POMDP-Based Worker Pool Selection for Crowdsourcing

Shreya Rajpal SHREYA.RAJPAL@GMAIL.COM
Indian Institute of Technology - Delhi

Karan Goel KGOEL93@GMAIL.COM
Indian Institute of Technology - Delhi

Mausam MAUSAM@CS.WASHINGTON.EDU
Indian Institute of Technology - Delhi

Abstract

In recent years, crowdsourcing has gained credibility as an effective and inexpensive method to solve large scale tasks by outsourcing to a large pool of human workers. However, because of variability in the quality of workers who perform these tasks, it is of paramount importance to optimize the quality of answers, while keeping the expenditure on these tasks to a minimum. Often, in crowdsourced settings, there are multiple worker pools, each having different mean worker skill level and a different asking price.

In this paper, we demonstrate the effectiveness of using decision theoretic approaches for optimizing the task of worker pool selection with respect to the quality of the responses received, as well as the total expenditure used for payments. In particular, we build a POMDP-based model that automatically decides which worker pool to query for a task based on its estimated difficulty, and the current confidence in the answer. Our experiments show that the net utility achieved by our worker pool selection model consistently outperforms baselines that use a single worker pool. We also achieve a better cost v/s quality tradeoff than the baselines, allowing us to get higher quality for a fixed budget.

1. Introduction

With the unprecedented rise of crowdsourcing, various market places for generic micro-crowdsourcing have emerged such as Amazon Mechanical Turk (AMT), oDesk, eLance, LeadGenius, and Microsoft's and Google's internal crowds. The worker pools in different market places have different strengths. For example, AMT workers have much more variable skill and usually output lower quality at a lower price compared to oDesk workers that often charge higher but also produce higher quality work (Ipeirotis, 2012). Then there are domain specific workerpools – ArcBazar is crowdsourcing specialized for architectural design, 99 Designs has expert artists, and TopCoder maintains a crowd of programmers and data scientists. In general specialized crowds charge significantly higher than generic crowds like AMT.

Even within a single marketplace the need for performance-based differentiation has resulted in assigning of badges or qualifications to subsets of workers. For instance, Amazon itself rewards a "Master" qualification to more experienced workers based on worker accuracies on gold data (Chen, 2012). AMT workers can also take qualification tests that make them eligible for certain tasks that need those skills. Requesters commonly use custom qualifications to compute their own metrics for identifying good workers. Similarly, CrowdFlower, a crowdsourcing consulting company, maintains "CrowdFlower Elite" – a group of workers for which CrowdFlower charges a higher price in return for a higher quality output.

The availability of several worker pools for a task can potentially create confusion for a new requester. Which platform should they select? Which qualification should they select within a platform? To our knowledge, no principled solution exists to this important problem; often requesters just go by their hunch, or at best test the various worker pools using gold questions and select the best *one* pool for the task. In this paper we argue that dynamically switching between the worker pools can be more effective than pre-selecting a single best worker pool.

As an example, suppose workers are required to critique

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

Hollywood cinema, and we have two worker pools – one general worker pool and one with a specifically strong knowledge of cinema. If a question is about some less popular cult movie then the pool with good movie knowledge is of paramount importance, whereas if it is about a common movie then even the general pool workers might give acceptable answers at potentially low costs. The asking prices of different pools and their quality profiles can be traded off to select the appropriate worker pool per question – this could provide significant cost savings or alternatively produce much higher quality for the same price.

Unfortunately, we do not know *a priori* whether a question is easy or difficult for a worker pool. However, we can continually estimate worker errors and task difficulties based on answer agreement and use this information to perform worker pool routing. We adapt existing literature on decision-theoretic crowdsourcing (Dai et al., 2013) to devise a Partially Observable Markov Decision Process (POMDP) formulation of the worker pool switching problem. Our solution is closest to that of *workflow* switching for crowdsourcing (Lin et al., 2012) – they switched between different workflows for the same task, but all work was given to the same pool and at the same price; our work is orthogonal in that it routes the same work to different pools possibly at different prices.

We perform simulation experiments to test our algorithm against the baseline of selecting a single best worker pool. The variations in quality and cost profiles of the pools guide the value of dynamic worker pool selection. If the two pools have low cost differential then the POMDP will almost always choose the better pool; whereas, if the cost differential is too high then then the better worker pool may be too costly to be useful. In intermediate situations our algorithm obtains significantly higher utility.

Additionally, we also perform a Mechanical Turk study for the task of named entity disambiguation using the dataset of Lin et al (2012). We used our two worker pools as normal AMT workers with high approval rating and AMT Master workers. We were surprised to find that Master workers are not much superior in quality than regular workers. More experiments are needed to determine whether this is generally the case or happened accidentally for our task.

