Combining Human and Machine Intelligence in Large-scale Crowdsourcing

Ece Kamar
Microsoft Research
Redmond, WA 98052
eckamar@microsoft.com

Severin Hacker*
Carnegie Mellon University
Pittsburgh, PA 15289
shacker@cs.cmu.edu

Eric Horvitz
Microsoft Research
Redmond, WA 98052
horvitz@microsoft.com

ABSTRACT

We show how machine learning and inference can be harnessed to leverage the complementary strengths of humans and computational agents to solve crowdsourcing tasks. We construct a set of Bayesian predictive models from data and describe how the models operate within an overall crowdsourcing architecture that combines the efforts of people and machine vision on the task of classifying celestial bodies defined within a citizens' science project named Galaxy Zoo. We show how learned probabilistic models can be used to fuse human and machine contributions and to predict the behaviors of workers. We employ multiple inferences in concert to guide decisions on hiring and routing workers to tasks so as to maximize the efficiency of large-scale crowdsourcing processes based on expected utility.

Categories and Subject Descriptors

I.2 [**Distributed Artificial Intelligence**]: Intelligent agents

General Terms

Design, Algorithms, Economics

Keywords

crowdsourcing, consensus tasks, complementary computing, decision-theoretic reasoning

1. INTRODUCTION

Efforts in the nascent field of human computation have explored methods for gaining programmatic access to people for solving tasks that computers cannot easily perform without human assistance. Human computation projects include work on crowdsourcing, where sets of people jointly contribute to the solution of problems. Crowdsourcing has been applied to solve tasks such as image labeling, product categorization, and handwriting recognition. To date, computers have been employed largely in the role of platforms for recruiting and reimbursing human workers; the burden of managing crowdsourcing tasks and making hiring decisions has relied on manual designs and controls. However, interest has been growing in applications of learning and planning to crowdsourcing.

We investigate principles and algorithms for constructing crowdsourcing systems in which computer agents learn about tasks and about the competencies of workers contributing to solving the tasks, and make effective decisions for guiding and fusing multiple contributions. As part of this investigation, we demonstrate how we can leverage the complementary strengths of humans and computer agents to solve crowdsourcing tasks more efficiently. We describe the operation of key components and overall architecture of a methodology we refer to as CrowdSynth, and demonstrate the operation and value of the methods with data and workload drawn from a large-scale legacy crowdsourcing system for citizen science.

We focus on solving *consensus tasks*, a large class of crowdsourcing. With consensus tasks the goal is to identify a hidden state of the world by collecting multiple assessments from human workers. Examples of consensus tasks include *games with a purpose* (e.g., image labeling in the ESP game) [13], paid crowdsourcing systems (e.g., product categorization in Mechanical Turk) [6], and citizen science projects (e.g., efforts to classify birds or celestial objects). Consensus efforts can be subtasks of larger tasks. For example, a system for providing real-time traffic flow and predictions may contact drivers within targeted regions for reports on traffic conditions [8].

We describe a general system that combines machine learning and decision-theoretic planning to guide the allocation of human effort in consensus tasks. Our work derives from a collaboration with the Galaxy Zoo citizen science effort [1], which serves as a rich domain and source of data for evaluating machine learning and planning methods as well as for studying the overall operation of an architecture for crowdsourcing. The Galaxy Zoo effort was organized to seek help from volunteer citizen scientists on the classification of thousands of galaxies that were previously captured in an automated astronomical survey, known as the Sloan Digital Sky Survey (SDSS). The project has sought assessments via the collection of multiple votes from non-experts. Beyond votes, we have access to a database of SDSS image analysis data, containing 453 image features for each galaxy, which were extracted automatically via automated machine vision.

We shall describe how successful optimization of the engagement of people with Galaxy Zoo tasks hinges on models learned from data that have the ability to predict the ul-

*Severin Hacker contributed to this research during an internship at Microsoft Research.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

timate classification of a celestial objects, including objects that are undecidable, and of the next votes that will be made by workers. Such predictions enable the system to balance the expected benefit versus the costs of hiring a worker. We formalize Galaxy Zoo as a Markov Decision Process with partial observability, using likelihoods of outcomes generated by the predictive models for states of ground truth and for worker assessments. We demonstrate that exact computation of the expected value of hiring workers on tasks is infeasible because of the long horizon of Galaxy Zoo tasks. We present approximation algorithms and show their effectiveness for guiding hiring decisions. We evaluate the methods by drawing votes from the dataset collected from the Galaxy Zoo system during its operation in the open world. The evaluations show that the methods can achieve the maximum accuracy by hiring only 47% of workers who voted in the open-world run of the system. The evaluations call attention to the robustness of different algorithms to uncertainties in the inferences from the learned predictive models, highlighting key challenges that arise in fielding large-scale crowdsourcing systems.

2. RELATED WORK

Modeling workers and tasks has been an active area of crowdsourcing research. Whitehill et al. apply unsupervised learning to simultaneously predict the correct answer of a task, the difficulty of the task and the accuracy of workers based on some assumptions about the underlying relationships between the answer, the task, and workers [14]. Dai et. el. assume that worker reports are independent given the difficulty of tasks, and learn models of workers and task quality under this independence assumption [3].

