Planning for Crowdsourcing Hierarchical Tasks

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

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

ABSTRACT

We show how machine vision, learning, and planning can be combined to solve hierarchical consensus tasks. Hierarchical consensus tasks seek correct answers to a hierarchy of subtasks, where branching depends on answers at preceding levels of the hierarchy. We construct a set of hierarchical classification models that aggregate machine and human effort on different subtasks and use these inferences in planning. Optimal solution of hierarchical tasks is intractable due to the branching of task hierarchy and the long horizon of these tasks. We study Monte Carlo planning procedures that can exploit task structure to constrain the policy space for tractability. We evaluate the procedures on data collected from Galaxy Zoo II in allocating human effort and show that significant gains can be achieved.

Categories and Subject Descriptors

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

General Terms

Design, Algorithms, Economics

Keywords

crowdsourcing; consensus tasks; complementary computing; MDPs; Monte Carlo planning

1. INTRODUCTION

Research on human computation has explored workflows for solving tasks that computers cannot solve in the absence of human input. The efficacy of these workflows depends on managing the accuracy and cost of scarce human intelligence allocated to tasks. Previous studies have demonstrated the value of using machine-learning inferential models coupled with decision-theoretic optimization techniques for managing hiring decisions. Work spans efforts on fundamental task structures and workflows such as consensus classification tasks [5] and iterative workflows [2], which led to significant efficiencies in solving crowdsourcing tasks [5, 2]. Interest has been growing in the application of these techniques to more complex workflows.

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

We focus here on the use of learning and planning to guide the complementary use of people and machines in *hierarchical tasks*–tasks involving the solution of a branching sequence of subtasks. We specifically study the solution of hierarchical consensus tasks (HCTs), where answers at higher levels gate the relevance of questions at lower levels of the hierarchy. HCTs pose a control problem where decisions are not only about hiring a new worker, but also about the portion of the hierarchy to cover with each worker to maximize the efficiency with which the larger problem is solved. HCTs can be used in numerous realms, including community sensing applications (e.g., asking drivers about traffic conditions) and paid crowdsourcing (e.g., detailed product categorization). Our studies make use of tasks and data from Galaxy Zoo II (GZ2), a citizen science project aimed at learning the morphology of celestial objects. Galaxy Zoo II acts as a canonical example of HCTs; annotations collected from citizen scientists are guided by a task hierarchy that seeks increasingly more detailed assessments of morphology from images. For a given task, the hierarchy determines the next subtask to show based on the worker's answer to the previous subtasks. We gained access to human annotations collected for Galazy Zoo II tasks as well as image features generated for each image with machine vision.

We shall describe a general system design that combines machine learning and decision-theoretic planning to optimize the allocation of human effort in HCTs. The system uses hierarchical classification to combine evidence acquired from humans about various subtasks in the hierarchy with inferences that come via machine perception to predict the correct answer hierarchy. It also employs discriminatively trained evidence models to make inferences about future human inputs. First, we formalize the decision making problem as a Markov decision process (MDP) with partial observability, where models for predicting the correct answer hierarchy are used for belief updating and models for inferring future evidence act as the transition model. Then, we demonstrate that the exact solution of HCTs is intractable due to the branching of the task hierarchy and the long horizon associated with these tasks and that even well-known approximations hit a combinatorial wall. We propose two new approximate planning algorithms that use Monte-Carlo planning techniques to explore the search space. The algorithms exploit the structure of HCTs to constrain the policy space to offer effective and tractable solutions. The first algorithm reasons about the cumulative value of a worker—the value associated with asking all applicable questions in the task hierarchy—to decide whether to hire a worker. The second

algorithm makes decisions at the subtask level by reasoning about the decoupled value of hiring a worker for a subtask independently of other subtasks in the hierarchy.

We evaluate the methods by drawing worker responses from logs of the GZ2 system in different conditions. The evaluations show that significant reductions in human effort can be achieved via use of decision-theoretic guidance of hiring in HCTs. We find that the proposed algorithms can cut through the complexity of solving HCTs and that other well-known planning approaches fail to scale.

2. RELATED WORK

Our research is related to previous work on the automated control of crowdsourcing tasks. Researchers have focused on tasks that can be decomposed into smaller tasks [15], tasks with infinite possible answers [12], and iterative workflows [2]. Other work has proposed decision-making models for choosing among different workflows [13]. DELUGE is a system proposed for optimizing a workflow for taxonomy creation [1]. In most cases, greedy and limited lookahead algorithms have been used to make hiring decisions. CrowdBudget is a system for allocating a fixed budget among crowdsourcing tasks using constraint satisfaction [19].

The most relevant prior studies are efforts to optimize hiring decisions for consensus tasks. While most approaches have employed heuristic methods or greedy analyses [16], the CrowdSynth effort has made use of planning procedures for analyzing comprehensive sequences of actions [5]. CrowdSynth learns from historical data to jointly predict user behavior and final answers and couples these inferences in a decision-theoretic planning procedure. The procedure makes decisions by trading off the costs of hiring workers with the long-term expected benefit. CrowdExplorer adapts these techniques for the adaptive control of consensus tasks when historical data is not available [7]. We build on the earlier studies to extend decision-theoretic methods to a new class of crowdsourcing tasks, where decisions are defined over a hierarchy of consensus tasks.

3. HIERARCHICAL CONSENSUS TASKS

Consensus tasks center on identifying a single correct answer based on pieces of evidence collected from a population of workers (e.g., image classification tasks) [5]. Hierarchical consensus tasks expand consensus tasks by seeking increasingly more detailed answers. HCTs are composed of a hierarchy of subtasks such that the answer of a subtask determines which subtask can be queried from a worker next. Galaxy Zoo II tasks are examples of HCTs that are designed to collect volunteers' input for detailed morphological classification of celestial objects recorded in the Sloan Digital Sky Survey (SDSS)[1]. The task hierarchy for GZ2 tasks is given in Figure 1 and is composed of 9 subtasks leading to 159 possible answer combinations (sequences).