Our paper makes the following contributions:

1. We identify the worker pool selection problem for crowdsourcing. Our proposed solution is a combination of unsupervised worker and question tracking with POMDPs for decision making.

2. We evaluate the value of our model with extensive simulation experiments. We find that our model always outperforms the baselines of selecting a single best worker pool. Its benefits are highest when the cost differentials between worker pools are intermediate.

3. We also perform live experiments on Mechanical Turk for switching between regular workers and Master workers. To our surprise we find that Master workers are not necessarily much better in quality for our task.

2. Related Work on Task Routing

There is substantial work on task routing in the literature, though we know of no work that models differential pricing, task difficulties and worker skills in the same model.

Ambati, Vogel and Carbonell (2011) developed a task routing framework that ranks tasks based on user preferences (which are, in their case, implicit as well as explicit) using a max-entropy NLP classifier. Donmez, Carbonell and Shneider (2010) developed a model for task routing which selects top quality labelers (workers) at each time step, and then aggregate their answers to estimate the true value of the answer. However, they assume all questions are equally difficult, and do not utilize responses from low-quality workers, which potentially alienates a large part of the pull-style crowdsourcing market. Another work by Donmez et al (2009) also selects the top quality workers for equal difficulty questions, along with assuming an equal wage system for all workers.

Karger, Oh and Shah (2014) also do not model question difficulty, and assume that workers are not persistent, so that each worker can only be used once. Other works (Karger et al., 2011a;b; 2013) continue to assume equal difficulty, and are non-adaptive in nature. Another work that assumes an equi-difficult distribution of questions (Yan et al., 2011) which improves the quality of the model by selecting the optimal task to be performed by the optimum worker. Other works (Bragg et al., 2014; Shahaf & Horvitz, 2010) assume a volunteer platform, i.e. workers do not get paid for their work, which makes the task routing cost agnostic.

Ho, Jabbari and Vaughn (2013) deal with the problem of adaptive task routing for heterogeneous task difficulty, where the worker's accuracy on some task is a function of worker skill and question difficulty. They however assume a homogeneous cost across workers of different skill levels, and the skill level of the worker is learned against a set of gold-labels, as opposed to our algorithm's unsupervised approach (which leverages a modified version of the expectation maximization algorithm (Whitehill et al., 2009). They also classify tasks based on different types, so that each worker's skill level varies with a particular task type. Lastly, their model assumes that each worker will announce the number of tasks they are willing to perform on arrival;

whereas in any real world online platform, worker retention is a complex problem to model, and assuming prior knowledge of it is inherently problematic. In Ho and Vaughn (2012), similar assumptions on worker wages are assumed, along with having prior information about question difficulty levels (in one scenario) and worker training for specific task types (in the other scenario).

No single work to our knowledge has attempted to tackle the task routing problem for open crowdsourcing marketplaces such as Amazon Mechanical Turk where worker pools have different strengths, different costs and tasks have different diffulties. In this work, we attempt to move towards this goal by modeling question difficulty, worker skill, and differential worker pricing in a single model.

3. Methodology

Since our model is an extension of the control model proposed by Dai, Lin, Mausam and Weld (2013), we first give a short description of their work, which will be helpful in describing our model.

3.1. Background

In their paper, Dai et al. (2013) demonstrate the value of using decision-theoretic techniques for the problem of optimizing workflows in crowdsourcing. Specifically, their model uses Partially-Observable Markov Decision Processes (POMDPs), and they demonstrate excellent cost-quality tradeoffs when performing dynamic switching of workflows using this model.

POMDPs provide a flexible framework with which to model decision-making when making noisy, or uncertain observations by relaxing the assumption that an agent has complete knowledge of his surroundings. The agent makes noisy observations about its current state, and the framework is used to model many single-agent, real-world problems, since the agent rarely has perfect information of its surroundings.

A POMDP is a six-tuple $< \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{P}, \mathcal{R} >$, where

- $\mathcal{S}$ is a finite set of discrete states.
- $\mathcal{A}$ is a finite set of all actions.
- $\mathcal{O}$ is a finite set of observations.
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition function describing the probability that taking an action in a given state will result in another state.
- $\mathcal{P} : \mathcal{S} \times \mathcal{O} \to [0, 1]$ is the observation function describing the probability that taking an action in a given state will result in an observation.
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward for taking an action in a state.

POMDPs extend MDPs by adding an observation set $\mathcal{O}$ along with an observation model $\mathcal{P}$. Since the agent has no direct knowledge of the world's current state, it maintains a probability distribution over states, called a *belief state* $\mathbf{b}$, reflecting its estimate of their corresponding likelihood(s):

$$\mathbf{b}(\mathbf{s}) \in [0, 1], \sum_{\mathbf{s} \in \mathcal{S}} \mathbf{b}(\mathbf{s}) = 1 \tag{1}$$

where $\mathbf{b}(\mathbf{s})$ is the probability that the current state is $\mathbf{s}$, inferred from the previous belief state, the most recent action, and the resulting observation.

