DIGITAL ACCESS TO SCHOLARSHIP AT HARVARD
DASH.HARVARD.EDU

HARVARD LIBRARY
Office for Scholarly Communication

Toward Automatic Task Design: A Progress Report

Citation
Eric, Huang, Haoqi Zhang, David C. Parkes, Krzysztof Z. Gajos, and Yiling Chen. 2010. Toward automatic task design: a progress report. In Proceedings of the ACM SIGKDD Human Computation Workshop: July 25-28, 2010, Washington DC, 77-85. New York, NY: Association for Computing Machinery.

Published Version
doi:10.1145/1837885.1837908

Permanent link
http://nrs.harvard.edu/urn-3:HUL.InstRepos:9938753

Terms of Use
This article was downloaded from Harvard University's DASH repository, and is made available under the terms and conditions applicable to Other Posted Material, as set forth at http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#LAA

Share Your Story

The Harvard community has made this article openly available.
Please share how this access benefits you. Submit a story .

Accessibility

Toward Automatic Task Design: A Progress Report

Eric Huang
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138 USA
ehhuang@post.harvard.edu

Haoqi Zhang
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138 USA
hq@eecs.harvard.edu

David C. Parkes
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138 USA
parkes@eecs.harvard.edu

Krzysztof Z. Gajos
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138 USA
kgajos@eecs.harvard.edu

Yiling Chen
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138 USA
yiling@eecs.harvard.edu

ABSTRACT

A central challenge in human computation is in understanding how to design task environments that effectively attract participants and coordinate the problem solving process. In this paper, we consider a common problem that requesters face on Amazon Mechanical Turk: how should a task be designed so as to induce good output from workers? In posting a task, a requester decides how to break down the task into unit tasks, how much to pay for each unit task, and how many workers to assign to a unit task. These design decisions affect the rate at which workers complete unit tasks, as well as the quality of the work that results. Using image labeling as an example task, we consider the problem of designing the task to maximize the number of quality tags received within given time and budget constraints. We consider two different measures of work quality, and construct models for predicting the rate and quality of work based on observations of output to various designs. Preliminary results show that simple models can accurately predict the quality of output per unit task, but are less accurate in predicting the rate at which unit tasks complete. At a fixed rate of pay, our models generate different designs depending on the quality metric, and optimized designs obtain significantly more quality tags than baseline comparisons.

Categories and Subject Descriptors

J.4 [**Computer Applications**]: Social and Behavioral Sciences; J.m [**Computer Applications**]: Miscellaneous

General Terms

Design, Economics, Experimentation

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
KDD-HCOMP'10, July 25, 2010, Washington, DC, USA.
Copyright 2010 ACM 978-1-4503-0222-7 ...$10.00.

Keywords

Human computation, Peer Production, Mechanical Turk

1. INTRODUCTION

In recent years there has been growing interest in the use of human computation systems for coordinating large-scale productive activity on the Internet. For example, a peer-production system like Wikipedia attracts tens of thousands of active editors that make millions of edits each month [13]. A game with a purpose like the ESP game attracts hundreds of thousands of players to label tens of millions of images through gameplay [12]. A crowdsourcing marketplace like Amazon Mechanical Turk allows requesters to post thousands of arbitrary jobs for hire each day, and attracts over a hundred thousand workers to complete these tasks [7].

A central challenge in designing human computation systems is in understanding how to design task environments that can effectively attract participants and coordinate the problem solving process. At a high level, the design of a human computation system consists of two components. One component is the design of incentives—social rewards, game points, and money—that help to attract a crowd and to encourage high quality work. The other component is the organization of individuals—the selection of participants, the assignment of subtasks, and the design of hierarchies—that help to usefully harness individual efforts to advance a system's purpose. In designing the environment for a particular task, the goal of the designer is to maximize the rate and quality of output, while minimizing costs.

In this paper, we consider a common problem that requesters face on Amazon Mechanical Turk: how should a task be designed so as to induce good output from workers? The problem exemplifies both the incentive and organizational aspects of the design challenge: in posting a task, a requester decides how to break down the task into unit tasks (called HITs, for human intelligence tasks), how much to pay for each HIT, and how many workers to assign to a HIT. These design decisions may affect the rate at which workers view and complete unit tasks, as well as the quality of the resulting work.

There are a number of challenges in effectively designing a task for posting on Mechanical Turk. The most notice-

able problem is that the effect of design on the rate and quality of work is often imprecisely known *a priori*, and likely dependent on the particular task and quality metric specified. While a designer may have some prior knowledge and be able to experiment with different designs, the design space is exponential in the number of design parameters and the number of experiments that can be performed is small. Furthermore, Mechanical Turk is inherently noisy, and any measurements obtained are affected in part by system conditions. Moreover, some statistics of interest, such as the number of currently active workers looking for tasks to perform, are unobservable by the requester.

In this work, we introduce a general approach for automatically designing tasks on Mechanical Turk. We construct models for predicting the rate and quality of work. These models are trained on worker outputs over a set of designs, and are then used to optimize a task's design. We demonstrate our approach on an image labeling task, for which we aim to maximize the number of quality labels received within a given amount of time, subject to budget constraints. We consider two measures of quality: one based on the number of distinct labels received, and another based on the number of distinct labels received that match an external gold standard.

Experimental results show that simple models can accurately predict the output per unit task for both quality metrics, and that the models generate different designs depending on the quality metric we care about. For predicting the rate of work, we observe that a task's completion time is correlated with the amount of work requested per dollar paid, and depends on the time during the day when a task is posted. But despite these effects, we find that the task completion time is nevertheless difficult to predict accurately and can vary significantly even for the same design. Focusing on using the quality prediction models for design, we find that for the same budget and rate of pay, optimized designs generated by our models obtain significantly more quality tags on average than baseline comparisons for both quality metrics.

