iSURE Program

Crowdscouring for Urgent Human Computing

Documentation

Ruizhongtai (Charles) Qi[1]

August 2012

Contact: CharlesQ34@gmail.com

Advisor: Zhi Zhai[2]**, David Hachen**[3]**, Tracy Kijewski-Correa**[4]**, Gregory Madey**[2]

Department of Computer Science and Engineering[2]**,**

Department of Sociology[3]**,**

Department of Civil and Environmental Engineering and Earth Sciences[4]**,**

University of Notre Dame

Department of Electronic Engineering[1]**,**

Tsinghua University

Contents

1 Introduction

1.1 Project Overview

"Urgent Computing" means providing prioritized and immediate access to supercomputers and grids for emergency computations [11]. We propose the concept of "Urgent Human Computing" where supercomputers and grids are replaced by on-demand citizen workers to solve problems that humans do better than computers.

On-line platforms like Amazon Mechanical Turk provide a scalable workforce for human computing purposes. These globally located citizen workers, or "citizen engineers" with great diversity in knowledge, working hours and motivations make it possible to harness suitable human computing power under urgent circumstances.

However, a challenging problem for Urgent Human Computing in open communities like Mechanical Turk is how to optimize speed, cost and quality. In this project, we explore both workflow design and algorithm design approaches to achieve reliable answers under time and cost limits.

In a Haiti earthquake photo-tagging task – classifying damage patterns in post-disaster photos, we examine efficiency, economy and quality control techniques by "crowdsourcing" it on Mechanical Turk. We use a hybrid of serial and parallel workflow to improve overall work quality. The results are compared with a previous experiment with college students [12]. Statistical worker grading algorithms and answer quality control algorithms are introduced. The effects of bonus and qualifying test on speed, cost, quality are also discussed. At last, we concluded the findings and insights revealed by the experiments and indicate the future study for Urgent Human Computing.

This documentation involves 3 chapters. The first is this introduction chapter, the second is a brief guide to Amazon Mechanical Turk and the third is about design and implementation of Haiti earthquake building image tagging experiments on Amazon Mechanical Turk. The lessons learnt are included in the third chapter. There are four appendixes of external question design and reports on trial experiments.

1.2 Related Work

Zhai Zhi, one of our prominent project members, used to set up an online platform [7] that recruited undergraduates at Notre Dame to indicate the damage pattern and severity in the Haiti earthquake building images [15] (the same set of images is used in this project). This project is different because this time we are publishing the task to an open community where workers are anonymous and with highly diversified background.

Many other researchers have done experiments on Mechanical Turk but they have their own focus. For example, Quinn studied a system that combine machine learning algorithm and human computation to achieve flexible control of speed-cost-quality [17], Greg Little and Robert Miller explored iterative and parallel workflow [18]. What we do in this project is to ask unknown citizens help in "engineering tasks" that require background knowledge of civil engineering. The building damage pattern evaluation task is quite different from common tasks in Mechanical

Turk, such as content creation, photo filtering, object detection (detect the existence of certain object in a photo) or audio/video/text transcription because most of these HITs require no special knowledge except human cognitive intelligence. For our project, we need to teach workers the concept in civil engineering (definitions of building elements and their damage patterns) and design qualification test to make sure the workers are able to work on our task. We need new design ideas to ensure work quality and high work responses in this complex task.

Summary:
This project explores new research problems and is different from existed work because it outsources "engineering task" that require special engineering knowledge to unknown citizens with diversified background.

2 A Brief Guide to Amazon Mechanical Turk

2.1 Overview

This chapter is a concentrated guide to Mechanical Turk. In 2.2 I will introduce the basic concepts used in Mechanical Turk and give several links to more detailed tutorials. In 2.3 a typical requester workflow (how a requester design, publish HITs and get results) is described. Section 2.4 is about 3 kinds of developer tools for Mechanical Turk and 2.5 includes the instruction on using the project framework.

2.2 Basic Concepts

Mechanical Turk is an on-line marketplace for work. Any Internet users could register as a "worker" or "requester".

Workers look for an interesting task, work on it and earn some money for the work. Their wage is usually lower than employees in real companies. With the advantage of Amazon ecosystem, the payment could directly go to a worker's Amazon account and he/she could shop on-line with the money. No bank account or credit could also be used but is not required. The best way to understand how a Turk worker works is to be a real Turk worker. Go to Mechanical Turk site [1] https://www.mturk.com/mturk/welcome and earn some money!

A requester is an "employer" who has human computation tasks or survey tasks and pay human worker to work on them. The requester would first fund his/her account and then design HITs (human intelligence task, a basic unit of work in Mechanical Turk) for a specific task. When the HITs are ready, the requester could publish the HITs and wait for Turk workers to respond. Note that multiple workers could work on the same HIT, i.e. a single HIT (or you can think it as a question) could have more than one "assignments". The requester can issue a qualification test to teach workers about the task or filter out unqualified workers. The workers have to pass the test to work on the HITs. Mechanical Turk also has some system qualifiers like HIT approval rate, worker location etc. To reward hard-working and high quality work, the requester could grant bonus to specific workers. To have a first impression on how a requester get work done, you can go to the site [1] , register as a requester and "Get Started" by publishing a template HIT.

Requester user interface on the web is enough for simple and typical tasks. But for tasks that need customized design or flexible control we need to use programming interfaces. Fortunately Amazon Mechanical Turk provides API and many developer tools for requesters to programmatically manipulate everything, i.e. publish HIT, publish qualification test, get HIT or test results, approve or reject work, grant bonus and so on. Mechanical Turk provides some official documentation on their developer tools [2] with introduction to Mechanical Turk. Another useful information source is the developer forum that is quite useful for detailed technical problems [3].

Amazon Mechanical Turk also has a sandbox [4] for requesters to test their design and programs. The sandbox is exactly like the real platform except it uses virtual currency.

Concept Summary:
Mechanical Turk, workers, requesters, HIT, qualification, assignment, bonus, requester user interface, developer tools, sandbox.

2.3 Requester Workflow

When a requester has a task for Turk workers, he/she would go through the following steps to release the task and get results.

Step 1: Define Task and HITs

Firstly the requester has to be clear about what he/she wants human workers to do. It could be an independent problem like image tagging or audio transcription. Also it could be a complex task composed of a series of HITs. In the later case, the requester has to decompose the large, complex task to small and simple ones. For example, the task may consist of two types of HIT, one is copying text from receipts in photos and another type of HIT is to verify if the text is the same as that in the photo. In this step the requester need to consider how much he would like to spend on the HITs and how long he would like to wait for the results. The design of tasks affects worker's interest and the quality of the overall work.

Step 2: Design worker's workflow

After one is clear what the task and the HITs would require workers to do, the requester has to design a workflow for workers to involve in the task. The workflow may include a qualification test, an entry survey, several types of HITs and rules on how a worker gets paid. For example, the requester could require the worker gain some qualifications to be able to work on the HITs. The workflow may have an instant paying rule where workers get paid as soon as they finished a HIT or the worker would get paid only after the requester have reviewed their answers. Also there could be a bonus incentive so workers know that serious work would help them earn more. Similar to step 1, the workflow design has great impact on work quality and worker's experience of the job.

Step 3: Choose a developing approach

For different types of task, requesters may have diversified requirements on HIT interface and flexibility of control (programmable manipulations). The requester needs to choose a developer approach that matches the characteristics of the task and programming background of the requester. The choice would affect functions a requester is able to use. For example, if the web interface is the only developing tool, then qualification test is impossible to be published. High-level tool kit (such as Turkits and our crowdsourcing framework for "engineering tasks") could simplify the process and save developing time but the sacrifice is flexibility in HIT control. In 2.3 we would closely discuss the pros And cons of several developing approaches.

Step 4: Implement HITs and qualifications

In this step the requester would design the HITs and qualification tests (if there is any) for the task. The design may be writing HTML files, writing XML files or using templates that only requires some blanks to be filled. It depends on the develop approach the requester choose in

step 3.

Step 5: Review algorithm

Mechanical Turk is only acting as a black box where questions come in and answers go out. There is no guarantee of answer quality. After the requester gets the answers he/she has to review them: filter out noise and get a high quality answer from multiple responses. Technique like majority rule, worker grading, golden standard and weighted voting may all raise the answer quality. Also, the requester has to be clear about the payment mechanism. How much to pay? Is there an extra bonus for a well-performing worker? What kind of answers should be counted as spams?

Step 6: Sandbox test

To ensure the system works as we think and there is no problem with HITs, and qualifications, the requester would better experiment in the sandbox before publishing them to the real platform. Note that the requester could also act as a worker and test the HITs.

Step 7: Release HITs

When the requester is sure everything works fine, he/she can pick up a time to release the HITs. The time HITs are published would affect the worker group working on them. For example, if the HITs are published at midnight in US, it would be daytime in India and Europe and there would be more Indian and Europeans and less US citizens working on the HITs in the first several hours.

Requester workflow summary:
step 1: Define Task and HITs
step 2: Design worker's workflow
step 3: Choose a developing approach
step 4: Implement HITs and qualifications
step 5: Review algorithm
step 6: Sandbox test
step 7: Release HITs

2.4 Developer Tools

Amazon Mechanical Turk provides requester user interface (web interface), APIs, web service description language (WSDL), command line tools and software development kits (SDK) for requesters. In this section, I will first introduce the commonly used RUI, CLT, and SDK and then summarize the typical way to design qualifications and HITs using these tools.

Note that this project uses CLT as the crowdsourcing framework's interface to Mechanical Turk and Python scripts to process data and capsulate CLT commands into higher-level programs. The entry survey and qualification test are defined with XML files and their answers are periodically reviewed, approved or rejected by programs. The HITs are defined by XML files with links to external HTML pages.

(1) Requester User Interface

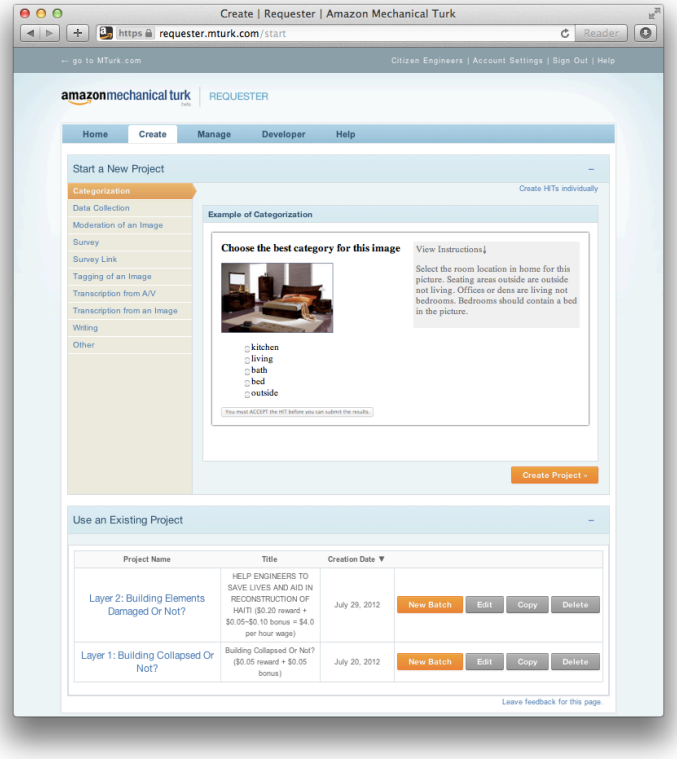


Fig. 1 requester user interface

The requester user interface is a web interface, straightforward and easy to pick up. Mechanical Turk provides several templates for common tasks, such as categorization, data collection, survey, transcription and writing and a blank template where a requester can upload HTML file for the HIT.

For detailed instructions on RUI, see official documentation in [4].

(2) Command Line Tools

The command line tools, as it says in its “Get Started” tutorial are “a set of command line interfaces allowing you to easily build solutions leveraging Amazon Mechanical Turk without writing a line of code. The goals of the Command Line Tools are to: abstract you from the "muck" of using web services, allow you to create solutions without writing a single line of code, allow you to focus more on solving the business problem and less on managing technical details.” It can be downloaded at https://requester.mturk.com/developer. A detailed set-up tutorial called “GetStarted” and instructions of supported functions along with a lot of example tasks are included in the tool folder.

The command line tools are the “wrapping-up” of APIs of Mechanical Turk. The CLTs support programmable control of loading HITs, granting bonus, publishing qualifications, getting results and so on. The pilot project described in chapter 3 is using the CLT for automatic control of crowdsourcing.

For detailed instructions on CLT, see official documentation in [4].

(3) Software Development Kit

Mechanical Turk provides SDK in languages of Java, PHP, Perl, Python (non-official), Ruby and .Net. As it is introduced in official site, Mechanical Turk SDK is a robust set of tools that support programmatic access to submit and approve tasks, and incorporate answers directly into your software applications. If designer hope to include Mechanical Turk workers into their system of program, SDK would be the best choice.

Three files define a HIT in CLT and SDK:

- The input file consists of variables for different HITs in a HIT batch.
- The question file is a XML file with strict requirements on structures. There are two ways to define a HIT, one is using a QuestionForm (XML file) with pure HTML code as formatted content. The other way is to use external question, where the XML file only contains an external URL that leads to the HTML page for HIT and annotation (the variable defined in the input file) names. The external HTML file may use Javascript to make HITs interactive.
- The properties file defines the HIT's title, description, key words, reward amount, assignment number for each HIT, allotted time (maximum time length a worker can work on the HIT), auto approval time, and required qualifications if there is any.