Formally the hierarchy for an HCT is composed of a set of subtasks from 1 to n. Let A_i be the set of possible answers for subtask i. We define the correct answer of a task as a sequence of answers $\{\bar{a_1}, ..., \bar{a_n}\}$, where $\bar{a_i} \in A_i$. We denote this sequence as $\{\bar{a_i}\}_1^n$. The answer of a parent subtask determines which child subtasks are applicable for a given task. For example, subtask $st2$ of the hierarchy, given in Figure 1, is applicable only if the answer of $st1$ is a_1^1: we can

[1]http://zoo2.galaxyzoo.org/

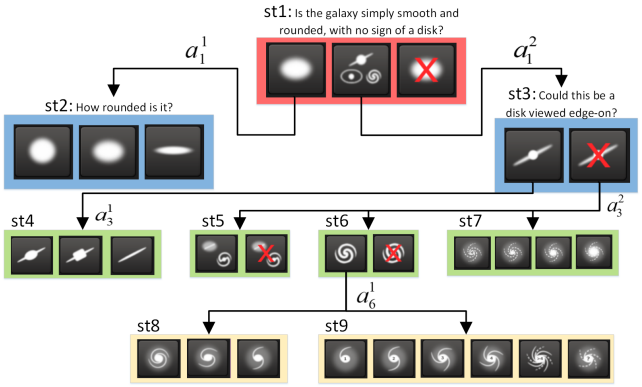

Figure 1: GZ2 subtask hierarchy: L2 hierarchy (top two levels) and L4 hierarchy (complete hierarchy).

only assess how rounded a galaxy is if the galaxy is rounded. Formally, given that subtask j is applicable only when the parent subtask i has the answer a_i ($st_i \xrightarrow{a_i} st_j$), $\bar{a_j}$ =n/a when $\bar{a_i} \neq a_i$ and $\bar{a_j} \in A_j \setminus \{\text{n/a}\}$ when $\bar{a_i} = a_i$.

For each task, the system has access to a population of workers. After hiring each worker, the system first asks about the top-level subtask. Conditional on the worker's answer, the system determines which subtask is applicable next based on the subtask hierarchy. Let $L_i \subseteq A_i$ be the set of answers the worker can choose from for subtask i, and $st_i \xrightarrow{a_i} st_j$; the system may present subtask j to the worker only if the worker report a_i for st_i.

We use the Galaxy Zoo II domain as our running example and employ a dataset released by the GZ2 team for evaluation [20]. The dataset includes 154,640 unique tasks and around 15 million responses collected for subtasks. The original Galaxy Zoo II system does not use optimization. Rather, all applicable subtasks are asked of every hired worker. The number of workers assigned to each task varies between 20 and 58. The dataset also includes visual attributes (SDSS features) identified by machine vision for each task.

4. SOLVING HCTS

A system for solving HCTs needs to make decisions about which set of evidence to collect from workers for inferring the correct answers of the hierarchy. For a new task, the system can either hire a new worker or may decide to end the task. If a decision is made to end the task, a prediction is provided about the correct answer sequence ($\{\hat{a_i}\}_{i=1}^n$). If the system decides to hire a worker, it queries an answer for the top level subtask (st1 for GZ2) and then deliberates about querying the worker for the next applicable subtask conditioned on the answer. For GZ2 tasks, if a worker has responded a_1^1 for subtask st1, the system deliberates next about querying the worker for subtask st2. The system repeats this deliberation until no more applicable subtasks exist for the current worker. At that point, the system reasons about hiring a new worker or terminating the task.

System decisions are guided by the tradeoff between making more accurate predictions about the correct answer sequence via collecting more evidence from workers (by hiring more workers and asking about more subtasks) and the costs of the additional human effort. We employ machine learned models and a decision-theoretic planner to optimize such de-

cisions. We use two sets of predictive models: *answer models* for predicting the correct answer sequence at any point in the execution, and *evidence models* for predicting the next set of evidence collected from workers. The answer models are used for assessing the system's confidence in predicting the correct answer sequence; the evidence models are used for inferring the next state of the system after taking a hiring action. Both models are constructed with supervised learning using the pipeline presented in Section 5.

The planner optimizes hiring decisions based on the current state of a task. The optimization problem is formalized as a Markov Decision Process (MDP) with partial observability as presented in Section 6, where answer models are used to generate the system's belief about the correct answer sequence at any given state and evidence models are used to build the transition model of the MDP. The MDP model is able to represent both the system's partial information about the correct answer sequence and uncertainty about future states if the system continues to query answers from workers. The system decisions are guided by the value of allocating a worker to a subtask to maximize the net utility of the system. The net utility is a combination of the system's utility for delivering the predicted answer sequence and cost for querying answers from workers. Two utility models are designed to express different preferences in evaluating the predicted answer sequence of a HTC, and are presented in Section 5.2.

5. MODELS FOR HCTS

5.1 Learning Pipeline

We built a learning pipeline that uses past data of instances collected for HCTs to make inferences about the correct answer sequences and evidence collection for future tasks. For each task, we create a new instance when new evidence is gathered from a worker for a subtask. Each instance contains two sets of evidence that have been collected for the task so far: f_m, features generated with automated machine analysis (SDSS features for GZ2 tasks), and $\{e_i\}_1^n$, the set of evidences collected from workers for all subtasks. f_m is available from the start of the task and $\{e_i\}_1^n$ is updated after each worker response. We convert e_i, the set of evidence collected for each subtask, into a set of features including number of total worker reports obtained on the task so far, number of reports for each answer, ratios of reports per answer, the count and ratio for the most popular answer and the difference between the most popular and the second most popular answers. We use the MART gradient boosting algorithm to learn an ensemble of regression trees for making predictions about HCTs [4]. Training, validation and testing sets include 500k, 250k, 250k GZ2 tasks respectively.

5.1.1 Learning Answer Models

For a given HCT with evidence set $(f_m, \{e_i\}_1^n)$, answer models make inference about the correct answer sequence of the task ($\{\bar{a_i}\}_1^n$). Labels for training answer models are ground truth answer sequences assigned to HCTs; they can either be collected from domain experts or majority opinion of a large number of workers can be taken as ground truth. For GZ2 tasks, labels are provided by astronomy experts and made public in a catalog given in [20].