1.1 Related work

Recent works have explored the effect of monetary incentives on worker performance on Mechanical Turk. Through a set of experiments, Mason and Watts [6] show that increasing monetary incentives induces workers to perform more units of a task, but does not affect the quality of work. In our image labeling task, we find that the quality of work can be accurately predicted without factoring in compensation. While related, our results are based on varying the task design and not the incentives for a particular design, and thus do not confirm or reject their claim. Focusing on labor supply, Horton and Chilton [2] study the effect of incentives on attracting workers to perform (multiple) HITs, and provide a method for estimating a worker's reservation wage.

Other works have considered designing Turk tasks by organizing workers and aggregating output. Snow et al. [10] consider a number of different natural language annotation tasks, and show that annotations based on the majority output among a group of Turkers is comparable in quality to expert annotations, but is cheaper and faster to obtain. Su et al. [11] consider the effect of qualification tests on worker output, and show that workers with higher test scores achieve higher accuracy on the actual task. Along an orthogonal direction, our work focuses on effectively distributing work across parallel subtasks.

An interesting example of organizing workers is TurKit [5], a toolkit for creating iterative tasks in which workers vote and improve upon work done by other workers. Little et al. [5] show that the use of voting and iteration allows for complicated tasks to be completed by a group of workers, even when the task is not easily divisible.

Building on TurKit, Dai et al. [1] propose TurKontrol, a system for controlling the request of additional voting or improvement tasks based on costs and the inferred work quality. Their work applies decision-theoretic planning techniques to optimizing sequential multi-HIT workflows. In contrast, our work focuses on a complementary challenge of learning about workers and on designing individual HITs. While the TurKontrol authors have yet to test their method on Mechanical Turk, we believe such decision-theoretic approaches can be effective when combined with learning about workers and within-HIT design.

Our work is inspired in part by theoretical work by Zhang et al. [14, 15, 16] on *environment design*, which studies the problem of perturbing agent decision environments to induce desired agent behaviors. The authors introduce models and methods for incentive design in sequential decision-making domains, and advance a general approach to design that learns from observations of agent behavior to (iteratively) optimize designs.

1.2 Outline

In Section 2 we introduce the Mechanical Turk marketplace and describe the image labeling task. Before exploring different designs for this task, we detail in Section 3 an experiment to capture the amount of variability on Mechanical Turk, where we post the same task design multiple times under varying system conditions. In Section 4 we discuss our initial experiments and report on the performance of models for predicting the rate and quality of work. We consider optimizing the task based on trained models in Section 5, where we compare the performance of optimized designs to baseline designs that pay at the same rate. We discuss the implications of our experiments for automatic task design, and outline the possibilities and challenges moving forward, in Section 6.

2. MECHANICAL TURK AND THE IMAGE LABELING TASK

Amazon Mechanical Turk (`www.mturk.com`) is a crowdsourcing marketplace for work that requires human intelligence. Since its launch in 2005, a wide variety of tasks have been posted and completed on Mechanical Turk. Example tasks include audio transcription, article summarization, and product categorization. Increasingly, Mechanical Turk is also attracting social scientists who are interested in performing laboratory-style experiments [3].

On Mechanical Turk, a *requester* posts jobs for hire that registered workers can complete for pay. A *job* is posted in the form of a group of *HITs* where each HIT represents an individual unit of work that a worker can accept. A requester can seek multiple *assignments* of the same HIT, where each assignment corresponds to a request for a unique worker to perform the HIT. The requester also sets the lifetime during which the HITs will be available and the amount of time

a worker has to complete a single HIT. Optionally, the requester can also impose a qualification requirement for a worker to be eligible to perform the task.

When choosing a task to work on, a worker is presented with a sorted list of available jobs, where for each job the title, reward, expiration time, and number of HITs available are displayed. The list can be sorted by the number of HITs available (the default), the reward, creation time, or expiration time. Workers can see a brief task description by clicking the title, or choose to 'view a HIT in this group' to see a preview of a HIT. At this point the worker can choose to accept or skip the HIT. If accepted, the HIT is assigned to that worker until the HIT is submitted, abandoned, or expired. Workers are not provided with additional information on the difficulty of tasks by the system, although there is evidence of workers sharing information on requester reputation via browser extensions and on Turk-related forums.[1]

Upon receiving completed assignments, the requester determines whether to approve or reject the work. If an assignment is rejected, the requester is not obligated to pay the worker. While tasks vary greatly in pay and amount of work, the *reward* per HIT is often between \$0.01 to \$0.10, and most individual HITs require no more than ten minutes to complete. There are thousands of job requests posted at any given time, which corresponds to tens and sometimes hundreds of thousands of available HITs. For each HIT completed, Amazon charges the requester 10% of the reward amount or half a cent, whichever is more.

2.1 Our approach to task design

An exciting aspect of Mechanical Turk as a human computation platform is that it allows for any requester to post arbitrary tasks to be completed by a large population of workers. A requester has the freedom to design his or her task as desired, with the aim of inducing workers to exert effort toward generating useful work for the requester. The task design allows a requester to optimize tradeoffs among the rate of work, the quality of work, and cost. While some of the qualitative aspects of the tradeoffs are well understood—e.g., paying more will increase the rate of work, both because more workers will want to accept HITs and that each worker will want to complete more HITs [2]—optimizing the design to achieve particular tradeoffs requires a quantitative understanding of the effect. For some non-monetary aspects of task design, e.g., the division of a task into HITs and assignments, the effect on the quality and quantity of work is less well understood, even qualitatively. Such effects are also likely to be specific to the task at hand, and depend on particular requester goals and constraints.

