Artificial Intelligence and Collective Intelligence

Daniel S. Weld
University of Washington (CSE)
Seattle, WA
weld@cs.washington.edu

Mausam
Indian Institute of Technology
Delhi, India
mausam@cse.iitd.ac.in

Christopher H. Lin
University of Washington (CSE)
Seattle, WA
chrislin@cs.washington.edu

Jonathan Bragg
University of Washington (CSE)
Seattle, WA
jbragg@cs.washington.edu

The vision of *artificial intelligence* (AI) is often manifested through an autonomous software module (agent) in a complex and uncertain environment. The agent is capable of thinking ahead and acting for long periods of time in accordance with its goals/objectives. It is also capable of learning and refining its understanding of the world. The agent may accomplish this based on its own experience, or from the feedback provided by humans. Famous recent examples include self-driving cars (Thrun 2006) and the IBM Jeopardy player Watson (Ferrucci et al. 2010). This chapter explores the immense value of AI techniques for collective intelligence, including ways to make interactions between large numbers of humans more efficient.

By defining *collective intelligence* as "groups of individuals acting collectively in an intelligent manner," one soon wishes to nail down the meaning of *individual*. In this chapter, individuals may be software agents and/or people and the collective may consist of a mixture of both. The rise of collective intelligence allows novel possibilities of seamlessly integrating machine and human intelligence at a large scale – one of the holy grails of AI (known in the literature as *mixed-initiative systems* (Horvitz 2007)). Our chapter focuses on one such integration – the use of machine intelligence for the management of *crowdsourcing* platforms (Weld, Mausam, and Dai 2011).

Crowdsourcing is a special case of collective intelligence, where a third party (called the *requestor*) with some internal objective solicits a group of individuals (called *workers*) to perform a set of inter-related tasks in service of that objective. The requestor's objective may be expressed in the form of a utility function to be maximized. For example, a requestor might wish to obtain labels for a large set of images; in this case, her utility function might be the average quality of labels subject to a constraint that no more than $ X dollars be spent paying workers. We assume that the workers act independently, interacting only through the shared tasks. Each worker has an individual utility function, which is often different from the collective's utility function. Furthermore, we assume that their utility functions are independent of each other. The AI subfield of *multi-agent systems* considers even richer models, in which individual agents may reason about the objectives of other agents, negotiate, and bargain with each other (Weiss 2013). We won't discuss these techniques here, but the chapter on game theory explores some of these issues.

There are two natural points of connection between AI & crowdsourcing: 1) AI for crowdsourcing and 2) crowdsourcing for AI. While this chapter centers on the former we note that in recent years crowdsourcing has had a significant impact on AI research as well – a great many projects use crowdsourcing to label training sets as input for data-hungry supervised learning algorithms (Snow et al. 2008a; Callison-Burch 2009; Hsueh, Melville, and Sindhwani 2009).

Why does crowdsourcing need AI? Crowdsourcing is an effective medium for congregating a large set of workers (usually virtually) who assist with a common goal. This allows for creative new applications that use the wisdom of crowds or the round-the-clock availability of people (e.g., (Bigham et al. 2010)). At the same time, the shear volume of tasks, and highly varying skills and abilities of workers typically make it infeasible to manually manage the task allocation as well as quality control. Moreover, the design of crowdsourced interfaces and workflows to accomplish a new task remains cumbersome and expensive. For example, often a task may get routed to a worker not skilled enough or interested in it. Different tasks may require slightly different workflows to achieve high quality. Different task instances may be individually easier or more difficult, requiring less or more work (iterations) on them. These and other challenges necessitate the use of automated techniques for the design and management of crowdsourcing processes.

A long-term vision of AI for crowdsourcing is to enable optimal design of workflows and management of task instances, thereby making crowdsourcing platforms highly efficient, saving thousands of man-hours and millions of dollars, and also making crowdsourcing really easy to use for a novice requestor. AI is a natural fit for this vision because, in general, AI algorithms are great at building models, drawing inferences, and detecting outliers from the data. They are also effective in taking decisions in uncertain environments towards maximizing an objective. In this chapter, we discuss several uses of AI in this space – we describe learning algorithms that model the accuracy of crowd members, aggregation methods for predicting true answers from error-prone and disagreeing workers, and AI control algorithms that choose which tasks to request and which individuals should work on them.

- *requestor* - an entity who is assembling a crowd for an objective.
- *objective* - what the requestor is trying to accomplish
- *worker* - an entity answering questions or performing tasks.
- *task* - what a worker is asked to do. Often responding to a multiple choice *question*.
- *response* - what a worker returns when given a question; also called a label.
- *answer* - the true, objective answer to a question, when one exists; ideally a worker's response will be the answer, but sometimes workers make mistakes.
- *workflow* - a set of tasks, usually interrelated, which are given to workers to perform. Some tasks may be performed automatically by programs, but most are given to human workers.

Figure 1: Terminology used in this chapter.

Preliminaries

Because different individuals in a crowdsourced collective may have differing priorities, it's important to be clear on terminology; please refer to Figure 1. We'll consider a variety of objectives in the chapter, but the most common objective is to accurately label a set of examples; here the requestor needs to choose how many workers should be given a labeling task and how their responses should be aggregated to estimate the best answer (i.e., the most likely answer). Usually, the requestor's objective includes minimizing the number of tasks given to workers, either because the workers are paid per task or just to avoid burdening volunteer workers. Sometimes, however, a requestor is interested in minimizing *latency* — *i.e.*, being able to compute an answer quickly — and this may require additional tasks, as we explain in the chapter's final section.

Either way, we focus on algorithms for helping the requestor decide what to do. As you'll see these methods include different types of machine learning, expectation maximization, optimization, policy construction for partially-observable Markov decision processes (POMDPs), and reinforcement learning.

The choice of algorithm depends not just on the requestor's objective but also on the *labor market*, an economic term we'll use even if the workers are volunteers and not being paid. In some markets, like Amazon Mechanical Turk, a requestor can post tasks to the market and workers get to choose which task they wish to attempt. In other markets, one can directly assign specific tasks to individual workers. In all cases it turns out to be useful to track workers' responses in order to construct a model of their accuracy and maybe which kinds of tasks they enjoy. In theory, a human worker is capable of performing an arbitrarily complex task, but we focus on simple jobs known as *microtasks*. For example, we consider small tasks like multiple-choice questions, writing or editing a short text description, or drawing a bounding box on an image. Often a complex objective can be achieved by a *workflow* comprised of these simple tasks. Finally, as mentioned in the introduction, we assume that individual workers are largely independent of each other; as a result, our approaches may not work well if malevolent workers collude in order to deceive the requestor. Fortunately, such behavior is extremely rare in practice.

The rest of this chapter has a simple organization. The next section discusses how to best interpret the output of error-prone workers. The final section discusses the control problem "What's the best job to do next and who should work on it?" considering objectives such as minimizing the amount of labor required or minimizing the time necessary to perform the task.

