CrowdWeaver: Visually Managing Complex Crowd Work

Aniket Kittur, Susheel Khamkar, Paul André, Robert E. Kraut
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA, 15213
{nkittur, pandre, kraut}@cs.cmu.edu, susheelkhamkar2@gmail.com

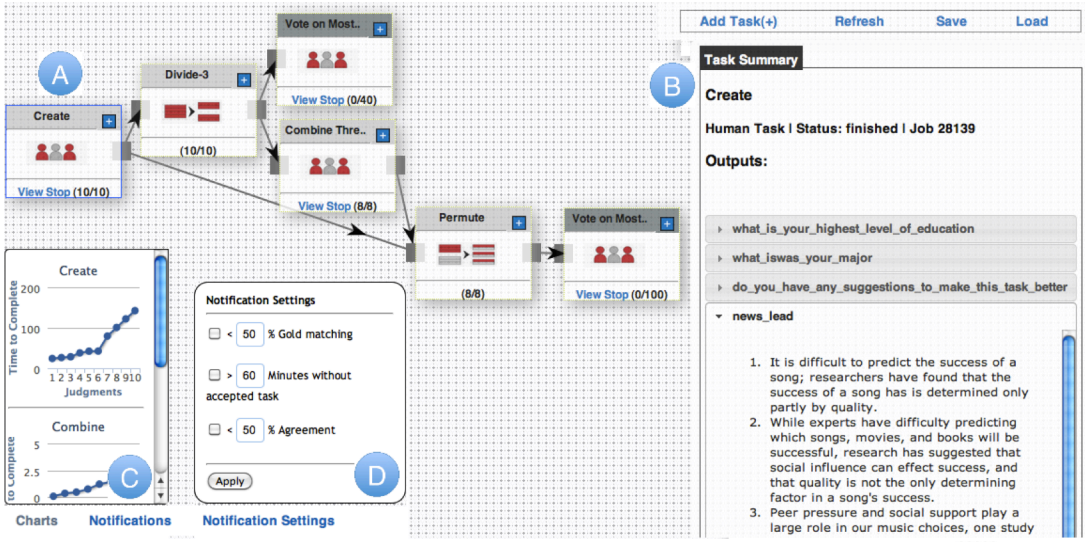

Figure 1. The CrowdWeaver workflow management interface. (A) The workflow consisting of human tasks ( ), e.g., "create (news leads)", and machine tasks (e.g., divide, permute). (B) The Task Summary pane details the selected task, with the "news lead" field expanded showing worker outputs. (C) Graphs to track 'crowd factors' such as arrival of workers to a task. (D) Settings to alert the requester if crowd factors such as uptake, quality or time to complete become cause for concern, allowing the requester to edit task design or instructions in real-time to optimize time and quality of results.

ABSTRACT

Though toolkits exist to create complex crowdsourced workflows, there is limited support for management of those workflows. Managing crowd workers and tasks requires significant iteration and experimentation on task instructions, rewards, and flows. We present CrowdWeaver, a system to visually manage complex crowd work. The system supports the creation and reuse of crowdsourcing and computational tasks into integrated task flows, manages the flow of data between tasks, and allows tracking and notification of task progress, with support for real-time modification. We describe the system and demonstrate its utility through case studies and user feedback.

Author Keywords

Crowdsourcing, workflow, visualization, coordination.

ACM Classification Keywords

H5.m. Information interfaces and presentation (e.g., HCI).

General Terms

Design

CSCW'12, February 11–15, 2012, Seattle, Washington.

INTRODUCTION

Crowdsourcing has become a powerful mechanism for online production, but is currently mostly used for simple, independent tasks. Recently, a number of toolkits have been created to programmatically support more complex tasks and workflows [4,5,6]. However, though a requester of work now has advanced tools and algorithms to *create* workflows, there remains limited support for *management* of those workflows. Managing workflows can be as or even more challenging than their creation: requesters must experiment and iterate with different tasks, instructions, rewards, and flows to optimize the process. For example, in interviews we conducted with employees of CrowdFlower (a commercial crowdsourcing platform), they noted that when working with a crowd "*a major issue is stringing tasks together.*" To achieve successful output requires a "*significant cost in time, [and involves experimentation with] what's the right [task] that gets to the most effective results most quickly and cheaply.*"

This discussion points to the unique challenges posed by managing crowdsourced workflows. In other forms of computational task flow management (e.g., visual programming languages [2], online mashups [7]) experimentation and execution is essentially free, instantaneous, and deterministic. In contrast, each run of a crowdsourced workflow requires an investment of time and

money, and may give different results each time. This suggests that crowd-based computational task flow systems should not only manage tasks but also address crowd-specific factors such as: *latency*, the delay between requesting and commencing work; *price*, understanding what price is appropriate, and how varying prices may affect work [1]; the varying *quality* of worker and output; and the varying *time* to completion of a unit of work.

