Revolt: Collaborative Crowdsourcing
for Labeling Machine Learning Datasets

Joseph Chee Chang
Carnegie Mellon University
Pittsburgh, PA 15213
josephcc@cs.cmu.edu

Saleema Amershi
Microsoft Research
Redmond, WA 98052
samershi@microsoft.com

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

ABSTRACT

Crowdsourcing provides a scalable and efficient way to construct labeled datasets for training machine learning systems. However, creating comprehensive label guidelines for crowdworkers is often prohibitive even for seemingly simple concepts. Incomplete or ambiguous label guidelines can then result in differing interpretations of concepts and inconsistent labels. Existing approaches for improving label quality, such as worker screening or detection of poor work, are ineffective for this problem and can lead to rejection of honest work and a missed opportunity to capture rich interpretations about data. We introduce *Revolt*, a collaborative approach that brings ideas from expert annotation workflows to crowd-based labeling. Revolt eliminates the burden of creating detailed label guidelines by harnessing crowd disagreements to identify ambiguous concepts and create rich structures (groups of semantically related items) for post-hoc label decisions. Experiments comparing Revolt to traditional crowdsourced labeling show that Revolt produces high quality labels without requiring label guidelines in turn for an increase in monetary cost. This up front cost, however, is mitigated by Revolt's ability to produce reusable structures that can accommodate a variety of label boundaries without requiring new data to be collected. Further comparisons of Revolt's collaborative and non-collaborative variants show that collaboration reaches higher label accuracy with lower monetary cost.

ACM Classification Keywords

H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

Author Keywords

crowdsourcing; machine learning; collaboration; real-time

INTRODUCTION

From conversational assistants on mobile devices, to facial recognition on digital cameras, to document classifiers in email clients, machine learning-based systems have became ubiquitous in our daily lives. Driving these systems are machine
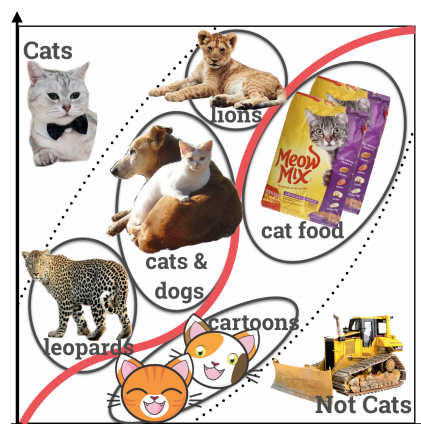
Figure 1. Revolt creates labels for unanimously labeled "certain" items (e.g., *cats* and *not cats*), and surfaces categories of "uncertain" items enriched with crowd feedback (e.g., *cats and dogs* and *cartoon cats* in the dotted middle region are annotated with crowd explanations). Rich structures allow label requesters to better understand concepts in the data and make post-hoc decisions on label boundaries (e.g., assigning *cats and dogs* to the *cats* label and *cartoon cats* to the *not cats* label) rather than providing crowd-workers with a priori label guidelines.

learned models that must be trained on representative datasets labeled according to target concepts (e.g., speech labeled by their intended commands, faces labeled in images, emails labeled as spam or not spam).

Techniques for collecting labeled data include recruiting experts for manual annotation [51], extracting relations from readily available sources (e.g., identifying bodies of text in parallel online translations [46, 13]), and automatically generating labels based on user behaviors (e.g., using dwell time to implicitly mark search result relevance [2]). Recently, many practitioners have also turned to crowdsourcing for creating labeled datasets at low cost [49]. Successful crowdsourced data collection typically requires practitioners to communicate their desired definition of target concepts to crowdworkers through guidelines explaining how instances should be labeled without leaving room for interpretation. The guideline generation process is similar but often less rigorous than the process used by expert annotators in behavioral sciences [37, 53] whereby experts independently examine a sample of data, generate guidelines especially around possibly ambiguous concepts discovered in the data, and then discuss and iterate over the guidelines based on feedback from others [33]. The guidelines are used as instructions in crowdsourced labeling

tasks given to multiple crowdworkers for redundancy. Label disagreements are commonly seen as noise or failure to carefully follow the guidelines, and later corrected through simple majority voting.

While traditional crowd-based labeling has produced many novel datasets used for training machine learning systems [17, 32, 45], a common assumption in labeled data collection is that every task has one correct label which can be recovered by the consensus of the crowd [5]. This assumption, however, rarely holds for every item even for simple concepts (e.g., *cat* vs. *not cat* as illustrated in Figure 1) and even experts have been shown to vary their labels significantly on the exact same data due to their evolving interpretations of the target concept [33]. Specifying comprehensive guidelines that cover all the nuances and subtleties in a dataset would require close examination of much of the data which is typically infeasible in the crowdsourcing settings. Crowdworkers are then often presented with incomplete guidelines and left to make their own decisions on items open to interpretation. Not only can this lead to poor quality labels and machine learning models with low accuracy, efforts to detect poor quality work (e.g., [25, 10, 22]) in these cases can actually be harmful due to rejection of honest work. More fundamentally, limiting crowdworkers to providing feedback only in terms of predefined labels, failing to capture their confusions and reasoning, presents a lost opportunity to discover and capture rich structures in the data that the crowdworkers had encountered.

In this paper, we present *Revolt*, a collaborative crowdsourcing system that applies ideas from expert annotation workflows to crowdsourcing (e.g., supporting flagging of ambiguous items and discussion) for creating high quality training labels for machine learning. Revolt enables groups of workers to collaboratively label data through three stages: *Vote* (where crowdworkers label as in traditional labeling), *Explain* (where crowdworkers provide justifications for their labels on conflicting items), and *Categorize* (where crowdworkers review explanations from others and then tag conflicting items with terms describing the newly discovered concepts). The rich information gathered from the process can then be presented to requesters at various levels of granularity for post-hoc judgments to define the final label decision boundaries.

Revolt requires no pre-defined label guidelines aside from the top-level concept of interest (e.g., faces, spam). As a result, this approach reverses the traditional crowdsourced labeling approach by shifting label requester efforts from guideline creation to post-hoc analysis. The rich structures provided by our approach has the additional benefit of enabling label requesters to experiment with different label decision boundaries without having to re-run label generation with the wide variety of possible label guidelines. For example, for collecting labels of images of *Cats* (Figure 1), Revolt produces structures that group together ambiguous sub-concepts such as *cartoon cats*, *cat food*, *leopards* and *lions* along with descriptive explanations about the structures. Requesters can then review these structures and experiment with machine learning models that are trained to identify *leopards* and *lions* as *Cats* or not.

This paper makes the following contributions:

- A new approach to crowdsourcing label collection that employs crowds to identify uncertain aspects of the data and generate rich structures for post-hoc requester judgements, instead of trying to clearly define target concepts beforehand with comprehensive guidelines.
- *Revolt*, an implementation of our collaborative crowdsourcing approach that builds structures containing rich enough information for generating training labels for machine learning. We present both real-time and asynchronous versions of our approach.
- An experiment comparing Revolt to traditional crowd-based labeling on a variety of labeling tasks showing Revolt can produce high quality labels without the need for guidelines.
- An experiment comparing Revolt to its non-collaborative variants showing the benefits of collaboration for reducing cost and increasing quality.