Collective Assessment & Prediction

To account for variability in worker skill, crowdsourcing requesters often ask multiple workers to perform the same (or related) tasks and then aggregate responses to infer the correct answers. In practice, the effectiveness of crowdsourcing can be highly dependent on the method for aggregating responses, and numerous strategies have been investigated. In this section, we describe a set of "post hoc" response aggregation methods that attempt to glean as much information as possible after task responses have been received. In the next section we turn to more advanced methods that seek to assign a given worker the most important task to perform or the most informative question to answer.

Suppose that an AI system is given a set of multiple-choice questions, a set of workers, and a set of their responses such that some questions are answered by more than one worker. We assume that the questions are objective, i.e. each has a unique correct answer, but workers may not answer correctly. Finally, we assume that the majority of workers are more likely to be correct than to make a mistake.[1] Artificial intelligence systems can work without information beyond the workers' proposed answers. The system's objective is to infer the correct answers from the noisy worker responses.

While we can mitigate the negative effects of imperfect workers in many ways, the techniques we consider in this section revolve around two common patterns. The first idea is to exploit redundancy, comparing different workers' responses to the same question. Indeed, Snow *et al.* (Snow et al. 2008b) found that simple majority voting allowed a crowd of novices to outperform an expert on natural language labeling tasks, like sentiment analysis and judging word similarity. The second common pattern is to learn and track the skills of the workers. Rather than a simple majority vote, these approaches weigh worker responses by using models of workers' abilities. In the simplest case, such a model might be a single "accuracy" number, but models can grow arbitrarily complex. For example, if one knew a worker was excellent at translating French to English, one might suspect that her English to French translations would also be of high quality.

Simple Approachs to Collective Assessment

We first consider algorithms that improve on vanilla majority voting by modeling worker skill. These are the bread

[1] If the majority of workers are likely to agree on an incorrect answer, then more sophisticated methods, like Bayesian Truth Serum, are necessary to reveal the right answer (Prelec and Seung 2007).

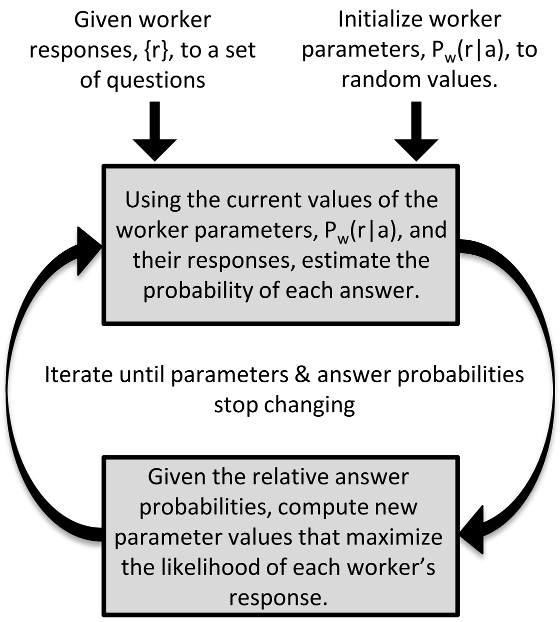
Given worker responses, {r}, to a set of questions
Initialize worker parameters, P_w(r|a), to random values.
Using the current values of the worker parameters, P_w(r|a), and their responses, estimate the probability of each answer.
Iterate until parameters & answer probabilities stop changing
Given the relative answer probabilities, compute new parameter values that maximize the likelihood of each worker's response.

Figure 2: Expectation maximization repeats two steps until convergence, alternately estimating the best answers, then updating its model of workers.

and butter algorithms. The simplest approach, and one commonly used in practice, uses supervised learning, which gives workers questions for which "gold standard" answers are already known (Dai, Mausam, and Weld 2011a). Workers who fail to correctly answer gold-standard questions are dismissed or have their weights lowered. To avoid gaming behavior (e.g., where a human might answer the first few questions and then, after convincing the system of his or her aptitude, unleash a simple bot to answer remaining questions, likely with greatly reduced accuracy), it is common to intermix questions with known and unknown answers. However, even this strategy is foiled by scammers building bots which utilize databases of known questions, leading to elaborate strategies for programmatically generating an unbounded number of gold-answer questions (Oleson et al. 2011).

Collective Assessment using Expectation Maximization

More sophisticated approaches eschew gold-answer questions entirely, instead using unsupervised learning to jointly estimate worker accuracy and consensus answers together. As a first example, we consider early work by Dawid and Skene (Dawid and Skene 1979). Although they originally pose their approach in terms of medical diagnosis, it clearly fits the crowdsourcing model presented above. There is a single question with an unknown correct answer and also parameters, $P_w(r|a)$, for each worker and each possible response, describing the probability that worker w will give response r when the true answer is a. These probabilities can be seen as a very simple model of worker abilities — an expert worker would have $P_w(r|a)$ close to zero for all $r \neq a$. Dawid and Skene make an important simplification — they assume that the worker's responses are conditionally independent of each other given the true answer. In other words, if we already knew the true answer, then our estimate of $P_w(r|a)$ should not be affected regardless of how other workers answer the question. David and Skene use an iterative algorithm called *expectation-maximization* (EM) to estimate which answers are correct at the same time that the algorithm learns the model of worker accuracies.

EM embodies the intuition that a good worker is one whose answers agree with those of other workers. More precisely, a great worker's answers agree with those of other good workers. Unfortunately, this idea yields a chicken-and-egg dilemma: how can you score one worker without already knowing the quality of her peers? EM solves this problem by computing better and better estimates until reaching a fixed point. It starts by taking a majority vote and using that to determine an initial guess of the correct answer for each question. EM then scores each worker (P_w) based on how many answers she got right. In subsequent iterations, EM weights each worker's votes based on her score, so better workers (with higher P_w scores) count more. Since weighted votes likely produce a different set of correct answers, the next step is to recompute each worker's score. This process repeats until quiescence. As EM assigns higher weights to good workers and lower weights to poor workers, it allows a single strong worker to overrule multiple weak workers and the predicted answer may no longer be the majority vote.

More precisely, EM is a general method for learning maximum-likelihood estimates of hidden parameters. As shown in Figure 2, it initializes the probability parameters to random values and then using the worker responses, repeats the following steps to convergence:

- **Expectation:** Given estimates of all the probabilities, $P_w(r|a)$, compute the probability of each possible answer using Bayes' rule and the assumption of conditional independence.
- **Maximization:** Given the posterior probability of each possible answer, compute new parameters $P_w(r|a)$ that maximize the likelihood of each worker's response.