Since the labels of answer models are organized in a subtask hierarchy in the form of a tree, we model the answer

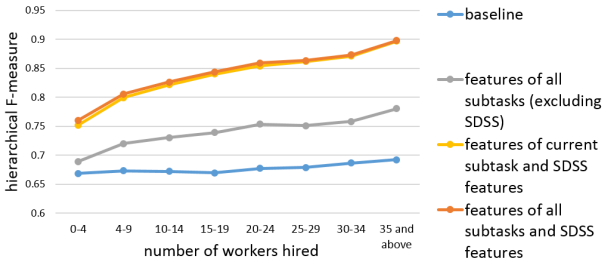

Figure 2: Performance of answer models.

prediction of HCTs as a task of hierarchical classification [17]. We implement the top-down approach, a popular approach for hierarchical classification [11] for ease of training and efficiency in making predictions. We train a classifier for each subtask of the hierarchy such that only the portion of the data that satisfies the precondition of the subtask is included into training the model. For example, only the set of instances with answer a_1^1 for subtask st1 are included into training a model for subtask st2.

Let $M_{A_i}(a_i, f_m, \{e_i\}_1^n)$ be the likelihood of the answer of subtask i being a_i. For any answer sequence $\{a_i\}_1^n$ consistent with the subtask hierarchy, $Pr(\{a_i\}_1^n)|f_m, \{e_i\}_1^n)$, the likelihood of the correct answer sequence being $\{a_i\}_1^n$, and $\{\hat{a_i}\}_1^n$, the predicted answer sequence, are computed as:

$$Pr(\{a_i\}_1^n)|f_m, \{e_i\}_1^n) = \prod_{i=1, a_i \neq \text{n/a}}^{n} M_{A_i}(a_i, f_m, \{e_i\}_1^n)$$

$$\{\hat{a_i}\}_1^n = \max_{\{a_i\}_1^n} Pr(\{a_i\}_1^n)|f_m, \{e_i\}_1^n)$$

We evaluate the quality of the hierarchical answer models using the hierarchical F-measure [9]. Given a correct answer sequence $\{\bar{a_i}\}_1^n$ and predicted answer sequence $\{\hat{a_i}\}_1^n$, precision (ppv) and recall (npv) values for this instance are computed as below, where δ is the Kronecker delta function:

$$ppv(\{\bar{a_i}\}_1^n, \{\hat{a_i}\}_1^n) = \frac{\sum_{i=1, \hat{a_i} \neq \text{n/a}}^{n} \delta(\bar{a_i}, \hat{a_i})}{\sum_{i=1}^{n} (1 - \delta(\hat{a_i}, \text{n/a}))}$$

$$npv(\{\bar{a_i}\}_1^n, \{\hat{a_i}\}_1^n) = \frac{\sum_{i=1, \bar{a_i} \neq \text{n/a}}^{n} \delta(\bar{a_i}, \hat{a_i})}{\sum_{i=1}^{n} (1 - \delta(\bar{a_i}, \text{n/a}))}$$

The hierarchical F-measure is computed as $\frac{2 \times hP \times hR}{hP + hR}$, where hP and hR are average precision and recall values computed over all instances of the testing set.

Figure 2 shows the hierarchical F-measure for predicting the correct answer sequence of GZ2 tasks for L2 subtask hierarchy when (1) models do not have access to machine generated SDSS features (grey line), (2) each subtask model M_{a_i} has access to SDSS features and the evidence for subtask i (yellow line) and (3) each model has access to all features (orange line). The blue line represents the baseline predicting the most common answer in the training set for each subtask. The results show that there is significant benefit from harnessing the SDSS features made available via machine vision. The quality of predictions improves with larger number of workers hired for a task. In addition, we find that providing access to evidence from other subtasks does not provide significant improvement as answer models created for different subtasks consistently choose SDSS and current subtask features as the most informative.

5.1.2 Learning Evidence Models

We create evidence models to infer the evidence that will be collected should a worker be queried about a particular subtask. We use the evidence models to predict how the set of evidence on each task changes as a system makes hiring decisions. We build an evidence model for each subtask. The labels for training evidence models are collected directly from the execution of HCTs; for each instance, the evidence collected for the current subtask becomes the label to train an evidence model for that subtask. We train evidence models with supervised learning. Our experiments show the same trends as the answer models; the learned models provide better predictions than a baseline model that predicts the most common evidence, and having access to features of other subtasks does not provide improvements in predictive performance.

5.2 Utility Models

When the system terminates evidence collection, it delivers a final inference about the correct answer sequence to the task owner. Utility models quantify the task owner's preferences about obtaining this inference as a function of its accuracy. Utility models are used in defining the reward function of the MDP model as described in Section 6.

Utility model, $U(\{\bar{a}_i\}_1^n, \{\hat{a}_i\}_1^n)$, quantifies the utility of predicting the correct answer sequence as $\{\hat{a}_i\}_1^n$ when the correct answer sequence is $\{\bar{a}_i\}_1^n$. Task owners may have different preferences about the completeness and accuracy of assessments for HCTs. In our experiments we focus on two utility models that express different preferences. The $F1$ utility model is defined as:

$$U_{F1}(\{\bar{a}_i\}_1^n, \{\hat{a}_i\}_1^n) = \frac{2 \times ppv(\{\bar{a}_i\}_1^n, \{\hat{a}_i\}_1^n) \times npv(\{\bar{a}_i\}_1^n, \{\hat{a}_i\}_1^n)}{ppv(\{\bar{a}_i\}_1^n, \{\hat{a}_i\}_1^n) + npv(\{\bar{a}_i\}_1^n, \{\hat{a}_i\}_1^n)}$$

The second utility model U_{EM} has value 1 if $\{\bar{a}_i\}_1^n$ and $\{\hat{a}_i\}_1^n$ are exact matches and has value 0 otherwise. Both utility models are bounded in [0,1]. U_{F1} rewards a prediction proportional to the goodness of match, whereas U_{EM} strictly rewards perfect matches.