We advance a particular design approach. For a given task, as a first step we consider a requester who experiments with a number of different designs, and uses the workers' output and measurements of system conditions to learn a *task-specific model* of the effect of design on the rate and quality of work. As a second step, we then consider the problem of using learned models to generate good designs based on their predictions. We would like to understand whether a learned model can be effective in providing a useful way to guide the search for better designs. For the rest of the paper, we detail an application of this approach for the design of an image labeling task.

[1]See `http://turkopticon.differenceengines.com/` and `http://www.turkernation.com/`, respectively.

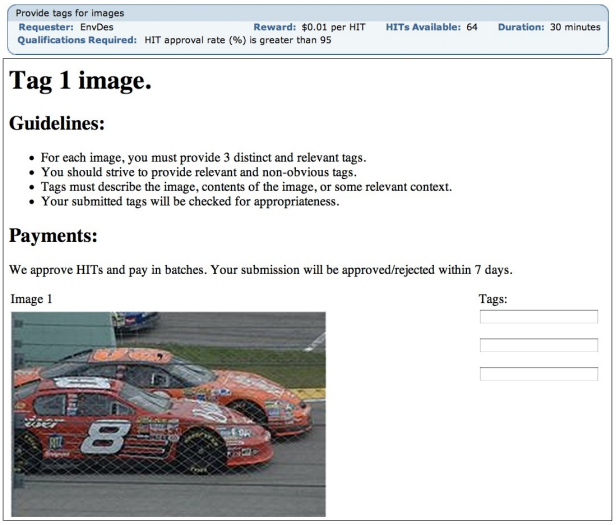


Figure 1: A HIT of the image labeling task

2.2 The image labeling task

We consider an image labeling task in which workers are asked to provide relevant labels (or equivalently, tags) to a set of images. Each HIT contains a number of images, and for each image, requests a particular number of labels for that image. Workers are informed of the number of images and number of labels required per image within the guidelines provided in the HIT, and are asked to provide 'relevant and non-obvious tags.' Workers can provide tags containing multiple words if they like, but this is not required nor specified in the instructions. See Figure 1 for a sample HIT that requests 3 labels for 1 image, for which possible labels may include 'NASCAR', 'race cars', 'red', 'eight', and 'tires.'

We obtain a large dataset of images for our task from the ESP game,[2] which contains 100,000 images and labels collected through gameplay. From this dataset we use images that contain at least 10 labels, of which there are 57,745. Of these, we have used 11,461 images in our experiments. Any particular image we use appears in only one HIT.

We consider two metrics for judging the quality of labels received from workers. One metric counts the number of unique labels received, and is thus concerned with the number of relevant labels collected. The other metric counts the number of such labels that also appear as labels in our gold standard (GS) from the ESP dataset. Since such labels are those most agreed upon in the ESP game, they are labels that are likely to capture the most noticeable features of an image. In computing these metrics, we first pre-process labels to split any multi-word labels into multiple single-word labels, and convert upper case letters to lower case. We then apply the standard Porter Stemming Algorithm [8] to normalize worker and gold standard labels. This ensures that labels such as 'dog' and 'dogs' are considered the same label, which is useful for our measure of uniqueness and for comparing received labels to the gold standard. Finally, we remove *stop words* like 'a' and 'the', which accounts for 0.9% of gold standard labels and 4.6% of distinct labels collected.[3]

[2]`http://www.cs.cmu.edu/~biglou/resources/`

[3]We used a fairly short, conservative list of stop words from `http://www.textfixer.com/resources/`.

In designing the image labeling task, a designer can decide on the reward per HIT, the number of images and tags requested per image per HIT, the total number of HITs, the number of assignments per HIT, the time allotted per HIT, and the qualification requirements. The goal of the requester is to maximize the number of useful labels received as judged by the quality metric of interest, subject to any time and budget constraints. For example, a requester may have $5 to spend, and aim to collect as many unique tags as possible within the next six hours. One can compare two different designs based on the amount of useful work completed within a certain time frame, or by examining the tradeoff between the work completed per dollar spent and the rate of work.

While all the design variables may have an effect on output, we focus our efforts on designing the reward per HIT, the number of images per HIT, the number of labels requested per image, and the total number of HITs. For all of our experiments, we fix the time allotted per HIT at 30 minutes (the default), but do not expect workers to spend more than a few minutes per HIT. We fix the number of assignments per HIT at 5; this gives us multiple sets of labels per image and will enable a study of the marginal effects of recruiting an additional worker to a HIT on the quality of output in future research. We require all workers to have an approval rate of at least 95%, such that only workers with 95% or more of their previously completed HITs approved are allowed to work on our task. This helps to keep chronic spammers out, but is not overly restrictive on the workers we can attract.

When posting tasks, we collect measurements of worker views and accepts over time, the amount of time a worker spends on a HIT, and the value of output as judged by our quality metrics. We also collect system conditions such as the time of day, the number of HITs available on Turk, the page position of our posting in different list orderings, and the number of completed HITs overall in Mechanical Turk. The last statistic is not available directly, and is estimated by tracking the change in the number of HITs available for tasks in the system at two minute intervals.

3. MEASURING OUTPUT VARIABILITY

Before considering the effect of design on output, we first report on the amount of variability in the output from Mechanical Turk when using *a fixed task design*. This lets us know how much variance to expect from the system, and allows us to study the effect of system conditions on output.

In particular, we consider a design for which each HIT has a reward of $0.01, contains 1 image, and requests 3 labels. We posted a group of 20 HITs at a time, and posted 24 groups of the same task design from 4/12/10 to 4/20/10. Each group of HITs was allowed to run for approximately 8 hours, and groups of HITs were posted sequentially around the clock. All groups had at least 75% of the assignments completed, with 18 of the 24 groups finishing before the time expired.