Previous work on decision-theoretic reasoning for crowdsourcing tasks focused on tasks that can be decomposed into smaller tasks [10], and on workflows that are composed of an improve and verify step, which can be solved via methods that perform short lookaheads [3]. In a related line of work, researchers proposed greedy and heuristic approaches for active learning in crowdsourcing systems [11]. Our work differs from previous approaches in the generality of the Bayesian and decision-theoretic modeling, and in our focus on learning and executing expressive models of tasks and workers learned from real-world data.

3. SOLVING CONSENSUS TASKS

A task is classified as a *consensus task* if it centers on identifying a correct answer that is not known to the task owner and there exists a population of workers that can make predictions about the correct answer. Formally, let t be a consensus task and A be the set of possible answers for t. There exists a mapping $t \rightarrow \bar{a} \in A$ that assigns each task to a correct answer.

Figure 1 presents a schematic of components and flow of analysis of a utility-directed *consensus system*. The consensus system takes as input a consensus task. The system has access to a population of workers, who are able to report their noisy inferences about the correct answer. Given that $L \subseteq A$ is a subset of answers that the system and workers are aware of, a report of a worker includes the worker's vote, $v \in L$, which is the worker's prediction of the correct answer. The system can hire a worker at any time or may decide to terminate the task with a prediction about the correct answer of the task based on reports collected so far ($\hat{a}$). The goal of the system is to predict the correct answer of a given task based on potentially noisy worker reports while considering the cost of resources.

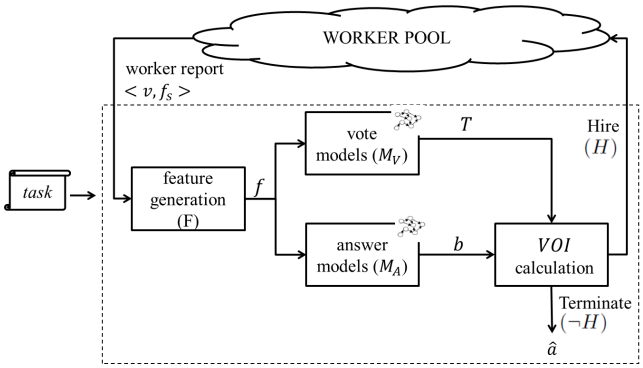

Figure 1: CrowdSynth: Key components and flow of analysis.

A successful system for solving consensus tasks needs to manage the tradeoff between making more accurate predictions about the correct answer by hiring more workers, and the time and monetary costs for hiring. We explore the opportunity to optimize parameters of this tradeoff by making use of a set of predictive models and a decision-theoretic planner.

The modeling component is responsible for constructing and using two groups of predictive models: *answer models* for predicting the correct answer of a given consensus task at any point during the process of acquiring votes, and *vote models* that predict the next state of the system by predicting the votes that the system would receive from additional workers should they be hired, based on the current information state. The modeling component monitors the worker population and task execution, and collects data about task properties and worker statistics, votes collected, and feedback received about the correct answer. A case library of execution data is used to build the answer and vote models.

We construct the answer and vote prediction models with supervised learning. Log data of any system for solving consensus tasks provides labeled examples of workers' votes for tasks. Labeled examples for training answer models are obtained from experts who identify the correct answer of a task with high accuracy. When expert opinion is not available, the consensus system may assume that the answer deduced from the reports of infinitely many workers according to a predetermined consensus rule is the correct answer of a given task (e.g., the majority opinion of infinitely many workers). To train answer models without experts, the system collects many worker reports for each task in the training set, deduces the correct answer for each task, and records either the consensus answer or the undecidable label.

Both answer and vote models are inputs to the planner. Vote models constitute the stochastic transition functions used in planning for predicting the future states of the model. The planner makes use of answer models for estimating the confidence on the prediction so that the planning component can decide whether to hire an additional worker.

The decision-theoretic planner models a consensus task as

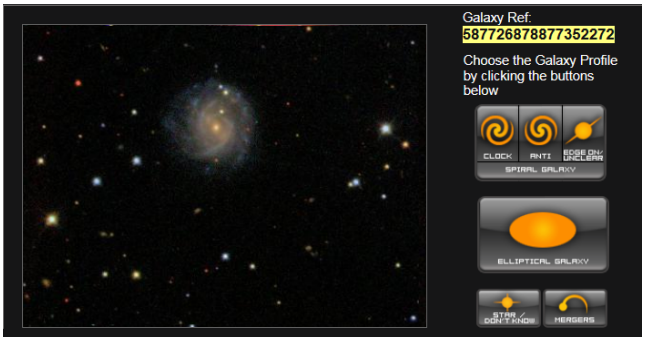

Figure 2: Galaxy Zoo interface for acquiring votes.

a Markov Decision Process (MDP) with partial observability. The MDP model is able to represent both the system's uncertainty about the correct answer and uncertainty about the next vote that would be received from workers. The planner computes the expected value of information (VOI) that would come with the hiring of an additional worker and determines whether the system should continue hiring (H) or terminate ($\neg H$) at any given state to maximize the total utility of the system. The utility is a combination of the reward (or punishment) of the system for making a correct (or incorrect) prediction and cost for hiring a worker.

3.1 Tagging Galaxies as a Consensus Task

In Galaxy Zoo, volunteers provide votes about the correct classifications of millions of galaxies that have been recorded in an automated sky survey [1]. Crowdsourcing provides a novel way for astronomers to reach a large group of workers around the world and collect millions of classifications under the assumption that the consensus of many workers provide the correct classification of a galaxy.