6. OPTIMIZING WORKER ALLOCATION

6.1 Problem Formalization

We model HCTs with an MDP, where the reward is uncertain; it depends on its belief about the correct answer sequence. An HCT is formalized as a tuple $< l, S, \mathbb{A}, T, R >$:

- l, the horizon of a task, is the maximum number of workers to be hired for a task.
- $s \in S$, a state of an HCT, is $< f_m, \{e_i\}_1^n, j >$, where f_m is the set of features generated with automated task analysis, e_i is the set of evidence collected for subtask i, and j is the current subtask under deliberation (current subtask as short). $S^T \subset S$ is the set of termination states, which includes states at the horizon.
- The set of actions are $\mathbb{A} = \{H, \neg H\}$ for assigning the current subtask to a worker and not doing so respectively. No more hiring action can be taken at a termination state. If the current subtask is top level (e.g., st1 for GZ2 tasks) and $\neg H$ action is taken, no further actions can be taken.
- $T(s, \alpha, s')$ is the transition probability from state s to s' after taking action α. The transition function is composed of models representing the execution of HCTs: Evidence model for subtask j is used to predict $Pr(\{e'_j\}|f_m, \{e_i\}_1^n)$, the likelihood of evidence set for subtask j to be updated to $\{e'_j\}$ when the current subtask is j and action H is taken. The subtask hierarchy is used to determine the next subtask to be considered, if any.
- $R(s, \alpha)$ is the reward associated with taking action α at state s. The reward for taking H action is equal to the cost of hiring for the current subtask. If taking action $\neg H$ leads to terminating evidence collection and delivering the predicted answer sequence, the associated reward is uncertain, depends on the quality of the prediction, and is estimated by querying the answer and utility models. b is the system's belief about the correct answer sequence, which is a probability distribution over $\{A_i\}_1^n$, the set of all answer sequences. At any state s, $b(\{a_i\}_1^n, s)$ is the likelihood of the correct answer sequence being $\{a_i\}_1^n \in \{A_i\}_1^n$, and $\{\hat{a}_i\}_1^n$ is the highest likelihood answer sequence as predicted by answer models based on the evidence in s. The reward for terminating evidence collection is calculated as $\sum_{\{\bar{a}_i\}_1^n} b(\{\bar{a}_i\}_1^n) \times U(\{\bar{a}_i\}_1^n, \{\hat{a}_i\}_1^n)$.

A policy π specifies the action that the system chooses at any state s. An optimal policy π^* satisfies the following:

$$V^{\pi^*}(s) = \begin{cases} R(s, \neg H) & \text{if } s \in S^T \\ max_{\alpha \in \mathbb{A}} \big(R(s, \alpha) + \\ \quad \sum_{s'} T(s, \alpha, s')\, V^{\pi^*}(s') & \text{otherwise} \end{cases}$$

A planner for solving HCTs faces the challenge of solving these tasks under the runtime limitations of a live platform such as GZ2. The state of an HCT includes a large number of machine generated features, some of which are defined on the continuous space. As a result, an HCT may have infinitely many possible initial states which makes offline policy computation infeasible. Thus, we focus on online solution algorithms that can incorporate any new evidence in real time to compute a decision for an HCT task. Another challenge for solving HCTs is the exponentially growing size of the action sequences in the horizon. HCTs differ significantly from consensus tasks in that taking $\neg H$ action does not necessarily end the task, leading to an exponential expansion of the space of possible policies. In addition, solving an HCT requires exploring a state space that may grow exponentially in the horizon and in the depth of the subtask hierarchy. For GZ2 tasks with L4 hierarchy, the number of the unique states is 38 thousand for a horizon of 2 and is 1.4 million for a horizon of 3.

When the system's initial belief is strong about the correct answer sequence (based on machine vision features for GZ2), long sequences of evidence may be needed to change the system's prediction about the correct answer sequence. GZ2 logs show that, on average, tasks employ 28 workers, and that the number of contributors can be as large as 58. Such a large horizon makes optimal approaches for solving these tasks intractable. Myopic and limited-lookahead approaches cannot explore deeper regions of the search space and consequently may fail to realize the value in longer-term evidence collection.

6.2 Solving HCTs Efficiently

Monte Carlo planning offers a way to address the challenges in solving HCTs; it is an online planning approach that can efficiently explore the search space of a large MDP.

Monte Carlo planning uses sampling to explore the search space and then builds a partial search tree to optimize decisions [8, 10, 18]. Despite theoretical results demonstrating the convergence of some Monte Carlo algorithms to optimal policies under infinitely many samples [10], in practice these general algorithms may fail to make effective decisions under runtime limitations. In particular, general Monte Carlo algorithms (e.g., UCT) hit a combinatorial challenge in solving problems with long horizons [6]. The number of samples for exploring state-action outcomes grows exponentially in the horizon. For problems where some actions may lead to early termination (e.g., taking $\neg H$ actions when current subtask is top level), these algorithms explore the part of the state space that is closer to the initial state and require immense number of samples to explore longer horizons.

Previous work on solving consensus tasks has shown that the sampling method in Monte Carlo planning can be customized to explore long horizons of these tasks [6]. When all $\neg H$ actions lead to terminating evidence collection, the MC-VOI algorithm can eliminate the action selection problem in sampling. For each sample, it can evaluate the utilities of hiring and not hiring simultaneously at each encountered state, which reduces the number of samples required to explore consensus tasks with long horizons.

We present two approximate algorithms that constrain the policy space in solving HCTs such that the sampling and evaluation techniques given in the MC-VOI algorithm can be used to make effective decisions for HCTs. The algorithms sample evidence paths to explore how evidence collection may proceed until termination. They differ in the way that they construct partial search trees and optimize decisions. The MC-VOW algorithm reasons about the cumulative value of a worker—the value associated with hiring a new worker and asking all applicable subtasks. It makes decisions about whether to hire a new worker. The MC-VOT algorithm reasons about the decoupled value of a subtask—the value associated with asking a worker about the current subtask independent of other subtasks in the task hierarchy. The decisions of this algorithm are on the subtask level.