Given this theoretical framework, Dai et al. (2013) apply this model to the problem of taking responses (or ballots) from workers in a crowdsourced setting in a dynamic fashion. Specifically, they consider the task of solving binary classification problems (where responses can be either of $0/1$). They model the worker's response using a generative model that is defined as:

$$a(d, \gamma) = \frac{1}{2}(1 + (1 - d)^{\gamma}) \tag{2}$$

where $d \in [0, 1]$ is the *intrinsic difficulty* of the task, and $\gamma \in [0, \infty]$ is the worker *error parameter*. A plate model representation of this generative model is given in Figure 1. Thus, a task with a difficulty approaching 1, leads to an accuracy near $1/2$ which is like a random guess from the worker. Similarly, as a worker's error parameter approaches ∞, the accuracy of the worker's responses approach random guesses. The accuracy function defined above thus gives the probability that a worker will give the correct response, given the difficulty of the question and the worker's error parameter.

Dai et al. then define their POMDP as follows,

- $\mathcal{S} = \{(d, v) | d \in [0, 1], v \in \{0, 1\}\}$ where d is the difficulty of the task and v is the true answer.
- $\mathcal{A}$ = {*query, submit true, submit false*}
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ contains the penalty for submitting an incorrect answer, and the cost of asking for a ballot.
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1] = ((d, v), a, (d, v)) \mapsto 1$. All other probabilities are 0.
- $\mathcal{O} = \{true, false\}$ is a Boolean response from a worker.
- $\mathcal{P} : \mathcal{S} \times \mathcal{O} \to [0, 1]$ is defined by their generative model.

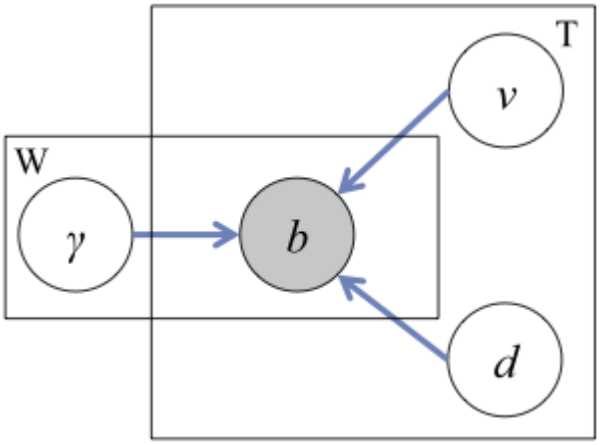

Figure 1. Plate Model Respresentation of the Generative Model. Reproduced from Dai, et al (2013).

The POMDP uses an average γ value that it uses to model new workers i.e., every new worker is assumed to be an average worker with value equaling $\bar{\gamma}$. The POMDP manages belief states over the cross-product of the Boolean answer, and the task difficulty. Dai et al (2013) discretize this difficulty into 11 possible values (to avoid a continuous state space), leading to a state space of $2 \times 11 = 22$ states.

They defined the POMDP's objective function as the minimization of expected cost, where cost is the sum of money paid to the workers and a requester-defined penalty for the wrong answer. This naturally trades off cost and accuracy, with higher penalties leading to possibly higher payments and higher quality output. To solve POMDPs, they run the ZMDP package for $300s$ using the default Focused Real-Time Dynamic Programming search strategy. We continue to use these details in our model.

3.2. Our Model

We extend Dai et al. (2013)'s model to the problem of task routing as outlined below. We restrict ourselves to binary classification tasks (of solving problems that have $0/1$ answers), and route tasks in a setting that has only k worker pools, $wp_1, wp_2, \ldots, wp_k$.

We model worker response accuracy using the same generative model as Dai et al (2013). We also extend their POMDP to allow for asking multiple worker pools as follows:

- $\mathcal{S} = \{(d, v) | d \in [0, 1], v \in \{0, 1\}\}$ where d is the difficulty of the task and v is the true answer.
- $\mathcal{A}$ = {*query* wp_1, *query* wp_2, ..., *query* wp_k, *submit true, submit false* }
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ contains the cost of asking for a ballot for each worker pool, and the penalty for submitting an incorrect answer.
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1] = ((d, v), a, (d, v)) \mapsto 1$. All other probabilities are 0.
- $\mathcal{O} = \{true, false\}$ is a Boolean response from a worker.
- $\mathcal{P} : \mathcal{S} \times \mathcal{O} \to [0, 1]$ is defined by our generative model.