Figure 2 displays the main interface between the system and workers for collecting worker reports. The system displays images of celestial objects taken from SDSS and asks workers to classify them into 6 possible classes: elliptical galaxy, clockwise spiral galaxy, anticlockwise spiral galaxy, other spiral galaxy, and stars and mergers.

The dataset collected to date includes over 34 million worker reports obtained for 886 thousand unique galaxies. We use a subset of this dataset to train and test predictive models. We use another subset to simulate the real-time execution of the methodology within a prototype system named CrowdSynth and evaluate its performance under varying domain conditions.

4. PREDICTIVE MODELS FOR CONSENSUS TASKS

We now focus on the construction of predictive models for answers and votes.

4.1 Datasets

We shall perform supervised learning from a case library that includes log entries collected during the operation of the Galaxy Zoo system. Each log entry corresponds to a worker report collected for a galaxy. A worker report is a combination of a vote ($v_i \in L$), and information and statistics (f_{s_i}) about the worker delivered v_i. v_i represents a worker's prediction of the correct answer (e.g., elliptical) and f_{s_i} includes the worker's identity, the dwell time of the worker, the time and day the report is received. In addition to v_i and f_{s_i}, a log entry for a galaxy includes the visual features of a galaxy (SDSS features). We divided the celestial objects in the Galaxy Zoo dataset into a training set, a validation set and a testing dataset, each having 628354, 75005, and 112887 galaxies respectively.

We defined sets of features that summarize task characteristics, the votes collected for a task, and the characteristics of the workers reported for the task. f, the set of features for a galaxy, is composed of four main sets of features: f_0, *task features*, f_v, *vote features*, f_w, worker features, and f_{v-w}, *vote-worker features*. *Task features* include 453 features that are extracted automatically from sky survey images by making use of machine vision [9]. These features are available for each galaxy in the system in advance of votes from workers. *Vote features* capture statistics about the votes collected by the system at different points in the completion of tasks. These features include the number of votes collected, the number and ratio of votes for each class in L, the entropy of the vote distribution, and the majority class. *Worker features* include attributes that represent multiple aspects of the current and past performance, behaviors, and experience of workers contributing to the current task. These features include the average dwell time of workers on previous tasks, average dwell time for the current task, their difference, mean and variance of number of tasks completed in past, and average worker accuracy on aligning with the correct answer. We use the training set to calculate features about a worker's past performance. Finally, *vote-worker features* consist of statistics that combine vote distributions with worker statistics. These include such attributes as the vote by the most experienced worker, the number of tasks responded by a worker, the vote of the worker who has been most correct, and her accuracy.

A feature extraction function F takes a galaxy task and a history of worker reports $h_t = \{< v_1, f_{s_1} >, ..., < v_t, f_{s_t} >\}$, and creates f, the set of features described here, as input to the predictive models.

Based on an analysis on the dataset, the designers of the Galaxy Zoo system identified the following consensus rule: After hiring as many workers as possible for a celestial object (minimum 10 reports), if at least 80% of the workers agree on a classification (e.g., elliptical, spiral, etc.), that classification is assigned to the celestial object as the correct answer. Experts on galaxy classifications note that the correct answers assigned to galaxies with this rule agree with expert opinions in more than 90% of the cases, and thus using this rule to assign ground truth classification to galaxies does not significantly diminish the quality of the system [9]. In our experiments, we consider galaxies with at least 30 votes and apply this rule to generate labeled examples.

Not all galaxies in the dataset have votes with 80% agreement on a classification when all votes for that galaxy are collected. We classify such galaxies as "undecidable" and we define $A = L \cup \{undecided\}$, where L is the set of galaxy classes. Having undecidable galaxies introduces the additional challenge for the predictive models of identifying tasks that are undecidable, so that the system does not spend valuable resources on tasks that will not converge to a classification. M_A, the answer model, is responsible for deciding if a galaxy is decidable as well as identifying the correct galaxy

class if the galaxy is decidable without knowing the consensus rule that is used to assign correct answers to galaxies. Because the number of votes each galaxy has in the dataset varies significantly (minimum 30, maximum 95, average 44), predicting the correct answer of a galaxy at any step of the process (without knowing how many votes the galaxy has eventually) is a challenging prediction task. For example, two galaxies with the same vote distribution after 30 votes may have different correct answers.

We perform Bayesian structure learning to data from the case library to build probabilistic models that make predictions about consensus tasks. For any given learning problem, the learning algorithm selects the best predictive model by performing heuristic search over feasible probabilistic dependency models guided by a Bayesian scoring rule [2]. We employ a variant learning procedure that generates decision trees for making predictions.

4.2 Predicting Correct Classification

We now explore the challenge of predicting the correct answer of a consensus task based on noisy worker reports. We first implement several basic approaches as proposed by previous work [12], and then present more sophisticated approaches that can better represent the dependency relationships among different features of a task.

The most commonly used approach in crowdsourcing research for predicting the correct answer of a consensus task is majority voting. This approach does not perform well in the galaxy classification domain because it incorrectly classifies many galaxies, particularly the tasks that are undecidable.