The model of Dawid and Skene is a relatively simple one, and researchers have created new models to address its various weaknesses. Whitehill *et al.* (Whitehill et al. 2009) note that worker responses are not really independent unless conditioned on both the correct answer *and* the question difficulty. To see this, suppose that on average students have an 80% chance of correctly answering textbook questions. Then we would expect that Jane, in particular, would have an 80% chance when confronted with question 13. However, if we were told that all 25 of the other students in the class had gotten the problem wrong, then we'd probably suspect that 13 is especially hard and we'd want to revise our estimate of Jane's chances downwards. Unfortunately, Dawid and Skene's model cannot make this inference, but Whitehill *et al.*'s uses information about workers' errors to update its belief about problem difficulty and hence about other workers' accuracies. The algorithm still follows the EM pattern shown in Figure 2, but the probability computations are a bit more complex.

Welinder *et al.* (Welinder et al. 2010) take Whitehill's approach a step further, designing a model with general multi-dimensional parameters. Questions have many features, one of which could be difficulty, and workers are modeled as linear classifiers who make their responses by weighting those features. This allows Welinder *et al.*'s model to account not only for worker skill and question difficulty, but also arbitrary worker and question features. For instance, they can learn that one worker is particularly good at discerning different types of birds, but only when viewed from the back. Surprisingly, these question features need not be specified in the model a priori; their algorithm learns the features! While this leads to excellent performance at answer assessment, it does have a drawback — it may be difficult or impossible for a human to understand the learned model.

All the models we have covered thus far assume that one knows the set of possible answers before giving tasks to workers. Lin *et al.* (Lin, Mausam, and Weld 2012a) address the case when either requesters cannot enumerate all possible answers for the worker or when the solution space is infinitely large. They use a model called the Chinese Restaurant Process (Aldous 1985) that often matches distributions seen in nature. Specifically, they specify a generative probabilistic model where workers answer a question by returning a previously-seen response with probability proportional to the number of other workers who have given that response and returning a novel response with some small fixed probablity. The benefits of this model include 1) the ability to handle open-ended questions, and 2) the capacity to deal with common mistakes that are repeated by multiple workers.

The AI literature abounds with various approaches to "post hoc" response aggregation. Kajino *et al.* (Kajino, Tsuboi, and Kashima 2012) note that using EM to learn parameters can lead to local optima. They propose instead to model the repeated labeling problem as convex optimization, so that a globally optimal solution can always be obtained. Prelec *et al.* (Prelec and Seung 2007) develop an algorithm that can find correct answers missed by the majority by asking workers to predict coworker mistakes. Liu *et al.* (Liu, Peng, and Ihler 2012) apply belief propagation and mean field approximation, techniques beyond the scope of this book, to perform the inference required to learn correct answers.

While we cannot hope to describe the entirety of the literature, we note that researchers are now beginning to apply the full force of state-of-the-art machine learning algorithms to this problem. To make comparison easier, Sheshadri and Lease (Sheshadri and Lease 2013) have developed an open-source shared task framework that allows benchmarking of response aggregation methods.

Gradually Moving to Fully-Automated Approaches

The next set of approaches seek to do more than simply reconcile multiple answers to a set of questions — they use machine learning to create an autonomous system that can answer questions itself. Raykar *et al.* (Raykar et al. 2010) propose a model that can not only learn about worker abilities and infer correct answers, but also jointly learn a logistic regression classifier that predicts future crowd responses or the answer — obviating the need to consult human workers in the future. They also use an EM pattern to learn their model.

Wauthier *et al.* (Wauthier and Jordan 2011) relax the idea that repeated labeling tasks must contain a "correct answer." Instead, they build a model that describes each worker's idiosyncrasies, which they use to predict *each worker's response to a future question*, q, as a function of features of q. Since this model handles subjective questions, it's quite powerful. Furthermore, it can be used to answer objective questions by adding an imaginary, virtual worker to define the gold standard, desired answers for a subset of the questions. Now one can simply use Wauthier *et al.*'s method to predict how this imaginary "always correct" worker would answer future questions. However, Wauthier *et al.*'s method has drawbacks as well. In contrast to the preceding techniques, which happened to all be *unsupervised*, it relies on *supervised machine learning*, which means that it requires that one already know the answers to some of the questions in order to predict the answers to others.

The ultimate objective of Raykar *et al.* and Wauthier *et al.* is to replace human workers rather than derive consensus from their answers. Their methods assume that there exist features for the question that would allow a learned classifier to accurately predict the answer. However, crowdsourcing is often used precisely to answer questions which are beyond AI's state of the art.

Instead of bootstrapping the learning process by using multiple workers to redundantly answer each question, Dekel and Shamir (Dekel and Shamir 2009b; 2009a) devise algorithms to limit the influence of bad workers. Specifically, they use workers' responses to train a support vector machine classifier, and they add constraints to the loss function such that no one worker, and no bad worker, can overly influence the learned weights. They also introduce a second technique to completely prune away any bad workers, re-solving the questions with the remaining, high quality workers.

Workflow Optimization

In the previous section we discussed the AI techniques used for predicting worker accuracy and determining the correct answers from a set of error-prone responses. We now shift our focus to the task of *optimizing* a given crowdsourcing process. Specifically, we consider the problem of dynamically controlling the execution of an interrelated set of tasks, which we call a *workflow*. A requestor typically has several objectives that must be jointly optimized. These often include the quality of the output, the total cost of the process (in case of economically motivated crowdsourcing) and/or other measures of efficiency such as number of workers required, total completion time, *etc.* Inevitably, there are tradeoffs between objectives. For example, one can usually increase output quality by enlisting more workers, but this increases cost. On the other hand, one can reduce cost by paying workers less, but this increases latency (Mason and

Watts 2009). In this section we describe various approaches useful for crowdsourcing optimization. While we focus on economically-motivated, micro-crowdsourcing platforms such as Amazon Mechanical Turk, the techniques also apply to other platforms, such as Zooniverse (Lintott et al. 2008), where one wishes to get the best quality output from a limited number of volunteers.

Researchers have taken two broad approaches for such optimizations. The first is to carefully design an efficient workflow for a given task such that the overall output is high-quality, and at the same time does not spend much money. An example is in sorting of items, such as images. One could use comparison between items or ask workers to rate an item on a numerical scale (Marcus et al. 2011). The latter approach spends less money but may not be as accurate. A hybrid scheme that first rates each item independently and later uses a comparison operator on an intelligently chosen subset of items to do fine-grained sorting produces a better tradeoff between cost and quality. There are several other examples of alternative workflows for a task, which achieve different cost-quality tradeoffs. These include computing the max from a set of items (Venetis et al. 2012), multiple-choice tasks (Sun and Dance 2012), soliciting translations (Sun, Roy, and Little 2011), writing image descriptions (Little et al. 2010), and taxonomy generation (Chilton et al. 2013). Designing an efficient workflow is usually task dependent. It requires involvement of a domain expert and typically much trial and error. To our knowledge, there are few general principles for this kind of work. The onus is on a human to creatively come up with a good workflow for the task.