Table 1 summarizes the mean and standard deviation of the rate and quality of output along a number of measurements.[4] The task took 5 hours and 30 minutes to complete on average, with the quickest run completing in just under 52 minutes and the longest run taking 8 hours and 37 minutes. Unlike task completion time, the number of unique labels received and the number of such labels that are in the gold standard vary much less, suggesting that the quality of output from workers remains relatively constant under different system conditions.

[4]We measure the completion time of an unfinished task as the time until the job expires (∼8 hours), and only measure the number of tags and unique workers for completed tasks.

Statistic	Mean	Std. Dev.
Time to 50% completion (min)	129.54	95.13 / 73%
Time to 100% completion (min)	330.44	124.93 / 38%
Total # of unique tags	264.56	18.06 / 7%
Total # of unique tags in GS	98.56	9.50 / 10%
# of unique workers	13.33	2.99 / 22%
Time to complete a HIT (s)	74.79	25.12 / 34%

Table 1: Statistics on an image labeling task with 20 HITs, where each HIT pays $0.01 and requests 3 labels for 1 image.

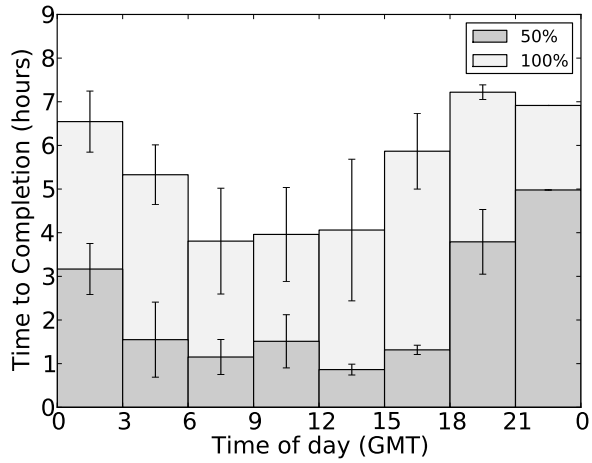
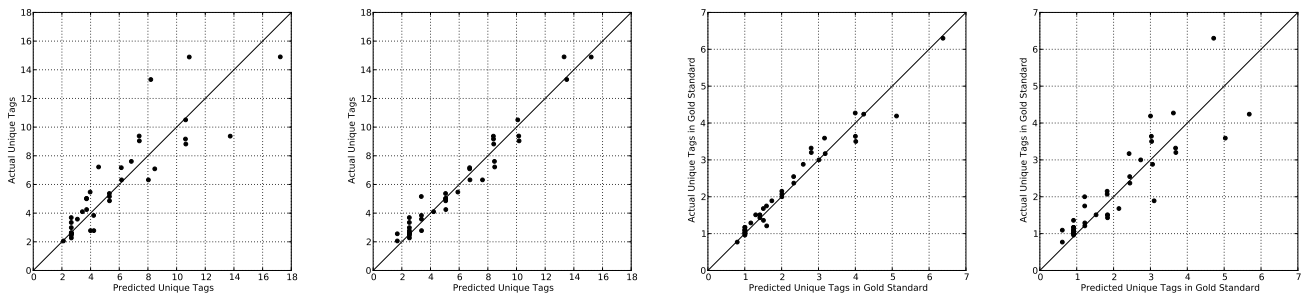


Figure 2: The effect of time of posting on time until 50% and 100% completion. Bins depict the average completion time of runs posted within a 3 hour period. Error bars represent the standard error.

One possible explanation for the significant variation in completion time is that the activity level of workers on Mechanical Turk varies over time. While we don't know how many workers are active on Mechanical Turk at any given time, it is reasonable to think that activity level is correlated with the time of day, where the system is likely to be more active during particular 'work hours' than at other times. In Figure 2 we plot the relationship between the time of posting and the time by which 50% or 100% of the task completes. We observe that jobs posted between 6AM GMT and 3PM GMT complete most quickly; this corresponds to posting between 2AM to 11AM EST in the United States and 11:30AM to 8:30PM IST in India, the two countries that provide 80% of workers on Mechanical Turk [4]. Given that these times correspond to waking hours in India, we expect most of the workers interested in this task to be from India. We geolocate workers based on their IP addresses by using the Linux shell command **whois**. Of the IP addresses for which we can determine the country of origin (247 out of 307), 62% were from India and 23% were from the US, which is consistent with our intuition.

	Estimated Model		R^2	RMSE	RRSE	MAE
	$N_{pic}N_{tag}$	$N_{pic}\log(N_{tag})$				
Diminishing returns in tags		1.9157 (0.0743)	0.7681	1.6617	0.4816	1.1341
Linear in tags	0.8426 (0.0140)		0.9576	0.7105	0.2059	0.5491
Diminishing returns in tags		0.7241 (0.0115)	0.9576	0.2574	0.2057	0.1743
Linear in tags	0.3050 (0.0115)		0.7652	0.6064	0.4853	0.4406

Table 2: Summary of model coefficients with standard errors and goodness of fit for predicting the average number of quality labels received per assignment. The top two models predict the number of unique tags collected, and the bottom two models predict the number of unique tags collected that are in the gold standard.



(a) Predicted number of unique tags based on diminishing returns in tags.

(b) Predicted number of unique tags based on total number of tags requested.

(c) Predicted number of unique tags in gold standard based on diminishing returns in tags.

(d) Predicted number of unique tags in gold standard based on total number of tags requested.

Figure 3: Predicted vs. actual number of quality tags received per assignment

4. INITIAL EXPERIMENTS AND BEHAVIORAL MODELS

From the variability measurements we learned that the completion time of a task may be highly variable, and may be difficult to predict accurately even for a fixed design. While some of the time variability can be explained by the time of day in which the task is posted, there is still a substantial amount of residual noise. In contrast, we find that the quality of work does not vary much with system conditions. Based on these observations, we expect the task design to only partially influence the rate of work, but account almost entirely for the quality of work.