	Pro.	**Con.**	**HIT design**	**Suitable task type**
RUI	Straightforward, easy to pick up, ready-to-go templates.	Limited functions., Non-programmable control.	Templates, HTML.	Typical and simple task like categorization, data collection, survey, transcription and writing.
CLT	Programmable control, diversified functions, universal to programming languages.	Overhead for familiarizing, Not suitable for embedding to another program.	XML (QuestionForm,ExternalQuestion)	Tasks that need dynamic and programmable control. Complex tasks that need more functions.
SDK	Programmable control, diversified functions, can be easily embedded to other programs.	Overhead for familiarizing, Restricted to certain programming languages.	XML (QuestionForm,ExternalQuestion)	Tasks that need dynamic and programmable control. Complex tasks that need more functions. Task publishing or answers are embedded to "father programs".

Table 1 Mechanical Turk developer tools

Next, common types of qualification and HIT design models are listed. A requester needs to choose a suitable model according to the task's requirements.

Qualification Models

1) Assigning Qualification

- Assign qualifications to workers by Worker-ID. No tests are involved.

2) Quiz Qualification

- Amazon Mechanical Turk automatically grades test answers with an answer key. A worker knows his score right after he completes it.

- Graded by requesters in local computers.

3) External Link

- The qualification test page has a link that leads a worker to an external webpage. The worker will finish the test in the external page.

Tips

a) If the qualification answers are automatically graded by the platform, requesters CANNOT download answers from the platform. So if a survey is involved, the results have to be reviewed locally, or the survey has to be transferred to an external link where the answers can be retrieved.

b) Except creating qualification type and assigning qualification to workers, all the other operations have to be done by API (i.e. CLT or SDK tools). The qualification page is designed by writing a QuestionForm XML file, no external question could be used, i.e. no Javascript is allowed.

HIT Models

1) Uploaded HTML page

- Tool: RUI
- Limits: Uploading by hand. Data stored on Amazon server and is updated once per hour.
- Strength: Straightforward, fast and easy to pick up.

2) QuestionForm XML

- Tool: CLT, SDK through API
- Limit: No JavaScript or CSS. Layout and UI is very limited. Strict structure of XML file makes debugging a tedious process.
- Strength: Program controlled uploading. Fast and easy to pick up.

3) ExternalQuestion HTML page

- Tool: CLT, SDK through API
- Limit: nearly no limits except the external HTML page should be stored online which may cost some money.
- Strength: Total control of UI and data retrieving. Javascript code can be used and

interaction is possible.

Tips

a) Personally, I highly recommend the ExternalQuestion approach since requesters can have total control of UI and are able to make HITs more interactive. Also Mechanical Turk will pass HIT ID and worker ID and many other parameters to the external page so the requester could write Javascript code to customize for different HITs or workers.

b) An example on how to create an ExternalQuestion HIT is appended in Appendix I.

Developer tools summary: Mechanical Turk provides developer tools of requester user interface (RUI), command line tools (CLT) and software development kit (SDK). CLT and SDK support programmable control. There are several models for qualification and HIT. The ExternalQuestion model for HIT design is highly recommended.

2.5 Crowdsourcing Framework

This section introduces our framework for crowdsourcing on Amazon Mechanical Turk and gives a step-by-step guide on how to use it for current HIT types and how to adapt the programs for other urgent human computing tasks.

Features of our framework:

- **Simple**: running the system you need only to type a few lines of command.
- **Safe**: file back-up, confirmation questions before running commands.
- **Automatic**: automatically review survey and qualification test answers, approve or reject test takers.
- **Smart**: HIT bumping help us get more worker responses.
- **Programmable**: program controlled HIT & Qual. loading, reviewing and worker grading.
- **Flexible**: Quick adaptation to different HIT type

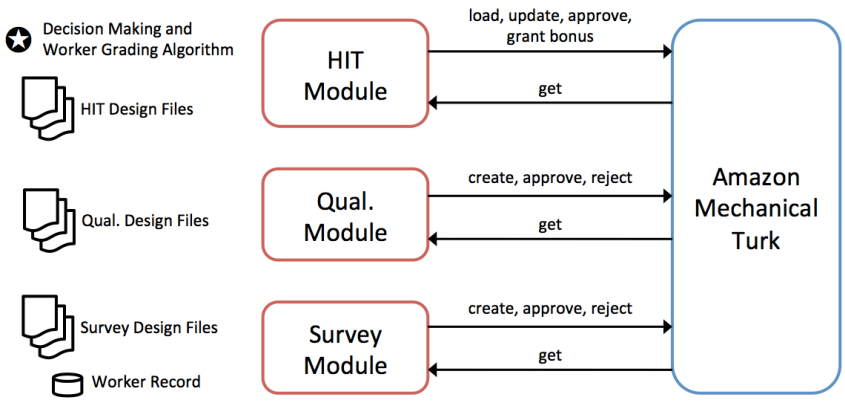


Fig. 2 Crowdsourcing framework

The figure above shows the crowdsourcing framework and the table and instructions give the steps a requester should follow to get a task done. STEP 1 is the most time-consuming step and when the STEP 1 is done the work left is just typing some short commands to start the task and wait for workers to respond, when enough workers have finished the HIT, you can simply type a few commands to review the results and pay the workers.

Any HIT in the task folder (in the same zip file with this documentation) can be published and reviewed in the following steps.

Steps	**Input**	**Output**	**Programs**
Publish Survey Qual.	survey.properties survey.question	survey.properties.success survey_worker_record.csv	survey_qual.py survey_qual_auto.py survey_review.py
Publish Test Qual.	qual.properties qual.question qual.key	qual.properties.success	qual.py qual_auto.py qual_review.py
Publish HITs	hit.properties hit.question hit.input external-hit.html	hit.input.success hit_results.csv	hit.py hit_bumping.py hit_init.py
Review HITs	hit_results.csv	worker_score.csv summary.csv worker_bonus.csv worker_toapprove.csv worker_toreject.csv output.csv	hit_review.py
Approving Workers	worker_bonus.csv	NULL	hit.py bonus.py

Table 2 steps to run the crowdsourcing system

Note: To successfully run the programs, the computer should have python and java installed and is using Linux or Mac OS or the command line in python scripts need to be changed to command lines for Windows OS.

The files in blue color are “static” files that would not change during the whole process; they are mainly the design files, properties files and resources files for HITs and qualifications. The files in orange color are those returned by Mechanical Turk, including the HIT answers and success file (indicating the success of HIT qualification loading, HIT and qualification ID stored in the file). The files in black are those generated by local programs.

The program files (i.e. python script files) in green color are high-level capsulation of CLT function that are irrelevant for task type. As long as the file directory structure meets the framework requirement, those scripts can work for different tasks. Other program files are in charge of HIT & qualification review or system initialization and they need to be modified or rewritten for different tasks (i.e. different HIT types and qualifications).

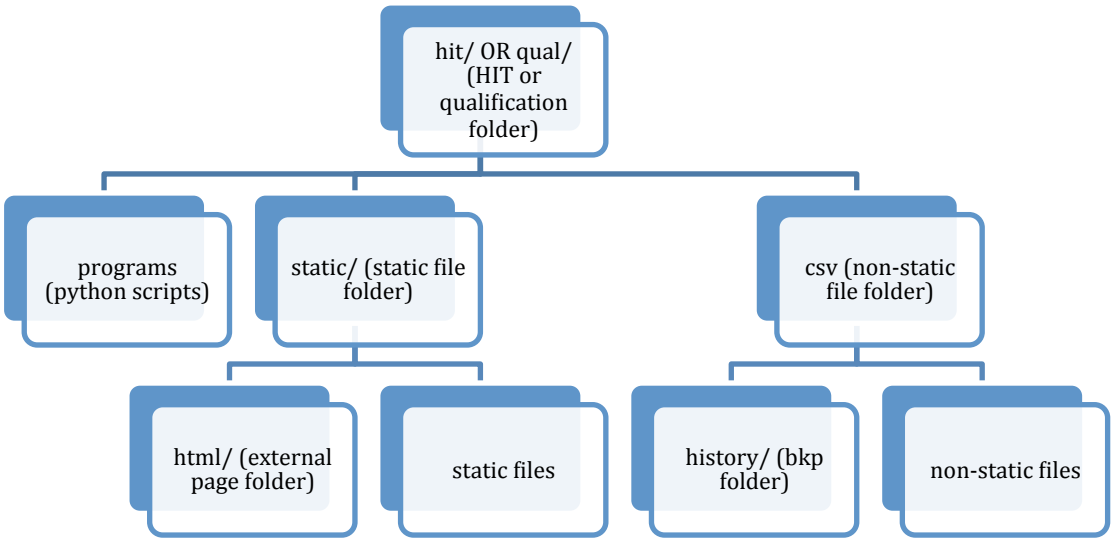


Fig. 2 file directory structure

To get a human computing task done in Amazon Mechanical Turk by using our crowdsourcing framework, a requester should follow the steps below.

STEP 1: Task customization

1) Design all the static files in blue color.

2) Design all the programs in red color.

STEP 2: Module test in Sandbox

Test entry survey, qualification test and HIT separately in sandbox.

STEP 3: Overall test in Sandbox

1) survey_qual.py –create *(publish entry survey)*

2) survey_qual_auto.py *(automatic review of entry survey)*

3) qual.py –create *(publish qualification test)*

4) qual_auto.py *(automatic review of qualification test)*

5) hit_init.py *(initialize some files)*

6) manually set HIT qualification type id in .properties file

7) hit.py –load *(load the batch of HITs)*

8) hit_bumping.py *(automatic updating of HITs)*

9) hit.py –get *(get HIT answer results)*

10) hit_review.py *(locally review HIT answer results)*

11) hit.py –approve –successfile *(approve all the workers = pay for them)*

12) bonus.py –grant *(grant bonus to well-performing workers)*

STEP 4: For Real

Just change the mturk.properties file (or move folders) and repeat everything in step 3.

To adapt existing HITs to other tasks, you need only to change the files in blue and red color. The blue color files determine the content, properties and UI of qualifications and HITs and the red color file determines how you review the results, grade and pay for workers.

An Example：

This example demonstrates how to publish an entry survey, a qualification test and a HIT type in sandbox.

Before trying out the example, you need to set the service_url, access_key and secret_key in mturk.properties file in “bin” folder mturk home directory (aws-mturk-clt-1.3.0 folder). Then you can change your current directory to the “test” folder and repeat each command in STEP 3 above. If the command line tool says qualification or HIT already exist, just change the title in properties file and try again. If the HIT is successfully published, you will see a HIT like the picture shows below in the sandbox HIT list.

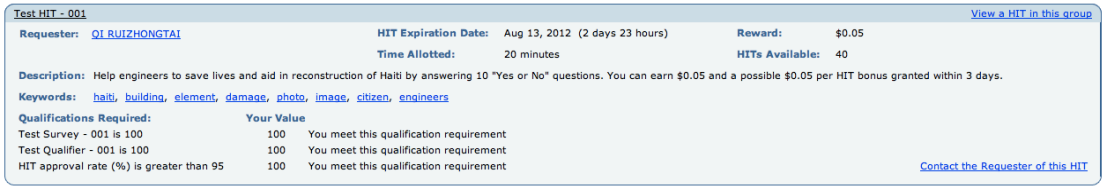


Fig. 3 an example HIT in Mechanical Turk sandbox

A typical way of operating the framework is to open several command windows with each responsible for one job. For example, in the snap shot below, the left-top window is in charge of survey review and approving, the right-top window is reviewing qualification, the left-bottom window is running HIT bumping script and the right-bottom window is used to get HIT results.

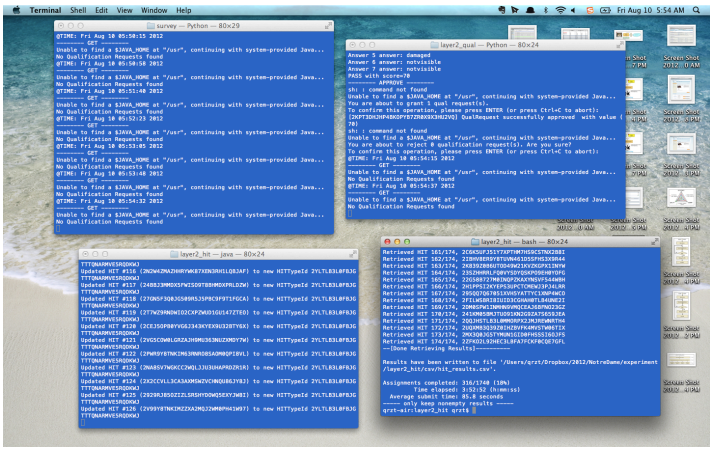


Fig. 4 screenshot of the typical working style when working with the framework

3 Haiti Building Photo Tagging Project

3.1 Overview

This chapter first sketches a "big picture" of the project (3.2-3.3) and then dive down to design and implementation details (3.4). In 3.2 we will talk about the motivation behind this project and in 3.3 the concepts involved will be introduced. In 3.4 I will describe the first round trials and the problems we met and how we solved the problems when implementing them. In 3.5 the lessons learnt are summarized and there will be some discussion on means to achieve efficiency, economy and high quality.

3.2 Motivation

When natural or man-made disasters happen, efficient post-disaster responses could help save lives and accelerate disaster recovery. Field reconnaissance generates thousands of images, both aerial and street level, of damaged infrastructure and debris that must be examined. However, common problems like image filtering and classification are hard for computers. It takes PhD students years to design and optimize computer algorithms and it takes experts weeks or even months to tag those huge amount of photos.