RELATED WORK

Data Labeling Techniques
Data labeling or annotation is a common practice for many research areas. In social and behavioral sciences, researchers annotate (or code) data to build up theories about collected data, and then analyze the annotated results to discover interesting phenomena [50]. This approach often involves multiple experts working in iterative and collaborative workflows. For example, annotators typically first examine and manually label a dataset (or subset of the dataset) independently and then compare and discuss their labels to iteratively refine a set of combined label guidelines [37, 53]. Multiple iterations of data examination, label discussion, and guideline refinement may also occur to ensure the quality and coverage of the final guidelines. Once the guidelines stabilize, annotators can then independently label additional data accordingly to produce consistent final labels with high agreement.

Similar collaborative and iterative workflows have been reported for creating high-quality labeled datasets used in natural language processing and machine learning (e.g., [39, 54, 33]). For example, Kulesza et al. [33] found that annotators often evolved their conceptual definition of a target concept and their corresponding labels throughout the course of observing more items in a dataset. Here, allowing annotators to create explicit structures designating ambiguous items discovered during labeling enabled them to gradually build up a better global understanding of the data and generate more consistent final labels. Wiebe et al. [54] also proposed an iterative and collaborative workflow that relies on comparing and discussing conflicting labels amongst expert annotators to construct and refine shared labeling guidelines for producing training labels for complex datasets. These iterative and collaborative processes provide expert annotators systematic ways to learn about and discuss different interpretations of data during labeling.

While expert annotation has been used in creating labeled datasets for machine learning, this process is often too costly and time consuming to scale to the large datasets required for modern machine learning algorithms. As an example, the Penn Treebank dataset that is commonly used in natural language processing for training part-of-speech sequence labelers

and syntactic parsers, was built by teams of linguists over the course of eight years [51]. Another example from a previous work showed labeling 1,000 English sentences took four experts nine hours each to iteratively refine their guidelines by labeling items independently then discussing together [54]. Many researchers and practitioners have therefore recently turned to crowdsourcing to label data for its scalability and relatively low cost [17, 32, 45]. However, despite its efficiency, researchers have also reported difficulty obtaining high quality labels using crowdsourcing [3, 16]. Multiple factors can contribute to poor quality labels, such as poor work from inattentive labelers, uncertainty in the task itself (resulting from poorly written guidelines or confusing interfaces), varying worker backgrounds and prior knowledge, or items that are difficult to understand by novice workers [30].

Improving the Quality of Crowdsourced Labels
While disagreements between expert annotators are typically resolved through discussing and refining guidelines [39], disagreements in crowdsourcing are commonly seen as labeling errors to be corrected through majority voting over independent redundant judgments of crowdworkers [26]. Methods for further improving the quality of crowdsourced labels can be mainly broken down into two camps [29]: techniques for preventing poor quality work and techniques for post-hoc detection of poor quality work. Prevention techniques include screening for crowdworkers capable of different tasks [18, 27], pre-training crowdworkers [19], maintaining quality while controlling for cost via dynamic task allocation [52, 8], or designing interfaces or payment structures to motivate good work [43, 47, 22]. Post-hoc identification techniques include probabilistic modeling based on crowdworker agreements for weighted voting [25], analyzing crowdworker behaviors during tasks [48], and using additional crowdworkers to review the work of others [10, 22].

A common assumption in previous work is that every item has one correct label, and conflicts among crowdworkers are the result of poor quality work from novice or inattentive workers. However, constructing comprehensive and clear instructions about how to correctly label a dataset is often not possible due to the large variety of nuances and subtleties that may exist in the data, even for seemingly simple topics. For example, requesters wanting to identify *cat* photos in a dataset might not be aware that the dataset also contains photos of *leopards*, and/or that leopards are sometimes referred to as *big cats*. As a result, crowdworkers often have to label with incomplete information. Concepts not specified in guidelines are then open to interpretation and confusion amongst crowdworkers (e.g., "should *leopards*, *lion cubs*, or *cartoon cats* be labeled as *cats*?" Figure 1), potentially leading to inconsistent labels (e.g., only some *leopard* items being labeled as *cats*). Methods for identifying poor work are ineffective in these cases and can be harmful to both crowdworker and requester reputations due to rejection of honest work. More fundamentally, this suggests a lost opportunity for requesters to discover interesting new concepts already identified by human computation during the labeling process since the crowdworkers are typically constrained to provide feedback in the form of predefined labels (e.g., *cats* or *not cats*, but not *leopards*).

Even if requesters attempt to create comprehensive guidelines, they often have to review large portions of a dataset to do so which can be prohibitively expensive. Moreover, as guidelines become more complete, they can also become longer and more complicated (e.g., [23] and [40]), requiring more crowdworker training or resulting in more errors. If the resulting label quality is inadequate, requesters will typically have to go through the tedious process of reviewing inconsistencies, identifying sources of confusion, updating the guidelines, and collecting entirely new labels given the updated guidelines [33], potentially doubling the monetary cost of the process.

Harnessing the Diversity of Crowdsourcing
Instead of treating crowdworker disagreement as noise introduced by poor work or lack of expertise, researchers have recently begun exploring methods to harness these disagreements as valuable signals. For example, researchers have found that disagreements amongst novice workers in syntactic labeling tasks often mirror disagreements amongst linguists [44] and are useful signals for identifying poor task designs [24]. In another example, Kairam and Heer [26] used label agreements to cluster crowdworkers into worker types (e.g., *liberal* and *conservative* labelers that identified different amount of targets in an entity extraction task). Manual analysis of these clusters were then used to improve future task designs. In contrast to this previous work, we use crowd disagreements to identify and explain ambiguous concepts in data for the purpose generating labeled data for machine learning.

Most closely related to our work is the MicroTalk system [21] which used crowd diversity to collect and present counterarguments to crowdworkers during labeling to improve label quality. However, this approach still assumes that a single correct label exists for every item and requires crowdworkers to pass a training stage to learn label guidelines before participating. In our work, we tackle the problem of labeling with untrained crowdworkers under the assumption that a single correct label might not exist for every item. Diversity in interpretation is then used to create rich structures of uncertain items for post-hoc label decisions, avoiding the burden of creating comprehensive guidelines beforehand and pre-training crowdworkers. We compare our approach to a condition inspired by MicroTalk called *Revote*, showing that Revolt can produce training labels with higher accuracy under the scenario where comprehensive guidelines are not available.

Structuring Unlabeled Data with the Crowd
Crowd structuring refers to tasks that make use of crowdsourcing for organizing information without predefined target concepts. For example, categorizing [12, 4] or create taxonomies [15] for a set of documents. In contrast, our work focuses on the task of *crowd labeling* which is the task of assigning predefined target concepts to each item in a dataset. In our approach, crowdworkers perform structuring within a labeling task, and only to resolve different interpretations of the same items for post-hoc requester review. Past work in crowd structuring also typically involves multiple stages completed by different crowdworkers working independently, while we
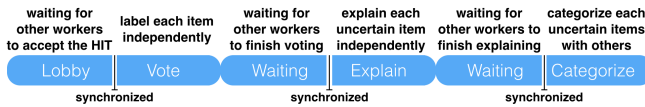
Figure 2. Overview of Revolt Stages: Synchronized stages requires all crowdworkers in the group to complete in order to move on.

took a different approach by utilizing real-time crowdsourcing to maintain a shared global structure synchronized across groups of crowdworkers working collaboratively.