There are three key differences between Dai et al's POMDP model and ours. First, we have many more actions, one for each worker pool, since the POMDP is now deciding which worker pool to query. Thus, the number of states in our POMDP formulation remain $11 \times 2 = 22$ and only the number of actions (and their associated costs) is increased. As a result, the overhead on computation is not significant and is due to extra actions, and not extra states.

Second, we also maintain several average skill parameters $\overline{\gamma}_i$, one for each pool. This allows a different $\mathcal{P}$, observation function, to be invoked based on the worker pool chosen. In real crowdsourcing situations, an average worker belonging to different worker pools will have a different expected skill level. Take the 2 worker pool setting as an example. In our model, the skill level (or the γ parameter) of a master worker will be better than (less than) the skill level (γ parameter) of a normal worker, which will be reflected in the observation function $\mathcal{P}$.

Finally, our POMDP provides a different cost for each worker pool. This is reasonable since typically AMT Master worker pools are recommended payments of about much higher than the pay to a normal worker. In general an expert's response is costlier than amateur's.

For learning the posteriors on each answer, true values of the worker γ parameters and question difficulty d, following previous research we utilize the Expectation Maximization approach (Whitehill et al., 2009). It jointly optimizes all parameters using alternating maximization.

Overall, our model is probably closest to the workflow switching model of Lin et al (2012). The main difference is that they switched between different workflows for the task but each workflow was priced the same and was given to the same worker pool. They had to model different difficulties in state space, but had the same costs. In our case, we have the same workflow but different worker pools, hence only one task difficulty d, but different $\overline{\gamma}$s and costs. In essence our models are complimentary and can be combined to have simultaneous workflow and worker pool switching.

Discussion: A key assumption in this formulation is that each worker pool is infinite, i.e., we can query a worker

pool for a question as many number of times. This is reasonable since Master AMT workers, or oDesk workers, etc., are large sets and usually we ask one question to only a handful of workers in a pool. For situations where worker pools are really small, we can incrementally remove the appropriate actions (when a worker pool exhausts) and query the next best worker pool. Overall, our problem and solution formulations are inherently different to works that allocate a question to a specific worker (e.g., (Bragg et al., 2014)).

4. Simulations

In this section, we perform several simulation experiments to test the efficacy of our model in a variety of experimental settings. For all of our experiments we restrict to simulating two worker pools, for example, Master (W_{Master}) and Normal (W_{Normal}) workers at AMT. We wish to understand when our model yields better performance compared against the baselines of pre-selecting either of the worker pools and sticking with them for all the tasks. To answer this, we vary the two distributions for worker parameters, the cost differentials and difficulty distributions of the tasks. For simulations we learn the POMDP with the true sampling distribution means as average γ values for the 2 worker pools while learning it (which are used by the POMDP as error parameters for future workers).

4.1. Simulation 1: Uniform Difficulty Distributions

We first test the setting in which questions have uniform difficulties between 0 and 1, i.e., there is no preference for a specific difficulty value. We draw 100 questions from 10 uniform difficulty distributions $\{U(0, 0.1), U(0.1, 0.2), ..., U(0.9, 1.0)\}$ with the constraint that we draw ten questions from each distribution.

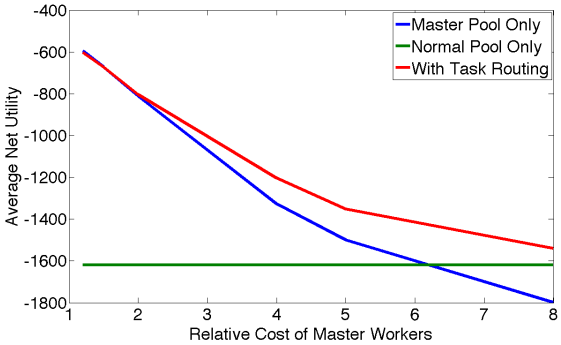

Figure 2. Variation in average net utility obtained on varying relative cost of the master pool for $W_{master} \sim \mathcal{N}(0.25, 0.05)$.

We wish to model that the master worker pool is significantly better than the normal pool. We sample 100 workers each from $W_{normal} \sim \mathcal{N}(1.0, 0.2)$ and $W_{master} \sim \mathcal{N}(0.25, 0.05)$. These gamma values are truncated above zero and are not allowed to be negative. A rough calculation reveals that on a question of medium difficulty ($d = 0.5$), the expected accuracy of a master worker will be around 92%, compared to 75% for a normal worker, in this scenario. In addition to this, this setting also models that master workers exhibit less variability in performance, and are thus reliably good.

We simulate worker answers using our generative model, and also shuffle the order in which the worker responses are received by the POMDP. We average our results over 100 runs. In all these experiments the POMDP and Bayes net models do not know the true difficulties of a question or the true parameters of the worker. However, they do have access to a prior difficulty distribution and γ averages. In future work, we can use reinforcement learning to learn these as well.