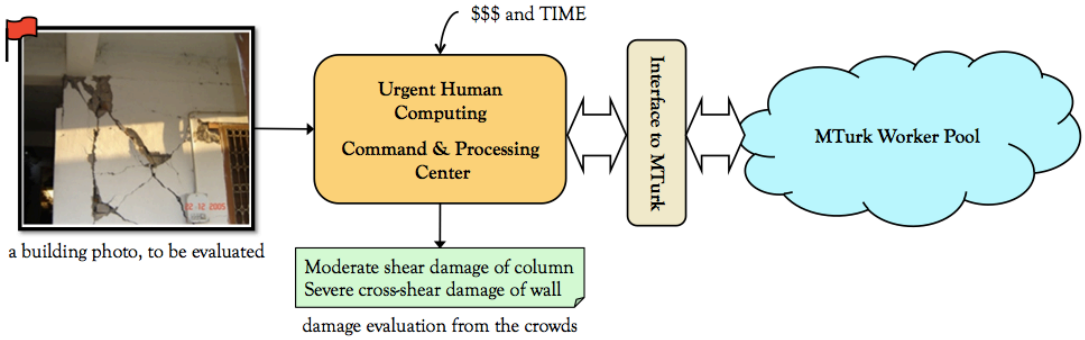


Fig. 5 the crowdsouring system's framework

To solve this problem, we propose the concept of "Urgent Human Computing" and explore to achieve it by "crowdsourcing" on Amazon Mechanical Turk platform. The system works as the picture shows above. The input is a large number of the earthquake building images and some money and time, the crowdsourcing system would embed the images to the predefined HITs and publish them to Mechanical Turk, and the outcome would be the damage evaluation of the buildings in the images.

3.3 Basic Concepts

- **"Urgent Computing"** means providing prioritized and immediate access to supercomputers and grids for emergency computations [11].
- **"Human Computation"** is "a paradigm for utilizing human processing power to solve problems that computers cannot yet solve."[12]
- **"Urgent Human Computing"** is an extension of "Urgent Computing" where supercomputers and grids are replaced by on-demand human workers. The central challenge to realize this concept is how to design system to guarantee efficiency, economy and work quality.
- **"Crowdsourcing"** for Urgent Human Computing releasing tasks to an open community rather than a designated agent. Note that the same task could be done by a small group of experts or members from a closed community. It is expected that the open community and online users from all over the world will finish the task in less time for less payment and give answers with comparable quality as experts or closed community.
- **"Citizen Engineering"** is a general concept that refers to using a cohort of physically dispersed citizens connected by the Internet to solve real-world "engineering tasks" [13][15]. The emphasis here is "engineering tasks" that require background knowledge in a specific area such as civil engineering.

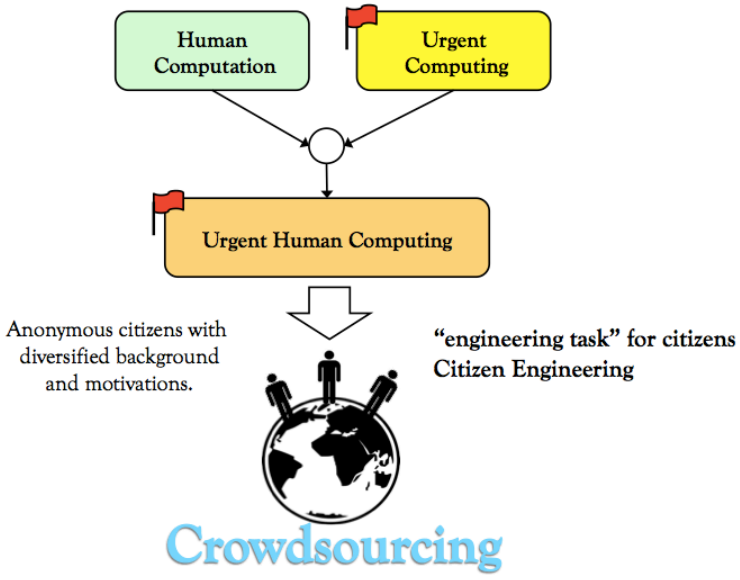


Fig. 6 relationships among concepts

3.4 Experiment Design

(1) Task Decomposition and Workflow Design

As the requester workflow in 2.3, our first define the task and decompose it to a 3 layered structure with 1 HIT type in layer-1, 1 HIT type in layer-2 and 4 HIT types in layer-3.

We decompose the complex task into 3 small ones (i.e. 3 layers) that workers without civil

engineering background are able to complete. Only after one layer is finished, do we go on to the next layer. In layer-3, four threads each for one building element are going on at the same time. This serial-parallel hybrid structure will help us get more reliable results since the decision of each layer is derived from the crowd-wisdom rather than an individual.

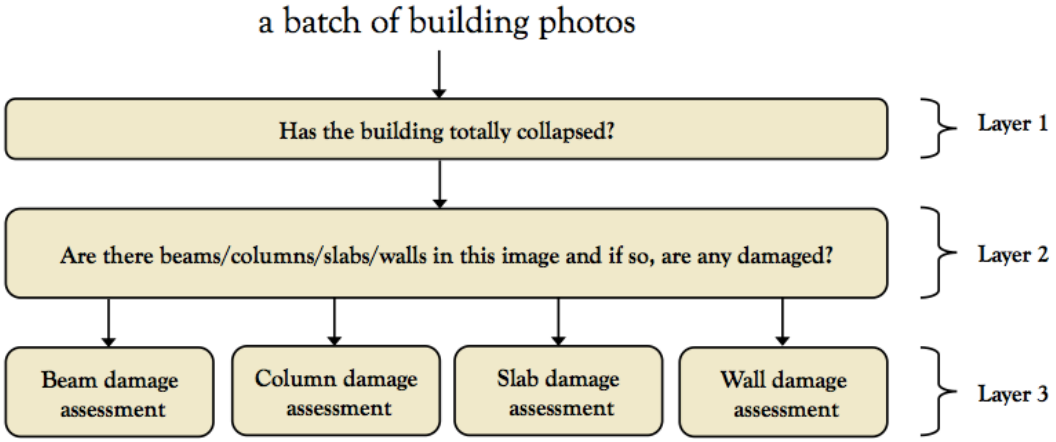


Fig. 7 task decomposed to a 3-layered structure

For detailed design of the HITs in each layer, please see the links below.

https://s3.amazonaws.com/citizen-engineers-hit-layer1/layer1-external-hit.html

https://s3.amazonaws.com/citizen-engineers-hit-layer2/layer2-external-hit.html

https://s3.amazonaws.com/citizen-engineers-hit-layer3/layer3-external-beam.html

https://s3.amazonaws.com/citizen-engineers-hit-layer3/layer3-external-column.html

https://s3.amazonaws.com/citizen-engineers-hit-layer3/layer3-external-slab.html

https://s3.amazonaws.com/citizen-engineers-hit-layer3/layer3-external-wall.html

Note that the links lead to external HTML pages of HITs that are published in the second round (large-scale experiments). The images do not show up because they are determined by ".input" files. To see the HITs with images you have to publish either in sandbox or in real platform. The HITs published in the first round (small-scale trials) do not look like the linked pages. To see how the first round HITs look like please go to 3.3.2 - 3.3.4.

In order to filter out unreliable workers and improve work quality, we design a workflow with qualification tests and a bonus incentive. A qualification test includes a tutorial and several questions that are exactly the same as those in real tasks. For each layer, a worker has to pass the test for that layer to be "qualified" to work on the task. Workers are also told that they would receive a bonus if they perform well on the tasks. Bonus is given to workers according to the estimated correction rate of their answers.

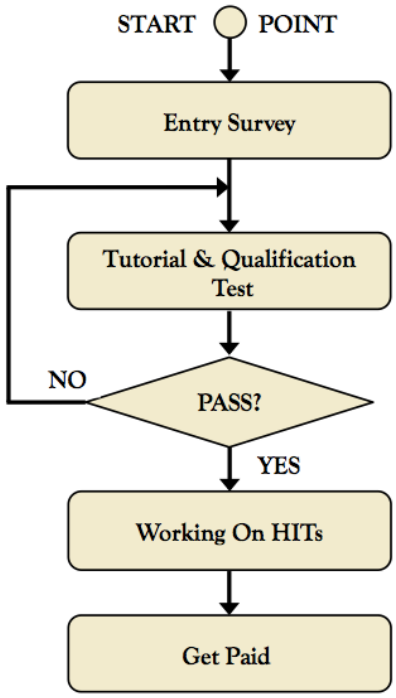


Fig. 8 workflow for Turk workers

Task and Workflow Design Summary:

- Decomposing the task to a 3-layered structure
- Entry survey & qualification test
- Bonus incentive

(2) Decision Making and Worker Grading

10 workers independently work on a single HIT, how should we make decision for the HIT question from 10 separate answers? There could be "noisy" inputs from low quality workers or careless clickers, but the assumption is that majority of the workers are serious with the questions and are likely to give the "right" answer. This assumption probably fails for ordinary workers working on "engineering tasks" that require special knowledge in a specific field. However, the pre-qualifying test helps filter out low quality workers and the bonus incentive will make qualified workers more intent to work seriously. So the assumption still stands in our project and a "simple voting" decision-making could retrieve accurate results. Simple voting or "majority rule" regards the majority answer as the "right" answer. It is easy to implement and effective under the assumption that the majority is right.

To implement the bonus granting, we need to grade the workers' performance. Well-performed workers should be granted with bigger bonus and poorly performed ones should not be granted any while spamming users should be rejected or blocked. Since there is no ground truth for the HIT questions, the results derived by simple voting are considered as the grading standard. Those who are more consistent with majority opinions are considered as better workers. This grading means is effective under the assumption above that the majority is right.

Note that to achieve more accurate worker grading, we can outsource a small portion of HITs to experts and save their answers as "ground truth" and use this standard answers to evaluate Turk workers' performance. This is not implemented in the following experiments but should be a promising way to increase grading accuracy.

On the other side, we could benefit from accurate worker grading in making HIT answer decisions. Simple voting could be replaced by "weighted voting" the final answer is more biased towards workers with higher grades. In one of Zhi Zhai's paper [14] a machine learning algorithm iteratively grades workers and do weighted voting until number of workers in different grade groups become stable. This weighted voting algorithm is not implemented in the pilot experiment but should be a included in a more delicate system.

Decision Making and Worker Grading Summary:

- Majority rule assumption stands because of the pre-qualifying requirement and bonus incentives.
- Decision making by simple voting
- Worker grading by quantify worker answer's consistency with the majority opinion.
- More delicate methods: expert sourcing to help worker grading; worker scores help improve decision making by weighted voting.

3.5 Experiments

In this section I will closely describe the trials and pilot experiments we have done on Mechanical Turk and present our "evolving" understanding of design principals. The experiment has two round where the first round is a small-scale trial and the second round is large-scale task releasing. The trials and experiments include:

1) Layer-1 Trial (first round)
2) Layer-2 Trial (first round)
3) Layer-3 Trial (first round)
4) Layer-1 Experiment (second round)
5) Layer-2 Experiment (second round)

For each trial or experiment I will first summarize the problems met and how we solved or avoided them. Then the model and content of the HIT and qualification test are introduced, with screen shots showing their UI (screen shots may only include part of a HIT or qualification).

(1) Layer-1 Trial

Layer-1 trial is asking workers to identify whether the building in the photo has totally collapsed. It is our first experiment on Mechanical Turk and everything works well. We are using RUI and CLT to interact with Mechanical Turk. RUI is used to design HIT, get results and approve

workers. CLT is used to control qualification test and entry survey. Data processing program and CLT capsulations are all written by python scripts.

The only thing to mention in this layer's trial is that when we design the layer-1 questions, we have two options, one is to make it like a photo filtering task (multiple choice question, each choice is a photo) and the other is "Yes or No" for every photo. Although working with the first design is more efficient but the work quality may fall down because the effort of serious work and careless clicking is quite different. Serious workers have to go over every photo but clicker might just check one or two photos and submit so they needn't to look at every photo carefully. Yet if there is a "Yes or No" question for each photo, both serious workers and potential clickers would have to look at the photo and select an answer. Gap between the effort to cheat and the effort to do serious work becomes small so workers are more likely to input a little more effort to gain a bonus.

Question design guideline: Make the difference between the effort it takes to click through and the effort it takes to answer seriously small.

There are 50 photos, 5 HITs with 10 photos per HIT. The payment for each HIT is $0.05 base reward plus a possible $0.05 bonus if the worker's answers are almost consistent with the "majority opinion". For more details of this trial, please go to appendix II to see a brief report on layer-1 trial.

Entry Survey and Qualification Test

Model: Quiz Qualification with two parts of entry survey and qualification test. Since we need to acquire survey answers, automatic grading is out of option. We use quiz qualification with local grading. To make the grading faster and shorten the workers' waiting time, the grading is controlled by a Python script which reviews, grades and uploads approval or rejection results every 30 seconds.

Content: The qualification's first part is an entry survey asking for worker's basic information, involving gender, age group, location, education level and motivation for doing this task (workers are informed that these survey data is only for research purposes and will be held as confidential and exempt from disclosure). The second part is qualification test. The test part first comes with an instruction and then several examples to illustrate the classification rules. The workers are required to read instruction and examples first and then answer 4 qualifying questions. The questions are exactly like the questions in HITs. If the worker answer all 4 questions correctly and doesn't miss any survey questions (this is told to workers before they start working on part two), he will pass the test. The grading would be completed in about 30 seconds (processing period of local grading program) after a worker takes the test. A notification email will be sent to the worker's mailbox indicating the result. An approval contains only the test title and test score whereas the rejection mail contains requester name, test title, test taking time and rejection comments written by the requester.

