Composing and Analyzing Crowdsourcing Workflows

PANOS IPEIROTIS, New York University and Google, Inc.
and GREG LITTLE, Digital Monk
and THOMAS W. MALONE, Massachusetts Institute of Technology

1. INTRODUCTION

When trying to crowdsource complex tasks, it is often better to break the big task into a set of smaller tasks and connect them into a production workflow [Little et al. 2009; Little et al. 2010]. Such workflows often allow a set of workers to work together in a task, creating an "assembly line for knowledge work" and allow the completion of tasks that are too complex or difficult for any participating individual to accomplish.

Recent research [Dai et al. 2013; Lin et al. 2012] has focused on analyzing and optimizing existing workflows, using Partially Observed Markov Decision Processes (POMDPs). The results indicate that using decision-theoretic approaches for dynamically adjusting the settings of crowdsourcing workflows and exchanging among multiple, existing workflows can lead to substantial improvements in both the quality and the overall cost of executing a workflow.

In this work, we present a method for creating and analyzing arbitrary workflows, and generating predictions about the resulting quality and cost of a given workflow. Specifically, we focus on the following questions:

(1) Given a workflow for a task, can we predict the cost and time for completing the task, together with the resulting quality of the generated product?
(2) Given a workflow for a task, can we generate alternative workflows that are expected to be better than the current one?

2. MODEL

For our modeling approach, we treat a workflow as a directed graph $G(V, E)$. Each node $v \in V$ corresponds to an "operation." Each operation node $v \in V$ has one or more inputs $\mathbf{in}(v) = \langle in_1(v), \ldots, in_n(v) \rangle$, and generates one or more outcomes $\mathbf{out}(v) = \langle out_1(v), \ldots, out_m(v) \rangle$. An edge $e \in E$ denotes the flow of data from one operation to another. The data that flow over the edges from one operation to another is described by probability distributions: These distributions describe our belief that the output of a given operation has a certain level of quality. Formally, the output of an operation is a set of pairs $\langle Q, \mathbf{F} \rangle$ where every quality value $q \in Q$ has an associated probability $\mathbf{F}(q)$ of occurring.

We describe four basic operations of a crowdsourcing workflow, which we believe to be sufficient to cover a wide range of scenarios: (a) Generation ($\mathcal{G}$), (b) Noisy measurement ($\mathcal{N}$), (c) Iteration ($\mathcal{I}$), and (d) Select top-r ($\mathcal{S}$).

2.1 Generation

The "*generation*" ($\mathcal{G}$) operation describes the creation from scratch of a particular artifact (e.g., an image description) and the output is the distribution of quality of the generated artifacts. This operation captures the beginning of the crowdsourcing process.The output of the $\mathcal{G}$ operation is typically latent, and not directly observable. We can only observe the output of $\mathcal{G}$ through a noisy measurement process, described next.

2.2 Noisy Measurement

The "*noisy measurement*" ($\mathcal{N}$) operation, captures the existence of noise when evaluating the quality of a given artifact. Every time that we have a measurement, the ouput of the measurement is a *noisy* representation of the underlying, latent, signal. For example, we can ask workers to evaluate the quality of a given image description: the answers provided by the workers are often close but not exactly identical to the "true" quality of the description.

To model the various types of measurement errors, we represent the measurement process using a joint probability distribution $N(m|y)$, which describes the probability of observing the measurement m when the underlying signal has value x. When visualizing the response using a 2-d matrix, a perfect, noiseless measurement is a diagonal matrix. An unbiased noisy measurement would correspond to an matrix where the responses are a band around the diagonal, with a width proportional to the noise.

The noisy measurement operation can describe the cost, behavior, and effect of many quality control algorithms [Welinder et al. 2010; Ipeirotis et al. 2010] that aggregate the votes from multiple workers, while trying to infer the true underlying value of the annotated objects. For example, we can pay more workers to lower the noise in the measurement, generating a more accurate representation of the original, latent signal, and see the effect of such a change in later stages of the workflow.

2.3 Iteration transformation

The "*iteration transformation*" ($\mathcal{I}$) operation captures the concept of iterative improvement. The basic idea of iterative improvement is that workers are rarely capable of producing a perfect artifact, but they are likely to improve upon an existing imperfect draft. To model of the iterative process, we use the concept of continuous state Markov Decision Processes. During an iteration, an artifact of quality i is entering as input and in the output we expect to see output of quality o, with probability $P(o|j)$; we call the 2-d matrix that represents the distribution $P(o|j)$ as "*transition matrix*". We assume, for simplicity, that the values of the transition matrix do not change over time. By modeling the iterative process as a stationary continuous-state Markov chain, we can estimate many properties of operation by simply analyzing properties of the transition matrix. For example:

—The principal eigenvector of the transition matrix corresponds to the *stable* distribution of quality that we will reach after iteratively improving the artifact many times.