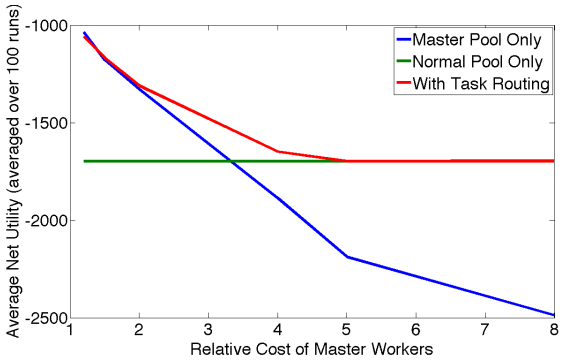

Figure 3. Variation in average net utility obtained on varying relative cost of the master pool for $W_{master} \sim \mathcal{N}(0.50, 0.10)$.

Figure 2 shows the performance of our model compared to the baselines on varying the relative cost of the master workers. We do utility calculations by summing up the costs of solving each question, and then adding a penalty for each wrong answer that is returned by the POMDP. We pick the penalty to be 100 for this case. For all experiments we keep payment of normal worker to be 1 and payment of master worker is varied from 1 to 8.

We find that our model equals or (Figure 2) outperforms both baselines for all choices of the relative cost for the master worker pool. In particular, we note that our model approaches the utility of using only the master worker pool for a low relative cost, and approaches the utility of using only the normal worker pool for a high relative cost. This is intuitively satisfying, since if the expert pool is almost as inexpensive as the normal pool, then there is no reason to switch to the normal pool (since experts give better quality answers). Similarly, if the experts are too expensive, then

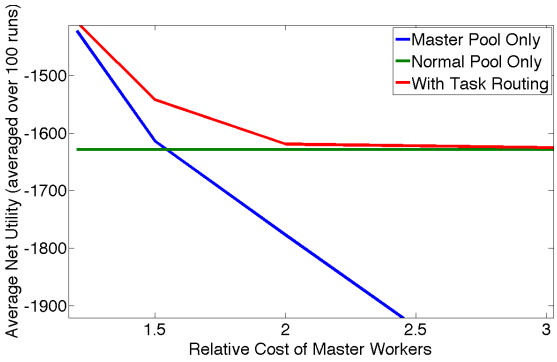

Figure 4. Variation in average net utility on varying relative cost of the master pool for $W_{master} \sim \mathcal{N}(0.75, 0.05)$.

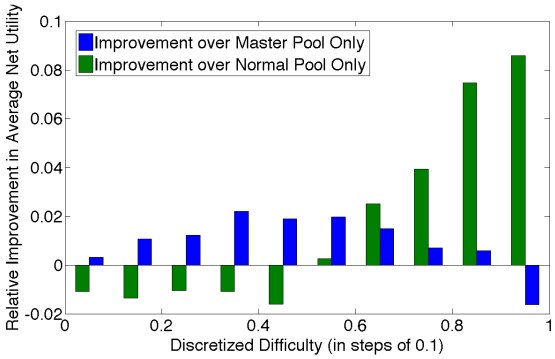

Figure 5. Improvement in net average utility (grouped into difficulty intervals of size 0.1) when $W_{master} \sim \mathcal{N}(0.25, 0.05)$ and relative master pool cost is 5. Averaged over 100 runs.

we would rarely (or never) want to use them, since it is unlikely we would get any clarity (subject to cost) from their answers.

We also examine the relative improvement in utility for different difficulty levels for the case when the relative master pool cost is 5. We observe that our model outperforms the master pool baseline for questions that have lower levels of difficulty (all except the highest difficulty level), and vastly outperforms the normal worker pool for questions that are more difficult, which demonstrates the efficacy of supplementing normal worker responses with those of experts.

We now vary the distribution of the master worker pool to $W_{master} \sim \mathcal{N}(0.5, 0.1)$, so that the master workers are closer to the normal workers in performance, and have higher variability (Figure 3). All other parameters are kept fixed at the same values. We find once again that our model outperforms or equals the baselines, and approaches the normal worker pool line at a lower relative cost than the previous case (Figure 3). This is natural since the expected quality differential between the pools is much lower than in the previous case, thus we would start overpaying the master pool much sooner in this case.

Lastly, we try $W_{master} \sim \mathcal{N}(0.75, 0.05)$. Our results follow the same pattern as the previous experiments (Figure 4). Thus, regardless of the difference in worker pool quality, our model is robust to changes in the relative pricing of the two baseline pools.

4.2. Simulation 2: Beta Difficulty Distributions

For the next set of experiments, we create a scenario that is likely closer in behavior to real crowdsourcing platforms. We draw question difficulties (for 100 questions) from a Beta distribution centered at 0.5 ($\sim \beta(2.0, 2.0)$), and simulate workers for each pool from the following gamma distributions: $W_{master} \sim \Gamma(3.5, 0.2)$ and $W_{normal} \sim \Gamma(4.0, 0.4)$.