In order to understand the effect of design on worker output, we consider learning models for predicting the quality of labels received per HIT and the completion time. We perform a series of 38 initial experiments—which serves as our training data—in which we vary the task's design (or configuration) by changing the reward (R), the number of images (N_{pic}) and number of labels per image per HIT (N_{tag}), and the number of HITs (N_{hits}). We consider rewards in the range of \$0.01 and \$0.10, and vary the number of images and tags requested between 1 and 10, respectively. In choosing the configurations, we aim to cover a large range of values along each dimension, and to vary the total number of tags requested per dollar pay, i.e., $N_{pic}N_{tag}/R$. For the most part we consider jobs that consist of groups of 20 HITs (in 31 configurations), but also include a few jobs containing 30, 150, 500, and 1000 HITs, respectively. Configurations were randomly ordered and allowed to run until completion, and were automatically posted in series over a three week period from 2/2/10 to 2/24/10 with no time gaps between postings. As stated previously, we fix the number of assignments (N_{asst}) requested per HIT at 5, and require all workers to have an approval rate of at least 95%.

In considering models for predicting the rate and quality of work, we measure the goodness of fit by reporting the coefficient of determination (R^2), the root mean square error (RMSE), the root relative square error (RRSE), and the mean absolute error (MAE), between predicted and actual output. All statistics are computed for the hold-out data via leave-one-out cross-validation.

4.1 Predicting label quality

We consider simple models for predicting the average number of quality labels received from workers. A summary of model coefficients and fitness is presented in Table 2.

4.1.1 Predicting unique tags

For predicting the average number of unique tags received per assignment (N_{unique}),[5] we hypothesize that we would experience diminishing marginal returns as we request more tags per image, suggesting the following model:[6]

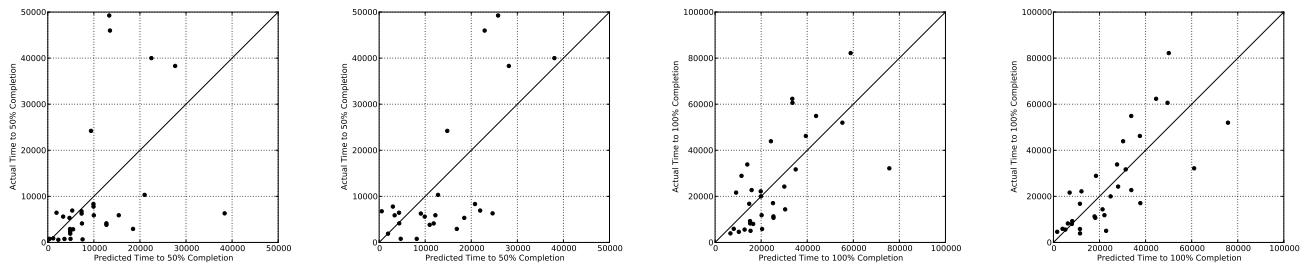
$$N_{unique} = \beta N_{pic} log(N_{tag}) + \epsilon$$

[5]We compute the per assignment contribution by dividing the number of quality tags collected per HIT by the number of assignments, which is fixed at 5.

[6]When taking a log, we smooth the input data by adding 1 to the number of tags (N_{tag}) to ensure the feature has weight instead of evaluating to zero.

	Estimated Model				R^2	RMSE	RRSE	MAE
	$\frac{N_{pic}N_{tag}}{R}$	$\cos(t)$	$\sin(t)$	Intercept				
Pay	25.58 (6.81)			-373.45 (3452.30)	0.45	12418.24	0.88	7898.78
Pay+Time	21.47 (5.26)	12173.01 (2542.22)	1019.55 (2265.39)	5177.23 (2864.85)	0.70	9819.85	0.70	7549.67
Pay	49.79 (8.06)			4955.10 (4081.98)	0.68	14914.62	0.72	11583.57
Pay+Time	44.71 (6.35)	13775.48 (3072.81)	-3103.65 (2738.20)	10801.1 (3462.78)	0.79	12630.06	0.61	9655.45

Table 3: Summary of model coefficients with standard errors and goodness of fit for predicting completion time (in seconds). The top two models predict the 50% completion time, and the bottom two models predict the completion time.



(a) Predicted time to 50% completion based on rate of pay.

(b) Predicted time to 50% completion based on rate of pay and time of posting.

(c) Predicted time to completion based on rate of pay.

(d) Predicted time to completion based on rate of pay and time of posting.

Figure 4: Predicted vs. actual time until 50% and 100% completion (in seconds).

We find that the model's predictions are somewhat accurate, with $R^2 = 0.77$ and $RRSE = 0.48$. We also consider a model without diminishing marginal returns in the number of labels requested:

$$N_{unique} = \beta N_{pic} N_{tag} + \epsilon$$

Surprisingly, we observe a significantly better fit, with $R^2 = 0.96$ and $RRSE = 0.21$; see Figures 3(a) and 3(b) for a comparison between the two models' predictions. This model suggests that the proportion of overlap in tags entered across the 5 assignments is invariant to the number of tags requested, and that at least within the range of values in our training data we don't observe workers running out of tags to describe an image.

4.1.2 Predicting unique tags in gold standard

For predicting the average number of unique tags received per assignment that are in the gold standard (N_{gs}), we again hypothesize that there would be an effect of diminishing marginal returns as we request more tags per image. Since there is a limited number of tags per image within the gold standard with which the collected tags can match, we would expect the effect of diminishing returns to be much stronger than for our other quality metric. We consider the following model:

$$N_{gs} = \beta N_{pic} \log(N_{tag}) + \epsilon$$

The prediction is highly accurate, with $R^2 = 0.96$ and $RRSE = 0.21$. The model's fit is significantly better than the fit of a model without diminishing returns ($R^2 = 0.77$, RRSE = 0.49); see Figures 3(c) and 3(d) for a comparison.