Next, we implement two approaches that predict the correct answer using Bayes' rule based on the predictions of the following models: $M_A(\bar{a}, F(f_0, \emptyset))$, a prior model for the correct answer, and $M_{V'}(v_i, \bar{a}, F(f_0, h_{i-1}))$, a vote model that predicts the next vote for a task conditional on the complete feature set and the correct answer of the galaxy. Because v_i is the most informative piece of a worker's report and predicting f_{s_i} is difficult, we only use $M_{V'}$ model to predict a worker's report.

The Naive Bayes approach makes the strict independence assumption that worker reports are independent of each other given task features and the correct answer of the task. Formally, $Pr(\bar{a}|f)$, the likelihood of the correct answer being $\bar{a}$ given feature set f, is computed as below:

$$Pr(\bar{a}|f) = Pr(\bar{a}|F(f_0, h_t)) \approx M_A(\bar{a}, F(f_0, \emptyset)) \prod_{i=1}^{t} M_{V'}(v_i, \bar{a}, F(f_0, \emptyset))/Z_n$$

where Z_n is the normalization constant.

Next, we introduce an *iterative Bayes update* model that relaxes the independence assumptions of the Naive Bayes model. The iterative Bayes update model generates a posterior distribution over possible answers at time step t by iteratively applying the vote model on the prior model as given below:

$$Pr(\bar{a}|f) \propto Pr(\bar{a}|F(f_0, h_{t-1}))Pr(< v_t, f_{s_t} > |\bar{a}, F(f_0, h_{t-1}))/Z_b \approx M_A(\bar{a}, F(f_0, \emptyset)) \prod_{i=1}^{t} M_{V'}(v_i, \bar{a}, F(f_0, h_{i-1}))/Z_b$$

where Z_b is the normalization constant.

Another approach is building direct models for predicting the correct answer of a task. A direct model takes as input f, the complete set of features, and predicts $\bar{a}$. Figure 3 shows an example of a direct model trained for predicting the correct answer of a galaxy task.

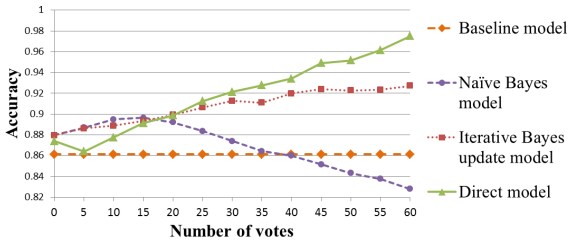

Figure 4: Comparison of accuracies of different models for predicting correct answer.

Figure 4 compares the accuracies of different answer models with a baseline model that classifies every task instance as the most likely correct answer in the training set. Both naive Bayes and iterative Bayes update models perform better than the direct models when the system has a small number of votes. However, the direct models outperform all others as the system collects more votes and as vote features become more predictive of the correct answer. When the system has 45 votes for a galaxy, the accuracy of direct models reach 95%. Based on the significantly stronger performance of the direct models for large numbers of votes, we use direct models in the consensus system.

4.3 Predicting the Next Vote

We also construct a model that predicts the next vote that a system would receive based on task features and worker reports collected so far. This model, symbolized as M_V, takes as input the complete feature set f. It differs from $M_{V'}$ in that the correct answer of a task ($\bar{a}$) is not an input to this model. M_V achieves 65% accuracy when 15 or more votes are collected. We compare the performance of M_V with a baseline model that simply guesses the most likely vote (elliptical galaxy), as the next vote. The comparison shows that M_V has better accuracy than the baseline when 10 or more votes are collected.

4.4 Predicting Termination

Although the system may decide to hire another worker for a task, the execution on a task may stochastically terminate because the system may run out of workers to hire or it may run out of time. Tasks logged in the Galaxy Zoo dataset are associated with different numbers of worker reports. The system has to terminate once all reports for a galaxy are collected. To model the distribution over votes received per task for Galaxy Zoo, we construct a probabilistic termination model from the training set (See Figure 5). The termination model predicts the probability of the system stochastically terminating after collecting different numbers of worker reports.

5. DECISIONS FOR CONSENSUS TASKS

At any time during the execution of the consensus system, the system needs to make a decision about whether to hire an additional worker for each task under consideration. If the system does not hire another worker for a task, it termi-