To address these challenges we present CrowdWeaver, a system to visually manage complex crowdsourcing tasks, with no need for programming knowledge. Our contributions include:

- A visual interface for task and workflow creation and monitoring that serves as an external mental model of a task flow;
- The management and reuse of templates, including both human and machine tasks;
- Tracking and notification of crowd factors (e.g., latency, price, time and quality);
- Support for real-time experimentation.

In the remainder of this paper we provide a brief overview of the architecture of the system, illustrate its features through an example task flow, and discuss feedback from task designers using the system.

THE CROWDWEAVER SYSTEM

CrowdWeaver is built on top of CrowdFlower, a commercial crowdsourcing platform that provides certain advantages over platforms such as Mechanical Turk, including the ability to post tasks on multiple markets (including MTurk), a rich API, and a robust template editor. The CrowdWeaver system consists of a MySQL database that stores information about the tasks, data, and connections between tasks; a JQuery-based web visual interface with which users view, create and manage tasks; and a Ruby controller that powers the interface, manages the database, and synchronizes with CrowdFlower.

Scenario

Imagine a task designer is trying to generate a news lead for a newspaper article as part of a larger task crowdsourcing the writing of the article. She wants to determine whether voting to select the best lead or merging leads together is a better task design. In one method, she has workers each generate a news lead, splits them into groups of three (to simplify the voting), and has workers vote on the best. In another method, she takes the same generated leads, splits them into groups of three, and has workers generate a new lead that merges the best aspects of the three they are given. Finally, she compares each combination of original and merged news lead and has workers vote on which is better.

Throughout the process, she wishes to monitor the workflow and be alerted if there are problems such as poor uptake or lack of agreement between workers. If so, she may step in and review the work to understand where the problem lies, and modify the task design or instructions. (This task is similar to that used in Kittur et al. [4].)

Figure 1-A presents this task flow as created and managed by CrowdWeaver, with the top path corresponding to the voting method and the bottom path the merging method, including permutation with the original leads and voting. The interface integrates both crowdsourced and machine tasks, and reflects all key data collection and manipulation steps, including the flow of data between tasks. Monitoring is achieved through real-time update of the number of completed assignments, and details of the task including outputs are shown in the Task Summary pane (Figure 1-B). Since manually keeping track of every detail could be overwhelming, notifications can be triggered (Figure 1-C) when crowd factors such as latency or worker agreement generate cause for concern. We discuss CrowdWeaver's core functionality and design in more detail below.

Task and Workflow Creation

Visual Representation

A key challenge is choosing how to visually represent a task, data fields, inputs, and worker outputs. For example, a simple image labeling task could include multiple data fields (e.g., each label, demographics), thousands of input images, and tens of thousands of worker outputs. Complex flows may involve many such tasks, each with potentially thousands of interconnected inputs and outputs. With workflows such as these, approaches which visually represent each input and output (e.g., [5]) can quickly get overwhelming. Instead, we chose a higher-level representation, visually depicting each task and the data flow between them, with details about inputs and outputs available on demand in a dynamic pane (Figure 1-B).

Another important consideration was to enable the interface to act as a user's external mental representation of the workflow. To support this, users can alter the spatial arrangement of tasks to match their mental model of the workflow, with spatial positions saved across sessions, enabling the construction of a persistent task landscape.

Creating and Reusing Templates

One of the most significant costs in time, effort, and money for crowdsourcing complex tasks is in experimenting and iterating with the instructions and materials. Even small changes in instructions can lead to dramatic differences in output quality [3]. To help with this costly process of iteration, CrowdWeaver saves each human task created and allows the user to create and modify (via the CrowdFlower editor) new instances of it for future tasks. Properties such as instructions and HTML, answer fields, pricing, market choice, and worker qualifications are inherited from the original template, while new input streams can be selected. A basic set of common human tasks are included as starter templates, including generate, vote, and merge tasks.

In addition to crowdsourced tasks completed by humans, CrowdWeaver supports tasks performed by machines, such as data manipulation, which use the same data structure

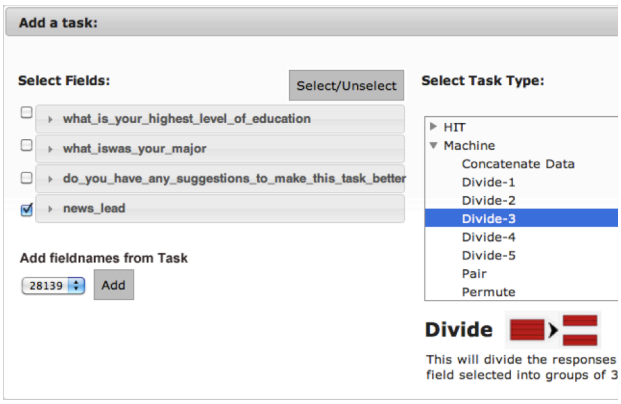

Figure 2. The Add Task screen, with the "news_lead" data field from the previous task selected as input, and a machine "divide" task chosen.