We describe the building blocks that are common in both algorithms. The *SampleEvidencePath* function samples one possible pathway that evidence collection may take while taking H actions until termination. The *SampleEvidence* function makes queries to the evidence models to sample future evidence to be collected at a given state. The *SampleNextState* function samples a next state from the transition model. Recursive calls to the *SampleEvidencePath* function provides a complete path from the initial state to a termination state. When a termination state is reached, the *SampleCorrectAnswer* function samples a correct answer sequence using the hierarchical answer models based on the evidence stored in the state. This sampled answer sequence is used to evaluate the utilities of all sampled states on the pathway simultaneously for hiring and not hiring actions.

The *UpdateSearchTree* function applies the evaluation technique introduced in [6] for building a partial search tree of visited states and for evaluating the values for taking different actions at these states. For each state s in the search tree, the algorithm keeps visitation counts, and estimates values for H action ($s.V^H$) and $\neg H$ action ($s.V^{\neg H}$). For a given state s and a sampled correct answer sequence $\{\bar{a}_i\}_1^n$, the value estimates are updated by the *UpdateSearchTree* function as follows: $s.V^H$ is computed as the weighted average of the V values of all states s has transitions to in the search tree. $s.V^{\neg H}$ is updated with $U(\{\bar{a}_i\}_1^n, \{\hat{a}_i\}_1^n)$, where $\{\hat{a}_i\}_1^n$ is predicted based on s, minus the cost of acquired evidence. $s.V$ is the maximum of $s.V^H$ and $s.V^{\neg H}$.

6.2.1 Decisions based on Cumulative Value of a Worker

The MC-VOW algorithm constrains the policy space such that, once a new worker is hired, all applicable subtasks are queried from the worker. Doing so eliminates decision-making at the subtask level and reduces the complexity of making decisions. The algorithm makes decisions only at states in which the current subtask is at the top of the subtask hierarchy (i.e., $TopLevel(s)$ is true). The decisions are guided by the estimated cumulative value of a worker (VOW), which quantifies the value for asking all applicable subtasks to a worker until reaching the bottom of the subtask hierarchy. For any state s^0, if VOW is estimated to be non-positive, the system ends the task and outputs the predicted answer sequence. Otherwise, the system collects evidence from the worker for all applicable subtasks in the hierarchy and updates s^0 accordingly. The reasoning is repeated for the updated state to decide whether to hire another worker. In computing VOW, the only actions available are for the top level of the hierarchy, whether to hire a worker or not, which allows for applying the sampling procedure suggested by the MC-VOI algorithm. VOW for a given state s^0 is computed as:

$$VOW(s^0) = R(s^0, H) + \sum_{s'} T(s^0, H, s')\, V_w^{\pi^*}(s') - R(s^0, \neg H), \text{ where}$$

$$V_w^{\pi^*}(s) = \begin{cases} R(s, \neg H) & \text{if } s \in S^T \\ max_{\alpha \in \mathbb{A}} \big(R(s, \alpha) + \sum_{s'} T(s, \alpha, s')\, V_w^{\pi^*}(s')\big) & \text{if TopLevel(s)} \\ R(s, H) + \sum_{s'} T(s, H, s')\, V_w^{\pi^*}(s') & \text{otherwise} \end{cases}$$

The details of an algorithm for computing VOW for s^0 is given in Algorithm 1. Each sampled evidence path contains evidence from all applicable subtasks. However, a state is added to the partial search tree only if the current subtask is the top level subtask or if the state is a termination state.

VOW quantifies the expected value of hiring a worker and asking the worker all applicable subtasks. This value takes into account that asking the top level subtask may lead to asking about other subtasks and gathering additional evidence. The shortcoming of this algorithm is that it cannot analyze the individual improvement that a subtask can provide for predicting the correct answer, and thus cannot optimize the subset of subtasks to ask to a worker to maximize utility. For a GZ2 task, this algorithm would never deliberately stop evidence collection after asking about subtask st1 even if other subtasks do not help with answer prediction.

6.2.2 Decisions based on Decoupled Subtask Value

Given the intractability of solving the large MDP associated with solving an HCT, we seek to decouple it into smaller, subtask specific MDPs. The idea of decomposing large MDPs into smaller problems has been studied in previous work for offline planning [3, 14]. We present an online planning algorithm that uses the special structure of HCTs to decompose it into individual consensus tasks.

```
CalculateVOW(s^0:initial state)
begin
    repeat
        SampleEvidencePath(s^0)
    until Timeout
    VOW ← s^0.V^H − s^0.V^¬H
    return VOW
end

SampleEvidencePath(s:state)
begin
    if ¬IsTerminationState(s) then
        e ← SampleEvidence(s)
        s' ← SampleNextState(s, e)
        {ā_i}_1^n ← SampleEvidencePath(s')
    else
        {ā_i}_1^n ← SampleCorrectAnswer(s)
    end
    if TopLevel(s) or IsTerminationState(s) then
        UpdateSearchTree(s, {ā_i}_1^n)
    end
    return {ā_i}_1^n
end
```

Algorithm 1: MC-VOW algorithm

We first describe the construction of a decoupled MDP for a subtask of an HCT. We show how solving these decoupled models still requires reasoning about the global task structure due to the subtask dependencies in the transition model. We define state equivalence between the decoupled model and the global model so that paths sampled over the global model can be used to make decisions about hiring a worker for a subtask. We outline the steps of this computation by presenting a Monte Carlo algorithm.

Each decoupled MDP optimizes hiring decisions for a subtask independently of other subtasks. This approach decomposes the decision-making of GZ2 tasks into 9 individual MDPs. It guides hiring decisions based on the *decoupled value* of hiring for the current subtask (VOT). This decoupled value is the value of asking the current worker and future workers only about the current subtask. It isolates the value for the current subtask, and does not capture the utility that can be generated by asking other subtasks as a follow-up to the current subtask. For a GZ2 task where the current subtask is st1, VOT captures the value in asking users about subtask st1, but does not capture the potential value in asking about st2. For a given state, this approach queries the MDP corresponding to the current subtask to estimate VOT. If VOT is estimated to be positive, the algorithm queries the worker about the current subtask and repeats the calculation for the updated state.