Real-time Crowdsourcing

Real-time crowdsourcing has been employed for a variety of purposes including minimizing reaction time in crowd-powered user interfaces [7, 6, 9, 34, 36], increasing the speed of data collection [35, 31], and improving data quality by incorporating expert interventions to guide novice workers [11, 20, 28]. In our work, we use real-time crowd-based collaboration to improve the quality of labeled data without expert intervention. Workers engage in real-time collaboration to build rich structures of data that incorporate different crowdworker perspectives and can be analyzed post-hoc. Moreover, while most previous real-time systems employ multiple crowdworkers in a single shared workspace, our approach dynamically coordinates crowdworkers to move synchronously between stages of different subtasks. Within each stage, crowdworkers make independent judgments to be revealed to others in subsequent stages. In this way, our approach still benefits from common crowdsourcing mechanisms of verification through redundant independent judgments, but can also capture and utilize diverse crowd perspectives through collaboration.

REVOLT

In this section, we describe Revolt, a collaborative crowdsourcing system for generating labeled datasets for machine learning. Throughout this section, we use the task of labeling images as being about "*Cats*" or "*Not Cats*" as a running example (Figure 1).

At a high level, Revolt divides a dataset into multiple batches and then coordinates crowdworkers to create labels for *certain* items (items receiving unanimous labels from multiple crowdworkers) in each batch and identify *uncertain* items (items receiving conflicting labels) for further explanation and processing. In the synchronized version (Revolt), the system coordinates small teams of three crowdworkers through three synchronized stages: Vote, Explain, and then Categorize (see Figure 2). In the asynchronous version (RevoltAsync), the system elicits different crowdworkers to work independently in the Vote and Explain stages, maintaining the same redundant judgement of three crowdworkers per item while eliminating the cost of coordinating crowdworkers in real-time. After collecting crowd judgments and explanations across all batches, both systems algorithmically produce structures (groups of semantically related items) at various levels of granularity for review by label requesters to determine final label decision boundaries (e.g., assigning the "*Cartoon Cats*" category as "*Not Cats*") before training a machine learning model. To minimize redundant information, the rest of this section describes
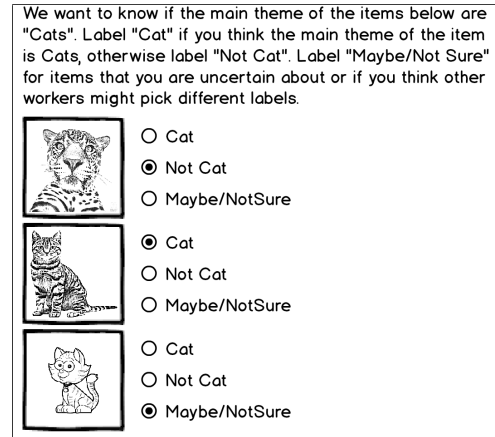

Figure 3. Human Intelligence Task (HIT) interface for the Vote Stage. In addition to the predefined labels, crowdworkers can also select *Maybe/NotSure* when they were uncertain about the item.

Revolt in the context of the synchronized version. We then describe the differences of the RevoltAsync condition.
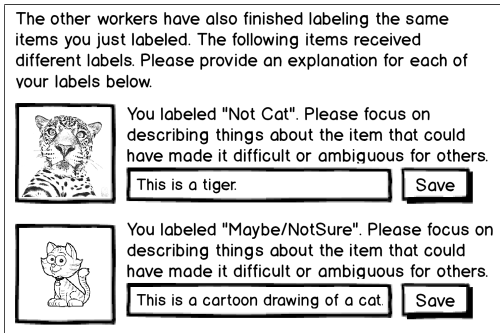
The Vote Stage

Revolt initially keeps crowdworkers in a lobby until enough crowdworkers have accepted the task and can begin as a group (Figure 2). The Vote stage then begins by collecting independent label judgments from multiple crowdworkers using an interface similar to that used in traditional crowdsourced labeling (see Figure 3). In addition to showing predefined labels as options at this stage (e.g., "*Cat*" or "*Not Cat*"), we also include a "*Maybe/NotSure*" option to ensure crowdworkers are not forced to make arbitrary decisions for uncertain items that should instead be explained further in subsequent stages. Through task instructions, crowdworkers at this stage are informed that others in the same group are also labeling the same items at the same time, and that they will be asked to compare their labels in subsequent stages. By allowing workers to express their uncertainty in the data and provide feedback in subsequent stages, Revolt avoids unfairly rejecting honest work [41].

Before Revolt can proceed to the next stage, all crowdworkers in a group must finish labeling all items in their batch. Crowdworkers who finish early are put into a waiting area where they can see in real-time how many crowdworkers in their group are still labeling items. Once the group is ready to continue, desktop and audio notifications are sent to all crowdworkers in case any stepped away while waiting. Once all labels are received, *certain* items are assigned their final labels as usual, and *uncertain* items (including items that received "*Maybe/NotSure*" labels) proceed to the Explain stage.

The Explain Stage

In the Explain stage, crowdworkers are asked to provide short explanations about their labels for items flagged as *uncertain* in the previous stage. Instructions informed each crowdworker that others in the group disagreed on the labels for these items and therefore their task was to describe their rationale for each label to the rest of the group (see Figure 4).

Figure 4. Human Intelligence Task (HIT) interface for the Explain Stage. Crowdworkers enter a short description for each item that was labeled differently in the Vote Stage. They were informed that disagreement occurred, but not the distribution of different labels used.
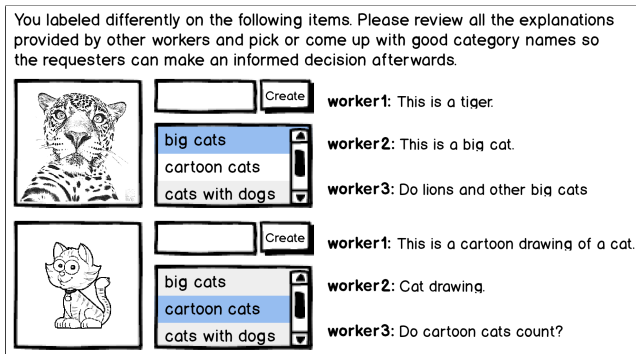
Figure 5. Human Intelligence Task (HIT) interface for the Categorize Stage. Crowdworkers select or create categories for items that were labeled differently in the Vote Stage, based on explanations from all three crowdworkers in the same group.

