Learning on the Job: Optimal Instruction for Crowdsourcing

Jonathan Bragg JBRAGG@CS.WASHINGTON.EDU
University of Washington, Seattle, WA

Mausam MAUSAM@CSE.IITD.AC.IN
Indian Institute of Technology, Delhi, India

Daniel S. Weld WELD@CS.WASHINGTON.EDU
University of Washington, Seattle, WA

Abstract

A large body of crowdsourcing research focuses on using techniques from artificial intelligence to improve estimates of latent answers to questions, assuming fixed (latent) worker quality. Recently, researchers have begun to investigate how best to actively improve worker quality through instruction (Basu & Christensen, 2013; Singla et al., 2014). However, none of the existing work considers the fundamental tradeoff between providing instruction and getting actual work done. In this work, we present a reinforcement learning agent capable of optimizing the instruction it provides, by learning the effectiveness of its teaching actions, the quality of the worker population, and the amount of work output it can expect from individual workers. Evaluations on synthetic data show that our agent learns adaptive instruction policies that significantly outperform common baseline strategies such as providing a tutorial of fixed length.

1. Introduction

Quality control is a central challenge faced by crowdsourcing systems. Researchers have proposed many sophisticated approaches that enable systems to produce higher-quality answers by estimating worker abilities. While these approaches can dramatically improve crowdsourcing results, they miss a key opportunity to *improve* worker abilities, thereby leading to further improvements in quality at lower cost. Better worker training has the potential to improve task performance, adherence to requester instructions, job satisfaction, and to enable crowdsourcing of more complex tasks.

Current approaches to worker training are largely ad-hoc. When establishing a crowdsourced workflow, most requesters first show workers an initial set of instructions, then administer a test to ensure that the worker paid attention to the instructions, and finally give the worker a set of tasks to perform. They may also scatter additional gold questions amidst the actual work tasks, to detect poor workers. Usually, they use A-B testing to tune the amount of instruction and number of tests (Kohavi et al., 2007), but even with A-B testing, this sequence is suboptimal for three reasons: (1) it is designed for the average worker, but most workers *aren't* average (different training and testing regimes are better for different people); (2) all the work done during A-B testing is typically wasted; and (3) the policy resulting from A-B testing is fixed and never changes if worker demographics (or behaviors) change, e.g., as a result of forum discussions or out-of-band conversations on how to game the system.

Our approach overcomes these challenges by using reinforcement learning and decision theory to optimally personalize and schedule instruction, test, and work activities. This method naturally (and optimally) inserts reminder lessons in a way that maintains worker effectiveness.

2. Problem

2.1. The Model

We assume that the requester has an unlimited number of gold questions, $\mathcal{Q}_{\text{gold}}$, with known answers, as well as questions with unknown answers, $\mathcal{Q}$. Intuitively, our objective is to answer the questions in $\mathcal{Q}$ with maximal accuracy, given a limited labor budget.

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

The task is defined by a set $\mathcal{R}$ of guidelines (rules) specified by the requester. Each question q requires that a worker know a subset of rules, $\mathcal{R}_q$, to answer it correctly. In general, we don't know a priori which questions require which rules, but we do know the subset of rules required for each gold question $q \in \mathcal{Q}_{\text{gold}}$.

We assume we know the prior probability, $P(r)$, of a rule r being required for an unknown question in $\mathcal{Q}$. This quantity can be estimated by computing the fraction of questions in $\mathcal{Q}_{\text{gold}}$ which require the rule (assuming $\mathcal{Q}_{\text{gold}}$ represents the overall distribution), or by sampling a small set of questions from $\mathcal{Q}$.

Following the *knowledge tracing* literature (Corbett & Anderson, 1994), we assume that worker w will answer a question correctly with probability $(1 - P_{\text{slip}})$ if s/he possesses the required skills and with probability P_{guess} otherwise. We represent worker w using a latent boolean vector of skills γ_w, where $\gamma_{w,r} = 1$ iff the worker knows how to apply rule r to questions. Assuming that each rule $r \in \mathcal{R}$ occurs with independent probability $P(r)$, the probability worker w answers an unknown question correctly is

$$P(\text{correct} \mid w) = P_{\text{guess}} \cdot (1 - z) + (1 - P_{\text{slip}}) \cdot z, \quad (1)$$

where $z = \prod_{r \in \mathcal{R}} 1 - P(r) \cdot (1 - \gamma_{w,r})$ is the probability that the worker possesses all the required skills. The probability a worker answers a gold question correctly follows the same equation, only we know $P(r) = 1, \forall r \in \mathcal{R}_q$, and 0 otherwise. Over time, we learn a prior distribution over γ_w for unseen workers. At any time step, a worker may leave the system with probability P_{leave}.