To formalize the computation of VOT for an initial state s^0 with current subtask i, we define a decoupled MDP, called MDP_i, that only models the decoupled decision process of asking workers about the subtask i. S_i is the set of states for MDP_i. $s_i \in S_i$ is composed of the initial state s^0, and evidence collected for subtask i following the initial state $\{e_i\}$. S_i^T is the set of termination states where future evidence collection is not possible. The set of actions include H, hiring for the current subtask, and $\neg H$ for not hiring. From the point of view of the decoupled MDP_i, choosing action $\neg H$ at any state s_i indicates that no positive value is associated with hiring a worker for the current subtask, leading to terminating evidence collection for that subtask. This structure allows us to use the sampling strategy used in the MC-VOI and MC-VOW algorithms to evaluate the values of each visited s_i for both actions simultaneously. The reward for hiring is equal to the cost of querying a worker for subtask i. The reward for not hiring is computed based on the belief about the correct answer sequence, as given in Section 6.1. VOT for initial state s^0 is computed as:

$$VOT(s^0) = R_i(s^0, H) + \sum_{s_i' \in S_i} T_i(s^0, H, s_i')\, V_i^{\pi^*}(s_i') - R_i(s^0, \neg H), \text{ where}$$

$$V_i^{\pi^*}(s_i) = \begin{cases} R_i(s_i, \neg H) & \text{if } s_i \in S_i^T \\ max_{\alpha \in \mathbb{A}}\big(R_i(s_i, \alpha) + \\ \quad \sum_{s_i'} T_i(s_i, \alpha, s_i')\, V_i^{\pi^*}(s_i')\big) & \text{otherwise} \end{cases}$$

States, actions, and the reward function of MDP_i can be easily decoupled from the general MDP. However, the transition model of MDP_i for taking action H cannot be completely decoupled from other subtasks. The model includes inferences from evidence models for subtask i, which can be queried based on the decoupled state s_i. However, the model should also predict the likelihood that there is going to be another opportunity of evidence collection for subtask i should evidence collection for higher level subtasks continue until termination. Since asking a worker about a subtask may be conditioned on the answers collected from a worker on other subtasks, this prediction should be performed on the general MDP modeling the complete HCT. For example in the GZ2 hierarchy, the evidence-collection opportunities for subtask st9 is low since it can be queried only when a worker answers subtask st1 as a_1^2, st3 as a_3^2 and st6 as a_6^1. Assessing this likelihood for st9 subtask requires reasoning about the likelihood of a worker answering subtasks st1, st3 and st6 in this particular way.

To describe the construction of the decoupled transition model from the global MDP model, we define a subtask equivalence between a global state s and subtask state s_i as follows: s and s_i are equivalent for subtask i ($s \cong s_i$) if (1) s and s_i share the same start state s^0, (2) the set of evidences for subtask i are identical in s and s_i, (3) s and s_i have the same termination flag and (4) the current subtask of s is i or $s \in S^T$. The first three conditions ensure that s and s_i have the same set of evidence with respect to the execution of subtask i. Given our observation from Section 5.1.2 showing that the likelihood of evidence for a subtask is independent of the evidence collected for other subtasks, the evidence model for subtask i produce identical predictions when queried with s and s_i. The fourth condition identifies the states of the general MDP deliberating about collecting evidence for subtask i or leading to the termination of MDP_i. Using the state equivalence definition, we can compute the transition probability of MDP_i using the general MDP model:

$$T_i(s_i, H, s_i') = \frac{\sum_{s \cong s_i, s' \cong s_i'} Pr(s) Pr(s'|s)}{\sum_{s \cong s_i} Pr(s)}$$

where s_i' is updated from s_i with a single evidence for subtask i or with termination. $Pr(s)$, the likelihood of being in state s and $Pr(s'|s)$, the likelihood of arriving to state s' from s by taking H actions can be computed by iteratively applying the transition model of the global MDP from s^0.

This derivation shows that evidence paths sampled over the global task model lead to valid samples and consequently a valid search tree for MDP_i.

CalculateVOT(s^0:initial state with current subtask i)
begin
  repeat
    $SampleEvidencePath(s^0, s^0, i)$
  until *Timeout*
  $VOT \leftarrow s^0.V^H - s^0.V^{\neg H}$
  return VOT
end

SampleEvidencePath(s:state, s_i:subtask state, i:subtask for which VOT is computed)
begin
  $j \leftarrow CurrentSubTask(s)$
  if $\neg IsTerminationState(s)$ **then**
    $e \leftarrow SampleEvidence(s)$
    $s' \leftarrow SampleNextState(s, e)$
    if $i = j$ *or* $IsTerminationState(s)$ **then**
      $s'_i \leftarrow UpdateSubTaskState(s_i, e)$
    else
      $s'_i \leftarrow s_i$
    end
    $\{\bar{a}_i\}_1^n \leftarrow SampleEvidencePath(s', s'_i, i)$
  else
    $\{\bar{a}_i\}_1^n \leftarrow SampleCorrectAnswer(s)$
  end
  if $i = j$ *or* $IsTerminationState(s)$ **then**
    $UpdateSearchTree(s_i, \{\bar{a}_i\}_1^n)$
  end
  return $\{\bar{a}_i\}_1^n$
end

Algorithm 2: MC-VOT algorithm

Algorithm 2 presents the details of an algorithm that estimates VOT by sampling execution paths from the global MDP model. Sampling execution paths eliminates the need to compute T_i and consequently $Pr(s)$ and $Pr(s'|s)$. As the algorithm generates the path, it keeps track of the corresponding state (s_i) of the decoupled MDP. s_i is updated when new evidence is gathered for subtask i or when task terminates. The partial search tree constructed by the *UpdateSearchTree* function is composed of the states of the decoupled MDP. The values estimated for the nodes of the tree capture the value of obtaining evidence for subtask i on predicting the answer sequence.

Figure 3 demonstrates how the algorithm generates a single execution path for a GZ2 task (L2 hierarchy) when the current subtask is $st2$. The upper portion shows a path sampled over the general model, the bottom portion shows the corresponding states of the decoupled model (dashed circles). States s_0, s^1_{st2} and s^2_{st2} are added to the partial search tree as a result of sampling this path.