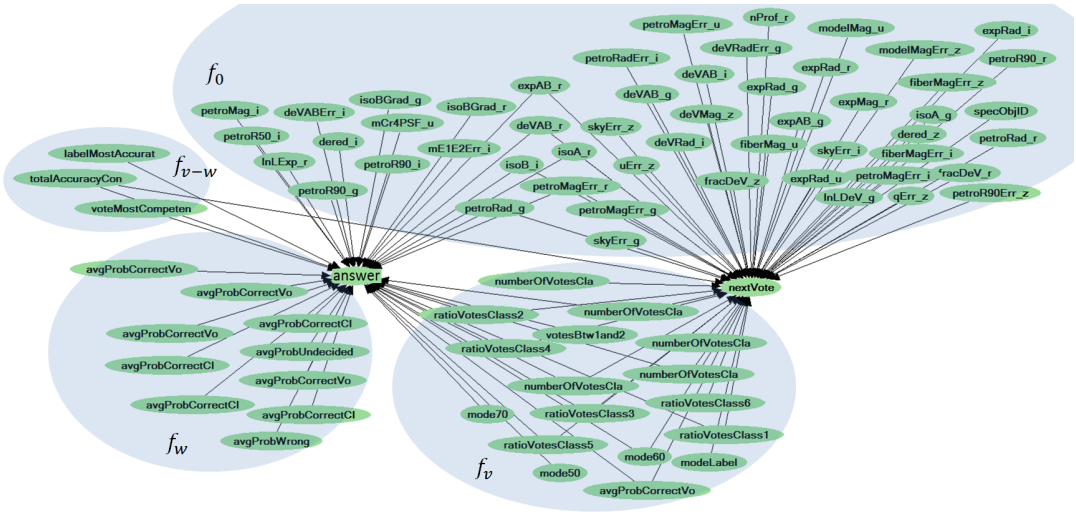

Figure 3: Direct model generated with Bayesian structure learning from Galaxy Zoo data. The model predicts correct answer of a task and next vote that the system would receive.

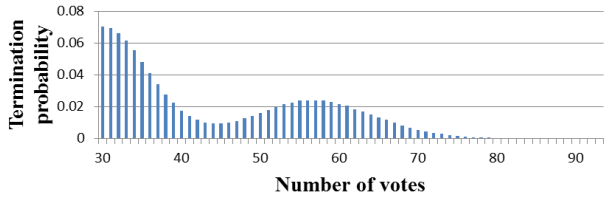

Figure 5: Termination probabilities estimated from training data.

nates and delivers the most likely answer that is predicted by the answer model. If the system decides to hire another worker, it collects additional evidence about the correct answer, which may help the system to predict the answer more accurately. But, hiring a worker incurs monetary and time costs. For solving consensus tasks effectively, the system needs to trade off the long-term expected utility of hiring a worker with the immediate cost. Deliberating about this tradeoff involves the consideration of multiple dimensions of uncertainty. The system is uncertain about the reports it will collect for a given task, and it is not able to observe $\bar{a}$, the correct answer of a consensus task. We shall model this decision-making problem as an MDP with partial observability, which makes calls to the answer and next vote models. We show that exact solutions of consensus tasks over long horizons is intractable and present approximate algorithms for estimating the expected value of hiring a worker.

5.1 Modeling Consensus Tasks

A consensus task is partially observable because the consensus system cannot observe the correct answer of the task. For simplicity of representation, we model a consensus task as an MDP with uncertain rewards, where the reward of the system at any state depends on its belief about the correct answer. A consensus task is formalized as a tuple $< S, \mathbb{A}, T, R, l >$. $s_t \in S$, a state of a consensus task at time t, is composed of a tuple $s_t =< f_0, h_t >$, where f_0 is the set of task features initially available, and h_t is the complete history of worker reports received upto time t.

$\mathbb{A}$, the set of actions for a consensus task include H, hire a worker, and $\neg H$, terminate and deliver the most likely answer to the task owner. $T(s_t, \alpha, s_{t+1})$ is the likelihood of transitioning from state s_t to s_{t+1} after taking action α. The transition function represents the system's uncertainty about the world and about worker reports. The system transitions to a terminal state if the selected action is $\neg H$. If the system decides to hire a worker, the transition probability to a next state depends on likelihoods of worker reports and the likelihood of termination. A worker report is a combination of v_i, worker's vote, and f_{s_i}, the set of features about the worker. To predict the likelihood of a worker report, we use the next vote model, and use average worker statistics computed from the training data to predict f_{s_i}.

The reward function $R(s_t, \alpha)$ represents the reward obtained by executing action α in state s_t. The reward function is determined by the cost of hiring a worker, and the utility function $U(\hat{a}, \bar{a})$, which represents the task owner's utility for the system predicting the correct answer as $\hat{a}$ when it is $\bar{a}$. For the simple case where there is no chance of termination, $R(s_t, H)$ is assigned a negative value which represents the cost of hiring a worker. The value of $R(s_t, \neg H)$ depends on whether the answer that would be revealed by the system based on task features and reports collected so far is correct. b_t is a probability distribution over set A that represents the system's belief about the correct answer of the task, such that for any $a \in A$, $b_t(a) = M_A(a, F(f_0, h_t))$. Let $\hat{a}$ be the most likely answer according to b_t, the reward function is defined as $R(s_t, \neg H) = \sum_{\bar{a}} b_t(\bar{a}) U(\hat{a}, \bar{a})$.

We model consensus tasks as a finite-horizon MDP. l, the horizon of a task, is determined by the ratio of the maximum reward improvement possible (e.g., the difference between the reward for making a correct prediction and the punishment of making an incorrect prediction) and the cost for hiring an additional worker.

A policy π specifies the action the system chooses at any state s_t. An optimal policy π^* satisfies the following equation for a consensus task of horizon l.

$$V^{\pi^*}(s_l) = max_{\alpha \in \mathbb{A}} R(s_l, \alpha)$$
$$V^{\pi^*}(s_t) = max_{\alpha \in \mathbb{A}} \big(R(s_t, \alpha) + \sum_{s_{t+1}} T(s_t, \alpha, s_{t+1})\, V^{\pi^*}(s_{t+1})\big)$$

Now, we can calculate the value of information (VOI) for any given initial state s_i.

$$\begin{aligned} VOI(s_i)) &= V^H(s_i) - V^{\neg H}(s_i) \\ &= R(s_i, H) + \sum_{s_{i+1}} T(s_i, H, s_{i+1})\, V^{\pi^*}(s_{i+1}) \\ &\quad -R(s_i, \neg H) \end{aligned}$$

VOI is the expected value of hiring an additional worker in state s_i. It is beneficial for the consensus system to hire an additional worker when VOI is computed to be positive.