2.2. The Decision Problem

At each time step, the system needs to decide whether to *teach* or *test* the worker (and if so which rule, r, to use), or whether to *ask* the worker to answer an unknown question $q \in \mathcal{Q}$.

We model the problem of making this choice as a partially observable Markov decision process (POMDP) (Kaelbling et al., 1998; Poupart, 2011). Formally, a POMDP is specified by a set of states, set of actions, transition function, observation function, and reward function, which we define as follows.

- The state space $\mathcal{S}$ can be factored as $< \Gamma_w, S_a >$, where Γ_w is a vector of boolean variables corresponding to the latent skill vector γ_w and S_a is a variable of size $|\mathcal{R}| + 1$ specifying the last rule tested (or none). [1] Additionally, $\mathcal{S}$ contains a special "submit" state.

- The set of actions, $\mathcal{A}$, consists of four actions: ask the worker an unknown question, select a rule to test (by asking a gold question that requires that rule), teach the worker a rule (we assume one or more pieces of instructional text associated with each rule), [2] and reject a worker (if the system deems him/her not worth employing). The current system may only teach immediately following a test action. The sequence of a test action followed a teach action corresponds to the *elicit* teaching action used in intelligent tutoring systems (Chi et al., 2008). [3] Finally, we assume that an external process selects questions from $\mathcal{Q}$ or $\mathcal{Q}_{\text{gold}}$.

- The transition function $P(s' \mid s, a)$ specifies a probability distribution over outcome states supposing that the system executes $a \in \mathcal{A}$ while in $s \in \mathcal{S}$.
  - Reject worker: Enter the submit state with probability 1.
  - Select a rule to test: Worker quits with probability P_{leave}. Otherwise, the worker answers and loses knowledge of each rule independently with probability P_{lose}. The state variable S_a is set to the rule tested.
  - Teach the rule that was just tested: The worker gains knowledge of the rule taught with probability P_{gain}. The state variable S_a is reset.
  - Ask an unknown question: Worker quits with probability P_{leave}. Otherwise, the worker answers and loses knowledge of each rule independently with probability P_{lose}.

- Observation function: The system can only directly observe the worker's answers to ask and test questions. The accuracy of the worker's answer is a function of which rules s/he knows and which are likely to be required by the question, as given in Equation 1. Additionally, we observe when workers leave.

- Reward function: The agent incurs a cost of c for each ask or test action (indistinguishable to the worker) and c_t for each teaching action. [4] The reward for asking question $q \in \mathcal{Q}$ is $\lambda f(q)$, where

$$f(q) = \mathrm{E}\left[\max_{z \in Z_q} P(Z_q = z \mid A_{q,w})\right] - \max_{z \in Z_q} P(Z_q = z)$$

is the expected accuracy gain produced by incorporating the new information provided by the worker into

[1] An additional variable specifying the last action is needed since our observation function does not include the previous state.

[2] A simple instructional action consisting of restating the rule along with the gold answer can be generated automatically.

[3] An alternative teaching action is to *tell* the worker the answer immediately, but this provides less information about worker knowledge.

[4] These labor costs can be computed as part of reinforcement learning, by estimating the average observed time to complete each action and multiplying by a fair wage.

the posterior predictions and λ is a parameter specifying the utility of information. In the above equation, Z_q is the random variable for the latent answer to question q, and $A_{q,w}$ is the random variable for worker w's response to q. The expectation is taken over possible worker responses, where $P(A_{q,w} = z \mid Z_q = z)$ is determined by Equation 1.