A second approach to crowdsourcing process-optimization is more computational. It assumes a given workflow (*e.g.*, developed using the first approach) that has some parameters or decision points. It then uses AI to optimize the workflow by learning the best values of the parameters and controlling the workflow to route a task through various decision points. Since these methods can get somewhat complex, we start with the simplest possible example.

Control of a Binary Vote Workflow

One of the simplest forms of control problem arises in crowdsourcing of a single binary choice question, where workers provide either a 'yes' or a 'no' response. Because worker responses are noisy, a common solution for quality control is to ask multiple workers and aggregate the responses using majority vote or the EM approaches described previously. But how many workers should be asked for each question? Choosing the optimal number requires making a trade-off between cost and the desired quality. We now focus on this control problem.

Typically, a requestor either decides this number based on the available budget or does some initial performance analysis to understand the average ability of the workers and then picks a number to achieve a desired accuracy. These approaches miss an extremely important insight: not all questions (nor workers) are equal. A fixed policy sacrifices the ability to shift effort away from easy problems to improve accuracy on hard tasks. A superior solution is to perform *dynamic* control, *i.e.*, decide for each question whether to take another judgment, based on the exact history of work so far; see Figure 3.

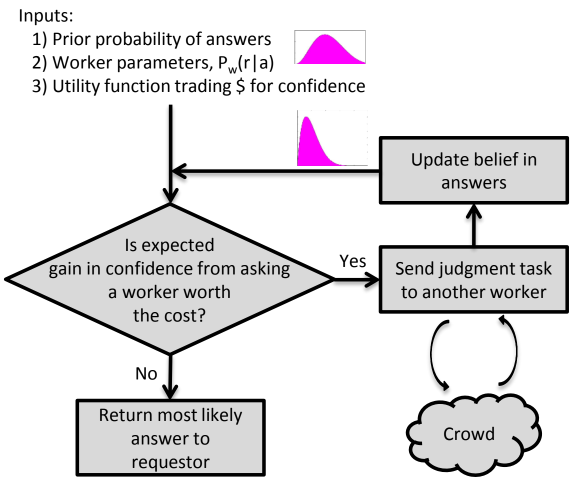
Inputs:
1) Prior probability of answers
2) Worker parameters, $P_w(r|a)$
3) Utility function trading $ for confidence
Update belief in answers
Is expected gain in confidence from asking a worker worth the cost?
Yes
Send judgment task to another worker
No
Return most likely answer to requestor
Crowd

Figure 3: The POMDP model for a binary vote workflow repeatedly decides if it is cost effective to ask another worker or whether the incremental reduction of uncertainty doesn't justify the cost.

As a case study we discuss the work of (Dai, Mausam, and Weld 2010; Dai et al. 2013), which models the problem of deciding whether to ask for another vote as a partially observable Markov decision process (POMDP), which is a popular technique for decision-theoretic optimization. Describing the POMDP representation and solution algorithms in detail is out of the scope of this chapter (see (Kaelbling, Littman, and Cassandra 1998; Poupart 2011)), but at a high level, a POMDP is a sequential decision making problem where all the actions (in our case, whether to ask for another worker's judgment or simply submit the answer) are known, the set of *possible* states is known but the exact state is not observable. For example, the true answer is not observable. The POMDP model defines system dynamics in terms of probability distributions for state transitions and observations. It also defines a reward that the agent tries to maximize.

The first step in defining the POMDP is specifying the state space as a pair (a, d), where a denotes the correct answer for the question ("yes" or 'no") and d is the question's difficulty (a number between zero and one). Since neither a nor d change over time, the POMDP transition probability is simply the identity function, making it a relatively simple problem to solve. However, the values of neither a nor d can be observed directly; at best, one can maintain a probability distribution (called a "belief") over the possible values of (a, d). We must now specify the POMDP observation model, which encodes the probability of various observations as a function of the (hidden) state. For example, what is the probability of a worker answering "yes" when the true answer is "no," *etc.*This is just another name for the worker models discussed previously, and we can use the methods of

Whitehill *et al.* (Whitehill et al. 2009).

A final input to the POMDP is a reward model, which in our case means the utility (or money equivalent) of obtaining the correct answer. For example, the requestor may specify that a correct answer is worth $\$1.00$ while an incorrect result (e.g. answering 'no' when the true hidden answer is 'yes') is equivalent to a $-\$1.00$ penalty. This utility is important, since it allows the controller to trade off between cost and quality. If a correct answer is very important to the requestor, the controller should enlist more workers. In essence, the POMDP controller can compare the cost of asking additional worker(s) and the expected utility gain due to increased likelihood of a correct answer, and only ask another worker when the net marginal utility is positive.

The literature describes numerous algorithms for solving POMDPs (Kaelbling, Littman, and Cassandra 1998; Poupart 2011), and many can be adapted for this problem. Dai *et al.* try lookahead search, discretized dynamic programming and UCT. The exact choice of algorithm is not important. Most algorithms will perform better than a static fixed number of instances per question or even a hand-coded policy like "ask two workers; if they agree, return that answer, otherwise break the tie with a third vote." Dynamic control is superior because it automatically adapts to question difficulty and worker accuracy, capturing some very valuable insights. First, if a question is easy (workers agree on its answer), the controller will not ask for many more judgments. On the other hand, if workers disagree the question may benefit from more judgments. Second, if a worker who is known to be excellent answers a question, the controller may not ask for more judgments. Finally, if a question is deemed very difficult (*e.g.*, good workers disagree on its answer), the system might decide that it is not worth finding the correct answer for this question as it might be too expensive. In such cases, the controller will quit early even if it is not very sure about the answer. These kinds of intelligent decisions make it quite valuable for requestors to use AI techniques for workflow optimization.

Other researchers have also studied this problem in slightly different settings. Parameswaran *et al.* (Parameswaran et al. 2010) investigate other budgeted optimization problems and their theoretical properties. Waterhouse (Waterhouse 2013) investigates information-theoretic measures and uses information content of a judgment as its value. Lin *et al.* (Lin, Mausam, and Weld 2012a) study controllers for data filling questions, where a worker must choose from an unbounded number of possible answers to a question instead of picking among two or a small number of known choices.

Kamar *et al.* (Kamar, Hacker, and Horvitz 2012) study the interesting problem of mixed initiative systems where there is a machine model, which provides its own judgment for each question (and possibly an associated confidence). Asking the workers can validate or challenge the machine answer. However, not all questions may need a worker response. If the machine is very confident then the controller may choose to ask no or very few human workers. Kamar *et al.* develop modifications of the UCT algorithm for creating their controller that can enhance machine answers with worker judgments for additional accuracy.

