CS 186 Assignment 7:
Programming The Crowd

Professor David C. Parkes
School of Engineering and Applied Sciences, Harvard University
Out Monday April 6, 2015
Due **5pm** sharp: **Tuesday April 14, 2015**
(Extension School: Wednesday April 15)

Total points: 70. Your goal is to use the TurkIt library in order to write tasks that will be executed on Amazon's Mechanical Turk (AMT). This is a group assignment to be completed by groups of **up to 2 students** (single-person for extension school). While you are permitted to discuss your designs with other students, each group must write their own code and explanations. If you want a partner and don't have one, post to Piazza as early as possible.

Submission Instructions:

1. Your submissions should be made to the assignment Dropbox on iSites, as a zip file.
2. Make sure to put the names of your team members in your write-up
3. Include in the zip file your write-up, primer.html, cs186-mturk-intro.js, cs186-mturk-sandbox.js, cs186-mturk-nuclei.js, cs186-mturk-sort.js, and other supporting files.
4. You will get points for completing the exercises, and **should present evidence of your work on AMT such as screenshots, .csv log files, worker/job IDs.**
5. Please start early on this assignment. The approval process for an account on Amazon Mechanical Turk might take up to 48 hours.

Amazon Mechanical Turk Norms: When you post tasks on Turk you have the option to review and reject responses from workers. Please accept all tasks completed in a reasonable way. [There is one exception in the nuclei counting experiment.] Otherwise, this will affect your reputation as a requester and possibly other fellow students who will be running similar experiments.

1. **[10 Points] Introduction**

   Your goal in this part of the exercise is to learn about AMT and the TurkIt framework and get acquainted with the basic terminology (worker, requester, HIT, etc)

   (a) **[4 Points]** Read about AMT from sources of your choice including Amazon's website. Why was that particular name chosen? Comment in a few sentences, on the slogan "Artificial Artificial Intelligence."

(b) **[3 Points]** Watch the demo video on TurkIt here `http://groups.csail.mit.edu/uid/turkit/` What did the last task involve and how was it accomplished?

(c) **[3 Points]** Report on a recent article in the news about AMT that you find interesting and briefly comment on it.

2. **[6 Points] Javascript primer**

Javascript is the scripting language that supports the vast majority of modern dynamic web applications (Web 2.0+). When a page loads, Javascript code works in the background, pushes and polls data from the server, and updates parts of the webpage, thus giving the impression of a normal desktop application.

There are numerous resources for JavaScript on the Web. A nice e-book can be found here `http://eloquentjavascript.net/`. A nice online IDE where you can run and test code can be found here `http://jsfiddle.net/` (not necessary for this assignment).

(a) **[2 Points]** Open the `primer.html` attached with this code package with a text editor. You will see the `<script..></script>` where Javascript code is located. Read and understand what the code is doing and then open the file with a browser. As an exercise rewrite Line 30 (`obj.add = X`) by replacing X with only 4 characters such that the code maintains the same functionality.

(b) **[4 Points]** Create an object `Algebra` with the following:
(i) a property `operation` equal to the string "multiplication"
(ii) a property `mul` that is a function to compute the product of its two arguments, i.e. `Algebra.mul(a,b)` calculates `a.b`
(iii) a property `genprod` that is a function accepting two arguments: n and f, where n is a positive integer and f is a function `Algebra.genprod(n, f)` computes the value: `f(1)` $\cdot$ `f(2)` $\cdots$ `f(n)`
(iv) What does `Algebra.genprod(n, function(x) { return x})` calculate?

3. **[8 Points] Login/Setup**

In this section you create an Amazon Web Services (AWS) account, which is necessary to work with the TurkIt platform, and an AMT account.

(a) **[1 Points]** Go to `http://aws.amazon.com/` and create an account (if you don't have one already). Also make a worker and requester account at AMT's home page here `https://www.mturk.com/mturk/welcome`

(b) **[2 Points]** Work for 20mins on AMT and make as much money as you can. Share your experience by providing brief details and evidence of your work. [**Note:** Some high-reward HITs require disclosure of private information like taxes or medical records. Avoid such tasks.]

(c) **[2 Points]** Download the TurkIt framework from `http://groups.csail.mit.edu/uid/turkit/` . It is a .jar file which should be easy to run once you have Java installed. Then watch the TurKit video again, crucially the part where the programmer sets the account credentials (00:27-00:37) and sets TurkIt "Properties." Follow the exact same steps but using your own AWS access key.

(d) **[1 Points]** Your master object in a TurkIt application is `mturk`. This is your main interface to TurkIt functionality. Review the functions associated with this object here: `http://groups.csail.mit.edu/uid/turkit/jsdocs/symbols/MTurk.html`

(e) **[2 Points]** Load $7 to your account through the VISA number that will be provided over email. As a next step, open the `cs186-mturk-intro.js` and fill in the missing code that prints the cash balance of your account (remember to set the mode from "sandbox" to "real" according to the TurkIt video).

4. **[10 Points] The "Sandbox"**

The "Sandbox" is a useful tool when developing AMT applications. The url is here `https://requester.mturk.com/developer/sandbox`. You can use the sandbox from TurkIt by selecting it as one of the choices in TurkIt just above the editor. As a first HIT, you will ask workers to pick a number from 1 to 10, first debugging in the Sandbox.

