A User Friendly Crowdsourcing Task Manager

Tomas Matera Jan Jakes Munan Cheng Serge Belongie

Department of Computer Science and Engineering
University of California, San Diego

Cornell NYC Tech
Cornell University

{tmatera,jajakes,mucheng}@cs.ucsd.edu

1. Introduction

Data collection via crowdsourcing often entails creating a user interface, a server-side system to store the collected information, data quality assurance algorithms and other repetitive, time-consuming tasks. Although there are some crowdsourcing web services that provide APIs such as Amazon Mechanical Turk (MTurk)[1], a lot of work still has to be done to deploy a customized Human Intelligence Task (HIT) and harvest the data.

We present a crowdsourcing task manager that allows task requesters to deploy such tasks with just a few clicks, share the task easily with their workers, deploy it to third-party crowdsourcing services, automatically collect additional data about the activity of the workers and perform quality assurance on the data. We demonstrate the capabilities of the system using an example implementation of a worker template for assessment of image similarity.

2. Related Work

Some crowdsourcing tools have been developed to work with Amazon MTurk API such as TurKit [2], a tool providing a programming API to run iterative tasks on MTurk. TurKit allows programmers to include HITs as a part of their algorithm and thus iteratively ask wokers to work on some data. A similar tool called Turkomatic [1] focuses on running recursive tasks on MTurk, automates the workflow of complex tasks and lets the workers subdivide the task into smaller units. Another toolbox called LabelMe [3] was developed especially for labeling objects in images. Currently it provides a web service, an iPhone application and Matlab bindings.

In contrast to the above examples, our tool does not require downloading, setup or programming to be used since it is built as a cloud service. Also, it does not focus on a specific problem but intends to be highly general.

[1] http://aws.amazon.com/mturk/

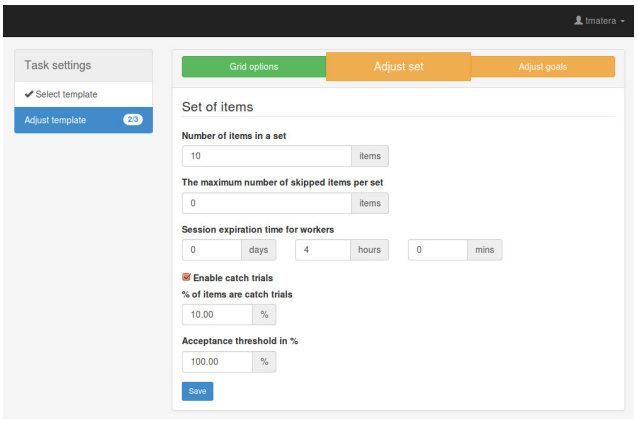

Figure 1. Screenshot of the requester environment.

3. System Architecture

The system is built as a user-friendly Software as a Service (SaaS) web application, running on top of Amazon Web Services (AWS). It exposes two separate user environments – a *task requester* interface to set up, manage and monitor HITs (see Fig. 1), and a *worker* environment for the actual data collection (see Fig. 2). The system cooperates with other modules that take care of user authorization, resource management and credentials storage for third-party crowdsourcing services in a secure way. In the remainder of this section we briefly describe each of the key components of the system.

3.1. Screen

A *screen* is a basic unit that exposes concrete data to the worker and asks her to perform an action. Such action can consist of anything that can be done in a web browser and sent back to the server. The actual form of the action is defined by a template (Sec. 3.3), the screens are grouped into a set (Sec. 3.2), some of them can have a special role (Sec. 3.4) and the time a worker spends on each screen is monitored by a timer (Sec. 3.5).

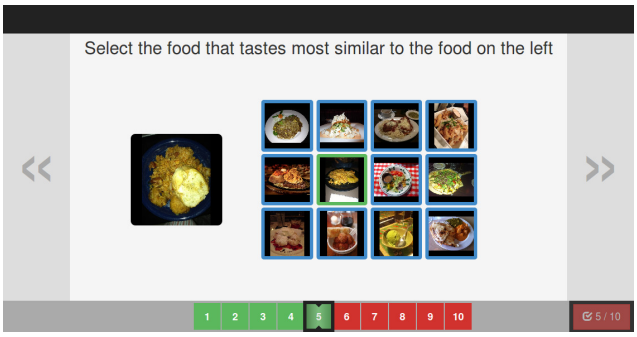

Figure 2. Screenshot of the worker environment.

3.2. Set of Screens

A *set of screens* is the smallest unit of work a worker must do to have his answers submitted. Each screen in the set requires the worker to perform the same action but on a different data. The total number of screens in the set can be configured by the task requester. Each set can contain a fixed number of special screens that serve to evaluate the quality of the collected data. This functionality is further explained in Sec. 3.4.

When deploying the crowdsourcing task to third-party services such as MTurk, a set of screens will be also the smallest unit for which a worker can be paid. The current capabilities of the system are presented in Sec. 4.

3.3. Templates

A *template* defines the specific action that the worker is requested to do and determines the appearance of the screen. In the case of collecting image similarities the action is implemented as a single or multiple item selection where each of the displayed items can be enlarged to inspect the details before answering.

From the internal perspective, the template also defines the type of data the tasks entail, which items will be shown on the screens and the possible mechanisms for quality assurance.

3.4. Quality Assurance

Quality assurance is an essential part of crowdsourcing since some of the workers may be unqualified or lack motivation to provide correct responses. Fine-tuned settings on third-party crowdsourcing services and worker redundancy can address this problem to some extent, but in practice a lot of wrong data can be still collected.

Our proposed system incorporates active detection of incompetent workers through the use of *catch trials* (also known as *gold data*), i.e., requester-prepared screens with known, unambiguous solutions that can be added to each screen set as desired. If a worker submits a wrong solution the requester can discard the entire screen to keep out bad data.

3.5. Timer

The time spent by the worker on a screen can reflect the difficulty of a task and thereby provide some information about the quality of the answer. Our system measures and stores the times for each worker on every screen and the interpretation of this information is up to the requester.

To achieve maximum precision the timers run on the client side and monitor worker activity. If the user is inactive or leaves the screen, the timer pauses and reverts back to the last activity. On any activity the timer is started again and continues from the last activity.

3.6. Goals

The task can be automatically terminated after reaching desired goals. Currently the task can be limited in time, by the number of submitted HITs or a combination of the two.

4. Third-party Platform Integration

To spread the task among the workers the system incorporates a *workforce* section that automates publishing tasks onto third-party platforms. Currently, we support deployment of the tasks on Amazon MTurk. To simplify result aggregation and analysis, the actual answers are stored in our system together with those submitted by workers from other sources. The requester can set up the automatic approval delay and choose to reject submissions that do not meet the quality standard.

5. Future Work

To turn our system into a truly general crowdsourcing tool, we plan to implement and test additional task templates and automate more steps of the data collection pipeline. We also intend to integrate with other third-party crowdsourcing services to allow requesters to benefit from alternatives to the very popular MTurk platform.

References

[1] A. Kulkarni, M. Can, and B. Hartmann. Turkomatic: Automatic Recursive Task and Workflow Design for Mechanical Turk, 2011. 1

[2] G. Little, L. B. Chilton, M. Goldman, and R. C. Miller. TurKit: Human Computation Algorithms on Mechanical Turk, 2014. UIST 2010. 1

[3] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. LabelMe: a Database and Web-based Tool for Image Annotation. *International Journal of Computer Vision*, 77(1–3):157–173, 2008. 1