Along the same lines, Lin *et al.* (Lin, Mausam, and Weld 2012b) use multiple kinds of evidence in the form of different workflows (or different ways of asking the same question). This exploits the observation that some questions may be best asked in one form and others in another. Lin *et al.*'s system can automatically switch between such workflows to dramatically improve the output quality without increasing the cost.

Another important aspect of control algorithms is model learning. So far we have assumed that the controller knows 1) the prior probability distribution for question difficulty, 2) ability parameters of workers, and 3) observation probabilities. In some cases, a human designer can estimate these numbers, but fortunately it is possible for the controller to *learn* these values even as it is controlling the workflow. A powerful formalism for balancing model learning with reward optimization is *reinforcement learning* (RL) (Sutton and Barto 1998). In RL-based control, the controller is in charge from the start; it naturally shifts its focus from model learning (exploration) in the beginning to reward maximization (exploitation) later on. Lin *et al.* (Lin, Mausam, and Weld 2012b) have used RL and found to have equivalent quality results without an explicit learning phase. Kamar *et al.* (Kamar, Kapoor, and Horvitz 2013) describe a similar approach applied to citizen science applications. We expect RL methods to become increasingly popular, since they dramatically reduce the entry barrier for the use of AI technology in crowdsourcing.

Selecting the Best Question to Ask

So far we have focused on the control of a *single* question where the agent's goal is to obtain a correct answer for a given question in a cost-efficient manner. However, requestors typically turn to crowdsourcing only when they have a large number of questions. An interesting decision problem arises in question selection in the context of unreliable workers. Given a fixed budget, should one divide resources equally, asking the same number of workers to tackle each question? Alternatively, it may be better to dynamically allocate workers to questions based on their answer uncertainty. The best policy is a function of how the answers will be used.

One common scenario is to use crowdsourced responses as a set of labeled examples for training a machine learning classifier. One can formalize the decision problem as follows. Suppose we have a large number of unlabeled questions, $u_1, \ldots, u_n$. Moreover, suppose that we have already asked workers for judgments to some of the questions $q_1, \ldots, q_k$. For each question q_i we may have asked multiple workers and hence we may have an aggregate answer (and associated confidence) for use in training the classifier. The decision problem is "Which question do we pick next for (re)-labeling?"

There are two competing sources of evidence for this decision: 1) an existing aggregate answer (labeled example) for a question may be inaccurate and may in fact hurt the classifier, suggesting we may want to ask for another judgment (relabeling), or 2) the classifier may be uncertain in some part of the hypothesis space and we may want to pick a question based on the classifier's uncertainty on the unlabeled data (active learning).

Active learning, the problem of choosing which unlabeled example should next be given to the oracle, has been widely studied in the AI and ML literature (Settles 2012). However, before the advent of crowdsourcing little work had considered active learning with noisy labels and hence the possibility of relabeling. Recently, several strategies have been explored for this problem. Sheng *et al.* (Sheng, Provost, and Ipeirotis 2008; Ipeirotis et al. 2013) focus on how best to relabel already-labeled questions, comparing multiple question selection strategies such as repeated round robin, answer entropy-based selection and others. Donmez *et al.* (Donmez and Carbonell 2008) focus on various two-worker scenarios. For instance, they develop an algorithm to determine the best examples to label and by whom if one worker is perfect but costly and the other worker is fallible but cheap. Wauthier and Jordan (Wauthier and Jordan 2011) propose a general utility-theoretic formulation to evaluate expected utility gain for each question and pick one with maximum gain.

Lin *et al.* (Lin, Mausam, and Weld 2014) approach understanding the tradeoff from a different direction, and instead consider conditions under which relabeling a small number of examples is better than labeling a large number of examples once. They find that properties like the inductive bias of the classifier and the accuracy of workers have profound effects on which strategy results in higher accuracies. We expect increasing attention to this area in coming years.

Selecting the Best Worker for a Task

Since some workers are more skilled or less error prone than others, it can be useful to match workers and tasks. Amazon Mechanical Turk is not an ideal platform for worker allocation, since it resembles a 'pull' model where the workers choose their next tasks themselves. However, other platforms (*e.g.*, Zooniverse (Lintott et al. 2008)) implement a 'push' model where the system decides which tasks to send to a worker.

There are various competing desiderata for such an allocation. The total budget or available time may be limited, so assigning every question to the best worker isn't typically feasible. More generally, one usually wishes to allocate tasks so as to achieve a high-quality result while ensuring worker satisfaction, which is especially important in citizen science or other applications with volunteer labor. This may imply an even distribution of tasks across workers and task diversity for any given worker. Parameters governing these distributions represent additional learning problems.

Various attempts have been made to study this problem in restricted scenarios. For example, Karger *et al.* (Karger, Oh, and Shah 2011a; 2011b; 2013) provide performance guarantees for a global worker assignment, but disallow adaptive task assignment, which would enable learning about worker skills over time to better utilize quality workers. On the other hand, Chen *et al.* (Chen, Lin, and Zhou 2013) learn worker skills and adaptively select promising workers, but do not bound the total number of tasks allowed per worker to ensure that no worker is overburdened.

Ho *et al.* (Ho and Vaughan 2012; Ho, Jabbari, and Vaughan 2013) and Tran-Thanh *et al.* (Tran-Thanh et al. 2012) assume constraints on the number of tasks that may be assigned to any single worker, and divide the control problem into two explicit phases of exploration and exploitation. Ho *et al.* study the scenario where there are multiple types of tasks and each worker has a hidden skill level for each task. Their model assumes that workers arrive randomly, one at a time. Tran-Thanh *et al.* allow the system to select workers, but assume that tasks are uniform and that a single worker completes each task, after which the controller is informed of the quality of the job performed. In most crowdsourcing settings, multiple workers are needed to ensure quality, and quality is not directly observable.

Others consider the worker selection problem in the context of active learning. Yan *et al.* (Yan et al. 2011) assume worker skills are known and adaptively select the most confident worker for a given question (using a model based on question features). Donmez *et al.* (Donmez, Carbonell, and Schneider 2009) facilitate a gradual transition from exploration (learning about worker parameters) to exploitation (selecting the best worker for a question) by modeling worker reliability using upper confidence intervals. Both Yan *et al.* and Donmez *et al.* first select the question and then choose the appropriate worker. By contrast, Wauthier and Jordan (Wauthier and Jordan 2011) design a joint question-worker selection algorithm geared towards learning about (latent) parameters.

The majority of question-worker selection methods seek to discover the best workers and use them exclusively, but in settings like volunteer crowdsourcing it is crucial to assign appropriate tasks to all workers regardless of their skill. Bragg *et al.* (Bragg et al. 2014) study the problem of routing questions in parallel to all available workers where tasks have varying difficulty and workers have varying skill, and develop adaptive algorithms that provide maximal benefit when workers and questions are diverse.