4.2 Predicting completion time

We turn to consider models for predicting completion time based on a task's design. A summary of model coefficients and fitness is presented in Table 3.

Intuitively, a task is more attractive if the pay is high but the amount of work is low. Given similar amounts of work, we would expect the number of tags requested per dollar pay (rate of pay) to be correlated with a task's completion time. We consider all 31 configurations with 20 HITs from our training data, and predict the 50% completion time ($T_{1/2}$) and 100% completion time (T) using the following model:

$$T = \beta_0 + \beta_1 \frac{N_{pic} N_{tag}}{R} + \epsilon$$

We see that the rate of pay is correlated with the completion time, with $R^2 = 0.68$ and $RRSE = 0.72$ for predicting 100% completion. The correlation is weaker for predicting 50% completion time, with $R^2 = 0.45, RRSE = 0.88$.

From the results of our variability study, we also expect the time of posting to affect the completion time. As we saw in Figure 2, the effect of time of day on completion time is sinusoidal. To incorporate this effect into our model, we convert the time of day to an angle t between 0 and 2π, corresponding to 0:00 GMT and 24:00 GMT respectively, and then encode it as two units, $\cos(t)$ and $\sin(t)$. This encoding scheme ensures that each time of day has a distinct representation and that the values for times around midnight are adjacent. Adding these time variables, we fit the following model:

$$T = \beta_0 + \beta_1 \frac{N_{pic} N_{tag}}{R} + \beta_2 \cos(t) + \beta_3 \sin(t) + \epsilon$$

	pay per tag	R	N_{pic}	N_{tag}	N_{hits}	N_{asst}	Posting Fees	Total Cost
'low pay' baseline	1/3¢	\$0.01	1	3	66	5	\$1.65	\$4.95
optimized for N_{unique} (low pay)	1/3¢	\$0.06	9	2	15	5	\$0.45	\$4.95
optimized for N_{GS} (low pay)	1/3¢	\$0.03	9	1	28	5	\$0.70	\$4.90
'high pay' baseline	1¢	\$0.04	1	4	22	5	\$0.55	\$4.95
optimized for N_{unique}, N_{GS} (high pay)	1¢	\$0.10	10	1	9	5	\$0.45	\$4.95

Table 4: Baseline and optimized designs for an image labeling tasks with a \$5 budget.

We observe an improvement in the fit, with $R^2 = 0.79$, $RRSE = 0.61$ for 100% completion time, and $R^2 = 0.70$, $RRSE = 0.70$ for 50% completion time; see Figure 4 for a comparison between the models' predictions. This improvement is more significant for predicting 50% completion time (R^2 from 0.45 to 0.70) than for 100% completion time (R^2 from 0.68 to 0.79). One possible explanation is that the effect of the time of posting diminishes as the HIT posting begins to span a longer time frame and overlap into other times of the day.

The fit of these models suggests that the rate of pay and the time of posting are correlated with the completion time, but that there is still a substantial amount of unexplained variance. To use these models for prediction and design, it would be useful to consider not only the expected completion time, but also to be mindful of the variance in the prediction. Furthermore, the current models are only trained on configurations with 20 HITs, and do not incorporate the effect of varying the number of HITs. We leave exploring these directions for future work, and for now focus on using the quality prediction models for design.

5. DESIGN EXPERIMENT

The initial experiments provide us with an understanding of how worker output responds to different design, and provide the building blocks for effective task design. Even at the same level of desirability to workers—e.g., as measured by the pay per tag, or more generally, the estimated pay per hour—we expect some designs to induce more quality output than other designs. The learned models can thus help us make informed design decisions, for particular quality metrics we care about.

We demonstrate the potential benefit of optimizing designs based on the learned models by considering a simple design experiment in which we compare different designs at a fixed pay per tag. We focus our comparison on the number of quality labels received (per dollar spent), and do not concern ourselves with the rate at which work completes.[7] Fixing the rate of pay allows us to compare designs based on the kind of work they request, and removes the effect of assigning more work at a lower rate of pay to get more quality labels from confounding the comparison.

We consider experiments at two pay rates: a low rate that pays $\frac{1}{3}$¢ per tag, and a high rate that pays 1¢ per tag. For each pay rate, we compare the output of *baseline designs* to designs optimized for each of our two quality metrics. Each design is given a budget of \$5, which must account for fees paid to Amazon as well as payments to workers. As in our initial experiments, the number of assignments per HIT (N_{asst}) is fixed at 5.

To optimize the task design, we choose values for the reward per HIT (R), number of images per HIT (N_{pic}), number of tags requested per image (N_{tag}), and the total number of HITs (N_{hits}) to maximize the total number of quality tags received as predicted by the model with the best fit, subject to budget and rate of pay constraints. We consider rewards in the range of \$0.01 and \$0.10 per HIT, and the number of images and tags requested per image in the range of 1 and 10, respectively. For example, the following formulation captures the optimization problem for finding a design that maximizes the total number of unique tags received as predicted by our model, subject to a \$5 budget and a pay rate of \$0.01 per tag:

$$\max_{R, N_{pic}, N_{tag}, N_{hits}} 0.8426 N_{pic} N_{tag} N_{hits} N_{asst}$$

$$N_{HIT} N_{asst} (R + \max(0.1R, 0.005)) \leq 5 \quad (1)$$

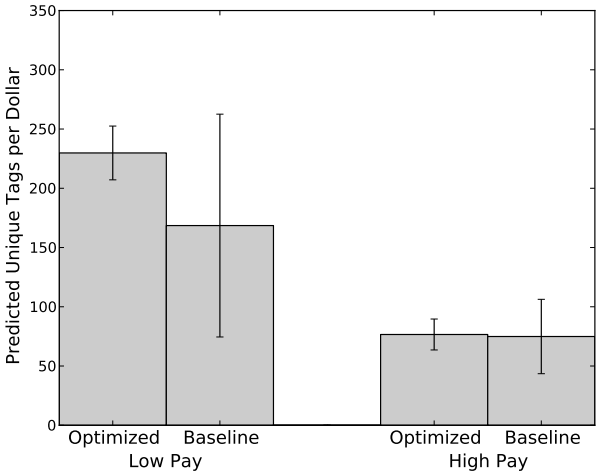
$$R / N_{pic} N_{tag} = 0.01 \quad (2)$$

Constraint 1 ensures that the cost of the design stays within budget, and constraint 2 ensures that the pay per tag is \$0.01. The max term in the budget constraint corresponds to Mechanical Turk's per assignment fees, which is 10% of the reward or half a cent, whichever is more.