5.2 Solving Consensus Tasks Efficiently

A state of a consensus task at any time step is defined by the history of observations collected for the task. The state space that needs to be searched for computing an optimal policy for a consensus task grows exponentially in the horizon of the task. For large horizons, computing a policy with an exact solution algorithm is infeasible due to exponential complexity. For example, an average Galaxy Zoo task includes 44 worker reports, and the horizon of such a task can be up to 93 time steps.

Myopic decision making and k-step lookahead search are approaches proposed by previous work for approximating the value of information efficiently [5, 4]. These approaches could perform well for solving consensus tasks, if collecting a small set of evidence changed the system's prediction of the correct answer. This condition is unlikely to be satisfied by consensus tasks where worker's reports each provide only weak evidence about the correct answer, and the system needs to reason about the value of collecting a large set of worker reports. For instance, there exists a set of Galaxy Zoo tasks with some particular initial features such that even obtaining 10 consecutive worker reports of the same galaxy label is not enough to change the system's current opinion about the correct answer. Thus, a limited lookahead search has little chance of improving the predictions of the system for this subset of tasks in a reasonable amount of computation time.

Monte-Carlo planning has been used to solve large *fully observable* MDPs [7]. We move to investigate sampling-based solution algorithms, which can be employed in *partially observable* real-world systems for solving consensus tasks accurately and efficiently. These algorithms use Monte-Carlo sampling to perform long lookaheads up to the horizon and to approximate the value of information. Instead of searching a tree that may be intractable in size, this approach samples execution paths (i.e., histories) from a given initial state to a terminal state. For each execution path, it estimates $V^{\neg H}$, the value for terminating at the initial state, and V^H, the value for hiring more workers and terminating later. The value of information is estimated as the difference of these values averaged over a large number of execution path samples. We introduce two algorithms that use this sampling approach to approximate VOI, but differ in the way they estimate V^H. The *lower-bound sampling* (LBS) algorithm picks a single best termination point in the future across all execution paths, V^H is assigned the expected value of this point. The *upper-bound sampling* (UBS) algorithm optimizes the best state for termination for each execution path individually. V^H is estimated by averaging over the values for following these optimal termination strategies. Both algorithms decide to hire an additional worker if VOI is computed to be positive. After hiring a new worker and updating the current state by incorporating new evidence, the algorithms repeat the calculation of VOI for the new initial state to determine whether to hire another worker.

For any given consensus task modeled as an MDP with partial observability, and any initial state s_i, a next state is sampled with respect to the transition function; the likelihood of sampling a state is proportional to the likelihood of transitioning to that state from the initial state. Future states are sampled accordingly until a terminal state is reached. Because sampling of future states is directed by the transition function, the more likely states are likely to be explored. For each state s_t^j on path j, $\hat{a}_t^j$ is the answer predicted based on the current state. When a terminal state is reached, the correct answer for path j, $\bar{a}^j$, is sampled according to the system's belief about the correct answer at this terminal state, when the system is most confident about the correct answer. An execution path from the initial state s_i to a terminal state s_n^j is composed of each state encountered on path j, the corresponding predictions at each state, and the correct answer sampled at the end. It is represented by the tuple: $p^j =< s_i, \hat{a}_i, s_{i+1}^j, \hat{a}_{i+1}^j, ..., s_n^j, \hat{a}_n^j, \bar{a}^j >$.

An execution path represents a single randomly generated execution of a consensus task. For any given execution path, there is no uncertainty about the correct answer or the set of observations that would be collected for the task. Sampling an execution path maps an uncertain task to a deterministic and fully observable execution. To model different ways a consensus task may progress (due to the uncertainty about the correct answer and the worker reports), a library of execution paths (P) is generated by repeating the sampling of execution paths multiple times. This library provides a way to explore long horizons on a search tree that can be intractable to explore exhaustively. If the library includes infinitely many execution paths, it constitutes the complete search tree.

Given an execution path p^j that terminates after collecting n reports, $V_k(p^j)$ is the utility for terminating on this path after collecting k-many worker reports. $V_k(p^j)$ is computed with respect to the answer predicted based on the worker reports collected in the first k steps and the correct answer sampled at the terminal state. Given that c is the cost for hiring a worker, $V_k(p^j)$ is defined as follows:

$$V_k(p^j) = \begin{cases} U(\hat{a}_k^j, \bar{a}^j) - kc & \text{if } k \leq n \\ U(\hat{a}_n^j, \bar{a}^j) - nc & \text{if } n < k \leq l \end{cases}$$

For simplicity of presentation, we assume a constant cost for hiring workers. The definition of $V_k(p^j)$ and consequently LBS and UBS algorithms can be easily generalized to settings in which worker costs depend on the current state.