Note that early prototypes of our system also revealed the individual votes from other crowdworkers on each item at this stage. However, pilot experiments showed that this resulted in less descriptive explanations that were more focused on reacting to other crowdworkers. For example, people who picked the majority vote labels often simply reaffirmed or expressed confidence in their original label (e.g., "*nothing says to me that this is a cat*") , whereas people who were in the minority often just yielded to the majority (e.g., "*this could be a cat, i might have messed this one up*"). Instead, hiding the labels and only stating that a disagreement had occurred resulted in more conceptual explanations helpful for the following stage (e.g., "*This is not a cat, but rather one of the big felines. Leopard or Cheetah I think.*" and "*Although leopards are not domesticated, they are still cats.*"). As in the Vote Stage, crowdworkers who finished early were placed in a waiting area before they could move on together.

The Categorize Stage
In the Categorize stage, crowdworkers were tasked with grouping uncertain items into categories based on their explanations. In this stage, we present the same uncertain items to each crowdworker again, but this time also reveal the explanations from others in the group (Figure 5). Crowdworkers were then instructed to categorize each item based on its explanations. Categories could either be selected from a list of existing cat-

egories presented next to each item or added manually via a text input field. Whenever a new category was added by a crowdworker, each list of categories was synchronized and dynamically updated across all items within the current group, and also across groups working on different parts of the dataset. To encourage category reuse and reduce redundancy we also implemented two mechanisms: First, the text field for creating categories also acts as a quick filter of the existing categories so that crowdworkers may more easily see and select an existing category rather than create a new one when appropriate. Second, the list of existing categories is sorted by the number of crowdworkers (across all batches of the same dataset) that have used each category, a similar strategy to that used to filter out low quality crowd categories in [15, 14]. After assigning categories, crowdworkers could submit their HITs independently without waiting for others to complete.

Post Processing of Crowdworker Responses
After crowdworkers in all groups have gone through all three stages, Revolt collects the crowd feedback for all batches. Revolt assigns labels to *certain* items directly, and then creates structures of *uncertain* items by applying simple majority voting on the category names suggested by crowdworkers for each item. In cases where all crowdworkers suggested a different category, a random crowdworker's category is used as the final category. At this point, structures can be presented to label requesters for review and to make final label decisions. For example, after reviewing structures, a label requester may decide that *leopards* and *lions* should be considered *Cats* while *cartoon cats* and *cat food* should be considered *Not Cats*. In this way, label assignments can be applied to the data in each category prior to training a machine learning system.

Revolt can also expand the crowd-generated categories to different numbers of clusters, supporting inspection at different levels of granularity. To do this, Revolt performs a hierarchical clustering method that uses the crowd categories as connectivity constraints. This post-processing approach works as follows: First, a term frequency-inverse document frequency (TF-IDF) vector is used to represent each uncertain item where each dimension is the count of a term in its explanations divided by the number of uncertain items with the same term mentioned their explanations. Then, hierarchical clustering with cosine similarity is applied. That is, initially, each item is treated as a cluster by itself. Clusters are then iteratively merged with the most similar clusters, prioritizing clusters with items in the same crowd category, until all items are in the same cluster.

Generating clusters at various levels of granularity allows label requesters to adjust the amount of effort they are willing to spend in making labeling decisions, allowing them to manage the trade-off between effort and accuracy. For example, labeling low level clusters allows for more expressive label decision boundaries, but at the cost of reviewing more clusters.

RevoltAsync
RevoltAsync removes the real-time nature of Revolt as follows: One set of crowdworkers label items independently in the Vote stage. RevoltAsync then still uses the results of three

crowdworkers per item to identify uncertain items. Uncertain items are then posted to the crowdsourcing market again for a different set of crowdworkers to explain the labels. That is, in RevoltAsync's Explain stage, crowdworkers are presented with an item and a label given by another crowdworker and then asked to justify that label given the knowledge that there were discrepancies between how people voted on this item.

RevoltAsync does not include a Categorize stage, which would require synchronization. Instead it uses the explanations collected at the Explain stage directly for clustering during post-processing. Clustering of explanations is still performed using hierarchical clustering, to produce structures at different levels of granularity, but without connectivity constraints based on the crowd categories provided by the Categorize Stage.

EVALUATION

In this section, we describe experiments we conducted to investigate the cost-benefit trade-off of Revolt compared to the traditional crowdsourcing approach for collecting labeled training data. We also examined several variants of Revolt to better understand the benefits of different components of the Revolt system and workflow.

To compare these workflows, we ran each condition on a variety of datasets and measured the accuracy of the resulting labels with respect to requester effort and crowdsourcing cost. To prevent learning effects, we do not reuse crowdworkers across conditions for the same dataset, and randomize posting order of condition and dataset combinations so that crowdworkers subscribed to postings from our requester account using third party services[1] were distributed across conditions.

Baselines and Conditions

Our conditions include Revolt, RevoltAsync, three variants, and two baselines representing traditional labeling approaches:

- *NoGuidelines*. A baseline condition where crowdworkers label items without guidelines. This condition should be considered a lower bound baseline, since in most real world scenarios requesters are likely to have some knowledge of the data or desired labels to create some initial guidelines.
- *WithGuidelines*. A baseline condition where crowdworkers label items according to provided guidelines. For this condition we endeavored to create comprehensive guidelines that left no room for subjective assessment as explained in the next Datasets and Guidelines section. Since creating comprehensive guidelines is often infeasible in realistic machine learning tasks, the results from this baseline should be considered an upper bound for what can be achieved with traditional crowdsourced labeling.
- *Revolt*. Our proposed Vote-Explain-Categorize workflow with synchronized real-time collaboration.
- *RevoltAsync*. Our Revolt variant with asynchronous collaboration mechanisms.
- *Revote*. A Revolt variant with similar strategies used in [21] wherein crowdworkers re-label uncertain items after considering each others' explanations instead of categorizing

them for post-hoc requester review. This variant replaces Revolt's Categorize stage with a Revote stage (without the *maybe* option) and uses simple majority voting to assign final labels to all items.

- *Solo*. A Revolt variant with no collaboration. In this condition, each crowdworker labels and explains their labels for all items independently. The system still computes uncertain items from three redundant labels and clusters uncertain items using their explanations.
- *SoloClusterAll*. A variant of *Solo* where the system clusters all items based on their explanations. Note that clustering all items (certain and uncertain) is only possible in the *Solo* variants where explanations were collected on all items. This approach creates categories for certain items as well as uncertain, requiring requester review of even items that reached consensus through crowd labeling.

Note that no post-hoc requester effort is required in the NoGuidelines, WithGuidelines and Revote conditions and only the WithGuidelines baseline requires requesters to create guidelines prior to crowd labeling. We implemented the Revolt, Revote, Solo, and SoloClusterAll conditions using the Turk-Server library [38], which provided the infrastructure for recruiting and coordinating crowdworkers in real-time. Labels for the RevoltAsync, NoGuidelines, and WithGuidelines conditions were collected through the Mechanical Turk form builder feature on the requester interface.

Datasets and Guidelines

We evaluated each of our conditions with eight tasks made up of different data types (images and webpages) and sizes (around 100 and 600 items, respectively). All of our datasets were obtained from the publicly available ImageNet [17] or Open Directory Project[2] databases, both commonly used for machine learning research.