We chose not to include latent answers to questions in the state space in the current work so that the state space does not grow with the number of questions. Moreover, including answers in the state space is unnecessary since we focus on maximizing the benefit provided by each worker separately. In expectation, the model we define is equivalent to a model with a state space that includes the latent answers, and which has a $2^{|\mathcal{Q}|}$ possible submit actions corresponding to submitting an answer for each question. Prior work on decision theoretic crowdsourcing (Lin et al., 2012a;b) has used such a model when $|\mathcal{Q}| = 1$; our setting is more challenging since we must consider the answers to many tasks simultaneously.

3. Experiments

We conducted experiments using synthetic data under a range of hand-estimated parameter configurations to demonstrate the potential benefits of our approach, which are twofold. First, learning a policy using our POMDP approach saves costly A-B testing to find the best fixed policy. Second, in some settings, the learned adaptive policy outperforms reasonable baseline policies, meaning that even extensive A-B testing will not yield a policy that is as good.

We assume in these experiments that worker skills are independent and thus the initial belief state probabilities are $P(\Gamma_w, S_a) = \prod_{r \in \mathcal{R}} P(\Gamma_{w,r})$ for states where S_a indicates that no rule has just been tested, and 0 otherwise. Simulations sample workers from this prior. Additionally, we assume binary multiple choice questions and unskewed data ($P(Z_q) = 0.5, \forall q \in \mathcal{Q}$), and set $c = c_t = 0.1$ and $\lambda = 1$. We used the ZMDP POMDP solver [5] with default configuration settings, maximum solve time of 1 minute [6], and discount factor of 0.99.

Our preliminary experiments compare to baseline policies that teach each rule k times during an initial tutorial phase. Our experiments assume that $P(r)$ and $P(\gamma_{w,r})$ are the same $\forall r \in \mathcal{R}$, since these configurations are most favorable to the baseline policies.

[5] https://github.com/trey0/zmdp

[6] Experiments were run on 6-core 2.4 GhZ processors.

3.1. Planning

Our first set of experiments assumes that the model parameters are known.

We experimentally obtain basic policies under a range of parameter settings. For instance, if workers already know the rules with high probability ($P(\gamma_w)$ is high) or the rules are infrequently needed to answer unknown questions ($P(r)$ is low), our system learns to begin asking unknown questions immediately (no teaching). The system also limits teaching if workers tend to leave quickly (P_{leave} is high). On the other hand, if workers need instruction but fail to respond to teaching ($P(\gamma_w)$ and P_{learn} are low), our system rejects workers immediately.

In addition to finding these policies, our system automatically tunes the amount of instruction. The POMDP policy earns at least as much reward as the best baseline policies in all the configurations tested. Additionally, the adaptive policies typically require significantly fewer teaching actions than baselines since they can better estimate when a worker knows a rule. Using fewer gold questions reduces the burden of creating gold questions for each rule as well as the risk that shared answers to gold questions will enable cheating.

3.2. Reinforcement Learning

When our system is first deployed in a new crowdsourcing environment, it must also learn the POMDP model. This necessitates an exploration-exploitation tradeoff. Preliminary experiments show that our system is able to learn the model using a simple epsilon-greedy strategy. [7] For episode e (the eth worker hired), we select actions randomly with probability $1/e$. We reestimate model parameters and replan prior to each episode. To estimate parameters, we treat the POMDP as an input output hidden Markov model (Bengio & Frasconi, 1995) and use the Baum-Welch (EM) algorithm initialized with parameters from the previous episode. We use a default prior of Beta(1.1, 1.1) [8] on parameters, but introduce a weak bias toward 0 (Beta(2, 5)) to match our intuition that P_{slip}, P_{lose}, and $P(\gamma_w)$ tend to be smaller than 0.5.

Figure 1 shows the learning performance given two rules and a set of plausible parameters ($P_{\text{leave}} = 0.01$, $P_{\text{learn}} = 0.4$, $P_{\text{lose}} = 0.05$, $P_{\text{slip}} = 0.1$, $P_{\text{guess}} = 0.5$, $P(\gamma_w) = 0.2$, $P(r) = 0.5$). POMDP (known), the model given true parameters, obtains 4.6 times as much reward as the best baseline policy in our space of baselines (teach each rule twice). Figure 2 shows a sample execution trace of this

[7] We also tried a Thompson sampling-based strategy, which performed worse due to high variance in the models and policies.