Sample Shots:

Viewing the qualification in the qualification tag page. Note that time allotted to the qualification test is 20 minutes and that to the survey is 5 minutes.

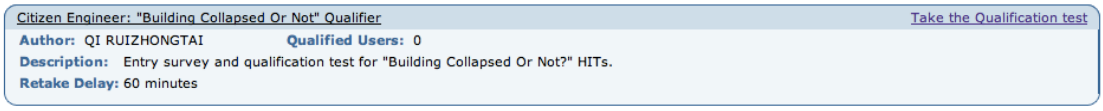


Fig. 9-1 layer-1 qualification screen shot

After one clicks “Take the Qualification test”:

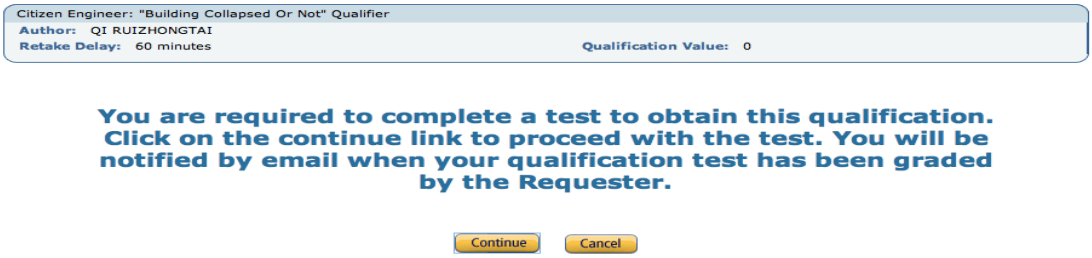
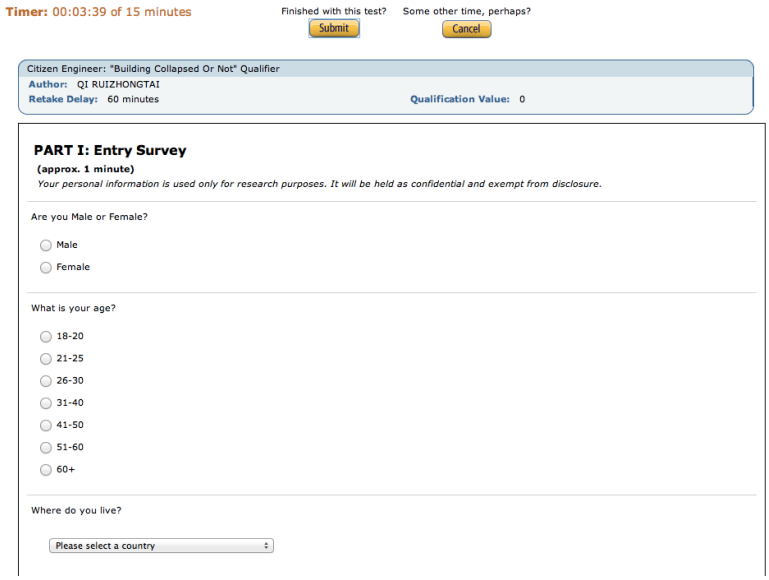


Fig. 9-2 layer-1 qualification screen shot

The test page:



What is the highest level of education that you have completed?

- Less than High School
- High School
- Some College
- 2-Year College Degree
- 4-Year College Degree
- Advanced Degree

What is the major reason you are doing this work?

- Money
- Fun
- Contributing to a good cause
- I have lots of free time
- The chanllenge

PART II: Qualification Test

(approx. 4 minutes)

Please read the instructions and examples first and then answer the questions in the qualifying test.

If you answer all 4 testing questions correctly you will be qualified to work on real HITs.

Instructions:

Your task is to judge whether the building in a photo has totally collapsed.

- Select **"Yes"** if the photo does not clearly show a building or part of a building (i.e. the building has totally collapsed).
- Selcet **"No"** if the photo shows an entire building or part of a building.

When more than one building is visible, make your judgement based on the most prominent building (i.e. the one in the center or front of the picture).

Examples:

Example 1. Has the building totally collapsed??

Yes. Photo does not clearly show a building or part of a building. It has totally collapsed.

Example 2. Has the building totally collapsed?

Yes. Photo does not clearly show a building or part of a building. It has totally collapsed.

Example 3. Has the building totally collapsed?

No. Photo shows an entire building.

Example 4. Has the building totally collapsed?

No. Photo zooms in to show only part of a building.

Example 5. Has the building totally collapsed?

No. Consider the building that is the focus of the photo, such as the building inside the red square.

Test:

Please answer the test question under each photo.

1. Has the building totally collapsed?

- Yes
- No

2. Has the building totally collapsed?

- Yes
- No

3. Has the building totally collapsed?

- Yes
- No

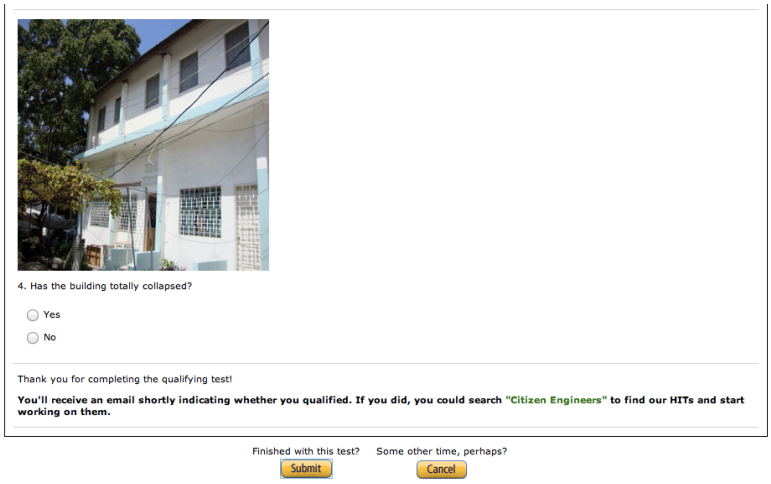


Fig. 9-3 layer-1 qualification screen shot

Workers will receive an email indicating whether they pass the test within 30 seconds ()The approval and rejection email are as follows (color changed distinction).

The approval mail looks like below.

Title: Qualification Granted Notification

Content:

Greetings from Amazon Mechanical Turk,

You have been granted the following Qualification:

- Citizen Engineer: Building Collapsed Or Not Qualifier, with a score of "4"

Qualifications you earn allow you to complete new kinds of HITs. Each Qualification has a score associated with it. You can complete HITs for which you have the required Qualifications and a sufficiently high score.

To begin working on HITs requiring the Qualifications you possess, click on the link below:

http://www.mturk.com/mturk/findhits?match=true

To view all of your Amazon Mechanical Turk Qualifications, please visit the page below:

http://www.mturk.com/mturk/findquals?earned=true

Sincerely,

Amazon Mechanical Turk

http://www.mturk.com

410 Terry Avenue North

SEATTLE, WA 98109-5210 USA

The rejection email looks like below.

Title:Qualification Request Rejection Notification

Content:

Greetings from Amazon Mechanical Turk,

One of your Qualification requests has been rejected by the Requester who maintains the Qualification. Please see details about the Qualification request below.

Qualification Rejected: Citizen Engineers: Building Photo Filtering's Entry Survey and Qualification Test
Requester: QI RUIZHONGTAI
Request Date: July 12, 2012
Rejection Date: July 12, 2012
Rejection Reason: Sorry, you didn't pass the qualification test. You can try it again when you are ready. Thank you!

You may request the Qualification again on or after July 12, 2012. For more information about Qualifications, please see the FAQ:

http://www.mturk.com/mturk/help?helpPage=main

Sincerely,
Amazon Mechanical Turk
http://www.mturk.com
410 Terry Avenue North
SEATTLE, WA 98109-5210 USA

Short summary of approval and rejection mail: (1) Approval email only contains test title and score. (2) Rejection email contains test title, requester, date and rejection reason.

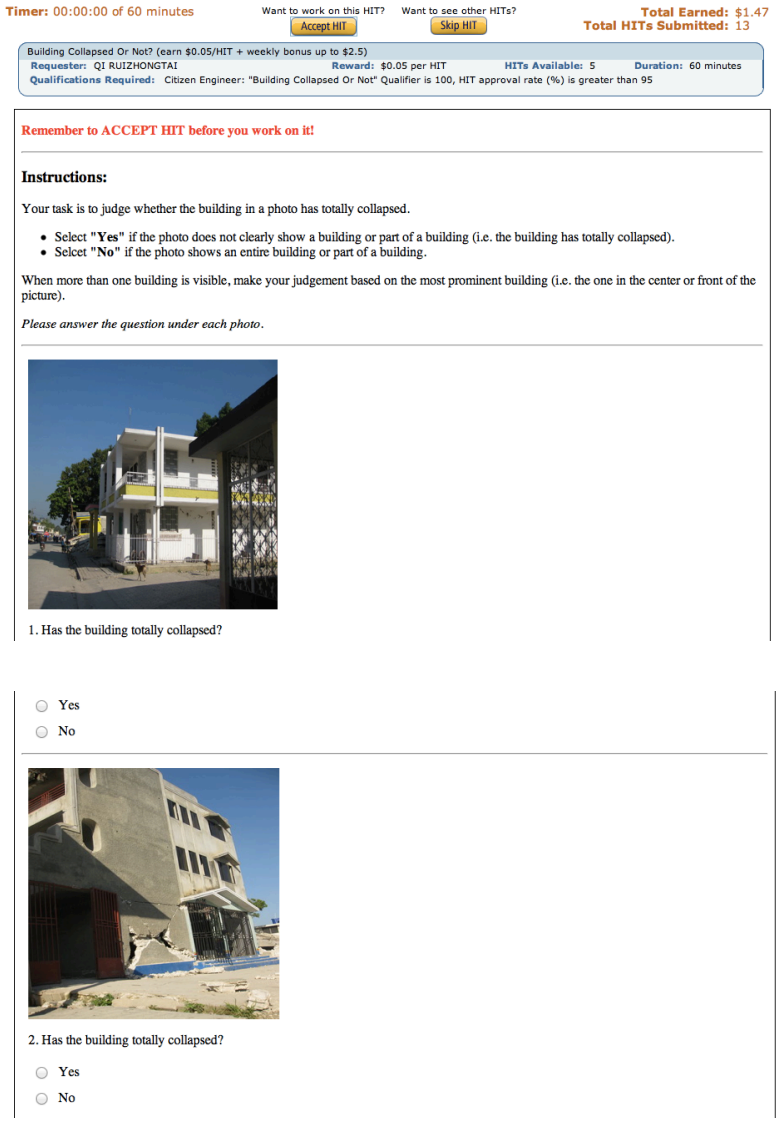
HIT

Model: HTML Page. Using external question requires that the page is stored on a web server and using QuestionForm XML file provides no benefits for current project so the most straightforward and appropriate way is to upload HTML files to RUI.

Content: The question page contains a simple instruction and 10 photo questions with a confidence question and a feedback textbox. We used to consider use a photo filtering template but it is against HIT design principal 1: make it hard to spam. If filtering is used, there could be clicker just select nothing or select everything so they can finish a HIT in very short time. However, if it is yes and no question and it is required to be answered, the effort it takes to spam will be nearly the same as to answer it seriously, so with the bonus incentives the worker would be more like to give serious answers.

Sample Shots:

Fig. 10-1 layer-1 HIT screen shot



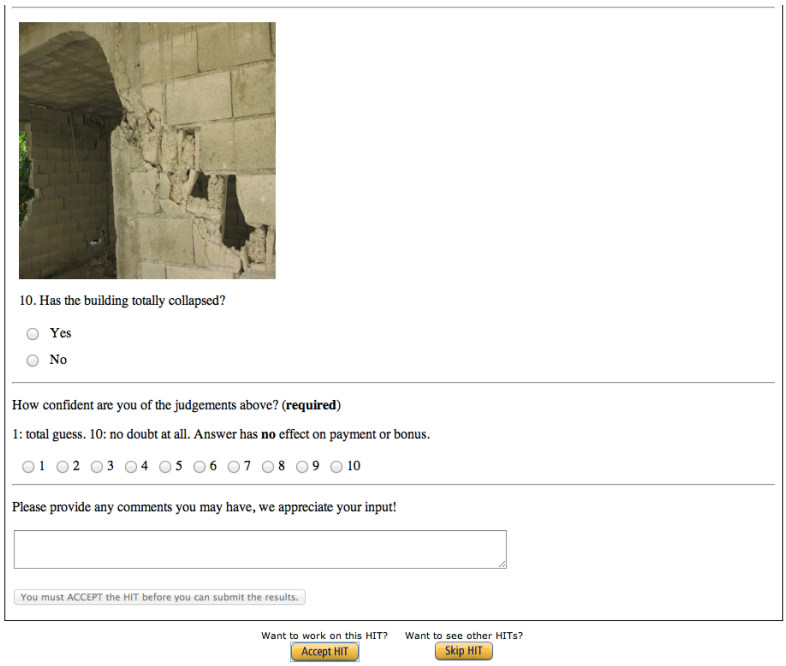


Fig. 10-2 layer-1 HIT screen shot

HIT answers are reviewed by a python script, which uses "majority rule" to make answer decision and use that decision to score workers. Workers with accuracy greater than 80% would receive a $0.05 per HIT bonus.

Layer-1 Trial Summary: First experiment on Mechanical Turk (non-sandbox), using RUI, CLT and Python scripts, using bonus incentives, entry survey and qualification test. HIT is controlled by web interface, i.e. RUI and survey, qualification are controlled by CLT commands.