Each labeling task asked crowdworkers to label each item in a dataset as belonging or not belonging to a given concept. We used target concepts of *Cars* and *Cats* for our image datasets and *Travel* and *Gardening* for our webpage datasets to show that interpretations can vary for even seemingly simple and generally familiar concepts (Table 1). For our *Cars* and *Cats* image datsets, we collected images from ImageNet [17] by first collecting all images that corresponded to WordNet [42] concepts containing the keyword "car" or "cat", and then sub-sampling the set down to approximately 600 images per dataset while ensuring no sub-concept (such as *sports car* or *cable car*) was overrepresented (>10%) in the final set. We obtained our *Travel* and *Gardening* webpage datasets from [33] which has approximately 100 webpages for each concept obtained from the Open Directory Project by selecting half from each category, "travel" or "gardening", and then selecting the remainder randomly from the database.

For each dataset, we generated two sets of gold-standard labels and corresponding guidelines (making eith datasets total) representing two different interpretations of the same concept in the following way: Independently, each author first manually labeled the datasets using a structured labeling process [33]
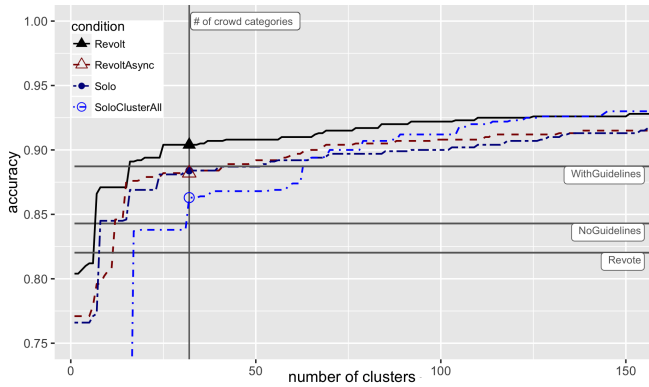
Figure 6. Accuracy of different approeaches as a function of post-hoc requester effort (i.e., number of clusters) for the Car1 dataset.

where they would categorize items as they examined them and then assign final labels at the end. This resulted in gold-standard labels and guidelines describing those labels (defined by rules each author would write down describing their categorizations and final label assignments) for that dataset. These guidelines can be considered comprehensive given that each item was examined during labeling. In realistic tasks with potentially large or complex datasets, it is often infeasible for label requesters to manually examine each item in order to create a set of guidelines (instead they typically examine a subset). Table 1 summarizes our datasets. To give some insights into the level of ambiguity that existed in each datasets, we report the proportions of items that received conflicting labels under the NoGuidelines conditions as μ. The average proportion of items being assigned the positive labels in each dataset is 0.41 ($\sigma = 0.12$).

RESULTS
In this section, we present our experimental results in terms of accuracy and cost of labels produced by each of our conditions. Final labels for the NoGuidelines, WithGuidelines, and Revote condition are assigned using simple majority voting. The Revolt, RevoltAsync, Solo, and SoloClusterAll conditions produce labels for unanimously voted items and categories (or clusters) for uncertain items. To simulate post-hoc requester judgments and measure accuracy of these conditions, we assign each uncertain category the label corresponding to the majority label of its items as defined by the gold-standards. As an example, in Table 2 we show the top eleven categories generated by Revolt for uncertain items in the *Cars* datasets and the proportion of the corresponding majority labels in two sets of gold-standard labels (e.g., 95% of the items in the *train car* category were labeled as *not car* in the gold-standard for both Car1 and Car2). This simulation allows us to produce final labels for all items which we can then compare directly to the gold-standard to compute accuracy. It is important to note that the gold-standard labels are only being used to simulate the post-hoc requester input and none of our approaches use gold-standard labels in their workflows.

In addition to presenting crowd-generated categories, Revolt (and its variant conditions) can also algorithmically produce clusters of uncertain items at various levels of granularity for

requesters to review (see the Revolt Section). As a result, requesters can vary the amount of effort they are willing to provide to produce final label decision boundaries in these conditions. Therefore, for these conditions, we also report on accuracy achieved at various levels of post-hoc requester effort. As an example, Figure 6 shows how the accuracy of Revolt changes for different amounts of requester effort required to assign labels (estimated by number of clusters needing labels) on the *Car1* dataset. For this example, receiving requester input for 32 categories produced by Revolt (see vertical line in Figure 6) achieved an accuracy higher than the upper bound WithGuidelines baseline, while other conditions did not.

We compare the accuracies of different conditions in two ways. In Table 1, we compare conditions at a fixed amount of post-hoc requester effort (i.e., the number of clusters needing examination by the requester). We fix effort to be the number of categories generated by the crowd under the Revolt condition for each dataset. For example, for the *Cars1* dataset, we compute accuracy at the point where 32 clusters would need to be examined. The accuracy results presented in the *Cars1* row in Table 1 therefore corresponds to a vertical cut of Figure 6 at the 32 number of clusters mark. To compare different conditions and baselines, we fit one generalized linear model per baseline, predicting correctness as a function of condition, with dataset as an additional factor to account for item variation. Both models significantly improve fit over a simpler model with dataset as the only variable ($X2(5)=160.1$, $p<0.01$, and $X2(5)=180.9$, $p<0.01$). Using the models, we ran general linear hypothesis tests for pairwise comparisons between conditions, and used Tukey's honestly significant difference as the test statistic. The models showed both Revolt and RevoltAsync to be significantly more accurate than the lower bound NoGuidelines condition (B=0.56 and 0.38, both p<0.01) while no significant differences were found when comparing to the upper bound WithGuidelines condition (B=0.05 and -0.13, p=0.99 and 0.63).

In addition to using a fixed numbers of clusters, Figure 7 shows the of accuracy improvement rate of each condition under different levels of requester effort relative to the NoGuidelines baseline. Since the smaller datasets only had less than 30 uncertain items, for conditions that generate rich structures (Revolt, RevoltAsync, Solo, and SoloClusterAll) we show the accuracy improvement rate for 10, 15, 20, and 25 post-hoc judgments for the smaller webpage datasets, and 10, 20, 30 for the larger image datasets. We also report the accuracy improvement for the WithGuidelines and Revote conditions that do not require post-hoc judgments.