The probability distribution functions for the distributions are shown in Figure 6. Our choice of worker error distributions is justified as follows: the normal worker pool is likely to contain a wide variability of workers, with a few 'rising' experts who have not been identified as such yet. On the other hand, on platforms such as Amazon Mechanical Turk, anecdotal evidence seems to suggest that master workers are not necessarily always perfect experts, and sometimes non-experts may in fact have expert qualifications. Our W_{master} distribution models this uncertain behavior. Lastly, our difficulty distribution suggests that most questions are neither too easy, and neither too hard, and should sufficiently model the average case scenario.

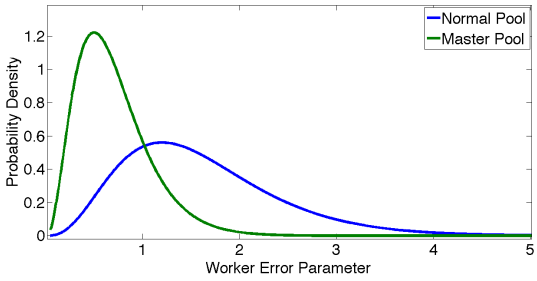

Figure 6. Gamma probability distribution functions of both worker pools.

We also vary the penalty incurred on returning an incorrect answer in these experiments. Increasing the incorrect answer penalty causes our POMDP to take more ballots (the high penalty is now offset by the extra cost of taking more ballots), and thus tend towards higher accuracy (although this may not always be the case).

Figures 7, 8 and 9 show the cost v.s. accuracy graphs for these simulations (with varying relative master pool costs). We find that our model lies both above (indicating higher accuracy at the same cost), and to the left (indicating lower

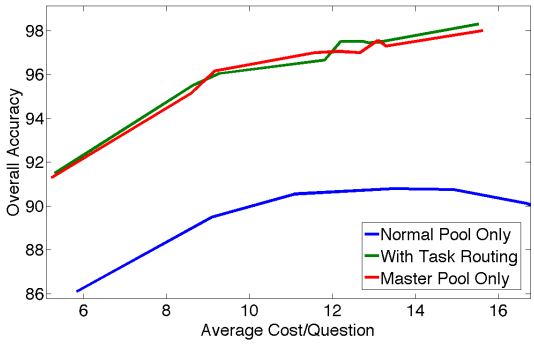

Figure 7. Overall accuracy v/s average cost/question for a $\beta(2, 2)$ difficulty distribution when relative master pool cost is 2. Averaged over 20 runs.

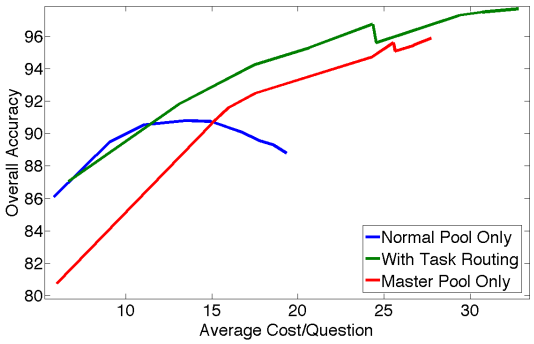

Figure 9. Overall accuracy v/s average cost/question for a $\beta(2, 2)$ difficulty distribution when relative master pool cost is 6. Averaged over 20 runs.

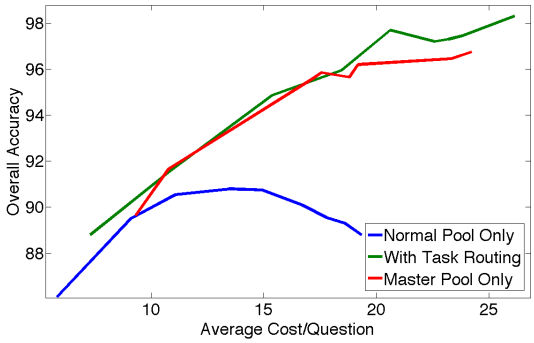

Figure 8. Overall accuracy v/s average cost/question for a $\beta(2, 2)$ difficulty distribution when relative master pool cost is 4. Averaged over 20 runs.