(2) Layer-2 Trial

We met two problems in layer-2 trial. The first problem is how task-decomposing. We are faced with several choices: (a) combine "element damaged or not" questions with "element visible or not" questions in layer-2 (b) put "element damaged or not" into layer-3 (c) make "element damaged or not" question a separate layer between original layer-2 and layer-3.

Finally we decide to combine the two questions of "element visible or not" and "element damaged or not" into layer-2 because identifying elements take much time but judging whether the elements are damaged is easy when the elements are identified.

The other problem is the unexpected low response rate. Twenty four hours after we published the HIT, not a single worker came to work on them.

Raising payment and lowering qualification requirement temporarily solved the worker response problem but HIT bumping shows much better effect in layer-3 trial and the second-round experiments. For a more detailed description on problems met in layer-2 trial, please see the report in appendix III.

There are 40 photos, 8 HITs with 5 photos per HIT. The payment for each HIT is $0.20 base reward plus a possible $0.05~$0.10 bonus if the worker's answers are almost consistent with the

“majority opinion”.

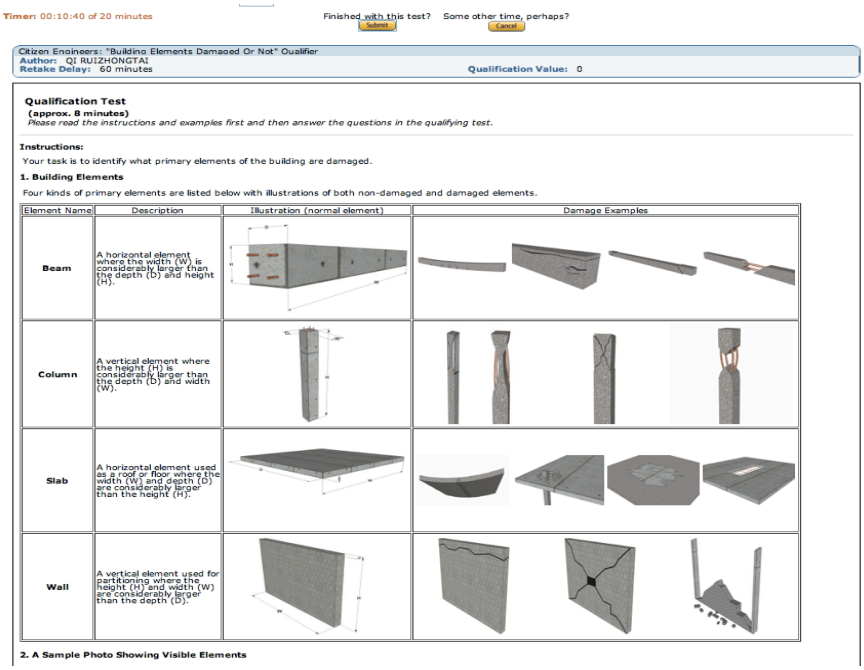
Entry Survey and Qualification Test

Model: Since each HIT type has its own qualification test but the survey should be the same for all HIT types in our project, we decided to separate survey and qualification test in layer-2 trial. So now there will be two qualifications, one is survey and another is the test.

Content: The standard to pass the survey is to answer all the survey questions. At first the requirement for passing the test is to answer all the 8 test questions correctly. However, worker response is extremely low so we have to lower the standard to equal to or more than 6 questions correct answered.

Sample Shots

Fig. 11-1 layer-2 qualification screen shot



3. Style of Question

For each photo, there are 4 questions, one for each element. For example, the question for column looks like:

Are there columns in this image and if so, are any damaged?

- *One or more columns are visible and at least one is damaged*
- *One or more columns are visible but none are damaged*
- *No columns are visible*

Note that when multiple elements of the same type are visible, if any of them are damaged, that element is counted as damaged.

Examples:

Example 1
Beam: One or more beams are visible and at least one is damaged
Column: No columns are visible.
Slab: No slabs are visible.
Wall: No walls are visible.

Example 5
Beam: One or more beams are visible but none are damaged
Column: One or more columns are visible and at least one is damaged
Slab: One or more slabs are visible but none are damaged
Wall: One or more walls are visible and at least one is damaged

Test:

Please answer the test questions under each photo.

For each photo, there are 4 questions one each on beams, columns, slabs and walls.

If you answer all 8 questions (4 for each photo) correctly you will be qualified to work on real HITs.

1. Are there **beams** in this image and if so, are any damaged?

- One or more beams are visible and at least one is damaged
- One or more beams are visible but none are damaged
- No beams are visible

2. Are there **columns** in this image and if so, are any damaged?

- One or more columns are visible and at least one is damaged
- One or more columns are visible but none are damaged
- No columns are visible

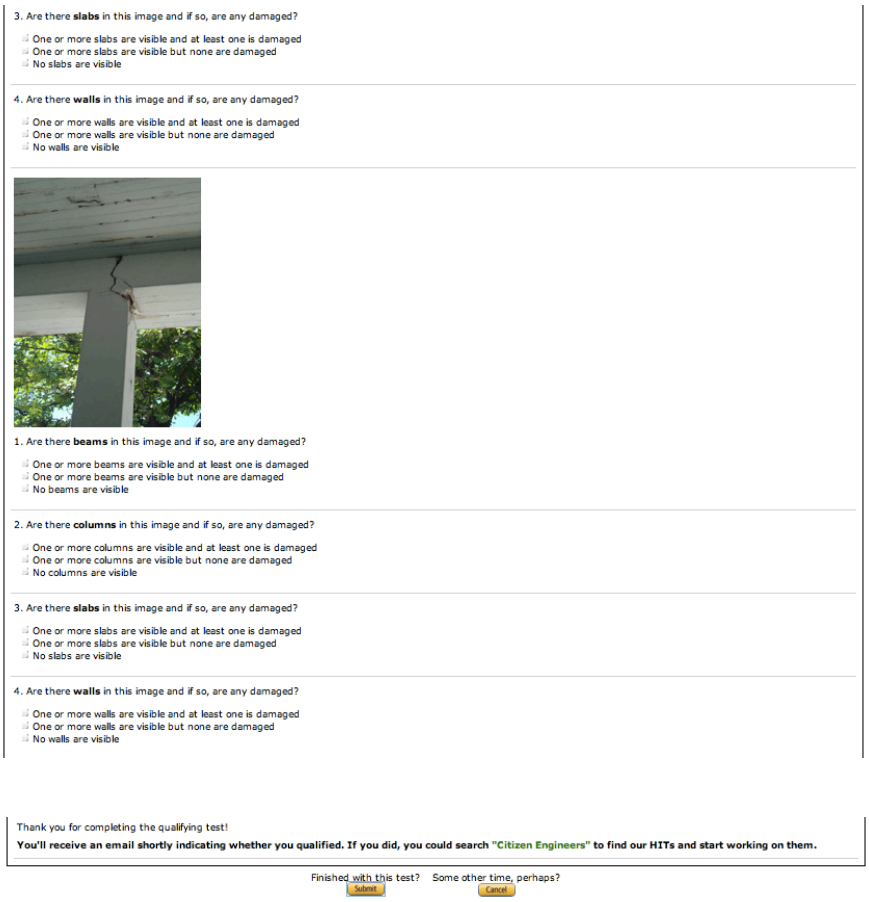


Fig. 11-2 layer-2 qualification screen shot

HIT

Model: The same model as layer-1 trial. HTML file uploaded to RUI.

Content: To raise more worker attention, the payment the original base reward is raised from $0.05 to $0.20. The HIT is similar to the qualification test except it does not include the full length instructions and examples (the instructions and examples are written in a HTML file stored in S3 server and the file can be accessed from HIT by clicking “here” to open a hyperlink).

HIT answers are reviewed by a python script, which uses “majority rule” to make answer decision and use that decision to score workers. Top ranked workers would receive a bonus for their good job.

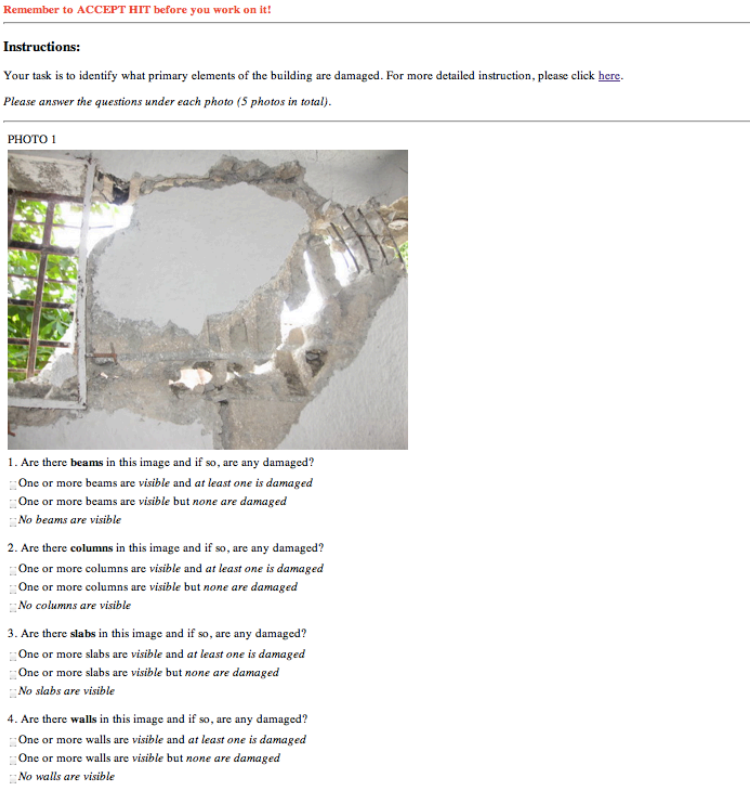
Each HIT contains 5 photos, 21 questions (each photo is with 4 questions and at last there is a subjective question asking for the worker’s confidence on hit/her work).

Layer-2 trial improvements:

Separate entry survey and qualification test, link from HIT to instructions and examples.

Layer-2 trial problems:

- *Task decomposing options. What we did:* combine "element visible or not" and "element damaged or not" questions into layer-2
- *Low worker responses. What we did: (1) Re-publish the HITs. (2) Raise payment (3) Make the HIT's title more attractive.*


Remember to ACCEPT HIT before you work on it!

Instructions:

Your task is to identify what primary elements of the building are damaged. For more detailed instruction, please click here.

Please answer the questions under each photo (5 photos in total).

PHOTO 1

1. Are there **beams** in this image and if so, are any damaged?
   - One or more beams are *visible* and *at least one is damaged*
   - One or more beams are *visible* but *none are damaged*
   - *No beams are visible*

2. Are there **columns** in this image and if so, are any damaged?
   - One or more columns are *visible* and *at least one is damaged*
   - One or more columns are *visible* but *none are damaged*
   - *No columns are visible*

3. Are there **slabs** in this image and if so, are any damaged?
   - One or more slabs are *visible* and *at least one is damaged*
   - One or more slabs are *visible* but *none are damaged*
   - *No slabs are visible*

4. Are there **walls** in this image and if so, are any damaged?
   - One or more walls are *visible* and *at least one is damaged*
   - One or more walls are *visible* but *none are damaged*
   - *No walls are visible*

Fig. 12 layer-2 HIT screen shot

(3) Layer-3 Trial

The low response rate problem is not solved in layer-2 trial. Actually after raising payment and lowering qualification standards only 91% of the HITs are finished when the HIT type expires. Efficiency is critical for urgent human computing, so worker responses to HITs must be raised up.

In order to raise worker response to our HITs, a HIT bumping technique is developed where a program periodically update the HITs to make them rank higher in the newest HIT list. (The idea comes from the observation on HITs published by crowdsourcing companies. Their HITs are always in the first few pages of newest HIT list. The only explanation is that the HITs are being updated very often. Thinking of "HIT bumping" mentioned in one of Robert Miller's presentation slides[20], I am sure that HIT bumping works in raising HIT's ranking in newest HIT list)

HIT bumping technique brings a dramatic growth in worker response. Four threads of HIT type in layer-3 are finished in less than one day, compared with only 91% of HITs finished in 3

days in layer-2 trial.

HIT answers are reviewed by a python script, which uses “majority rule” to make answer decisions and use that decision to score workers. Workers with accuracy greater than 80% would receive a $0.05 per HIT bonus.

There are 20 photos, 7 HITs in beam threads, 22 photos, 8 HITs in column threads, 25 photos, 9 HITs in wall threads and 11 photos, 4 HITs in slab threads. The payment for each HIT is $0.05 base reward plus a possible $0.05 bonus if the worker’s answers are almost consistent with the “majority opinion”.

Entry Survey and Qualification Test

Model: QuestionForm and CLT control, the same with layer-2 trial.

Content: There are four threads in layer-3 (i.e. four HIT type and four qualification tests) for elements of beams, columns, slabs and walls. Qualification test for each element has the same style. It contains a short instruction on element type, element damage pattern and a single photo test asking for damage pattern recognition and subjective damage severity assessment.

