Human Computation
AAAI Technical Report WS-12-08

Dynamically Switching between Synergistic Workflows for Crowdsourcing

Christopher H. Lin Mausam Daniel S. Weld
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195
{chrislin,mausam,weld}@cs.washington.edu

Introduction

Crowdsourcing marketplaces (*e.g.*, Amazon Mechanical Turk) enable rapid construction of complex workflows (Little et al. 2009; Bernstein et al. 2010) that seamlessly mix human computation with computer automation to accomplish practical tasks. One of the biggest challenges facing designers of crowdsourced applications is the variability of worker quality.

Frequently, a task designer will experiment with several alternative workflows to accomplish the task, but choose a single one for the production runs (*e.g.* the workflow that achieves the best performance during early testing). In the simplest case, alternative workflows may differ only in their user interfaces or instructions. For a more involved task like text-improvement, one workflow may present workers with several different improvements of a text and ask them to select the best one. A second workflow might instead present workers with one improvement and ask them to rate it on a scale from 1 to 10.

After choosing a single workflow, in order to ensure quality results, task designers often aggregate worker responses on redundant runs of the workflows (Snow et al. 2008; Whitehill et al. 2009; Dai, Mausam, and Weld 2010). For instance, to determine the best text improvement from the results of the second workflow, the task designer might select the one with the highest average rating.

Unfortunately, this seemingly natural design paradigm does not achieve the full potential of crowdsourcing. Selecting a *single* best workflow is suboptimal, because alternative workflows can compose *synergistically* to attain higher quality results.

Suppose after gathering some answers for a task, one wishes to further increase one's confidence in the results; which workflow should be invoked? Due to the very fact that it is different, an alternative workflow may offer independent evidence, and this can significantly bolster one's confidence in the answer. If the "best" workflow is giving mixed results for a task, then an alternative workflow is often the best way to disambiguate.

Instead of selecting one a priori best workflow, a better solution should reason about this potential synergy and *dynamically* switch between different workflows. We now present a short summary of Lin *et al.* (2012).

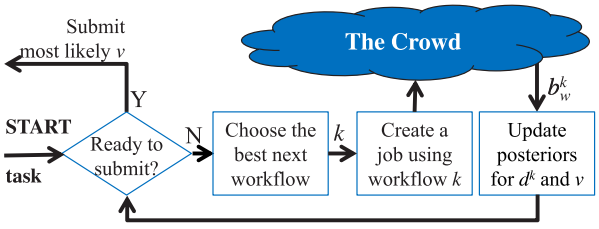

Figure 1: AGENTHUNT's decisions when executing a task.

Probabilistic Model for Multiple Workflows

We follow and extend Dai *et al.*'s probabilistic generative model (2010; 2011). We assume that the task has 2 possible answer choices. For our model, there are K alternative workflows that a worker could use to arrive at an answer. Let $d^k \in [0, 1]$ denote the inherent difficulty of completing a task using workflow k, and let $\gamma_w^k \in [0, \infty)$ be worker w's error parameter for workflow k. Notice that every worker has K error parameters. Having several parameters per worker incorporates the insight that some workers may perform well when asked the question in one way but not so well when asked in a different way.

The accuracy of a worker w, $a(d^k, \gamma_w^k) = \frac{1}{2}\left(1 + (1 - d^k)^{\gamma_w^k}\right)$, is the probability that she produces the correct answer using workflow k.

We assume that given the workflow difficulty, d^k, workers' answers are independent of each other. We also assume that the workers do not collaborate with each other and that they are not adversarial.

Decision-Theoretic Agent

Using our model, we design an automated agent, named AGENTHUNT, that uses a Partially-Observable Markov Decision Process (POMDP) to dynamically switch between workflows (Sondik 1971; Russell and Norvig 2002).

For AGENTHUNT, a state in the POMDP is a $K + 1$ tuple $(d^1, d^2, \ldots, d^K, v)$, where d^k is the difficulty of the k^{th} workflow and v is the true answer of the task. At each time step, AGENTHUNT has a choice of $K + 2$ actions — it can submit one of two possible answers or create a new job

	AGENTHUNT	TURKONTROL	AGENTHUNT$_{RL}$
Avg Accuracy (%)	92.45	84.91	93.40
Avg Cost	5.81	6.26	7.25
Avg Net Utility	-13.36	-21.35	-13.85

Table 1: Comparisons of accuracies, costs, and net utilities of various agents when run on Mechanical Turk.

with any of the K workflows. The process terminates when AGENTHUNT submits any answer. When AGENTHUNT creates a new job using workflow k, it will receive an observation containing one of the 2 answers chosen by some worker w. This information, along with knowledge of γ_w^k, allows AGENTHUNT to update its belief. After submitting an answer, AGENTHUNT can update its records about all the workers who participated in the task using what it believes to be the correct answer. Figure 1 is a flow-chart of decisions that AGENTHUNT needs to make.