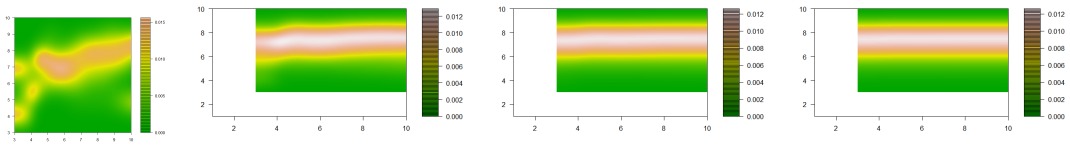
—The second eigenvalue of the transition matrix is indicative of the speed of convergence. Small eigenvalues mean that we only need a few iterations to reach the stable state, while large eigenvalues indicate the opposite.

2.4 Select top-r

The "comparison" ($\mathcal{I}$) operation corresponds to the action where a worker looks at n objects and selects the r-th ranked element.[1] The operation receives as input a set of artifacts, described by their respective quality distributions; in the simplest case we have two inputs, in the general case n inputs. The operation returns the r-th element, and following the conventions of our modeling approach, we want to know the quality distribution of that artifact.

To model the expected outcome of this operation, we rely on results from *order statistics*. Using existing results, we can *analytically* infer the distribution of the r-th ranked element, when ranking n elements (the inputs) that are distributed according to a set of given probability distributions. For example, if we try to select the maximum out of n input elements that have qualities uniformly dis-

[1] In many common scenarios, $r = 1$ or $r = n$, corresponding to the case of picking the minimum or the maximum element.



(a) 1 iteration (b) 2 iterations (c) 3 iterations (d) 4 iterations

Fig. 1: Example of the transition matrices that correspond to various numbers of iterations. The matrix has a second eigenvalue of only 0.2 and quickly converges to a stable state after only a couple iterations.

tributed distributed, then the r-th ranked element has a quality that follows the $Beta(r, n - r + 1)$ distribution.

3. EXAMPLE ANALYSES

Example 1: Define number of iterations in a workflow. When analyzing workflows with iterations, it is useful to know how many times we will need to iterate to reach a desirable outcome. Figure 1a illustrates the transition matrix for a single iteration. Using results from the analysis of Markov Chains, we can use the principal eigenvector of the matrix to see the quality distribution after iterating infinite number of times. Additionally, by looking at the second eigenvector of the matrix we can estimate how quickly the process will converge to that distribution. The second eigenvalue of that particular matrix is only 0.2, indicating that even after the second iteration the divergence from the stable state is going to be pretty small (only $0.2 \times 0.2 \approx 4\%$ error).

Example 2: Discovering the max element, tournament vs. sequential. Consider the case of trying to find the maximum element out of set of n, using only binary comparisons. The conventional thinking is that a set of binary, tournament-style elimination comparisons would be the best way to do so. Our analysis indicates that a better alternative is to use a *sequential* comparison strategy where we first pick the best out of two elements, then we compare the winner with the third element, the winner of this comparison with the fourth and so on. The basic intuition: In tournament-style comparisons, the winners after each round are getting closer and closer together, making the comparisons costlier as we need more workers for a reliable comparison. In contrast, with sequential comparisons, the most difficult comparison is the first one, and then the expected distance between the compared elements is expected to increase making the comparisons easier.

REFERENCES

Peng Dai, Christopher H Lin, Daniel S Weld, and others. 2013. POMDP-based control of workflows for crowdsourcing. *Artificial Intelligence* (2013).

Panagiotis G Ipeirotis, Foster Provost, and Jing Wang. 2010. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD workshop on human computation*. ACM, 64–67.

Christopher H Lin, Daniel S Weld, and others. 2012. Dynamically switching between synergistic workflows for crowdsourcing. In *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Greg Little, Lydia B Chilton, Max Goldman, and Robert C Miller. 2009. Turkit: tools for iterative tasks on mechanical turk. In *Proceedings of the ACM SIGKDD workshop on human computation*. ACM, 29–30.

Greg Little, Lydia B Chilton, Max Goldman, and Robert C Miller. 2010. Turkit: human computation algorithms on mechanical turk. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*. ACM, 57–66.

Peter Welinder, Steve Branson, Pietro Perona, and Serge J Belongie. 2010. The multidimensional wisdom of crowds. In *Advances in Neural Information Processing Systems*. 2424–2432.