(a) **[4 Points]** Open the `cs186-mturk-sandbox` file. Complete it to create the "Pick-a-number" HIT, choose Sandbox and then "Run." TurkIt will give you the link where the HIT is posted. Click on it and attach a snapshot of it. Here is our HIT for reference:

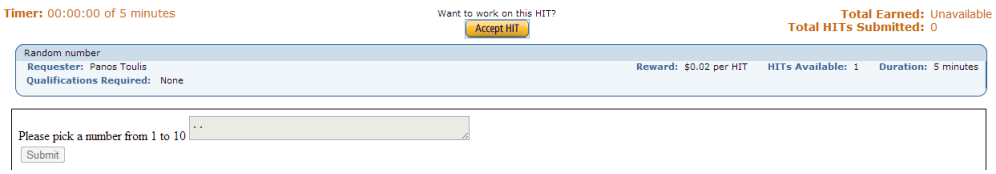

Figure 1: A Posted task in the Sandbox

[**Note:** If you are having trouble viewing your HIT content in your browser, please try another one. HIT cannot display properly in some browsers, e.g. Chrome and Firefox, due to security settings.]

(b) **[6 Points]** As a "Requester" you can monitor your HITs. Here is the link: At the "Manage" tab (`https://requester.mturk.com/manage`), click on the "Manage HITs individually." There you can see the open HITs, review submissions from workers and agree on payments. In this task, ask 5 workers to pick a random number. You will need to run in "Real" mode and attach a snapshot)e.g., see the one below.) Also don't forget to review the submissions and pay your workers. Write down the answers and comment on how "random" the choices were. Also write down how long it took to get all 5 answers and comment. A suitable payment for a simple HIT like this one is probably around 2 cents.

Figure 2: Posted tasks in the "Manage" console

5. **[16 Points] Nuclei Counting**

Some tasks are computationally hard but easy for people. An example of such a task is image analysis for biological studies. In this exercise, your task is to use AMT to count nuclei in an image depicting many different cells. The code resides in `cs186-mturk-nuclei.js`, and includes a link to the image. When ready, it will first create a HIT that will post an image and ask a worker to count the number of nuclei.

When a response arrives, you will run the same script again to process the results. There are two ways to do this: (i) Run the script once, check periodically your MTurk Requester home page and run the script once again when there is an answer, and (ii) Run the script in "Run Repeatedly"-mode which will get the answer soon after it arrives without having to check the MTurk website yourself. *We recommend (i) because we found (ii) to be buggy in our runs but feel free to experiment!*

(a) **[1 Points]** Read the `createHIT(.)` function from the TurKit API. What is the object used as an argument? List the parameters and explain what they do.

(b) **[3 Points]** Open the source file `cs186-mturk-nuclei.js` in the TurkIt editor. Fill in the code in the function `createNucleiHIT(argCost)` to create the nuclei counting task (the `argCost` is the money you are willing to spend for this HIT).

– Do not run your HIT yet!

(c) **[5 Points]** It is usually the case that HIT responses are of low quality, especially for tasks that do not require worker qualifications. This can be mitigated using the TurkIt `mturk.vote()` function, which allows a requester to ask for other workers to vote on the quality of the response. If the majority vote is positive you should accept the output of the count HIT, and otherwise you should reject it (in this particular case you would end-up rejecting a HIT.)

In the same source file, fill in the appropriate code in function `vote(argNucleiCount, argVoteCost)`. (Note that `argNucleiCount` is the number of nuclei given in the response to the nuclei-counting HIT.)

(d) **[3 Points]** Read the `vote(.)` function, use `maxAssignments` to define the minimum number of votes necessary for a single choice, and determine the payments for the nuclei-counting HIT and the voting HITs.

Make sure everything is fine and press "Run"!

– Save at least 70% of your budget for the rest of the assignment!

[**Note:** After the counting HIT is created, TurKit will exit, waiting for an answer. As long as there is no answer for this HIT, the voting HITs **will not** be created, even if you click the "Run" button again. Only one voting HIT will be created but with multiple assignments.]

(e) **[4 Points]** Visit your AMT homepage as Requester, and wait until a response is submitted. When this happens, run `cs186-mturk-intro.js` to access the HIT response(s) and check the status of your HITs.

What did you find? Explain what you see, how much you spent, the count obtained, the votes, and give snapshots for your HITs.

6. **[20 Points] Image Sorting**

Now that you're familiar with TurkIt and AMT, the goal in this exercise is to use AMT to sort 8 images of Harvard year into temporal order (earliest in time first). The goal is to get a reliably high quality sort for the budget available. There is no need to save any budget.

This exercise is deliberately open-ended and you can be imaginative in your design. To get a good quality output, you will need to use multiple workers and have a clever way to process responses. However, more workers will cost you more, and you need to stay on budget. Before you start, you will need to think about how many image comparisons will be needed, how many workers to request per comparison, and how much to pay.

The file `cs186-mturk-sort-images.js` contains code that places one image next to another image and asks a question of a worker. The code also includes the images themselves as `imgur.com` links.

You can get 15 points for completing the assignment based on the code provided, and an extra 5 points for implementing an an extension to the design. For example, it might be more cost-effective to use a different HIT design; e.g., you might want to place more than 2 images on the same page, have different sizes, or different HTML inputs, etc.

Clearly describe your approach by including:

(a) The HTML code that was shown to workers (with a snapshot)

(b) The payments and/or voting mechanisms you had in place

(c) The submissions from workers (with a few snapshots)

(d) The final ordering obtained by your human computation algorithm, and some of the intermediate algorithmic steps in obtaining the ordering.

(e) Any experiments you performed to pick the optimal design.

Remember to provide evidence of work on AMT in order to receive full credit. But be reasonable in deciding how much information to include. We're looking for enough to understand your approach, understand what it did on this input, and why you ended up with this design. For this you can use a few snapshots in various places. But it might not be reasonable to provide a full documentation.

[**Note:** No prior knowledge on the order of images should be used in your design!]