In our experiments, $3 were paid to each worker for participating in a batch of Revolt, Revote, Solo, SoloClusterAll conditions, where $1 was paid as base payment for completing the first stage, and $2 bonuses were added for completing the rest of the stages. For the RevoltAsync condition, $1 was paid for each Vote and Explain task. We adjusted the number of items in each batch so that crowdworkers could finish batches under 20 minutes including time spent waiting for other crowdworkers. Each batch in the image datasets contained around 60 items while each batch in the webpage datasets contained

Dataset	Type	N	μ	NoGdlns.	WithGdlns.	Revote	Revolt	Solo	SoloAll	RevoltAsync	#Categories
Cars1	image	612	.27	.843	.887	.820	**.904**	.863	.884	.882	32
Cars2	image	612	.27	.756	.804	.775	**.827**	.794	.807	.820	32
Cats1	image	572	.12	.844	**.939**	.845	.916	.720	.900	.902	14
Cats2	image	572	.12	.920	**.962**	.904	.935	.787	.916	.918	14
Travel1	webpage	108	.24	.759	.870	.787	**.880**	.815	.806	.870	22
Travel2	webpage	108	.24	.769	.870	.759	**.889**	.796	.796	.870	22
Garden1	webpage	108	.12	.806	.843	.787	**.889**	.861	.759	.852	8
Garden2	webpage	108	.12	.778	.833	.787	**.843**	.815	.787	.787	8

Table 1. Accuracy of different labeling conditions. The number of clusters of the Solo, SoloClusterAll, and RevoltAsync conditions were fixed to the number of categories observed under the Revolt condition. Bold numbers indicate the best performing condition for each dataset.

Category	Size	Car1GdStdLabel		Cars2GdStdLabel	
train car	19	95%	not car	95%	not car
train	19	100%	not car	100%	not car
military vehicle	16	100%	not car	100%	not car
car	15	73%	car	53%	not car
vehicle mirror	14	100%	car	100%	not car
bumper car	12	100%	not car	100%	not car
tow truck	11	91%	car	91%	car
wheel	8	100%	car	88%	not car
truck	8	100%	car	75%	car
trolley	7	86%	car	86%	car
vehicle interior	6	100%	car	100%	not car

Table 2. Revolt categories for the Car datasets and the corresponding gold-standard label determined with majority voting for each category.

around 27 items. For the baseline conditions, we paid $0.05 for labeling one image, and $0.15 for labeling one webpage.

We also compared cost of each condition in terms of crowdworker work duration (Figure 8). For our Revolt, Revote, Solo and SoloClusterAll, we measure work duration directly by tracking crowdworker behaviors using our external HIT interface, tracking mouse movements to identify the exact time crowdworkers started working after accepting the HIT. Our NoGuidelines, WithGuidelines, and RevoltAsync conditions were implemented via the Mechanical Turk form builder feature. While Mechanical Turk does report work duration, crowdworkers often do not start work immediately after accepting a HIT. To correct for this, we approximate the work duration for these interfaces in the following way. We approximate the work time of the NoGuidelines and WithGuidelines conditions (our baseline conditions) using the timing statistics collected from the Vote Stage of the Revolt workflow, as the crowdwork involved in these baselines are of the same nature as the Vote stage. We similarly approximate the total work duration for the RevoltAsync condition by using the timestamps from the Solo condition (where crowdworkers provided explanations for each item), and multiplying the average duration with the number of uncertain items identified for each dataset in this condition.

DISCUSSION

Revolt vs Traditional Crowdsourced Labeling
In both traditional crowd-based labeling and Revolt, requesters examine uncertain items to refine the label boundaries. However, in Revolt, this is done at the category level in a post-processing step as opposed to reviewing items, refining in-

structions, and launching more tasks in a loop. The latter may lead to wasted work, particularly when refinements require workers to relabel the same items. In Revolt, structures captured from crowdworkers during labeling allow requestors to refine label boundaries post-hoc without having to launch more tasks.

When compared against the NoGuidelines condition (the lower bound of traditional crowdsourced labeling), Revolt was able to produce higher accuracy labels across all eight datasets (Table 1). The generalized linear models also showed both Revolt and RevoltAsync to be significantly more accurate than the NoGuidelines condition. The comparison of the NoGuidelines and WithGuidelines conditions shows that comprehensive guidelines indeed increase labeling accuracy across all eight datasets (Figure 7), but at the cost of the effort needed to create comprehensive guidelines in advance. In contrast, Revolt was able to produce comparable accuracy without any guidelines. In fact, in 6 out of the 8 datasets we tested, Revolt was able to produce labeling accuracies slightly higher than the upper bound baseline (Table 1). The generalized linear models also showed that neither Revolt nor RevoltAsync were significantly different than the upper bound condition (B=0.05 and -0.13, p=0.99 and 0.63). This suggests that Revolt can outperform current best practices for crowdsourcing training labels where guidelines provided by the requesters are likely to be less comprehensive than the ones provided in the WithGuidelines condition. That is, Revolt shows promise to improve the quality of labeled data collection while removing the burden of comprehensive guideline generation by making use of collaborative crowdsourcing approaches.

Forcing Crowdworkers to Revote
An alternative way of explaining why we see uncertain items with conflicting labels in Revolt's Vote Stage is that crowdworkers could converge on true concepts for all items but they are simply making mistakes while labeling. To test this, the Revote condition allowed crowdworkers to reconsider their labels after seeing explanations from others, providing the opportunity to correct mistakes. While previous work has shown accuracy improvement using this strategy under a scenario where clear guidelines were given to pre-trained crowdworkers [21], results from our study showed that the Revote condition did not improve labeling accuracy compared to the NoGuidelines lower bound baseline (B=0.03, p>0.99), with near zero median accuracy improvement (Figure 7). This suggest that in scenarios where it is infeasible to generate comprehensive
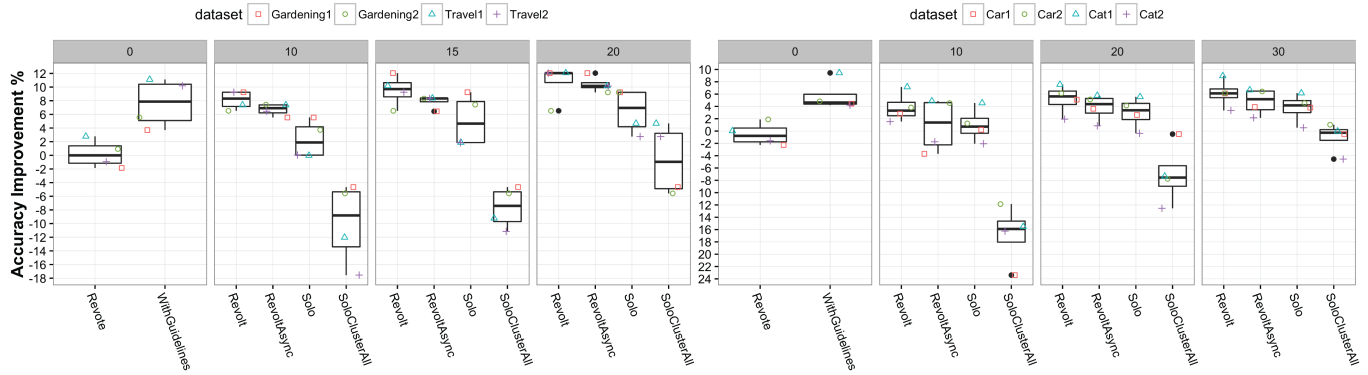
Figure 7. Accuracy improvement of different conditions over the NoGuidelines baseline as a function of requester effort.
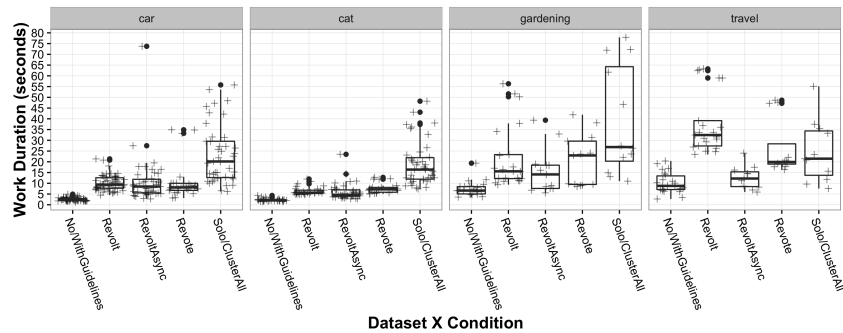

Figure 8. Work duration of each crowdworker under different conditions, normalized by the number of item in each batch.

guidelines to guide workers towards a single correct answer, accuracy cannot simply be improved by quality control on individual workers; instead allowing crowdworkers to inform the requesters about their confusions and discoveries may be a better strategy than forcing them to make arbitrary decisions and then performing post-hoc quality control.