Screen Shots:

Fig. 13-1 layer-3 qualification screen shot

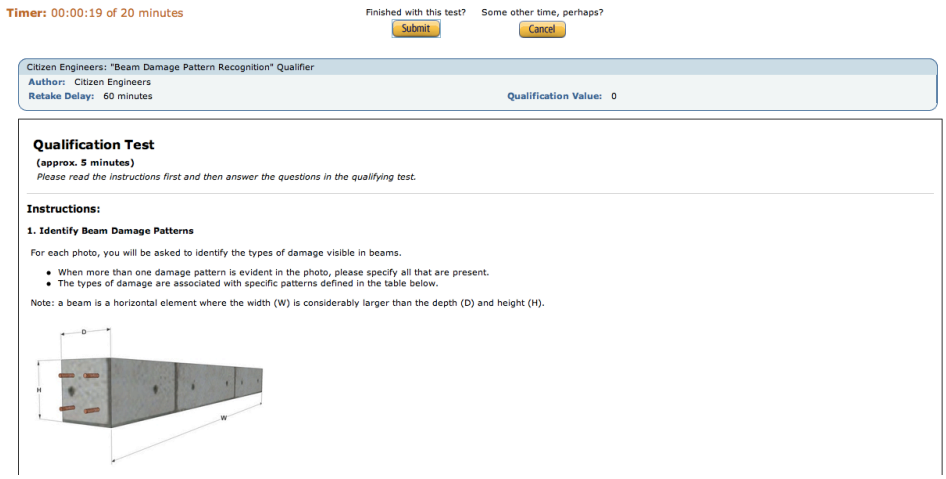


Fig. 13-2 layer-3 qualification screen shot

HIT

Model: ExternalQuestion with CLT control. HIT bumping technique requires the HIT model (three HIT models described in 2.4) must be QuestionForm or ExternalQuestion and HIT must be manipulated by CLT or SDK. The HIT model in layer-1 and layer-2 trial is HTML page in RUI, so we need to make a change. Since QuestionForm (XML) does not support more flexibility than

HTML in RUI, and debugging with XML file is not a pleasant experience, and ExternalQuestion could support Javascript, HIT model in layer-3 trial is changed to ExternalQuestion and CLT is used to control HIT activities.

Content: With support of Javascript, the instructions and examples are included in the question HTML file and a button at the top of the page controls its display. Each HIT contains 3 photos, 7 questions (each photo is with 2 questions and at last there is a subjective question asking for the worker's confidence on hit/her work).

See an example (without photos) page at:

https://s3.amazonaws.com/citizen-engineers-hit-layer3/layer3-external-beam.html

The HIT title, description etc. are defined in the "properties" file under the experiment->layer3_beam_hit folder.

Layer-3 trial improvements:

(1) New system structure. HIT in ExternalQuestion model. HIT and qualification are both controlled by programs. Javascript is supported and interactive UI is realized.

(2) HIT bumping technique dramatically raises worker responses.

(4) Layer-1 Experiment

After the first round of trials, the crowdsourcing frame is tuned to its best status as it is introduced in "2.5 Crowdsourcing Framework". Another round of experiment with the full set of photos is ready to go.

In the second round's layer-1 experiment, survey and qualification test are separate as layer-2 and layer-3 trials. HIT model is ExternalQuestion as that in layer-3 trial. With existed design files of layer-1 trial it's easy to adapt them to current framework.

In layer-1 experiment, there are 400 photos in 40 HITs (i.e. 10 photos in one HIT), 400 assignments (10 assignments per HIT) with payment of $0.05 base plus a possible $0.05 per HIT. All the HITs are finished in just 3 hours!

(5) Layer-2 Experiment

Layer-2 experiment is different from layer-2 trial in 5 aspects:

a) HIT bumping technique is used.

b) Number of available HITs increases to 348 while it is 8 in the trial.

c) One HIT includes 2 photos, thus 8 photo questions plus 1 confidence question while a HIT includes 5 photos, 21 questions in the trial.

d) Answers to the question are simplified. Instructions are clearer. HIT title is changed to be more attractive as "HELP ENGINEERS TO SAVE LIVES AND AID IN RECONSTRUCTION OF HAITI - Layer - 2 ($0.05 reward + $0.05 bonus = $4.0 per hour wage)" while the trial HIT's title is "Building Elements Damaged Or Not? ($0.05 reward + $0.05 bonus)".

e) There are more qualified workers or survey passers when the second round experiment is running.

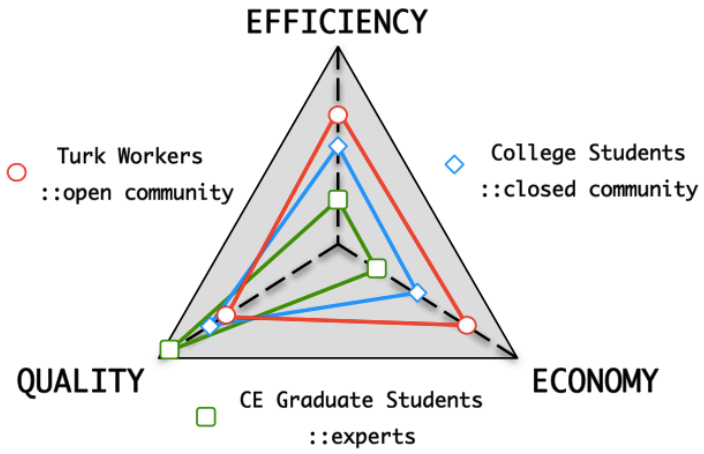
The worker response to layer-2 experiment is really fast. 1740 assignments are completed in 12 hours! This dramatic growth of worker response is due to the four changes above.

Lesson learnt: Keep the HIT clear and lite. Limit question numbers in a single HIT.

3.6 Results and Analysis

This section is on the to-do list, key points listed as below:

- Worker demographics
- Work quality comparison
- Response time statistics
- Work answer patterns (if there is any)



3.7 Lessons Learned and Discussions

We are exploring the concept of Urgent Human Computing by prototyping this project.

(1) Amazon Mechanical Turk

In this pilot project, we learnt a lot about Amazon Mechanical Turk. AMT is a marketplace for online work with the strengths of inherent payment system, abundance of developer tools and large worker base (world wide anonymous could also be a weakness). However, it's not a perfect place for crowdsourcing or urgent human computing tasks. Mechanical Turk is still growing and function limits do impair its power as a crowdsourcing platform.

a) **Lack of task routing**. No smart task search engine or task recommendation system is available in Amazon Mechanical Turk. Workers are usually searching for HITs by the newest HIT or most HIT available ranking [19]. To get enough worker response, a requester has to "game" with the system by HIT bumping or similar technique. It's

possible that a third party take the responsibility of task searching. The task search engine should be a "machine learning" one that is able to infer a worker's expecting jobs from his/her working and test history.

b) **Inconsistency within the platform**. AMT is a large platform, so it is normal that many inconsistencies exist. For example, AMT supports ExternalQuestion model for HITs but not for qualifications even these two are quite similar and people do need that model for qualifications. Also the csv file structures are not the same for RUI, CLT and API.

c) **Lack of communication support among requesters and workers**. The only way for workers and requesters to communicate is through email. That is obviously inconvenient esp. for those workers who are not used to use emails on a daily basis. Requesters could notify workers by email if the developer tools are SDKs but this function is not available for RUI or CLT developers. Even so, it could be better if an IM system is embedded into the AMT platform.

(2) Task and Workflow Design

In this pilot project, we tried many new task and workflow design ideas to fit "engineering task" into a crowdsourcing context. Without enough experiments to give comparative data, we cannot say which design idea has which impact, so we just discuss the possible benefits gained from our new design ideas, where the inferences are based on the lessons learnt from trials and experiments described in 3.5.

a) Qualification test

"Engineering tasks" require engineering knowledge but we know nothing about workers' background in a specific engineering area**.** Thus, we have to arrange a series qualification tests to make sure only qualified workers join our HITs. This is important because if ordinary workers could also work on the HITs when they have no idea what to answer, the quality for answers would be seriously impacted. Besides the function of qualifying "good" workers and filtering out not so good workers, a qualification test could also act as a tutorial that teaches workers how to work on the HITs. When the instructions are in the qualification test, the worker is more likely to read them carefully when they know that they have to give right answers to pass the test. However, besides the benefits of qualifying and teaching, the qualification test may reduce worker responses thus make the task much less efficient. Fortunately with HIT bumping technique or future task search engine, the response rate could be easily raised up as long as the task is meaningful and the wage is fare.

b) Bonus incentives

We there is as less noise as possible in the input answers. One incentive for workers to work seriously is that requesters have the power to reject any work that does not meet the requirements of requesters. If a worker is often rejected, he would not be qualified to a lot of HITs. However, rejecting workers too mush it not good for the requester either. A requester who rejects workers frequently would have a bad reputation and most of the workers would not like to work for that requester. So

rejecting low quality work is not a win-win situation. On the other side, bonus incentives do push the workers to work more seriously and at the same time keep requesters from having a bad reputation. When a worker can earn a considerable amount of bonus if he works just a little harder, he would probably do it. For easy work, the proportion of base reward should be large and for tough HIT the proportion of bonus should be raised and even to the amount that is larger than the base reward. It is because tough work requires more effort, we need a bigger bonus incentive to push worker work harder, and thus work quality would be acceptable.

c) Layered structure

For a complex "engineering task", we cannot show everything to the workers at once. They would not be patient enough to learn that much because they may not dare to take the consequence of spending a lot of time on a long tutorial and qualification test but fail it at last without earning any money. So, the complex task needs to be decomposed to small ones. A small task, or a single layer of a series of task, would be easy to pick-up and it takes a worker less time to learn the new knowledge and become qualified. However, when we decompose the task, we have to think carefully about how many sub-tasks or layers we want to use and how to assign the overall task to these layers. Too many layers might not be a good idea because it's a "serial" process for workers going through the layers. Only after one layer is completed can we issue the next layer.

(3) HIT and Qualification Test Design

In trials and experiments, esp. layer-2 trial and layer-2 experiment, we learnt a lot about HIT and qualification design. It can be summarized to the following points:

a) Balance effort-spend and reward-gained ratio for serious and careless workers

In layer-1 we could design the HIT as image filtering task, in layer-2 we could design the HIT as multiple-selections-permitted questions asking "which elements are visible and damaged?" but we didn't because by doing so the "clicking time" of careless workers could be much lower than serious workers let alone "thinking time". For either image filtering or element visibility question, careless workers could just check one or none of the options and go on to the next HIT while a serious worker have to first think carefully and then move his/her mouse and check multiple options to complete the same question. For the requester who is reviewing the worker's answer, he cannot simply assume that workers who check less options give answers with lower quality, therefore it becomes hard to distinguish careless workers from serious workers and it is even harder to pay fair wages for these workers with mixed working attitudes.

To solve this problem we use two HIT design methodologies to balance the effort-spend and reward-gained ratio of workers, i.e. the relation between how hard and effectively they work and how much they are paid. The first method is trying to lower the percentage of careless workers by question design. The general rule is that clickers and serious workers should spend comparable amount of "clicking time" on the HIT. To carry out this rule, we decompose a multiple-selections-permitted question to multiple

single-selection questions, so all workers have to spend same amount of "clicking time" to go through the test. This is effective in reducing spamming rate because the design "wastes" the clicker's time. The second methodology is to give workers enough incentives to be serious on our HITs. One incentive is moral impetus, i.e. wish to help and work for a good cause and the other incentive is economic driving force, i.e. wish to earn more by working less. Using "help engineers to save lives and aid in reconstruction of Haiti ($0.05 reward + $0.05 bonus = $4.0 per hour wage)" as the title exactly presents these two incentives.

b) Keep the HIT size down

Psychologically, small-size HIT makes it looks easier and friendlier. For the same question type, if there are less questions in a single HIT, a worker can complete the HIT in less time and get each portion of reward in less time (approval may not be in real time, but the worker feels he earns some money every time he completes a HIT). Besides, for the same amount of task, reducing HIT size increases HIT number which helps it rank higher in the most HIT available list on Mechanical Turk.

In layer-2 trial one HIT has 21 questions (5 photos and each with 4 questions on visibility and damage of beams, columns, slabs and walls and a subjective confidence question), with clear but long question options, the HIT looks a little tedious. We infer that too many questions are in a single HIT is one of the reasons for layer-2 trial's low worker response. Re-design of layer-2 keeps only 9 questions (2 photos each with 4 questions and a subjective confidence question) in one HIT. We also shorten the answer options (original answers are in the instruction section as detailed explanation for the options) and design a button to control the display of the instruction section and keep it hide in default mode (this interactive function is realized by Javascript). These three modifications make the layer-2 HIT look much more user-friendly.

Note that we decide to put 2 photos in layer-2 HIT because that would make the total number of questions close to 10. As the difficulty of questions rises in 3 layers, number of question should decrease. There are 11 questions in layer-1, 9 questions in layer-2 and 7 questions in layer-3.

(4) Workers

This section is on the to-do list, key points listed as below:

- Highly educated
- Biased answers towards to check elements as "damaged"
- Professional workers
- Morality driven workers

4 Contributions

This section is on the to-do list, key points listed as below:

Contributions:

- Pilot project in Urgent Human Computing, validating its feasibility
- Understand Mechanical Turk and Turk workers
- Set up a framework for crowdsourcing on Mechanical Turk
- First attempt in open-community crowdsourcing for “engineering task”, insights gained on task and workflow design

What’s new in this project?

- Bonus as an incentive for quality work, Qualification Test for worker filtering -> need more comparative experiments!
- Layered work flow design -> also need more experiments to prove the design is better than another form. (1,2 lead to a problem: how to tune parameters and optimize the system when we are unable to repeat the experiments??)
- *** Group work comparison: experts, closed community, open community. As many aspects as possible. Human computation -> large scale, quick and reliable.
- ** (metaphysical) Explore "crowdsourcing" for Urgent Human Computing for “engineering tasks”. (concepts, why this is feasible and design principals)
- (metaphysical) General model/toolkit/principal for urgent human computing.

5 Future Work

This section is on the to-do list, key points listed as below:

Future work:

- **EFFICIENCY-ECONOMY-QUALITY:** How to coordinate speed, economy and quality? Difference among experts, closed community and open community?
- **RECRUITING:** How to attract workers and reduce response time? Maybe push task news, give fair payment and keep a long-term business relationship?
- **SIMULATION:** How to optimize parameters and validate design ideas? Is human computation simulation possible?