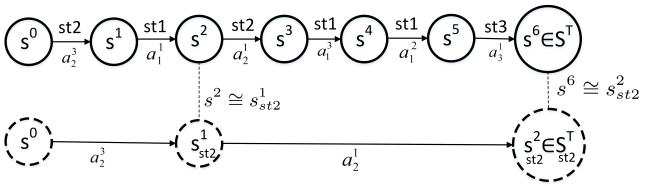

Figure 3: Execution path generated by the MC-VOT algorithm.

7. EXPERIMENTS

We evaluate the ability of the proposed system to guide hiring decisions for HCTs on a subset of the testset sampled from the GZ dataset. The testset contains 1000 randomly selected tasks, which overall hired 28131 workers and collected 70359 responses cumulatively for all subtasks given in the GZ2 subtask hierarchy. 55195 of these responses are for the subtasks in the top two levels of the hierarchy. In our experiments, the system is given access to answer models trained with all features, and evidence models trained with SDSS features and current subtask features.

Each GZ2 task in our dataset has answers from a limited number of workers, which vary between 20 and 58. While simulating our system with this data, evidence gathering may abruptly stop due to running out of data. To model this stochastic event, we expand the transition model of our MDP with a probabilistic termination model, which predicts the likelihood of running out of worker answers conditioned on the number of workers reported.

We compare the performance of the system when using different baselines and decision-theoretic algorithms to guide hiring decisions in addition to the MC-VOW and MC-VOT algorithms. The *no hire* baseline collects no responses from workers and delivers the predicted answer sequence based on the SDSS features only. The *hire all* baseline delivers a prediction based on SDSS features and all worker responses available in the dataset. We implement a limited-lookahead approach which forms a search tree up to a limited horizon to optimize decisions. The limited lookahead algorithm becomes too expensive to experiment with for lookahead sizes larger than 5 and 1 for the L2 and L4 hierarchies, respectively. We also implement the UCT algorithm, a popular Monte Carlo algorithm which uses regret analysis for action selection in its sampling. We experiment with different constants for the UCT algorithm and report the results for the best performing constant. All algorithms are given access to the same set of answer, evidence and termination models. MC-VOW, MC-VOT, and UCT algorithms can be stopped anytime to produce a result. In our experiments, we provide the same running time to all Monte-Carlo algorithms for a fair comparison. All algorithms are tested on a machine with 2.50GHz CPU and 64 GB RAM.

To understand the performance of these algorithms in varying conditions, we experiment with two different utility functions, U_{F1} and U_{EM}; we consider two different sizes for the subtask hierarchy, L2 and L4; we vary the cost of asking a subtask to a worker. In our experiments, this cost is kept constant for all subtasks.

Figure 4 summarizes the net utilities (i.e., utility of answer predictions minus the cost of hiring) obtained from different algorithms in varying conditions averaged over multiple runs. The bars in the figure represent the maximum and minimum utilities achieved through the different runs of the algorithms. In these experiments, 1.6 seconds are used in each run of the Monte Carlo algorithms, but similar trends are observed when larger running times are provided (up to 13 secs per decision). The running time of the limited-lookahead algorithms are 2.5 and 8.8 seconds for hierarchy sizes of L2 and L4 respectively. The results show that the MC-VOW algorithm performs better or as well as

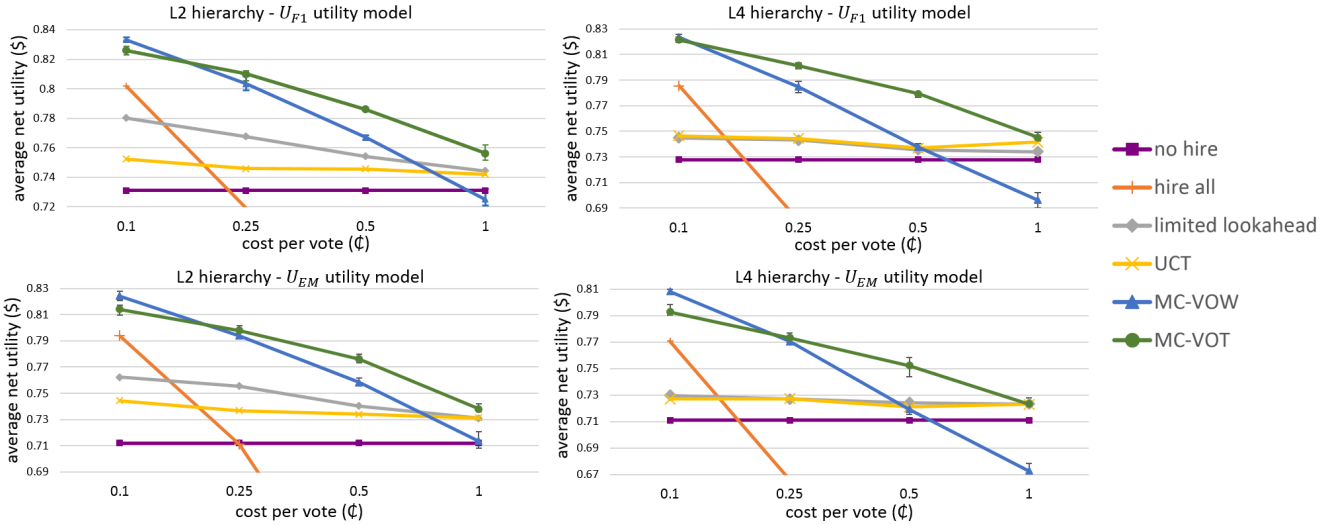

Figure 4: Each plot shows the performance of different algorithms for varying subtask costs. Plots on the top and bottom are for U_{F1} and U_{EM} utility models respectively. Plots on the left and on the right are created for different sizes of subtask hierarchies.

all other algorithms when the cost per subtask is low, since it can successfully identify when the combinatorial evidence collected from all subtasks can improve the system's predictions. However, not being able to optimize worker allocation for individual subtasks negatively affects its performance as the cost of a subtask grows. On the other hand, the MC-VOT algorithm performs better than the MC-VOW algorithm and better or equal to all other algorithms and baselines when the cost of a subtask is higher, since it can optimize worker allocation for subtasks and can identify which subtasks help to make better predictions for HCTs. The difference between MC-VOW and MC-VOT for larger costs grows when the subtask hierarchy is larger, since it becomes more costly to hire a worker for all subtasks and there are more opportunities for optimizing access to individual subtasks. The results also show that average utilities gained by all decision-theoretic algorithms are lower when the utility function is U_{EM}, as this utility function is less forgiving to the mistakes made by approximate optimization. When the cost of a subtask is not very high, both MC-VOW and MC-VOT algorithms outperform UCT and limited-lookahead algorithms. In most cases, UCT and limited-lookahead algorithms fail to efficiently explore the large horizon of HCTs and fail to identify when collecting more evidence can help with solving HCTs. Even larger running times are insufficient for UCT to explore action sequences that do not stop evidence collection prematurely.