Benefits of Collaborative Crowdsourcing

Traditionally, crowdsourcing techniques require independent crowdworker judgments and do not permit knowledge sharing. In this work, we investigated the benefits of collaborative crowdsourcing by allowing limited and structured communications between crowdworkers. Our collaborative conditions (Revolt, RevoltAsync, Revote) presented crowdworkers with different combinations of conflicting judgements, justifications for the conflicting judgements, and proposed structures (i.e., category names) from other crowdworkers, either synchronously or asynchronously. On the other hand, in NoGuidelines, WithGuidelines, Solo, and SoloClusterAll conditions, workers were not presented with any judgments from others.

Comparing Revolt to RevoltAsync, Revolt with synchronous stages performed slightly better than RevoltAsync at the cost of slightly higher worktime (Figure 8), but the difference was not significant (B=0.18, p=0.28). Comparing collaborative and non-collaborative conditions, results show that both Revolt and RevoltAsync outperformed the non-collaborative Solo condition for each dataset we tested (Figure 7). Based on the generalized linear models, the real-time collaborative Revolt condition achieved significantly higher accuracies than

the non-collaborative Solo condition, while the RevoltAsync variant did not (B=0.24 and 0.06, p=0.04 and 0.97). Interestingly, we initially expected the RevoltAsync condition would yield poorer results compared to the non-collaborative Solo condition due to cognitive dissonance (i.e., asking one crowdworker to explain the label of another). However, the results showed no significant difference between the two conditions. On the other hand, the non-collaborative SoloClusterAll condition, where explanations were collected for all items to cluster both certain and uncertain items, performed worse than the lower bound baseline. These results suggest that identifying and presenting disagreements is an important factor for eliciting meaningful explanations, even when the disagreements were presented to different crowdworkers in an asynchronous setting (Figure 7).

Cost Analysis

In general, items in the webpage datasets took longer to label compared to items in the image datasets. This is to be expected since webpages typically contain both text and images and therefore often require more effort to comprehend (Figure 8). Comparing different conditions, traditional labeling (WithGuidelines, NoGuidelines) that only required crowdworkers to label each item had the lowest work times, and the real-time collaborative conditions, Revote and Revolt, had similar and higher work times. This suggests categorizing or re-labeling items has similar costs, but creating rich structures for post-hoc requester judgments can lead to better accuracies. The RevoltAsync condition showed lower work time compared

to the Revolt condition. This is also to be expected since crowdworkers did not need to wait for others during the task for progress synchronization. The non-collaborative workflow conditions Solo and SoloClusterAll has the highest work time since it required explanation of all certain and uncertain items. Therefore, using the voting stage to identify uncertain items and guide efforts on structure generation can improve accuracy while also lowering cost.

One concern for synchronous collaborative crowdsourcing is crowdworkers idling or returning the HIT before the task is competed. This was especially important since we did not include a method for using labels from groups with missing labels. In sessions with drop-outs, we paid the crowdworkers and discarded their labels. In fact, the first prototype of Revolt had a high dropout rate of around 50% (i.e., half of the crowdworkers did not complete the three stages), making it infeasible for practical use. Through an iterative task design process, we observed the following mechanisms being effective for reducing drop-outs: Explaining the collaborative nature of the task, providing time estimates in the task instructions, adding example items in the preview screen so crowdworkers knew what to expect if they accepted the HIT, sending desktop and audio notifications to help coordinate workers, and giving clear progress indicators throughout the tasks (e.g., current bonus amount, number of remaining stages, and the amount of bonus for completing each stage). In the final version of the task with these mechanisms, the dropout rate was lowered to an average of around 5% for the eight datasets presented in this paper.

FUTURE WORK

In this work we focused on designing Revolt's collaborative crowdsourcing workflow and evaluating whether the generated structures contain information with enough richness for label requesters to define accurate label decision boundaries. While we believe the proposed mechanisms of identifying uncertainty with disagreements and creating structures with explanations can generalize to multi-class scenarios, the added complexity to both the crowdworkers and the interfaces should be studied further. Research is also needed to design a requester-facing interface depicting these structures and to compare requester effort in post-hoc analysis with guideline creation.

To gain insights into these future directions, we conducted a small follow up experiment where we ran Revolt on data needed by a group of practitioners from a large technology company for a real machine learning research problem. This group required 300 items from the publicly available 20 Newsgroup Dataset [1] to be labeled as being about "Autos" or "Not Autos". Prior to our study, the group of three practitioners already spent approximately 2-3 hours each to browse through some of the data and then about 1-2 hours to generate and iterate over the guidelines. This is a typical process for guidelines creation analogous to previous work [54], and should represent a realistic scenario somewhere between our lower bound NoGuidelines and upper bound WithGuidelines conditions. Because the practitioners already had some idea of how they wanted the dataset labeled, we ran Revolt with their guidelines.

We presented Revolt's results to one member of the research group and asked them to examine the resulting structures. Interestingly, 93 out of the 300 items (31%) were inconsistent even though we gave crowdworkers guidelines about how to label, underscoring the difficulty of creating comprehensive guidelines covering the subtleties in a dataset. These items surfaced 23 unique categories and, to the practitioner's surprise, 70% were not covered in the original guidelines (e.g., auto accessories, insurance, intoxication). 7 of the categories were mentioned in the guidelines with explicit instructions about how to label (e.g., driving, buying/selling, auto repair), indicating either failure of some workers to learn the guidelines or failure of the guidelines to capture the complexity of these categories. For example, one of the items categorized as driving was about which side of the road people should drive on. While this could be considered about auto driving, it could also be about driving other types of vehicles. Reading crowdworker explanations helped the practitioner to better understand this ambiguity, and led them to second guess their original guideline about labeling driving related items as autos. The practitioner we interviewed also suggested additional ways they wanted to use Revolt's results, such as removing ambiguous items or certain categories before training a model, or creating features around categories that surfaced. Further research is necessary to examine the potential for Revolt's rich structures to support these tasks.