(data fields, inputs and outputs). A basic set of machine tasks are included, such as divide, concatenate, pair, and permute; users can add new machine tasks by adding a wrapper function within the controller code, and a class to implement the new machine task.

Data Flow Between Tasks

CrowdWeaver uses a dataflow paradigm [2] supporting the visualization and flow of data between tasks. For example, in our scenario, the designer wishes to determine whether voting on news leads or merging them together is a better workflow. To create the voting flow (top path in Figure 1), the designer passes the news leads generated by workers in the original task to new *divide* and *vote* steps. She wants to group the data into sets of three, so she selects the "news lead" data field and the *divide* task. For convenience, descriptive text and images of machine tasks are shown in a preview when selected (see Figure 2). The designer creates the *vote* task in a similar way, passing through data from *divide* and repurposing the generic *vote* template.

The system also supports branching and combining multiple data flows. As illustrated in the scenario, the designer creates the alternate branch in which workers merge the best aspects of three news leads and vote on the merged version (bottom path in Figure 1). These results are compared to the originally generated leads by combining the original and alternate data flow branches (specifically, by adding additional fields from the original task; see bottom left of Figure 2).

Tracking and Notification of Task Progress

As identified in the Introduction, crowd work poses challenges in terms of latency, price, time, and quality. Rather than wait for a task to complete (or never attract enough workers) to determine if there is a problem, CrowdWeaver offers a task progress system to easily view and be alerted of common signifiers of task problems.

Monitoring

As data comes in, summary statuses are viewable at a glance. The bottom of each task shows the number of outputs collected, and graphs updated in real-time (Figure 1-C) help monitor crowd factors. To drill down into the data, one can expand the Task Summary pane (Figure 1-B). This makes it easy to trace data across tasks and to compare different task versions as all data is immediately available from the main interface.

Notifications

Figure 1-D shows the current notifications available in CrowdWeaver. In terms of latency, if a task does not attract new workers for a period of time, it may be that the instructions are unclear or unattractive, or the task is incorrectly priced [1]. Identifying this early could save hours or days waiting for workers. Similarly, quality can either be compared against a gold standard (as CrowdFlower currently does) or inferred from the percentage of workers whose answers agree with each other. In cases where a problem occurs in a task, the requester can stop a poorly functioning flow and create a new branch to fix the problem while preserving their existing work.

SYSTEM USAGE

We have experimented with creating and managing a variety of tasks using CrowdWeaver, including product research, article generation, and science journalism. Furthermore, functionality from crowdsourcing frameworks such as CrowdForge [4] including partitioning, map, and reduce tasks can be replicated in CrowdWeaver. For example, Figure 3 shows a flow involving two *create* tasks (generating eReader models and features to evaluate them); a *permute* task creating all combinations of models and features; a *find information* task gathering evidence about each combination; and a *divide* task splitting the data into groups relevant to each model (in this example, two).

Interviews and Usage

To gather initial feedback about the usefulness of CrowdWeaver for complex workflow management, and reaction to and usage of the tool in a workflow creation task, we conducted two interviews and recruited two participants for a formative study.

In interviews with two CrowdFlower employees who create workflows for corporate clients, the general approach of CrowdWeaver for dealing with time-consuming workflow creation and management was appreciated ("*awesome concept!*"). The employees were particularly enthusiastic about the potential to "*string jobs together, visualize all running jobs, and [see] stats on a given workflow.*"

To gather feedback about the usefulness of CrowdWeaver, we recruited two participants (graduate students) familiar with designing tasks for MTurk but with no prior exposure to CrowdWeaver. Participants were assigned the example problem described earlier: given a set of news leads, decide whether a vote or a merge task resulted in better outcomes. They were asked to draw a flow indicating how they would complete the task using MTurk and their current tools. They were also given a 10 minute tutorial on CrowdWeaver's

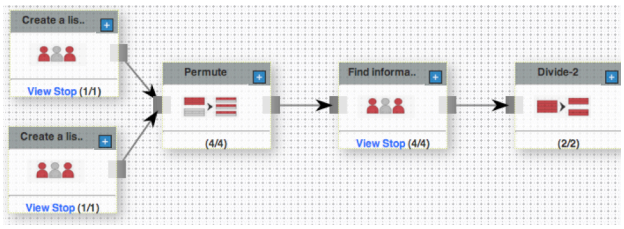

Figure 3. Task flow for researching an eReader.

functions, then asked to create the flows in CrowdWeaver, and finally to fill out a questionnaire.