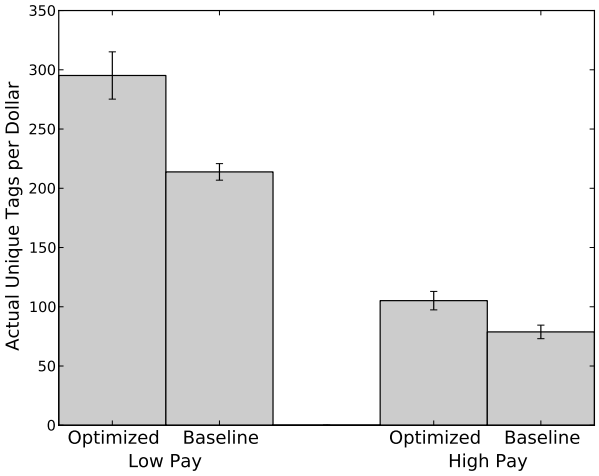
Table 4 summarizes the baseline and optimized designs for both pay rates and quality metrics. For the low pay rate, we use as the baseline the same design that we had considered previously for measuring variability (but with more HITs). For maximizing the number of unique tags collected, we see that the optimized design attempts to save on posting fees by putting more work into a HIT and paying more per HIT, which allows for more tags to be requested. For maximizing the number of unique tags that are in the gold standard, the optimized design avoids diminishing returns by requesting 1 tag per image, and also saves on posting fees by putting more work in a HIT. For the high pay rate, we consider a baseline design that requests 4 tags for one image. Here the optimized designs for the two quality metrics are the same; more work is put into each HIT to save on posting fees (hitting the upper bound on reward per HIT), and only 1 tag is requested per image to avoid diminishing returns.

Figures 5(a) and 5(c) show the models' predictions with bars representing the 95% prediction interval for these designs. We see that the difference in the predicted numbers of unique tags per dollar spent between baseline and optimized designs is small, since the benefits of the optimized design is only from savings in posting fees. By avoiding diminishing returns in tags, designs optimized for the numbers of unique tags that are in the gold standard are expected to perform significantly better.

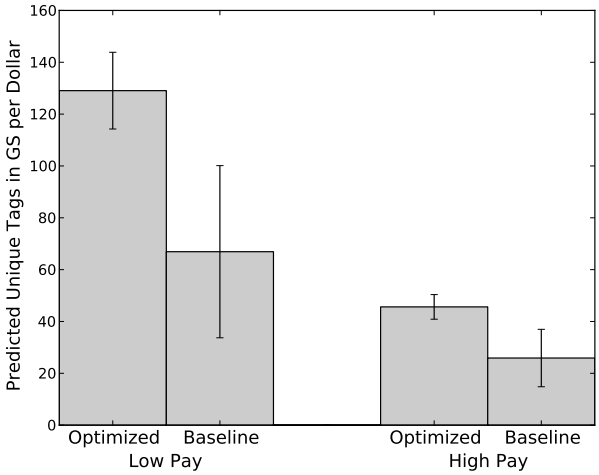
[7] In practice, we can set the rate of pay based on how quickly we want work to get done. But since time is not considered in this experiment, fixing the rate of pay allows for a fair comparison between designs.



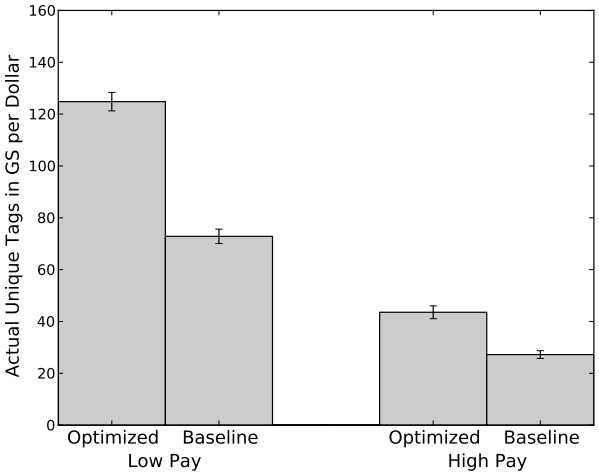
(a) Predicted number of unique tags per dollar spent.



(b) Actual number of unique tags per dollar spent.



(c) Predicted number of unique tags in gold standard per dollar spent.



(d) Actual number of unique tags in gold standard per dollar spent.

Figure 5: Predicted and actual number of quality tags received per dollar spent for baseline and optimized designs. Error bars in predictions indicate the 95% prediction intervals, and error bars in results represent the standard error over 5 runs of each design.

5.1 Results

We post 5 groups of each design in round-robin order. Each group was initially ran for 6 hours; if it didn't finish within 6 hours, it was resumed at a later time.[8] Figures 5(b) and 5(d) show the average number of unique tags and unique tags in the gold standard received per dollar spent, with bars capturing the standard error of the mean.