[8] This prior gives initial parameter estimates of 0.5.

Figure 1. Cumulative rewards for a simple (two rule) problem, averaged over 1000 simulations and shown with 95% confidence bands. POMDP (known) uses the true parameters, POMDP (learned) reestimates parameters each episode, and the best fixed policy is the baseline policy that teaches each rule twice. Each episode corresponds to hiring a new worker.

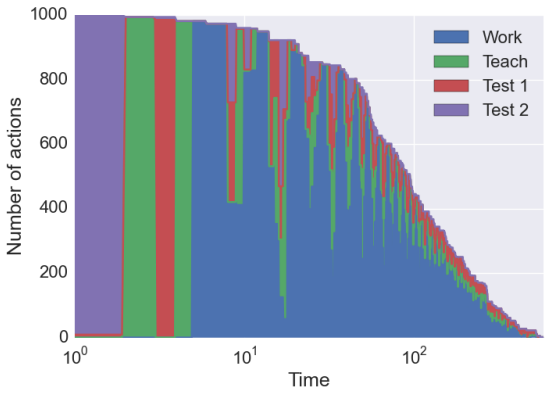

Figure 2. Sample execution trace for the POMDP (known) model in Figure 1. The number of actions (Y axis) is summed over 1000 workers and drops over time as workers abandon the task. Note that the system executes different (adaptive) teaching sequences depending on test results. Work corresponds to asking an unknown question and Test 1 and Test 2 correspond to testing rules 1 and 2, respectively.

model. After having seen 100 workers, the reinforcement learning agent accumulates 2.8 times as much reward as the best baseline policy. While one could make the space of baseline policies arbitrarily more complex (e.g., by introducing an additional parameter controlling intervals between teaching sessions), the POMDP approach is simple to specify, does not require extensive A-B testing, and can perform significantly better even than the best candidate baseline.

4. Related Work

Researchers have begun to investigate how best to improve crowdsourcing worker quality by actively selecting examples for instruction (Basu & Christensen, 2013; Singla et al., 2014). Unlike our approach, this work focuses solely on teaching and does not consider the tradeoff between providing more instruction and assigning work that helps the system answer unknown questions.

Optimizing instruction for crowdsourcing shares many of the same challenges faced by intelligent tutoring systems. We make use of knowledge tracing (Corbett & Anderson, 1994), one of the earliest probabilistic models of student learning, but a number of more sophisticated models have also been proposed (Koedinger et al., 2013). Researchers have developed techniques for optimizing instruction in intelligent tutoring systems, including using POMDPs to produce better teaching policies (e.g., (Rafferty et al., 2011)). However, our system must consider the long-term implications of learning gains (on future tasks), rather than simply minimizing the time to achieve those learning gains.

Our approach is also inspired by recent work on optimizing the order in which algebraic lessons are taught in educational mathematics games (Liu et al., 2013; Mandel et al., 2014; O'Rourke et al., 2015). Like intelligent tutoring systems, these games seek to maximize student knowledge (or as a proxy, the time spent interacting with the game). In our case, worker knowledge is irrelevant in itself, since workers can leave at any time. Instead, we seek to optimize our accuracy over a set of questions that need answering.

5. Conclusions and Future Work

In the future, we plan to conduct live experiments using an NLP data annotation task called relation extraction. Although the relations are familiar ones, such as *lived in*, the annotation guidelines specified by benchmarks like the NIST Text Analysis Conference KBP challenge (Surdeanu & Ji, 2014) include numerous rules, some of which are counter-intuitive and must be learned by workers.

There are many possible directions to extend this work. We plan to include the latent answers as part of our state, to enable reasoning about answers across workers. Additionally, we plan to scale the model so that it is capable of handling more complex tasks with many more rules. While researchers have successfully solved POMDPs with large state spaces by exploiting factored representations (e.g., (Hoey et al., 2007)), scaling the number of state variables and actions is challenging, as is learning parameters for such a model. Furthermore, reinforcement learning is made more challenging by the fact that rewards for asking unknown questions are not directly observable and depend

on our estimates of the skills possessed by workers. Still, our experiments suggest that learning is possible in this setting.

