamentally modify the types of human intelligence tasks that requesters seek, dramatically changing the crowdworking economy to an industry with more emphasis on in-person data collection and the skills that humans are uniquely suited for, rather than mere image recognition.

**Ethical & Privacy Considerations**

There are certainly challenging ethical questions around perpetuating a system that rewards workers pennies for their time and effort. The hope for this project, however, was to improve conditions within an already enormous economy, rather than creating a superior system that doesn’t disrupt Amazon’s ecosystem. Unlike some industries that exploit workers in developing nations, Amazon’s workforce is distributed across the globe, including the United States. Workers enter willingly into the working arrangement, and we sought to facilitate increased wages for those workers.

There are certain tasks on Mechanical Turk that consist of finding the contact information (phone number, email address, mailing address) of individuals associated with particular educational institutions or other organizations. This raised some questions about privacy as we tested tasks. It is straightforward to block these tasks within the application, and use the mobile interface as a tool with higher standards and respect for privacy than the original platform itself. At this point, however, the responsibility for the contents of any task currently lies with the requester paying for it and Mechanical Turk’s guidelines for acceptable use of the platform.

**Discussion**

The decision to develop in Android was a good one because of the array of libraries and packages available for language parsing and the ease with which we were able to integrate supporting pieces like the database and server. If we were to initiate this project over again, we would need to put more effort into maintaining our databases of users and task metadata, since Parse has been discontinued by Facebook. However, they have open-sourced it, which offers more individual control in exchange for a slightly more arduous setup process. Flask was also a straightforward choice for our server, and Amazon’s EC2 hosting was a good choice for us because they offer free hosting under a particular volume of requests (one million - for our testing purposes, we stayed below this threshold). It was quick to setup and test our ideas before investing too much time and effort into incompatible technologies.

As our project progressed, and we encountered consistent difficulties in obtaining a signature in making the server requests to Amazon, we considered creating our own crowdsourcing platform. We reached out to several graduate student researchers with our advisor to better understand the problem, and one of them had heard about Selenium webdriver
at a conference. Without that recommendation, we would likely have needed to pivot to construct our own system and spend less time on mobile development and more time on creating a good requester-side API.

If we had more time to continue to build up MobileTurk, in creating a scalable platform moving forward, having internal collaboration from Mechanical Turk or Crowdflower would be immensely helpful. Without that, in order to get a critical volume of users and tasks, you would need to build and promote your own system. If we moved forward with this, we’d like to include a machine learning element in determining what constitutes a good HIT for mobile by asking workers to rate their experience after completing a task, then using that data to train a classifier and re-ranking tasks by mobile-friendliness rather than recency of posting and volume of available work. In this way, users could spend less time searching for tasks or completing uncooperative tasks, get straight to the best HITs, and earn a higher effective hourly wage.

Additionally, we believe that the introduction of a mobile crowdworking platform will cause an increase in mobile-friendly or mobile-specific HITs. Taking pictures of certain things, recording the temperature, recording audio of events, and other similar tasks that involve using a phone in a place other than a home will become possible HITs once a mobile platform is introduced. This increases the breadth of possibilities for crowdworking and makes crowdworking an even more useful tool for data collection.

Citations

Various sources for Android OS Market Share in SE Asia.