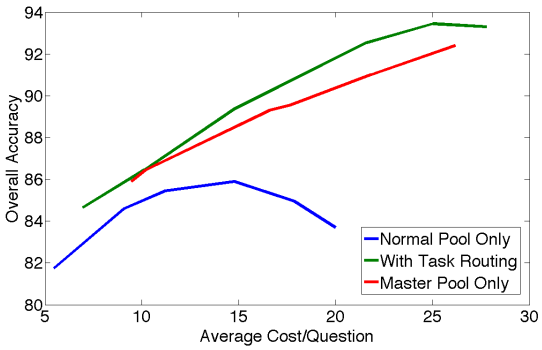

Figure 10. Overall accuracy v/s average cost/question for a $\beta(0.5, 0.5)$ difficulty distribution when relative master pool cost is 4. Averaged over 20 runs.

cost for the same accuracy) of the baselines in most cases.

As the relative cost of the master pool increases, our model outperforms the master pool baseline by an increasing margin. This is clearer at higher average costs (consequently higher budgets), since our model is able to make use of a large number of normal worker ballots (in the ratio of the relative master pool cost) in conjunction with answers given by the master workers, to achieve high levels of accuracy. On the other hand, our model performs similar to the normal pool baseline at low costs (when the relative master pool cost is 6), since to keep overall cost low, task routing to master workers becomes difficult when the requester budget is small (modeled using a low penalty for a wrong answer). Our model is thus robust to changes in relative pricing of both worker pools.

Also note the inability of the normal pool to cross the 91% overall accuracy mark. The performance of the normal pool is inhibited by the quality of workers in that pool, and thus taking a larger number of ballots from exclusively normal pool workers should not lead to an overall improvement in accuracy.

To achieve 95% accuracy in the case where the relative master pool cost is 6, we observe that it costs our model \$19.6, while the master pool baseline takes \$25.0 (the normal pool baseline doesn't achieve this accuracy at any price). Thus, our model gives savings of more than 20% in this setting (which translates to 500 for 100 questions).

Lastly, we look at the effect of changing the difficulty distribution to a Bi-Modal Beta Distribution ($\sim \beta(0.5, 0.5)$), where questions either have very high or very low difficulty. We find that for a relative master pool cost of 4 (Figure 10), our model continues to outperform both baselines by considerable margins.

5. Live Experiments

To carry out live experiments, we collected data from Amazon Mechanical Turk on 150 Named Entity Disambiguation questions (Lin et al., 2012) (which have $0/1$ answers) from the following 2 mutually exclusive worker pools:

- **Master Pool:** Categorization Master Workers with more than 5000 HITs and a 98%+ approval rate
- **Normal Pool:** Non-Master Workers with more than 100 HITs performed and a 95%+ approval rate

We paid 2.4 cents per question to each worker from the master pool, and 1.6 cents per question to each worker from the normal pool and collected 30 ballots per question. Using this data, we carried out live testing offline. We first estimated the $\overline{\gamma}$, average worker parameters, for each worker pool by running a modified version of Whitehill et al (2009)'s EM algorithm to estimate the individual error parameters of each worker, and then averaging over each pool. Surprisingly, the averages turned out to be extremely close to each other (0.94 v.s. 0.95), indicating that there was not much difference between the quality of the 2 pools. In fact, the worse average error belonged to the master pool!

To separate the two worker pools artifically we enforce that Masters are better than normal workers by pick the $\overline{\gamma}$ of the master pool to be 0.90, and that of the normal pool to be 1.00, thus making the master pool less error-prone. We learned the POMDP using these values for the average error parameters of the workers, and solved the 150 Entity Disambiguation questions multiple times using our learned POMDP. The relative cost of the master worker pool is fixed at 1.5 units (since we paid 8 cents over the normal pool) in all our experiments.

We find that the cost differential is much higher than the quality differential in this case, and our model only routes tasks to the normal worker pool, and gives an identical cost-quality tradeoff to that pool (as shown in Figure 11). However, we find, quite surprisingly that our master pool baseline performs *much worse* than the normal pool, indicating that Amazon Masters probably gave worse output in our experiments.

Overall, this is surprising and violates conventional wisdom. While Lin et al (2012)'s task set is not particularly difficult, it is definitely not extremely easy either, as observed in their prior work. This could be a first evidence suggesting that Master worker accuracy or sincerity is going down. Of course, more experiments are needed to verify if ours is an accidental observation or points to a general phenomenon. However, we at least did prove through live experiments that our model chooses not to switch much in case a worker pool is dominated by the other (higher price, not much higher or even worse quality).

While our live experiments could not thoroughly test our hypothesis, since the Master worker pool ended up being worse than Normal, extensive simulation experiments do give us strong belief that in cases where one worker pool is much better but also costlier our model will judiciously switch between the two to obtain significantly better cost-quality tradeoffs.

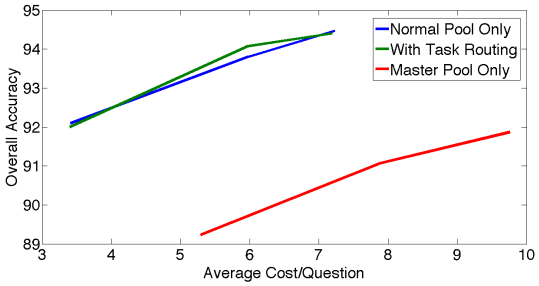

Figure 11. Overall accuracy v/s average cost/question when relative master pool cost is 1.5 in the live experiment. Averaged over 20 runs.