Finally, Shahaf and Horvitz (Shahaf and Horvitz 2010) study generalized task markets where the abilities of various workers are known, but workers charge different rates for their services. They study worker selection given a task and desired utility so that the workers with appropriate skill levels and payment profiles are chosen for a given task. Zhang *et al.* (Zhang et al. 2012) take a different approach entirely and shift the burden of finding the appropriate worker from the system to the workers, noting that workers themselves may be best equipped to locate another worker with the appropriate skills for completing a task. Overall, worker-task allocation is an exciting problem, for which there does not yet exist a satisfactory solution. We expect considerable progress in the next few years, given the problem's importance.

Controlling Workflows for Complex Objectives

Most AI research on workflow control has focused on simple multiple choice questions, as a natural first step. But the true power of crowdsourcing will be realized when we

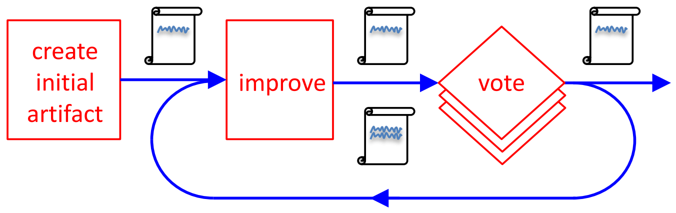

Figure 4: Control flow for an interative improvement workflow (adapted from (Little et al. 2010)).

optimize workflows for more complex tasks. A wide variety of complex tasks have already been explored within this framework. Examples include computing a max from a set of items (Guo, Parameswaran, and Garcia-Molina 2012), multi-label classification and generating a taxonomy of items (Bragg, Mausam, and Weld 2013), iterative improvement workflow for writing image descriptions (Dai, Mausam, and Weld 2010; 2011b), creating plans for achieving a goal (Kaplan et al. 2013), and selecting between multiple alternative workflows for a given task (Lin, Mausam, and Weld 2012b). In most of these settings the general abstraction includes defining a state space that encapsulates the agent's current belief about progress towards the requestor's objective, estimating the value of each possible worker task, issuing the best task, and repeating the process based on the expected outcome and any new information observed.

The exact mechanism for computing the value of human actions depends on the high-level objective. In cases where the exact POMDP can be solved, the POMDP policy is used to select the worker tasks. In other cases simpler strategies have been used to reduce the computation involved. For example, greedy action selection was used to guide multi-label classification (Bragg, Mausam, and Weld 2013), and limited lookahead search was used to control iterative improvement.

As a case study we briefly discuss the iterative improvement workflow (Figure 4), introduced by Little *et al.* (Little et al. 2010) and optimized by Dai *et al.* (Dai, Mausam, and Weld 2010; 2011b; Dai et al. 2013). Iterative improvement has been used to accomplish a wide range of objectives, such as deciphering human handwriting (Little et al. 2010), but for concreteness we discuss the case where the objective is to generate high-quality English captions for images. The workflow starts by asking a worker to write an initial caption for the picture. Then, at each iteration a worker is shown the image and current caption and is asked to improve the caption by smoothing the writing or adding details. Another set of workers is shown the two descriptions (original and "improvement") and is asked to select the best caption. These votes are aggregated and the best description is adopted for the next iteration.

From the point of view of AI control, there are three actions that can be performed during execution of this workflow: (1) issue a human task asking for another improvement, (2) issue a ballot task, requesting another comparison vote, or (3) submit the current description. To pose the control problem as a POMDP we first define the world state: the qualities of the two image descriptions. Let's use q_1 to denote the quality of the base description and q_2 to denote the quality of the newly "improved" description. If no improvement has yet been requested, then q_2 is undefined. We can constrain the qualities to be real numbers in $[0, 1]$, where one represents an idealized perfect description of the image and zero denotes the worst imaginable description.

Next, we define the POMDP actions corresponding to asking a worker to improve a description or compare two descriptions. The improvement model computes a probability distribution for possible values of $q_2 \in [0, 1]$ given that a worker with ability γ_{imp} tried to improve a description of quality q_1. Similarly, the voting model computes the probability that a worker of ability γ_{vote} will respond that the description number one is better when shown descriptions whose qualities are q_1 and q_2. Naturally, the probability of a mistake (saying "description one is better" when $q_2 > q_1$) increases if $|q_1 - q_2|$ is small and is inversely related to the worker's skill, γ_{vote}.

So far we have defined the dynamics of a POMDP. The final step is defining the utility function for the system to maximize, which will have two parts — the benefit due to returning a good description and the cost paid to workers. Clearly, the utility of returning a description with quality q should be a monotonically increasing function of q, though different requestors will assign different values to different qualities. Most people find it hard to articulate their precise utility function, and this has led to techniques for *preference elicitation*, which usually try to induce a general utility function from a small set of concrete judgements that are easier for people to answer (Boutilier 2002; Gajos and Weld 2005).

The definition of the POMDP is complete; now we need to solve it to produce a *policy* that says which action to execute as a function of the agent's beliefs. Because the state space of this POMDP is continuous (qualities are continuous variables), it is difficult to solve exactly. Dai *et al.* implemented this model and tried several approximate solution techniques, using supervised learning to induce the probabilistic transition functions from labeled training data. They found that POMDP-based control produced descriptions with the same quality as the original hand-coded policy, using 30% less labor.

An even more interesting observation is the manner in which the AI policy achieved its savings — by issuing voting jobs in a dramatically asymmetrical fashion. Little *et al.*'s hand-coded policy always asked two workers to compare the original and "improved" descriptions. If the assessments agreed, they adopted the consensus description for the next iteration. If the two workers disagreed, then the policy asked a third worker to break the tie. Thus on the average, the hand-coded policy issued about 2.5 voting tasks per iteration. In contrast, the POMDP policy, shown in Figure 5, issues *no* voting tasks in the early iterations, allowing it to issue five or more in later iterations. In hindsight, this allocation makes sense — since it is relatively easy for workers to improve a description early on, there is little reason to waste worker time verifying the improvement's quality. After a few cycles of improvement, however, the description

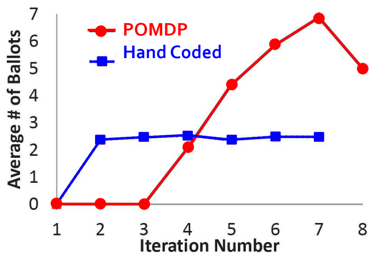

Figure 5: The POMDP controller for an iterative improvement workflow allocates resources differently than the hand-engineered policy, issuing more vote tasks (ballots) in later iterations, when the comparisons are harder to make and additional opinions are needed for accurate decisions (Dai et al. 2013).

has become increasing good and therefore harder and harder to improve. Now the POMDP chooses to spend additional resources issuing comparison votes, since it wants to be sure about which description to adopt for the next iteration.