Learning the Model

In order to behave optimally, AGENTHUNT needs to learn worker error parameters and a prior difficulty distribution. We consider two unsupervised approaches to learning — offline batch learning and online RL. In the first approach we first collect training data by having a set of workers complete a set of tasks using a set of workflows. Then we use an EM algorithm, similar to that proposed by Whitehill *et al.* (2009) to learn all parameters jointly. In our second approach, we create AGENTHUNT$_{RL}$, which is able to accomplish tasks right out of the box by learning parameters on the fly. When it begins a new task, it uses the existing parameters to recompute the best policy and uses that policy to guide the next set of decisions. After completing the task, AGENTHUNT$_{RL}$ recalculates the maximum-likelihood estimates of the parameters using EM as above.

Experiments

We choose a named-entity recognition task, for which we create $K = 2$ alternative workflows. We compare two agents: TURKONTROL, a state-of-the-art controller for optimizing the execution of a single (best) workflow (Dai, Mausam, and Weld 2010), and our AGENTHUNT, which can switch between the two workflows dynamically.

We develop two workflows for the task. Both workflows begin by providing users with a body of text and an entity. The first workflow, called "WikiFlow," displays the titles and first sentences of Wikipedia articles and asks workers to choose the one that best describes the entity. Then, it returns the Freebase[1] tags associated with the selected Wikipedia article. The second workflow, "TagFlow," asks users to choose the best set of Freebase tags directly.

After gathering training data using Mechanical Turk, and confirming our suspicions with successful simulated experiments, we perform real-world experiments using Mechanical Turk. We set TURKONTROL to use the TagFlow since our data show it to be easier for workers.

We generate 106 new tasks for this experiment, and use gold labels supplied by a single expert. Table 1 shows that AGENTHUNT fares remarkably better in the real-world than TURKONTROL given similar budgets.

Finally, we compare AGENTHUNT to AGENTHUNT$_{RL}$. We test it using the same 106 tasks from above. Table 1 suggests that the two agents are comparable and that AGENTHUNT can perform in an "out of the box" mode, without needing a training phase.

More details can be found in Lin *et al.* (2012). Our system is available for use at http://cs.washington.edu/node/7714.

Related Work

The benefits from combining disparate workflows have been previously observed. Babbage's Law of Errors suggests that the accuracy of numerical calculations can be increased by comparing the outputs of two or more methods (Grier 2011). AGENTHUNT embodies the first method for *automatically* evaluating potential synergy and dynamically switching between workflows.

Acknowledgements

We thank Xiao Ling for generating our training and testing data. This work was supported by the WRF / TJ Cable Professorship, Office of Naval Research grant N00014-12-1-0211, and National Science Foundation grants IIS 1016713 and IIS 1016465.

References

Bernstein, M. S.; Little, G.; Miller, R. C.; Hartmann, B.; Ackerman, M. S.; Karger, D. R.; Crowell, D.; and Panovich, K. 2010. Soylent: A word processor with a crowd inside. In *UIST*.

Dai, P.; Mausam; and Weld, D. S. 2010. Decision-theoretic control of crowd-sourced workflows. In *AAAI*.

Dai, P.; Mausam; and Weld, D. S. 2011. Artificial intelligence for artificial intelligence. In *AAAI*.

Grier, D. A. 2011. Error identification and correction in human computation: Lessons from the WPA. In *HCOMP*.

Lin, C. H.; Mausam; and Weld, D. S. 2012. Dynamically switching between synergistic workflows for crowdsourcing. In *AAAI*.

Little, G.; Chilton, L. B.; Goldman, M.; and Miller, R. C. 2009. Turkit: tools for iterative tasks on mechanical turk. In *KDD-HCOMP*, 29–30.

Russell, S., and Norvig, P. 2002. *Artificial Intelligence: A Modern Approach*. Prentice Hall.

Snow, R.; O'Connor, B.; Jurafsky, D.; and Ng, A. Y. 2008. Cheap and fast - but is it good? evaluating non-expert annotations for natural language tasks. In *EMNLP*, 254–263.

Sondik, E. J. 1971. *The Optimal Control of Partially Observable markov Processes*. Ph.D. Dissertation, Stanford.

Whitehill, J.; Ruvolo, P.; Bergsma, J.; Wu, T.; and Movellan, J. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS*.

[1] *www.freebase.com*