Self Reflections

I am glad to have learnt a lot from this summer program. By doing independent research I experienced how research is like walking in a wild unknown land. The researcher should gain experience by studying other researchers' work and he should do experiments to validate ideas and discover new problems. The obstacles (or challenges) for doing research could be technical issues (knowledge on how to use tools) at first and unexpected problems in the middle and urgent time schedule in the end. What a researcher needs to do is to be perseverant in looking for solutions and never give up easily. In the middle of the program, I am lucky to have the chance to join a symposium held by Computer Science Department. By designing the first research project poster in my life and presenting our work to audience with various background, I learned not only the skills to design a good poster but also the "tricks" of how to attract visitor's eyes and how to effectively communicate ideas to the audience. Writing project abstract and documentation is another valuable experience, which teaches me the importance of thinking in the manner of writing. Not until I write do I really start to think about the goals and innovative points of our project. However, writing is a good documentation or paper is no easy job. It takes time and effort. That is another lesson I learned when I am carrying up with schedule in the last few days at Notre Dame.

In summary, in this program I learned how research is like and what obstacles there could be during a research project. I learned how to summarize a research project into a poster and more importantly how to present it to audience with different backgrounds. At last I learned the significance of writing-as-a-manner-of-thinking and that writing a good documentation is not that easy as I thought.

Acknowledgements

The research presented in this documentation was supported in part by the Notre Dame iSURE program and an award from the National Science Foundation, under Grant No. CBET-0941565 for a project entitled *Open Sourcing the Design of Civil Infrastructure (OSD-CI)*.

I'd like to give special thanks to our group members -- Dr. Madey, Dr. Kijewski-Correa, Dr. Hachen and Zhai Zhi. Thank you all for constructive discussions, insightful suggestions to the project. Thank you for kind help and encouragement to me.

Dr. Madey, thank you for "apprenticing" me to be an independent researcher, maybe I have not yet become a qualified researcher but I am approaching to be one under your guide.

Dr. Kijewski-Correa, thank you for showing us how meaningful our project is and I am honored to be working with such a brilliant professor like you.

Dr. Hachen, you are the first sociology professor I have ever met. Thank you for giving detailed comments and feedbacks when we are designing HITs and qualification tests. Btw, I like the style of your personal webpage and the photos you took!

At last, thank you my dear friend Zhai Zhi. Having a crowdsourcing expert and professional researcher nearby really help me solve handy problems and keep our project moving on.

References

[1] Amazon Mechanical Turk official site

https://www.mturk.com/mturk/welcome

[2] Amazon Mechanical Turk official documentation

HTML version:

http://docs.amazonwebservices.com/AWSMechTurk/latest/AWSMechanicalTurkRequester/Welcome.html

PDF version:

http://aws.amazon.com/documentation/mturk/

[3] Amazon Mechanical Turk official developer forum

https://forums.aws.amazon.com/index.jspa

[4] Amazon Mechanical Turk Sandbox

https://requestersandbox.mturk.com

https://workersandbox.mturk.com

[5] Amazon S3 Management Console

https://console.aws.amazon.com/s3/home

[6] OCD-CI; Open Sourcing the Design of Civil Infrastructure.

http://www.nd.edu/~opence/index.html

[7] OCD-CI Experiment

http://www.crowdstudy.org/login.php

[8] CrowdFlower

http://crowdflower.com

[9] Crowd computing and human computation, a talk by Rob Miller

http://video.mit.edu/watch/crowd-computing-and-human-computation-algorithms-7704/

[10] mturk tracker

http://mturk-tracker.com/general/

[11] SPRUCE; Special PRiority and Urgent Computing Environment. http://spruce.teragrid.org/ , Retrieved Jul. 2012.

[12] Lius von Ahn, L. *Human Computation*. Doctoral Thesis. UMI Order Number: AAI3205378, CMU, 2005.

[13] Z. Zhai, P. Sempolinski, D. Thain, G. R. Madey, D. Wei, and A. Kareem. *Expert-citizen*

engineering: "crowdsourcing" skilled citizens. In DASC, page 879-886. IEEE, 2011.

[14] Z. Zhai, D.Hachen, T. Kijewski-Correa, F. Shen, and G. Madey. *Citizen engineering: Methods for "crowd sourcing" highly trustworthy results*. In Proceedings of the Forty-fifth Hawaii International Conference on System Science (HICSS-45), Maui, HI, USA, Jan. 4-7 2012.

[15] Z. Zhai, T. Kijewski-Correa, D. Hachen, Greg Madey. *Haiti Earthquake Photo Tagging: Lessons on Crowdsourcing In-Depth Image Classifications.*

[16] A. J. Quinn, B. B. Bederson. *Human Computation: A Survey and Taxonomy of a Growing Field.*

[17] A. J. Quinn, B. B. Bederson, T. Yeh, J. Lin. *CrowdFlow: Integrating Machine Learning with Mechanical Turk for Speed-Cost-Quality Flexibility.*

[18] G. Little, L. B. Chilton, M. Goldman, R. C. Miller. *Exploring Iterative and Parallel Human Computation Processes.*

[19] L. B. Chilton, J. J. Horton, R. C. Miller, S. Azenkot. *Task Search in a Human Computation Market.*

[20] Rob Miller. *Crowdsourcing: Research & Best Practice.* Presentation slides.

Appendix I An Example for the ExternalQuestion HIT

A detailed description of ExternalQuestion can be found in the webpage http://docs.amazonwebservices.com/AWSMechTurk/2007-03-12/AWSMechanicalTurkRequester/ApiReference_ExternalQuestionArticle.html.

The best way to learn how to create an ExternalQuestion HIT is trying to do it by yourself. The following are a few example files for creating an ExternalQuestion HIT.

FILES to prepare:

(1) external-question.html (stored on server, url in .question file)

(2) hit.question

(3) hit.properties

(4) hit.input (csv file with delimiter='/t')

1. external-question.html

This file needs to be stored online. For example, it can be stored on Amazon S3 server: https://s3.amazonaws.com/citizen-engineers-hit-layer2/layer2-external-hit.html.

The instructions on how to use Amazon S3 is here: http://docs.amazonwebservices.com/AmazonS3/latest/gsg/GetStartedWithS3.html

Since the HTML file in the link is too large to be included here, we only show the most important part of it in the textbox below.

```
<!-- This file needs to be hosted on an external server.  -->
<html>
<script language="Javascript">
// This method Gets URL Parameters (GUP)
function gup( name )
{
  var regexS = "[\\?&]"+name+"=([^&#]*)";
  var regex = new RegExp( regexS );
  var tmpURL = window.location.href;
  var results = regex.exec( tmpURL );
  if( results == null )
    return "";
  else
    return results[1];
}
</script>
<body><form id="mturk_form" method="POST"
action="http://www.mturk.com/mturk/externalSubmit">
<!-- These are required hidden fields, used by Amazon -->
<input type="hidden" id="assignmentId" name="assignmentId" value="">
```

```
<!-- HIT QUESTION AREA BELOW-->
<img alt="oops..." id="image1" />
<img alt="oops..." id="image2" />
<input type="checkbox" name="input_name">
<!-- HIT QUESTION AREA ABOVE-->
<!-- This is the submit button -->
<input id="submitButton" type="submit" name="Submit" value="Submit">
</form>
<script language="javascript" >
   document.getElementById('assignmentId').value = gup('assignmentId');
   document.getElementById('image1').src="http://www.crowdstudy.org/repository/"+gup("
image1")+".JPG";
   document.getElementById('image2').src="http://www.crowdstudy.org/repository/"+gup("
image2")+".JPG";
</script>
</body>
</html>
```

2. hit.question

```
<?xml version="1.0"?>

<!-- Note the inclusion of the $urls variable which is defined as a field in the input file.
Apache Velocity is the template engine that is

used to perform the merging of variables into template files.  You can learn more about
Velocity's capabilities at http://velocity.apache.org.-->

<ExternalQuestion
xmlns="http://mechanicalturk.amazonaws.com/AWSMechanicalTurkDataSchemas/2006-07-14/Ext
ernalQuestion.xsd">

<ExternalURL>https://s3.amazonaws.com/citizen-engineers-hit-layer2/layer2-external-hit
.html?image1=${image1}&amp;image2=${image2}</ExternalURL>

<FrameHeight>1200</FrameHeight>

</ExternalQuestion>
```

3. hit.properties

```
#####################################

## Basic HIT Properties

#####################################

title:Test - 0003

description:Test - Publishing - Description - 0001

keywords:test publish
```

```
reward:0.01

assignments:3

annotation:${image1},${image2},${image3},${image4},${image5}

#######################################

## HIT Timing Properties

#######################################

# this Assignment Duration value is 60 * 60 = 1 hour

assignmentduration:3660

# this HIT Lifetime value is 60*60*24*7 = 1 week

hitlifetime:604820

# this Auto Approval period is 60*60*24*3 = 3 days

autoapprovaldelay:259210

#######################################

## Qualification Properties

#######################################


# this is a built-in qualification -- user must have > 95% approval rate

qualification.1:00000000000000000L0

qualification.comparator.1:greaterthan

qualification.value.1:95

qualification.private.1:false
```

4. hit.input (csv file)

```
image1   image2

DSC00388 IMG_0482

DSC00482 IMG_0329

IMG_0839 DSC00643
```

When these files are ready (i.e. they are put in the right place, html page on server; input, properties and question file in under same directory), you can use either CLT command or SDK program to load the HIT. For example, use CLT to load the HIT by typing command (filename should contain absolute address which is omitted here):

./loadHIT.sh –input hit.input –question hit.question –properties hit.properties

Appendix II Report on Layer-1 Trial

First Trial on Amazon Mechanical Turk – Summary

By Ruizhongtai Qi (Charles) Email: CharlesQ34@gmail.com

This short report gives a brief summary on the trial. It consists of a Overview part introducing the experiment and 4 parts of demographics, qualification, time and answer quality presenting the analysis results of the experiment. In the last part is conclusion, indicating a few lessons learnt and plan for next move.

Overview

On last Saturday morning (July 21st), we published our first set of HITs to Amazon Mechanical Turk platform. The first trial consists of 50 photos to be tagged as totally collapsed or not. Photos are randomized ordered so photos of the same building have little chance to appear in one HIT. Each HIT involves 10 different photos and has 10 assignments (i.e. 10 different workers would work on this single HIT).

Until Monday morning (July 23rd), all the HITs have been completed. 24 workers took our qualification tests and entry survey and 19 of them passed. 12 qualified workers worked on the HITs and most of them (8->5HITs, 2->4HITs, 2->1HIT) finished all 5 HITs available to an individual.

Analysis Results

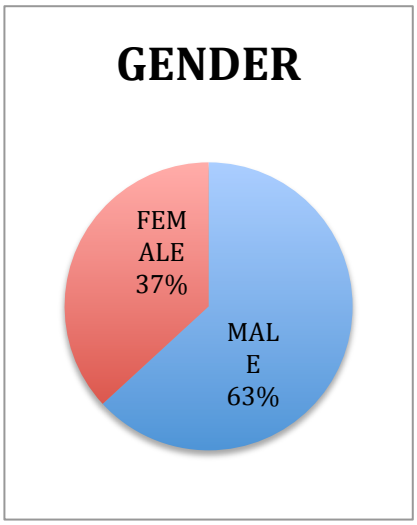
I. Demographics

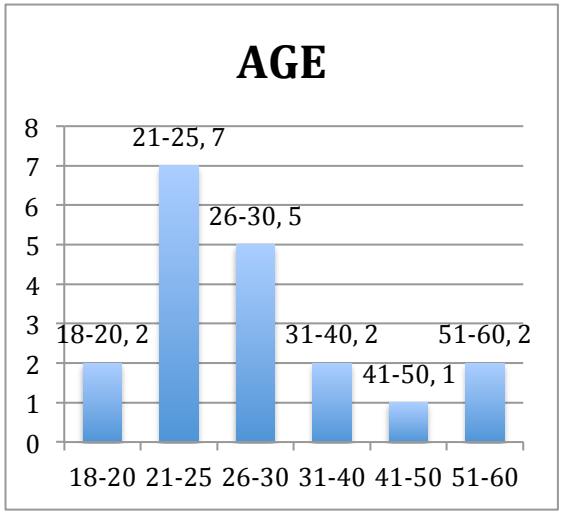
Most of our workers are young (<30) and with good education background (18 out of 19 with college education). As expected, most workers are from India and US and they work for money, killing time, challenge and fun.