This example points to the value of using AI control technology for complex tasks. For such tasks, often a human designer is unable to think through all possibilities, and hand-engineering a control policy that consistently exhibits optimal behavior can be difficult. Data-driven control approaches prove much more robust to corner cases and often end up saving large amounts of money. In other work, Bragg *et al.* (Bragg, Mausam, and Weld 2013) show that they can categorize a large number of items into multiple categories with the same accuracy as hand-engineered policies while using less than 10% of the labor.

A key limitation of this technology is its dependence on AI practitioners. The AI models and algorithms change somewhat based on the task at hand. They require a level of mathematical sophistication that is often too great a barrier for typical requestors considering crowdsourcing. Weld *et al.* (Weld, Mausam, and Dai 2011) sketch the architecture of a general purpose system, which will take a new workflow written in a high-level description language and automatically optimize it to control the workflow intelligently. If researchers can implement systems of this form and make them easy to use, then AI methods may transform crowdsourcing practice in the years to come.

Minimizing Latency

Although most research has focused on minimizing cost, there are a number of important situations where latency, the time to complete a task or workflow, is especially important. For example, when crowdsourced workers are used to interpret a smartphone photo of a street sign for visually-challenged users (Lasecki et al. 2013), a quick response is essential. Other low-latency applications include text editing (Bernstein et al. 2011), the selection of key frames from video, and captioning (Lasecki et al. 2012).

We may able to obtain near instantaneous work if we pre-employ several workers so that, as work arrives, they are already waiting to work on it (Bigham et al. 2010). This technique, termed the *retainer model*, has been studied analytically to determine the minimum number of retained workers required to achieve a quick response and the effect of delay on worker attentiveness. Queueing theory may be used to model the probable arrival times for user requests and thus the expected wait times for workers. We can then choose the number of workers to optimize the total cost subject to a constraint on expected delays or probability of missing a request (Bernstein et al. 2012). If there are several tasks requiring retainers then these can share the waiting workers thus amortizing the wait costs across tasks.

Conclusion

In summary, crowdsourcing, a popular form of collective intelligence, has close connections to artificial intelligence. An increasing number of machine-learning applications are trained with data produced by crowd annotation. Furthermore, many AI methods can be used to improve crowdsourcing. In particular, expectation maximization may be used to aggregate the results of multiple imprecise workers, learning worker accuracies at the same time. Partially-observable Markov decision processes (POMDPs) and related decision-theoretic approaches may be used to optimize the types and number of tasks given to workers. Since these AI methods are a very active area of research, we expect to see even more powerful methods in the coming years.

Acknowledgements: We appreciate many helpful conversations with Lydia Chilton, Peng Dai, Shih-Wen Huang, Stephen Jonany, and Andrey Kolobov. This work was supported by the WRF / TJ Cable Professorship, Office of Naval Research grant N00014-12-1-0211, and National Science Foundation grants IIS 1016713 and IIS 1016465.

References

Aldous, D. J. 1985. Exchangeability and related topics. In *École d'Été de Probabilités de Saint-Flour XIII 1983*, volume 1117 of *Lecture Notes in Mathematics*. Springer Berlin / Heidelberg. 1–198. 10.1007/BFb0099421.

Bernstein, M. S.; Brandt, J.; Miller, R. C.; and Karger, D. R. 2011. Crowds in two seconds: Enabling realtime crowd-powered interfaces. In *UIST*.

Bernstein, M.; Karger, D.; Miller, R.; and Brandt, J. 2012. Analytic methods for optimizing realtime crowdsourcing. In *Collective Intelligence*.

Bigham, J. P.; Jayant, C.; Ji, H.; Little, G.; Miller, A.; Miller, R. C.; Miller, R.; Tatarowicz, A.; White, B.; White, S.; and Yeh, T. 2010. Vizwiz: nearly real-time answers to visual questions. In *UIST*, 333–342.

Boutilier, C. 2002. A POMDP formulation of preference elicitation problems. 239–246.

Bragg, J.; Kolobov, A.; Mausam; and Weld, D. S. 2014. Parallel task routing for crowdsourcing. In *HCOMP*.

Bragg, J.; Mausam; and Weld, D. S. 2013. Crowdsourcing multi-label classification for taxonomy creation. In *HCOMP*.

Callison-Burch, C. 2009. Fast, cheap, and creative: evaluating translation quality using amazon's mechanical turk.

In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, 286–295. Association for Computational Linguistics.

Chen, X.; Lin, Q.; and Zhou, D. 2013. Optimistic knowledge gradient policy for optimal budget allocation in crowdsourcing. In *ICML*.

Chilton, L. B.; Little, G.; Edge, D.; Weld, D. S.; and Landay, J. A. 2013. Cascade: crowdsourcing taxonomy creation. In *CHI*, 1999–2008.

Dai, P.; Lin, C. H.; Mausam; and Weld, D. S. 2013. Pomdp-based control of workflows for crowdsourcing. *Artificial Intelligence* 202:52–85.

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

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

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

Dawid, A., and Skene, A. M. 1979. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied Statistics* 28(1):20–28.

Dekel, O., and Shamir, O. 2009a. Good learners for evil tecahers. In *ICML*.

Dekel, O., and Shamir, O. 2009b. Vox populi: Collecting high-quality labels from a crowd. In *COLT*.

Donmez, P., and Carbonell, J. G. 2008. Proactive learning: cost-sensitive active learning with multiple imperfect oracles. In *CIKM*, 619–628.

Donmez, P.; Carbonell, J. G.; and Schneider, J. 2009. Efficiently learning the accuracy of labeling sources for selective sampling. In *KDD*.

Ferrucci, D. A.; Brown, E. W.; Chu-Carroll, J.; Fan, J.; Gondek, D.; Kalyanpur, A.; Lally, A.; Murdock, J. W.; Nyberg, E.; Prager, J. M.; Schlaefer, N.; and Welty, C. A. 2010. Building watson: An overview of the deepqa project. *AI Magazine* 31(3):59–79.

Gajos, K., and Weld, D. S. 2005. Preference elicitation for interface optimization. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, 173–182. New York, NY, USA: ACM Press.

Guo, S.; Parameswaran, A. G.; and Garcia-Molina, H. 2012. So who won?: dynamic max discovery with the crowd. In *SIGMOD Conference*, 385–396.

Ho, C.-J., and Vaughan, J. W. 2012. Online task assignment in crowdsourcing markets. In *AAAI*.

Ho, C.-J.; Jabbari, S.; and Vaughan, J. W. 2013. Adaptive task assignment for crowdsourced classification. In *ICML*, 534–542.

Horvitz, E. 2007. Reflections on challenges and promises of mixed-initiative interaction. *AI Magazine* 28(2):13–22.

Hsueh, P.-Y.; Melville, P.; and Sindhwani, V. 2009. Data quality from crowdsourcing: A study of annotation selection criteria. In *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, HLT '09, 27–35. Stroudsburg, PA, USA: Association for Computational Linguistics.