Comparison of MC-VOW and MC-VOT algorithms with the *hire all* baseline shows that decision theoretic optimization offer benefits for solving HCTs. The algorithms can adapt their hiring policies to different cost values, task hierarchies, and utility functions. Figure 5 shows how the two algorithms allocate worker resources to subtasks when the cost of a subtask is 1.0 and 0.1 cents. When the cost is low, the MC-VOW algorithm is able to capture 99.6% of the raw utility captured by the *hire all* baseline by only hiring 40% of available workers. The MC-VOT algorithm achieves 97.5% of the raw utility by hiring 30% of workers and asking 17% of available subtasks, since it can dynamically select which

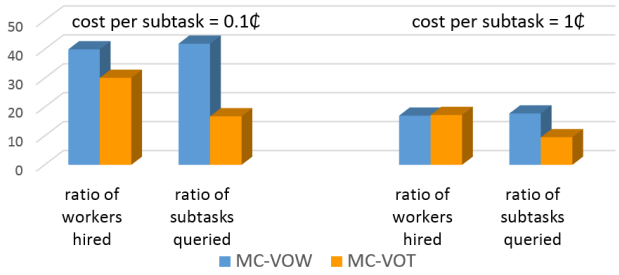

Figure 5: Hiring behaviors of MC-VOW and MC-VOT algorithms for U_{F1} utility model and L4 hierarchy.

subtasks to query. The algorithms adapt to higher costs by reducing the amount of evidence they collect from workers.

8. CONCLUSION AND FUTURE WORK

We presented our efforts to develop a methodology for making principled decisions for solving hierarchical consensus tasks. We studied a workflow where every worker answers a task by starting from the top-level subtask. However, other workflows are possible, including sequences where effort is saved via workers starting from an intermediate subtask in the hierarchy. To start from an intermediate subtask, the system and the worker would need to reach an agreement on the answers for the higher-level subtasks. Without such an agreement, conflicts could occur. Such an experience would require a new interaction design for HCTs, and is an interesting future direction for this work. We are exploring these and other task structures in crowdsourcing that can benefit from the techniques presented in this work. We are also investigating new Monte Carlo planning algorithms that can combine the strengths of MC-VOW and MC-VOT algorithms while preserving computationally efficiency.

REFERENCES

[1] J. Bragg, Mausam, and D. Weld. Crowdsourcing multi-label classification for taxonomy creation. In *First AAAI Conference on Human Computation and Crowdsourcing*, 2013.

[2] P. Dai, C. Lin, Mausam, and D. Weld. POMDP-based control of workflows for crowdsourcing. In *Artificial Intelligence*, 2013.

[3] T. Dean and S.-H. Lin. Decomposition techniques for planning in stochastic domains. In *Proceedings of the 14th international joint conference on Artificial intelligence-Volume 2*, pages 1121–1127. Morgan Kaufmann Publishers Inc., 1995.

[4] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[5] E. Kamar, S. Hacker, and E. Horvitz. Combining human and machine intelligence in large-scale crowdsourcing. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 467–474. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[6] E. Kamar and E. Horvitz. Light at the end of the tunnel: A Monte Carlo approach to computing value of information. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 571–578. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[7] E. Kamar, A. Kapoor, and E. Horvitz. Lifelong learning for acquiring the wisdom of the crowd. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2313–2320. AAAI Press, 2013.

[8] M. Kearns, Y. Mansour, and A. Ng. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. In *Proceedings of the 16th international joint conference on Artificial intelligence-Volume 2*, pages 1324–1331. Morgan Kaufmann Publishers Inc., 1999.

[9] S. Kiritchenko, S. Matwin, and A. F. Famili. Functional annotation of genes using hierarchical text categorization. In *in Proc. of the BioLINK SIG: Linking Literature, Information and Knowledge for Biology (held at ISMB-05*. Citeseer, 2005.

[10] L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. *Machine Learning: ECML 2006*, pages 282–293, 2006.

[11] D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 170–178. Morgan Kaufmann Publishers Inc., 1997.

[12] C. Lin, M. Mausam, and D. Weld. Crowdsourcing control: Moving beyond multiple choice. In *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[13] C. Lin, M. Mausam, and D. Weld. Dynamically switching between synergistic workflows for crowdsourcing. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[14] R. Parr. Flexible decomposition algorithms for weakly coupled Markov decision problems. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 422–430. Morgan Kaufmann Publishers Inc., 1998.

[15] D. Shahaf and E. Horvitz. Generalized task markets for human and machine computation. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[16] V. S. Sheng, F. Provost, and P. G. Ipeirotis. Get another label? Improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 614–622. ACM, 2008.

[17] C. N. Silla Jr and A. A. Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1-2):31–72, 2011.

[18] D. Silver and J. Veness. Monte-Carlo planning in large POMDPs. *Advances in Neural Information Processing Systems (NIPS)*, 2010.

[19] L. Tran-Thanh, M. Venanzi, A. Rogers, and N. R. Jennings. Efficient budget allocation with accuracy guarantees for crowdsourcing classification tasks. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 901–908. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[20] K. W. Willett, C. J. Lintott, S. P. Bamford, K. L. Masters, B. D. Simmons, K. R. Casteels, E. M. Edmondson, L. F. Fortson, S. Kaviraj, W. C. Keel, et al. Galaxy zoo 2: Detailed morphological classifications for 304 122 galaxies from the Sloan Digital Sky Survey. *Monthly Notices of the Royal Astronomical Society*, 435(4):2835–2860, 2013.