In the drawing task, both sketched flows very similar to Figure 1, although their projected data manipulation steps were done either through scripting (e.g., Python) or by hand (copy-paste). Both participants were able to generate the entire working flow, including splitting and merging tasks with multiple connections. Participants rated the usefulness of CrowdWeaver very highly, with answers between 6 and 7 on a 7-point Likert scale for "viewing and managing tasks visually was helpful", "connecting tasks to each other was helpful", "I enjoyed using CrowdWeaver", and "I would like to use software like CrowdWeaver in the future". One participant noted that it "*fit [his] intuition about workflows nicely*", and that "*the visual component helps me model what I'm doing with workers and data*", suggesting that it did support his mental model of the task flow.

DISCUSSION AND FUTURE WORK

One of our design goals was to visualize task flows such that they provided useful externalized mental representations. To do so we chose to represent tasks at the level of the subtask rather than visualizing each individual workers' output. However, an important lesson we learned is that even this level of representation may be too low level for many users, suggesting an opportunity for future research to better visually represent multiple levels of detail (i.e., task, data fields, inputs/outputs). A related challenge is that both participants found it difficult to adapt to the CrowdFlower nomenclature, editing interface, and task structure, giving CrowdWeaver relatively low scores on ease of use (mean = 4.5 on a 7-point Likert scale) and suggesting that making a visual task management system such as CrowdWeaver easier to use for those less experienced with existing crowdsourcing paradigms may be a valuable area for future research.

We further investigated these issues with employees of CrowdFlower who were responsible for managing the design of tasks for corporate clients but not actually programming the tasks themselves. We found that for these users, the level of abstraction presented in CrowdWeaver was lower-level than they desired. While our system enables the user to stay "close to the data", these employees thought about flows in higher-level steps such as "generate", "vote", etc., without the intermediate linking data manipulation tasks.

This suggests that instead of the visualization of individual workers' outputs or even of subtasks, a fruitful area of research may be how to enable the aggregation and visualization of subtasks as meaningful patterns of work (e.g., combining divide, vote, match and filter into a single useful pattern for interacting with the crowd). Such an approach could be instantiated as a "simple" interface for CrowdWeaver in which users create and manage entire subflows, with data manipulation steps (such as splitting the data into threes) surfaced as user-configurable parameters. Supporting interaction at both the pattern level and the task level could be helpful for more advanced users who need greater customization, and enable sharing and customization of useful patterns.

CONCLUSION

We present CrowdWeaver, a system for visually creating and managing crowd workflows. CrowdWeaver functions as a "mental model" for the task designer, integrates human and machine tasks, supports template reuse, manages dataflow, and does not require programming knowledge. In particular we address unique factors of crowd work, allowing monitoring and alerting based on task progress in terms of: latency, price, time to completion and quality.

CrowdWeaver can benefit task designers through easier iteration and experimentation for complex flows, leading to time and cost savings. We describe how the system can be used for a variety of tasks, gather feedback from participants using CrowdWeaver, and discuss future opportunities for research on the visual management of crowdsourcing flows, notably in the level of representation, and aggregating subtasks into reusable patterns of work.

ACKNOWLEDGMENTS

This research was supported by NSF grants OCI-09-43148, IIS-0968484, IIS-1111124 and a grant from Carnegie Mellon's Center for the Future of Work.

REFERENCES

1. Faridani, S., Hartmann, B., & Ipeirotis, P. G. What's the Right Price? Pricing Tasks for Finishing on Time. *Proc. HCOMP 2011.*
2. Johnston, W.M., Hanna, J.R., & Millar, R.J. Advances in dataflow programming languages. *ACM Computing Surveys (CSUR) 36*, 1 (2004), 1–34.
3. Kittur, A., Chi, E., & Suh, B. Crowdsourcing user studies with Mechanical Turk. *Proc. CHI 2008.*
4. Kittur, A., Smus, B., Khamkar, S. & Kraut, R. E. CrowdForge: Crowdsourcing complex work. *Proc. UIST 2011.*
5. Kulkarni, A., Can, M., & Hartmann, B. Collaboratively Crowdsourcing Workflows with Turkomatic. *Proc. CSCW 2012.*
6. Little, G., Chilton, L.B., Goldman, M., & Miller, R.C. Turkit: Human computation algorithms on Mechanical Turk. *Proc. UIST 2010*, 57–66.
7. Yu, J., Benatallah, B., Casati, F., & Daniel, F. Understanding mashup development. *IEEE Internet Computing*, (2008), 44–52.