We define $V^{\neg H}$ with respect to execution path library P as given below:

$$V^{\neg H}(s_i) = \sum_{p^j \in P} V_i(p^j)/|P|$$

The lower-bound sampling (LBS) algorithm approximates V^H as given below:

$$V^H(s_i) = \max_{i<k\leq l}(\sum_{p^j \in P} V_k(p^j)/|P|)$$

LBS picks the value of the best termination step in average for all execution paths. This algorithm underestimates V^H because it picks a fixed strategy for future, and does not optimize future strategies with respect to different worker reports that could be collected in future states. LBS is a pessimistic algorithm; given that the MDP model provided to the algorithm is correct and the algorithm samples infinitely many execution paths, all hire (H) decisions made by the algorithm are optimal.

The upper-bound sampling (UBS) approximates V^H by optimizing the best termination step individually for each execution sequence:

$$V^H(s_i) = \sum_{p^j \in P}(\max_{i<k\leq l} V_k(p^j)/|P|)$$

In distinction to the LBS algorithm, the UBS algorithm overestimates V^H by assuming that both the correct state of the world and future state transitions are fully observable, and thus by optimizing a different termination strategy for each execution sequence. The UBS algorithm is an optimistic algorithm; given that the MDP model provided to the algorithm is correct and the algorithm samples infinitely many execution paths, all not hire ($\neg H$) decisions made by the algorithm are optimal. In the next section, we empirically evaluate the performance of LBS and UBS algorithms on a dataset collected from the Galaxy Zoo system.

6. EXPERIMENTS

We evaluated the ability of the CrowdSynth prototype to guide the solution of consensus tasks on a subset of the testset collected from the Galaxy Zoo project. The testset includes 44350 votes collected for 1000 randomly selected galaxies, and 453 SDSS image features describing each galaxy. We evaluated variations of the system by employing different decision-making algorithms.

The MDP used in CrowdSynth for modeling Galaxy Zoo tasks includes the belief update functions and transition functions learned from real-world data, as described earlier. These predictive models are not perfect; they can be noisy, there can be inconsistencies between consecutive beliefs. This study also helps to evaluate the effect of the noise in the building blocks of an MDP on the performance of different MDP solution algorithms.

6.1 Results

We compare the performance of the decision-theoretic CrowdSynth methodology to two baselines. The first baseline is named *no hire* as it hires no workers and delivers the most likely answer prediction based on the features extracted digitally. The second baseline collects all worker reports available for a task and makes a prediction about the correct answer afterwards. We name this baseline *hire all*. We also implemented myopic and k-step lookahead algorithms, which have been proposed by previous work to estimate VOI. In these experiments, the system is rewarded \$1 for correctly predicting the correct answer of a task (including predicting undecidables), and the cost of hiring a worker is varied between \$0.1 and \$0.0001. The LBS and UBS algorithms used in our investigations terminate after 2000 samples.

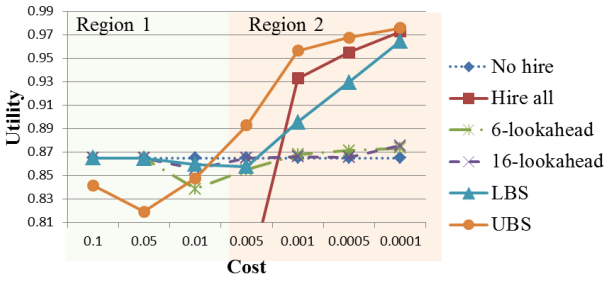

Figure 6: Performance of decision-making models with variation of worker costs.

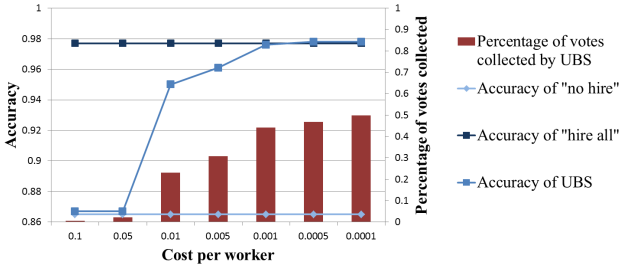

Figure 7: Analysis of behavior of UBS algorithm for varying worker costs.

Figure 6 summarizes the performances of different decision-making algorithms and baselines as a function of the cost of a worker. We divide the figure into two regions of worker costs: high worker costs (Region 1) and low worker costs (Region 2). For high worker costs, none of the decision-theoretic algorithms are able to perform better than the *no hire* baseline, because the expected cost for hiring enough workers to change the system's prediction of the correct answer is as high as or higher the expected benefit.

As shown in Figure 3, predictions of direct answer models are noisier when a few worker reports are collected than when no reports are collected. In Region 1, all decision-theoretic algorithms are affected by this noise because the lookahead depth is relatively short. In addition, the UBS algorithm is affected negatively by the overestimation of VOI in this region.

When the cost of a worker is low, the UBS algorithm performs significantly better than all other algorithms and baselines. The performance of the LBS algorithm is negatively affected by the underestimation of VOI in this region. k-step lookahead algorithms are outperformed by UBS and by LBS (except when cost is 0.005), because for many Galaxy Zoo tasks, even having 16 steps lookahead may not be enough to properly estimate VOI. Overall, the UBS algorithm outperforms the default policy used in the Galaxy Zoo effort (*hire all*) for all worker costs, including high worker costs.