Another possible direction for future work is to adopt more sophisticated models of worker learning and engagement. In order to facilitate learning parameters from a small amount of data, we assume that a worker learns or forgets any rule with the same probability. However, in some settings, rules may require different amounts of instruction. Additionally, modeling individual learning rates (Lee & Brunskill, 2012), individual probabilities of leaving (Mao et al., 2013), or classes of workers (like spammers) may further improve performance. Behavioral traces may also be useful for predicting disengaged workers (Rzeszotarski & Kittur, 2011), signaling the need for interventions with tests or warnings.

This work is a first step toward our vision of a comprehensive system capable of providing a variety of scaffolding actions that maximize individual worker contributions and overall answer quality. We believe that work on optimizing worker training will lead to important advances in crowdsourcing capabilities, as well as planning and reinforcement learning algorithms.

References

Basu, Sumit and Christensen, Janara. Teaching classification boundaries to humans. In *AAAI*, 2013.

Bengio, Yoshua and Frasconi, Paolo. An input output HMM architecture. In *NIPS*, 1995.

Chi, Min, Jordan, Pamela, VanLehn, Kurt, and Hall, Moses. Reinforcement learning-based feature selection for developing pedagogically effective tutorial dialogue tactics. In *Educational Data Mining*, 2008.

Corbett, Albert T and Anderson, John R. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4):253–278, 1994.

Hoey, Jesse, Von Bertoldi, Axel, Poupart, Pascal, and Mihailidis, Alex. Assisting persons with dementia during handwashing using a partially observable markov decision process. In *International Conference on Vision Systems (ICVS)*, 2007.

Kaelbling, Leslie Pack, Littman, Michael L., and Cassandra, Anthony R. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1): 99–134, 1998.

Koedinger, Kenneth R, Brunskill, Emma, Baker, Ryan SJd, McLaughlin, Elizabeth A, and Stamper, John. New potentials for data-driven intelligent tutoring system development and optimization. *AI Magazine*, 34(3):27–41, 2013.

Kohavi, Ron, Henne, Randal M, and Sommerfield, Dan. Practical guide to controlled experiments on the web: listen to your customers not to the hippo. In *KDD*, 2007.

Lee, Jung In and Brunskill, Emma. The impact on individualizing student models on necessary practice opportunities. In *EDM*, 2012.

Lin, Christopher H, Mausam, and Weld, Daniel S. Dynamically switching between synergistic workflows for crowdsourcing. In *AAAI*, 2012a.

Lin, Christopher H, Mausam, and Weld, Daniel S. Crowdsourcing Control: Moving Beyond Multiple Choice. In *UAI*, 2012b.

Liu, Yun-En, Mandel, Travis, Butler, Eric, Andersen, Erik, O'Rourke, Eleanor, Brunskill, Emma, and Popovic, Zoran. Predicting player moves in an educational game: A hybrid approach. In *Educational Data Mining*, 2013.

Mandel, Travis, Liu, Yun-En, Levine, Sergey, Brunskill, Emma, and Popovic, Zoran. Offline policy evaluation across representations with applications to educational games. In *AAMAS*, 2014.

Mao, Andrew, Kamar, E, and Horvitz, Eric. Why stop now? predicting worker engagement in online crowdsourcing. In *HCOMP*, 2013.

O'Rourke, Eleanor, Andersen, Erik, Gulwani, Sumit, and Popovic, Zoran. A framework for automatically generating interactive instructional scaffolding. In *CHI*, 2015.

Poupart, Pascal. *Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions*, chapter 3, pp. 33–62. IGI Global, 2011.

Rafferty, Anna N, Brunskill, Emma, Griffiths, Thomas L, and Shafto, Patrick. Faster teaching by POMDP planning. In *International Conference on Artificial Intelligence in Education (AIED)*, 2011.

Rzeszotarski, Jeffrey M and Kittur, Aniket. Instrumenting the crowd: using implicit behavioral measures to predict task performance. In *UIST*, 2011.

Singla, Adish, Bogunovic, Ilija, Bartók, Gábor, Karbasi, Amin, and Krause, Andreas. Near-optimally teaching the crowd to classify. In *ICML*, 2014.

Surdeanu, Mihai and Ji, Heng. Overview of the english slot filling track at the TAC2014 knowledge base population evaluation. In *Text Analysis Conference (TAC)*, 2014.