Ipeirotis, P. G.; Provost, F.; Sheng, V. S.; and Wang, J. 2013. Repeated labeling using multiple noisy labelers. *Data Mining and Knowledge Discovery*.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101(1-2):99–134.

Kajino, H.; Tsuboi, Y.; and Kashima, H. 2012. A convex formulation for learning from crowds. In *AAAI*.

Kamar, E.; Hacker, S.; and Horvitz, E. 2012. Combining human and machine intelligence in large-scale crowdsourcing. In *AAMAS*.

Kamar, E.; Kapoor, A.; and Horvitz, E. 2013. Lifelong learning for acquiring the wisdom of the crowd. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*.

Kaplan, H.; Lotosh, I.; Milo, T.; and Novgorodov, S. 2013. Answering planning queries with the crowd. In *VLDB*.

Karger, D. R.; Oh, S.; and Shah, D. 2011a. Budget-optimal crowdsourcing using low-rank matrix approximations. In *Conference on Communication, Control, and Computing*.

Karger, D. R.; Oh, S.; and Shah, D. 2011b. Iterative learning for reliable crowd-sourcing systems. In *NIPS*.

Karger, D. R.; Oh, S.; and Shah, D. 2013. Efficient crowdsourcing for multi-class labeling. In *Proceedings of the ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems*, 81–92.

Lasecki, W. S.; Miller, C. D.; Sadilek, A.; Abumoussa, A.; Borrello, D.; Kushalnagar, R. S.; and Bigham, J. P. 2012. Real-time captioning by groups of non-experts. In *The 25th Annual ACM Symposium on User Interface Software and Technology, UIST '12, Cambridge, MA, USA, October 7-10, 2012*, 23–34.

Lasecki, W. S.; Thiha, P.; Zhong, Y.; Brady, E. L.; and Bigham, J. P. 2013. Answering visual questions with conversational crowd assistants. In *The 15th International ACM SIGACCESS Conference on Computers and Accessibility, ASSETS '13, Bellevue, WA, USA, October 21-23, 2013*, 18.

Lin, C. H.; Mausam; and Weld, D. S. 2012a. Crowdsourcing control: Moving beyond multiple choice. In *UAI*.

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

Lin, C. H.; Mausam; and Weld, D. S. 2014. To re(label), or not to re(label). In *HCOMP*.

Lintott, C.; Schawinski, K.; Slosar, A.; Land, K.; Bamford, S.; Thomas, D.; Raddick, M. J.; Nichol, R. C.; Szalay, A.; Andreescu, D.; Murray, P.; and VandenBerg, J. 2008. Galaxy zoo : Morphologies derived from visual inspection of galaxies from the sloan digital sky survey. *MNRAS* 389(3):1179–1189.

Little, G.; Chilton, L. B.; Goldman, M.; and Miller, R. C.

2010. Turkit: human computation algorithms on mechanical turk. In *UIST*, 57–66.

Liu, Q.; Peng, J.; and Ihler, A. 2012. Variational inference for crowdsourcing. In *NIPS*.

Marcus, A.; Wu, E.; Karger, D. R.; Madden, S.; and Miller, R. C. 2011. Human-powered sorts and joins. *PVLDB* 5(1):13–24.

Mason, W. A., and Watts, D. J. 2009. Financial incentives and the "performance of crowds". *SIGKDD Explorations* 11(2):100–108.

Oleson, D.; Sorokin, A.; Laughlin, G. P.; Hester, V.; Le, J.; and Biewald, L. 2011. Programmatic gold: Targeted and scalable quality assurance in crowdsourcing. In *Human Computation Workshop*, 11.

Parameswaran, A.; Garcia-Molina, H.; Park, H.; Polyzotis, N.; Ramesh, A.; and Widom, J. 2010. Crowdscreen: Algorithms for filtering data with humans. In *VLDB*.

Poupart, P. 2011. Chapter 3. In Sucar, E.; Morales, E.; and Hoey, J., eds., *Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions*, 33–62. IGI Global.

Prelec, D., and Seung, H. S. 2007. An algorithm that finds truth even if most people are wrong. http://www.eecs.harvard.edu/cs286r/courses/fall12/papers/Prelec10.pdf.

Raykar, V. C.; Yu, S.; Zhao, L. H.; and Valadez, G. 2010. Learning from crowds. *Journal of Machine Learning Research* 11:1297–1322.

Settles, B. 2012. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Shahaf, D., and Horvitz, E. 2010. Generlized markets for human and machine computation. In *AAAI*.

Sheng, V. S.; Provost, F.; and Ipeirotis, P. G. 2008. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the Fourteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Sheshadri, A., and Lease, M. 2013. Square: A benchmark for research on computing crowd consensus. In *HCOMP*.

Snow, R.; O'Connor, B.; Jurafsky, D.; and Ng, A. 2008a. Cheap and fast — but is it good? evaluating non-expert annotations for natural language tasks. In *EMNLP'08*.

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

Sun, Y.-A., and Dance, C. R. 2012. When majority voting fails: Comparing quality assurance methods for noisy human computation environment. *CoRR* abs/1204.3516.

Sun, Y.-A.; Roy, S.; and Little, G. 2011. Beyond independent agreement: A tournament selection approach for quality assurance of human computation tasks. In *Human Computation*.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning*. The MIT Press.

Thrun, S. 2006. A personal account of the development of stanley, the robot that won the darpa grand challenge. *AI Magazine* 27(4):69–82.

Tran-Thanh, L.; Stein, S.; Rogers, A.; and Jennings, N. R. 2012. Efficient crowdsourcing of unknown experts using multi-armed bandits. In *ECAI*, 768–773.

Venetis, P.; Garcia-Molina, H.; Huang, K.; and Polyzotis, N. 2012. Max algorithms in crowdsourcing environments. In *WWW*, 989–998.

Waterhouse, T. P. 2013. Pay by the bit: an information-theoretic metric for collective human judgment. In *CSCW*, 623–638.

Wauthier, F. L., and Jordan, M. I. 2011. Bayesian bias mitigation for crowdsourcing. In *NIPS*.

Weiss, G., ed. 2013. *Multiagent Systems, second edition*. MIT Press.

Weld, D. S.; Mausam; and Dai, P. 2011. Human intelligence needs artificial intelligence. In *HCOMP*.

Welinder, P.; Branson, S.; Belongie, S.; and Perona, P. 2010. The multidimensional wisdom of crowds. In *NIPS*.

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

Yan, Y.; Rosales, R.; Fung, G.; and Dy, J. G. 2011. Active learning from crowds. In *ICML*.

Zhang, H.; Horvitz, E.; Chen, Y.; and Parkes, D. C. 2012. Task routing for prediction tasks. In *AAMAS*, 889–896.