6. Conclusions and Future Work

In this paper, we have demonstrated the effectiveness of using a decision theoretic framework that leverages POMDPs for the problem of worker pool selection for binary classification tasks in a crowdsourced setting. Our contributions are firstly, that we identify the worker pool selection problem for crowdsourcing, and highlight its importance in achieving better cost-quality tradeoffs. Our proposed solution is a combination of unsupervised worker and question tracking with POMDPs for decision making. We find that our model always outperforms the baselines of selecting a single best worker pool and that its benefits are highest when the cost differentials between worker pools are intermediate. We also perform live experiments on Mechanical Turk for switching between regular workers and Master workers. To our surprise we find that Master workers are demonstrably worse than regular workers.

However, we have limited ourselves to providing information about average mean error parameter values for each worker pool, while learning individual worker gammas in an unsupervised fashion. An immediate area of focus would be to use Reinforcement Learning (RL) to allow online relearning of the POMDP that is being used for decision-making, based on changing information about the mean skill levels of different worker pools. This has the benefit of not requiring any information about the worker pools (except for costs). An RL approach would be able to determine (using an appropriate exploration-exploitation tradeoff) the individual performances of each worker pool and make decisions based on the expertise of each pool and their associated costs.

References

Ambati, Vamshi, Vogel, Stephan, and Carbonell, Jaime G. Towards task recommendation in micro-task markets. In *Human computation*, pp. 1–4. Citeseer, 2011.

Bragg, Jonathan, Kolobov, Andrey, and Weld, Daniel S. Parallel task routing for crowdsourcing. In *Second AAAI Conference on Human Computation and Crowdsourcing*, 2014.

Chen, Edwin. Making the Most of Mechanical Turk: Tips and Best Practices, 2012. URL blogs.echen.me.

Dai, Peng, Lin, Christopher H, Weld, Daniel S, et al. Pomdp-based control of workflows for crowdsourcing. *Artificial Intelligence*, 202:52–85, 2013.

Donmez, Pinar, Carbonell, Jaime G, and Schneider, Jeff. Efficiently learning the accuracy of labeling sources for selective sampling. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 259–268. ACM, 2009.

Donmez, Pinar, Carbonell, Jaime G, and Schneider, Jeff G. A probabilistic framework to learn from multiple annotators with time-varying accuracy. In *SDM*, volume 2, pp. 1. SIAM, 2010.

Ho, Chien-Ju and Vaughan, Jennifer Wortman. Online task assignment in crowdsourcing markets. In *AAAI*, volume 12, pp. 45–51, 2012.

Ho, Chien-Ju, Jabbari, Shahin, and Vaughan, Jennifer W. Adaptive task assignment for crowdsourced classification. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 534–542, 2013.

Ipeirotis, Panos. A computer scientist in a business school. Mechanical Turk vs oDesk: My experiences, 2012. URL http://www.behind-the-enemy-lines.com/.

Karger, David R, Oh, Sewoong, and Shah, Devavrat. Budget-optimal crowdsourcing using low-rank matrix approximations. In *Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on*, pp. 284–291. IEEE, 2011a.

Karger, David R, Oh, Sewoong, and Shah, Devavrat. Iterative learning for reliable crowdsourcing systems. In *Advances in neural information processing systems*, pp. 1953–1961, 2011b.

Karger, David R, Oh, Sewoong, and Shah, Devavrat. Efficient crowdsourcing for multi-class labeling. In *ACM SIGMETRICS Performance Evaluation Review*, volume 41, pp. 81–92. ACM, 2013.

Karger, David R, Oh, Sewoong, and Shah, Devavrat. Budget-optimal task allocation for reliable crowdsourcing systems. *Operations Research*, 62(1):1–24, 2014.

Lin, Christopher H, Daniel, Mausam, and Weld, S. Dynamically switching between synergistic workflows for crowdsourcing. In *In Proceedings of the 26th AAAI Conference on Artificial Intelligence, AAAI12*. Citeseer, 2012.

Shahaf, Dafna and Horvitz, Eric. Generalized task markets for human and machine computation. In *AAAI*, 2010.

Whitehill, Jacob, Wu, Ting-fan, Bergsma, Jacob, Movellan, Javier R, and Ruvolo, Paul L. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in neural information processing systems*, pp. 2035–2043, 2009.

Yan, Yan, Fung, Glenn M, Rosales, Rómer, and Dy, Jennifer G. Active learning from crowds. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 1161–1168, 2011.