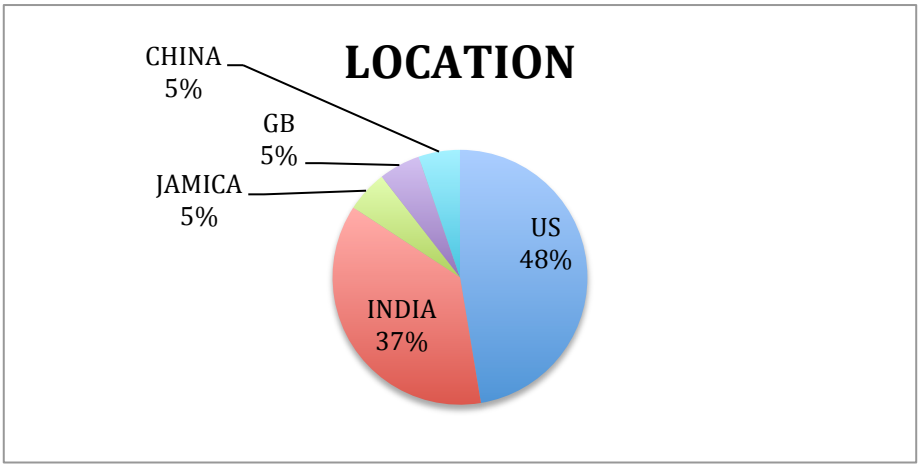
WorkerId	Gender	Age	Location	Education	Motivation
A171B77S5U22B7	male	21-25	CN	somecollege	fun
A2WVRLFUYWHGKA	female	41-50	US	4yearcollege	fun
A2DPIDAK0IZN3E	male	18-20	IN	2yearcollege	killingtime
A185GTV9OKOBE3	female	31-40	US	highschool	money
AHLADH7JTJRO8	male	26-30	IN	4yearcollege	chanllenge
A1SR1SZA08EP2V	female	21-25	IN	4yearcollege	money
A20YQRBJNKS090	male	26-30	IN	advanced	chanllenge
A25OK203N0QYH2	male	26-30	US	advanced	money
A21K2QPSYQH7EN	female	21-25	IN	2yearcollege	chanllenge
A11UWQSS3CS0QB	male	21-25	IN	4yearcollege	money
A2QLSHXNCHBRN4	male	26-30	US	somecollege	money
A1CYTIZPVGCVCZ	male	21-25	JM	4yearcollege	morality
A2VN8Q8X7WWIM	male	51-60	US	4yearcollege	killingtime
A1KFAHW9IOH88O	female	18-20	GB	somecollege	killingtime
A1WWARKKWORAV0	female	51-60	US	2yearcollege	money
A1J5YO3Z66CZVE	male	21-25	US	4yearcollege	money

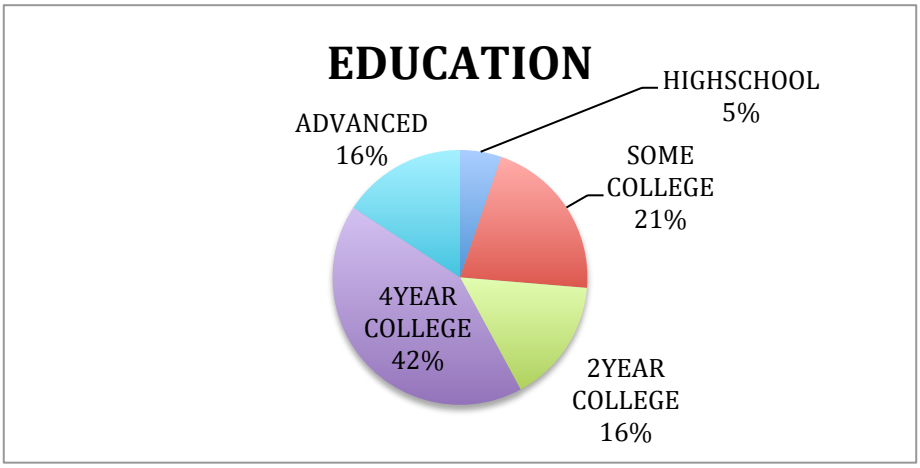
A1WWD77BECLEF6	male	31-40	US	somecollege	money
A3VOV4F5L8131I	female	26-30	IN	advanced	money
A1QAJCJNOQK4PB	male	21-25	US	4yearcollege	money

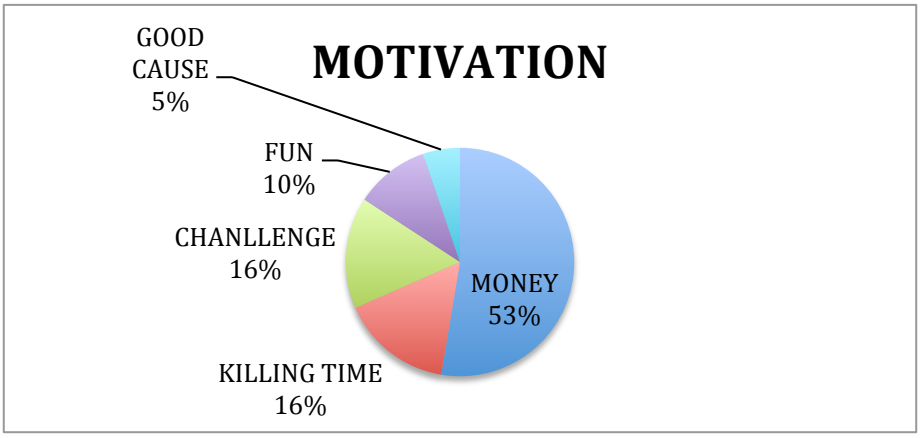
worker information (only those who passed)











II. Qualification

24 workers have taken the qualification test and entry survey. 19 of them passed with all survey quesitions answered and all 4 test questions correct. The other 5 workers failed. 3 out of the 5 failed workers didn't answer the test question 2 (image shown below) correctly. The other 2 failed for that they forgot to or intended not to answer one of the survey questions (education for one worker and location for another).

2. Has the building totally collapsed?

- ○ Yes
- ○ No

Two test takers think it has totally collapsed.

III. Time

(1) When did workers come to our HITs?

Due to the small size of the trial HIT set (only 5 HITs), workers tend to answer all the HITs in the set at once. The time they came to the HITs is as follows,

Jul 21 14:21 GMT *HITs CREATED*

Jul 21 18:30 GMT

Jul 22 16:40 GMT
Jul 22 17:30 GMT
Jul 22 18:40 GMT
Jul 22 18:50 GMT
Jul 22 19:05 GMT
Jul 22 19:40 GMT

Jul 23 02:50 GMT
Jul 23 03:30 GMT
Jul 23 05:00 GMT
Jul 23 06:44 GMT

All HITs Finished
Jul 24 15:21 GMT *HITs EXPIRES*

(2) What is the average time spent per HIT?

What is the average hourly payment?

There are 5 HITs with ten photos in each HIT. Each HIT has 10 assignments to 10 different workers. So we have (5 HITs * 10 assignments/HIT) = 50 assignments in total.

Average Time per Assignment: **1 minute 43 seconds**

Average Hourly Rate: **$1.748**

Actually, some workers spend unordinary length of time on the HIT, for example 18 or 12 minutes. After filtering out that kind of data, we get

Average Time per Assignment: **1 minute 5 seconds**

Average Hourly Rate: **$2.77**

Average Hourly Rate + Bonus: **$2.77*2 = $5.64**

(3) OPEN QUESTION: How to attract workers to our HITs?

Some ideas in my mind:

Means	Pro.	Con.
Advertise HITs on other websites or through mail-list.	Direct and effective esp. for post-disaster response. May be quickly spread by Facebook, twitter or other on-line communities.	Require active on-line volunteer working as "advertiser". Not suitable for ordinary HIT (lack of HIT search or routing market).
"HIT Bumping": keep cancelling	A technical "trick". Effective and	Require comparatively

and publishing the HIT to make it rank high in the newest HIT list.	universal for all HIT type.	complex implementation.
Increase number of available HIT to make it rank high in most HIT available list.	Useful and easy to realize.	Objective requirement: there must be large amount of HITs. Unsuitable for small amount of HITs.
Raise HIT payment.	Um… Money makes the world go round.	"We are short of budget!"

IV. Answer Quality

Our workers are working seriously and no spam worker is detected. By simple voting, all of them gain answer accuracy larger than 90%. Most photos reached consensus where 9 or 10 of 10 workers gave the same answer. Answers with 7 photos (14%) show some ambiguity. 4 of these 7 photos has a 7:3 score and 1 photo has 2:8, 1 photo has 4:6 and another has 5:5.

I listed the ambiguous photos below:

Collapsed: 4, Not Collapsed: 6

Collapsed: 5, Not Collapsed: 5

Collapsed: 7, Not Collapsed: 3

Collapsed: 7, Not Collapsed: 3

CONCLUSION

1. No spams and 90%+ accuracy suggests that our pre-qualifying strategy may have filtered out some unqualified workers. OR, maybe the HIT number is too small for a spam to notice.

2. Entry survey should be separated from qualification test, or a worker has to answer the same questions three times for the three layers. We should do everything we can to make it more convenient for worker to work on our HITs.
3. Some comments that our HIT is "informative" and "amazing" which made us feel relieved ☺ (all comments listed below)

   One comment suggests adding a "not sure" button to the choice, but I think this will make some spam easier, by just clicking on the "not sure" buttons.
4. Unexpected slow HIT-completed-speed pushes us to think and experiment ways to "sell our HITs out".

Next Move:

1. Come up a way to "sell" our HITs faster.
2. Separate survey and qualification test.
3. Move on to experiment in large scale Layer-1 or small scale Layer-2 HITs.

Comments Received

1. it give different angle of views

2. SOME PICTURES ARE CONFUSING

3. Quite Informative

4. amazing

5. good one

6. Very Informative.

7. Interesting survey.

8. interesting topic,with nice picture

9. I think you should have a not sure button as on image 4 there was an image of a concrete/block. i was not sure what that was.....

11. nice task

Trial UPDATES

1) Published the Qualification in the morning. Started the automatic grading Python program.
2) Published the "Building Collapsed Or Not?" HITs (50 photos, 5 HITs, each 10 assignments) in the morning right after the posting of qualification.
3) Until 11:16 PM, there are ONLY 3 qualification takers including me. One set of HITs (i.e. 5

HITs of 50 photos) is answered by a American female worker.

4) Until 4:35 PM, July 22nd, 2012 (1) 15 workers passed the qualification test, and we have new guests from Great Britain and Jamaica! (2) 64% of the HITs have been finished. And, we have received some comments! (3) some comments: "very informative", "nice task", "I think you should have a not sure button as on…", "it give different angles of views"

5) Until 9:56 PM, 68% of the HITs have been finished.

6) Until 11:09 PM, "We are not getting any new guests since early this afternoon. I found our HIT is on page 35 sorting by the newest and page 29 sorting by the most HITs available. So it becomes too hard for workers to find it."

7) Until 9:32 AM, JOB IS DONE!

Appendix III. Report on Layer-2 Trial

Second Trial on Amazon Mechanical Turk – Summary

By Ruizhongtai (Charles) Qi

Email: CharlesQ34@gmail.com

Overview

Layer-2 trial was published on the evening of July 29th, Sunday and it was re-published several times in order to get more worker response. The task is asking workers to identify visible building elements and judge whether a visible element is damaged.

In this trial, we met two problems. The first is how to design questions to make everything clear and to prevent from clickers. The other problem is how to raise worker response to our "engineering task" that requires a qualification. We will discuss them in the "Problems" part.

The trial is closed on Aug 4th with 91% of the task finished. The entry survey is separate from qualification test in Layer-2 and 45 workers including those from layer-1 finished it. 20 workers passed the qualification test which requires 6 correct answers out of 8.

Problems

1. Question Design

The first problem is how to design questions to make everything clear and to prevent from clickers. Since looking for building elements take efforts but judge whether an element is damaged is effortless, we decided to combine the "element visibility" and "element damaged or not" questions into one layer – layer-2.

We are faced with several choices and at last we choose Plan B because Plan A is unreasonable when there are no elements visible we still ask about the elements' damage. We could code Javascript to control the second question in Plan A, but that would make it possible that workers intend to check nothing in the first question and then they don't need to answer the second question. We didn't choose Plan C because of the design cost is larger and identifying beam, column, slab and wall are highly related tasks so it is easier to do them together.

PLAN A

What elements are visible?

A. Beam

B. Column

C. Slab

D. Wall

What elements are damaged?

A. Beam

B. Column

C. Slab

D. Wall

E. None of above

PLAN B

1. Is there any beam visible AND damaged?

A. Yes

B. No, there is at least one beam visible in the photo, but none is damaged.

C. No, there is no beam visible in the photo.

(The same for columns, slabs and walls)

PLAN C

Separate Plan B questions to 4 threads (4 HIT types) i.e. only ask workers to identify one kind of element in a task.

2. Recruiting

The second problem is low worker response. It could be that workers find the qualification test too complex or think the payment is too low.

In order to raise worker response, we did a few things.

At first we try to republish the HIT on July 30th afternoon to make it rank higher in the newest HIT list. It didn't work well since the HIT "sank" quickly in the list.

On July 30th evening, only 3 workers are qualified to our HITs. So we doubled the payment to \$0.10+\$0.05~\$0.15 bonus and lowered the requirement of qualification test, from 8/8 correct answers to 6/8 correct answers. We also changed the title to emphasize the raised wage by noting the task provided \$4.0 per hour wage.

By Aug 1st, only 7 out of 15 qualification test takers passed. We raised the wage again to \$0.20 base payment + \$0.05~%0.10 bonus. After that, the number of worker slowly grew and until Aug 2nd 10:11AM 48% of the HITs have been finished and until Aug 4th, 20 workers passed and 91%

of the task finished.

"Engineering task" is not easy. The "information gain" for "engineering task" is larger than most other tasks asking workers to use their prior knowledge or common senses. So turk workers are more reluctant to participate.

To tackle with the problem, we have an "emergency" solution and a long-term solution. The "emergency" solution is to use the HIT bumping technique, where we use programs to update HITs periodically in order to make it rank higher in the newest HIT list. The long-term one is to "reserve" a group of workers who are willing to learn new things and help us in post-disaster response.

(To realize HIT bumping, I updated the crowdsourcing system. It's now more flexible and programmable.)

Work Quality

Compared with a civil engineering graduate student's answers as ground truth, turk workers have a 71% accuracy on layer-2 questions (visibility of elements and whether a type of element is damaged). The worker answer is derived from simple voting.