The decision-theoretic approach can perform better than the hire all baseline for varying cost values as it successfully trades off the estimated utility for hiring a worker with the

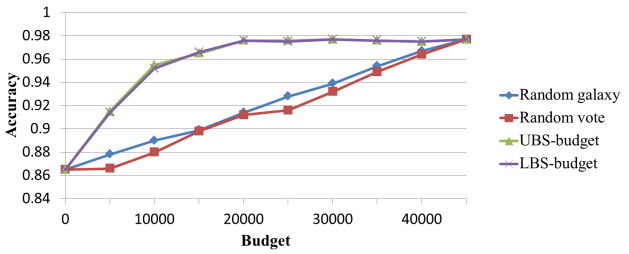

Figure 8: Comparison of decision-making policies under a fixed budget.

cost of doing so. Figure 7 reports the accuracy of the UBS algorithm and the percentage of votes collected by the algorithm for varying cost values. When the cost for hiring a worker is very high, the algorithm hires very few workers (less than 1 worker per celestial object), which results in a slight improvement in accuracy. The algorithm gradually improves its accuracy in predicting the correct answer by hiring more workers as the cost decreases. The algorithm reaches 95% accuracy by collecting only 23% of the reports, it reaches the accuracy of the *hire all* policy by collecting only 47% of the votes. The algorithm is able to improve its accuracy by hiring only a subset of the votes because it can distinguish the set of galaxies for which its decision is likely to change in the future.

For the next set of experiments, we modify the problem so that hiring decisions are not influenced by the cost of hiring a worker, but instead by varying limitations in budget. The budget constrains the total number of workers that can be hired for 1000 celestial objects. The challenge for the system is distributing a limited budget among 1000 different objects to achieve the highest prediction accuracy. We experiment with four decision-making models. The first model, *random galaxy*, randomly selects a celestial object and collects all available votes for that object. The *random vote* model is the approach followed by the original Galaxy Zoo system. This model selects a random object and collects a single vote for that object at each iteration until it runs out of budget. *UBS*-budget and *LBS*-budget models calculate VOI for each celestial object with the UBS and LBS algorithms, and hire a worker with the highest VOI.

Figure 8 compares the performance of these models for varying budgets. Both UBS and LBS models outperform other approaches for all budget sizes. After collecting 20000 votes, the accuracy of the VOI models converge as the system has collected all the evidence it needs to make accurate predictions.

7. DISCUSSION AND FUTURE WORK

We reviewed our efforts to take a principled end-to-end approach to consensus crowdsourcing. We composed a system that combines machine vision, machine learning, and decision-theoretic planning to make effective decisions about when to hire workers and how to perform classifications when observations cease. We constructed a prototype system and evaluated our learning and decision-making techniques on real-world data collected during the operation of the Galaxy Zoo system in the open world. The experiments demonstrate that the methodology can solve consensus tasks accurately and achieve significant savings in worker resources by intelligently allocating resources.

We are exploring extensions of the methods that can reason about optimal timing and pricing of tasks. We have been investigating models that can make predictions about *individual* workers, so that the decision-theoretic planner can make effective decisions about the best worker to hire and the best task to assign to workers who come available. We are also investigating Monte-Carlo approaches that can more accurately estimate VOI. Finally, we are studying challenges with the development of online crowdsourcing services that have the ability to continue to learn from data, combine reports in a coherent manner, and ideally route people and tasks with the Bayesian and decision-theoretic procedures that we have described.

8. ACKNOWLEDGMENTS

We thank Paul Koch for assistance with accessing Galaxy Zoo data, Chris Lintott for sharing the Galaxy Zoo data, and Dan Bohus, Rich Caruana, Paul Koch, and Chris Lintott, for discussions and feedback.

9. REFERENCES

[1] Galaxy Zoo, 2007, http://zoo1.galaxyzoo.org/.

[2] D. Chickering, D. Heckerman, and C. Meek. A Bayesian approach to learning Bayesian networks with local structure. In *UAI'97*, 1997.

[3] P. Dai et al. Artificial intelligence for artificial artificial intelligence. In *AAAI*, 2011.

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

[5] D. Heckerman, E. Horvitz, and B. Nathwani. *Toward normative expert systems: The Pathfinder project.* Stanford University, 1991.

[6] P. Ipeirotis. Analyzing the Amazon Mechanical Turk marketplace. *XRDS: Crossroads, The ACM Magazine for Students*, 17(2):16–21, 2010.

[7] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. *Machine Learning: ECML 2006*, pages 282–293, 2006.

[8] A. Krause, E. Horvitz, A. Kansal, and F. Zhao. Toward community sensing. In *IPSN*. IEEE, 2008.

[9] C. Lintott et al. Galaxy Zoo: Morphologies derived from visual inspection of galaxies from the Sloan Digital Sky Survey? *Monthly Notices of the Royal Astronomical Society*, 389(3):1179–1189, 2008.

[10] D. Shahaf and E. Horvitz. Generalized task markets for human and machine computation. In *AAAI*, 2010.

[11] V. Sheng, F. Provost, and P. Ipeirotis. Get another label? Improving data quality and data mining using multiple, noisy labelers. In *ACM SIGKDD*, 2008.

[12] R. Snow, B. O'Connor, D. Jurafsky, and A. Ng. Cheap and fast but is it good?: Evaluating non-expert annotations for natural language tasks. In *EMNLP*, 2008.

[13] L. Von Ahn and L. Dabbish. Designing games with a purpose. *Communications of the ACM*, 2008.

[14] J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. *NISP*, 2009.