In all comparisons, we find that the optimized models received more quality tags than baseline comparisons. The optimized designs for unique tags received 38% more tags in the low pay condition, and 33% more in the high pay condition. For collecting unique tags that are in the gold standard, the optimized designs received significantly more quality tags than the baseline comparisons, with 71% more in the low pay condition and 60% more in the high pay condition. For all baseline and optimized designs, the actual number of gold standard tags received is very close to our model's predictions (within 11%), and well within the prediction intervals.

Interestingly, our optimized designs received significantly more unique tags than our models predicted, by 28% in the low pay condition and 38% in the high pay condition. One possible explanation is that our model underpredicts the number of unique tags when the number of tags requested per image is low as is the case in our designs. After checking the model's predictions on the training data, we notice that our model underpredicts for 10 out of the 11 configurations that request one or two tags per image (by 15% on average). Our model also underpredicted the number of unique tags obtained by the baseline in the low pay condition by 27%,

[8]We initially posted the baseline designs between 3/25/10 and 3/29/10, and the optimized designs between 4/22/10 and 4/26/10. While almost all trials of the high pay configurations completed within this time frame, many of the low pay configurations did not; these configurations were ran to completion between 4/29/10 and 5/7/10.

suggesting that the model may need to be refined to obtain more accurate predictions.

6. DISCUSSION

By learning about how workers respond to designs in our initial experiments, we are able to construct models that can accurately predict worker output in response to different designs. These models are then used to optimize the task's design, subject to designer constraints such as budget and rate of pay, to induce quality output from workers. Our results show that optimized designs generated by our models obtain significantly higher quality labels than baseline comparisons.

We are interested in extending the current work along a number of directions. We would like to understand the effect of distributing work across multiple assignments on the quality of output, and to include the number of assignments as a design variable. We are also interested in revisiting models for predicting the rate of work, and figuring out how to incorporate them to design with respect to time-related tradeoffs. One possible direction is to learn the relative rates at which work completes for different designs, which may be sufficient for accurately predicting the relative output between designs. Furthermore, while this work focuses on the design of a task with identical, parallel subtasks, we are also interested in the design of 'organizational hierarchies' [9] among workers, where workers can leverage each others' work and perform different roles in the task.

We believe the same approach of learning from observations of behavior to optimize designs can be effectively used to design other tasks, with respect to different performance metrics, and in richer design spaces. While linear regressions were used for this work, other modeling approaches and learning techniques can be similarly incorporated into the design process. Models of behavior need to be specific to the particular task and performance metric at hand; constructing accurate models will likely require drawing from social science theories, gaining an understanding of the task domain and the population of workers, and learning from experimentation.

In addition to accurate models, we need methods that help to discover effective designs after few experiments. While we trained our models on a set of manually picked designs and then used these models to design once, we can imagine a process that automatically picks subsequent experiments in a way that drives the search for better designs. One such method is suggested by Zhang et al. [14], which iteratively optimizes a design by selecting subsequent designs to experiment with based on the model's current predictions of the best design. Methods can also take into account the cost of exploration, and make tradeoffs between exploiting known designs and conducting more experiments. The development of such methods for task design—methods that automate the process of experimentation, learning, and optimization—presents an exciting area for future investigation.

Acknowledgments

The second author acknowledges support from the Department of Defense (DoD) through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program. In addition, research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

7. REFERENCES

[1] P. Dai, Mausam, and D. S. Weld. Decision-theoretic control of crowd-sourced workflows. In *the 24th AAAI Conference on Artificial Intelligence (AAAI'10)*, Atlanta, Georgia, 2010.

[2] J. Horton and L. Chilton. The labor economics of paid crowdsourcing. *CoRR*, abs/1001.0627, 2010.

[3] J. Horton, D. G. Rand, and R. J. Zeckhauser. The Online Laboratory: Conducting Experiments in a Real Labor Market. *SSRN eLibrary*, 2010.

[4] P. G. Ipeirotis. Demographics of mechanical turk. *CeDER Working Papers*, 2010.

[5] G. Little, L. B. Chilton, M. Goldman, and R. C. Miller. Turkit: tools for iterative tasks on mechanical turk. In *KDD-HCOMP '09*, Paris, France, June 2009.

[6] W. Mason and D. J. Watts. Financial incentives and the "Performance of Crowds". In *KDD-HCOMP '09*, Paris, France, June 2009.

[7] J. Pontin. Artificial intelligence, with help from the humans. *The New York Times*, March 2007.

[8] M. F. Porter. *An algorithm for suffix stripping.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[9] H. A. Simon. The sciences of the artificial. *Cambridge, Ma: MIT Press*, 1969.

[10] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng. Cheap and fast - but is it good? evaluating non-expert annotations for natural language tasks. In *the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 254–263, Honolulu, October 2008.

[11] Q. Su, D. Pavlov, J.-H. Chou, and W. C. Baker. Internet-scale collection of human-reviewed data. In *the 2007 International World Wide Web Conference*, Banff, Alberta, Canada, May 2007.

[12] L. von Ahn and L. Dabbish. Designing games with a purpose. *Commun. ACM*, 51(8):58–67, 2008.

[13] Wikipedia statistics, January 2010. http://stats.wikimedia.org/EN/TablesWikipediaEN.htm.

[14] H. Zhang, Y. Chen, and D. C. Parkes. A general approach to environment design with one agent. In *the 21st International Joint Conference on Artificial Intelligence (IJCAI-09)*, Pasadena, CA, 2009.

[15] H. Zhang and D. C. Parkes. Value-based policy teaching with active indirect elicitation. In *the 23rd AAAI Conference on Artificial Intelligence (AAAI'08)*, Chicago, IL, 2008.

[16] H. Zhang, D. C. Parkes, and Y. Chen. Policy teaching through reward function learning. In *the 10th ACM Electronic Commerce Conference (EC'09)*, Stanford, CA, 2009.