The practitioner also made several suggestions with respect to how one might design the presentation of Revolt's structures. First, an indicator of category quality or confidence based on the distribution of labels assigned by individual workers would have helped the practitioner prioritize which categories to look at first and how much to examine each category before making a decision (e.g., by reading individual explanations or viewing a few of the individual items within a category). Other suggestions included blacklisting certain items from the category list (e.g., "autos" surfaced as a category), presenting structures within hierarchical categories, and searching explanations for to find related items under different categories. Future research should consider these insights in determining how to efficiently present Revolt's structures to label requesters.

CONCLUSIONS

In this paper, we presented Revolt, a new approach for generating labeled datasets for machine learning via collaborative crowdsourcing. Our experimental results comparing Revolt to traditional crowd labeling techniques demonstrates Revolt can shift the efforts of label requesters from a priori label guideline creation to post-hoc analysis of crowd-generated conceptual structures. This has several benefits including potentially surfacing new or ambiguous concepts unanticipated by label requesters, reducing the amount of crowdworker training and effort required to learn label guidelines, and allowing label requesters to change their minds about label decision boundaries without having to re-collect new data.

REFERENCES

1. 1995. The 20 Newsgroups Dataset. (1995).
http://people.csail.mit.edu/jrennie/20Newsgroups/

2. Eugene Agichtein, Eric Brill, and Susan Dumais. 2006. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 19–26.

3. Omar Alonso, Catherine C Marshall, and Marc Najork. 2013. Are some tweets more interesting than others?# hardquestion. In *Proceedings of the Symposium on Human-Computer Interaction and Information Retrieval*. ACM, 2.

4. Paul André, Aniket Kittur, and Steven P Dow. 2014. Crowd synthesis: Extracting categories and clusters from complex data. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*. ACM, 989–998.

5. Yoram Bachrach, Thore Graepel, Tom Minka, and John Guiver. 2012. How to grade a test without knowing the answers—A Bayesian graphical model for adaptive crowdsourcing and aptitude testing. *The proceedings of the International Conference on Machine Learning* (2012).

6. Michael S. Bernstein, Joel Brandt, Robert C. Miller, and David R. Karger. 2011. Crowds in Two Seconds: Enabling Realtime Crowd-powered Interfaces. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA, 33–42. DOI: http://dx.doi.org/10.1145/2047196.2047201

7. Jeffrey P Bigham, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C Miller, Robin Miller, Aubrey Tatarowicz, Brandyn White, Samual White, and others. 2010. VizWiz: nearly real-time answers to visual questions. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*. ACM, 333–342.

8. Jonathan Bragg, Daniel S Weld, and others. 2013. Crowdsourcing multi-label classification for taxonomy creation. In *First AAAI conference on human computation and crowdsourcing*.

9. Michele A. Burton, Erin Brady, Robin Brewer, Callie Neylan, Jeffrey P. Bigham, and Amy Hurst. 2012. Crowdsourcing Subjective Fashion Advice Using VizWiz: Challenges and Opportunities. In *Proceedings of the 14th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '12)*. ACM, New York, NY, USA, 135–142. DOI: http://dx.doi.org/10.1145/2384916.2384941

10. Chris Callison-Burch and Mark Dredze. 2010. Creating speech and language data with Amazon's Mechanical Turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*. Association for Computational Linguistics, 1–12.

11. Joel Chan, Steven Dang, and Steven P Dow. 2016. Improving crowd innovation with expert facilitation. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*. ACM, 1223–1235.

12. Joseph Chee Chang, Aniket Kittur, and Nathan Hahn. 2016. Alloy: Clustering with Crowds and Computation. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 3180–3191.

13. Joseph Z. Chang, Jason S. Chang, and Jyh-Shing Roger Jang. 2012. Learning to Find Translations and Transliterations on the Web. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL '12)*. Association for Computational Linguistics, 130–134.

14. Lydia B Chilton, Juho Kim, Paul André, Felicia Cordeiro, James A Landay, Daniel S Weld, Steven P Dow, Robert C Miller, and Haoqi Zhang. 2014. Frenzy: collaborative data organization for creating conference sessions. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*. ACM, 1255–1264.

15. Lydia B Chilton, Greg Little, Darren Edge, Daniel S Weld, and James A Landay. 2013. Cascade: Crowdsourcing taxonomy creation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1999–2008.

16. Gianluca Demartini, Djellel Eddine Difallah, and Philippe Cudré-Mauroux. 2012. ZenCrowd: Leveraging Probabilistic Reasoning and Crowdsourcing Techniques for Large-scale Entity Linking. In *Proceedings of the 21st International Conference on World Wide Web (WWW '12)*. ACM, New York, NY, USA, 469–478. DOI: http://dx.doi.org/10.1145/2187836.2187900

17. Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 248–255.

18. Djellel Eddine Difallah, Gianluca Demartini, and Philippe Cudré-Mauroux. 2013. Pick-a-crowd: Tell Me What You Like, and I'Ll Tell You What to Do. In *Proceedings of the 22Nd International Conference on World Wide Web (WWW '13)*. ACM, New York, NY, USA, 367–374. DOI: http://dx.doi.org/10.1145/2488388.2488421

19. Shayan Doroudi, Ece Kamar, Emma Brunskill, and Eric Horvitz. 2016. Toward a Learning Science for Complex Crowdsourcing Tasks. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 2623–2634.

20. Steven Dow, Anand Kulkarni, Scott Klemmer, and Björn Hartmann. 2012. Shepherding the crowd yields better work. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*. ACM, 1013–1022.

21. Ryan Drapeau, Lydia B Chilton, Jonathan Bragg, and Daniel S Weld. 2016. MicroTalk: Using Argumentation to Improve Crowdsourcing Accuracy. (2016).

22. Derek L Hansen, Patrick J Schone, Douglas Corey, Matthew Reid, and Jake Gehring. 2013. Quality control mechanisms for crowdsourcing: peer review, arbitration, & expertise at familysearch indexing. In *Proceedings of the 2013 conference on Computer supported cooperative work*. ACM, 649–660.

23. Google Inc. 2016. Google Search Quality Evaluator Guidelines. (2016). http://static.googleusercontent.com/media/google.com/en//insidesearch/howsearchworks/assets/searchqualityevaluatorguidelines.pdf

24. Oana Inel, Khalid Khamkham, Tatiana Cristea, Anca Dumitrache, Arne Rutjes, Jelle van der Ploeg, Lukasz Romaszko, Lora Aroyo, and Robert-Jan Sips. 2014. Crowdtruth: Machine-human computation framework for harnessing disagreement in gathering annotated data. In *International Semantic Web Conference*. Springer, 486–504.

25. Panagiotis G Ipeirotis, Foster Provost, and Jing Wang. 2010. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD workshop on human computation*. ACM, 64–67.

26. Sanjay Kairam and Jeffrey Heer. 2016. Parting Crowds: Characterizing Divergent Interpretations in Crowdsourced Annotation Tasks. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*. ACM, 1637–1648.

27. Ece Kamar, Severin Hacker, and Eric Horvitz. 2012. Combining human and machine intelligence in large-scale crowdsourcing. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 467–474.

28. Joy Kim, Justin Cheng, and Michael S Bernstein. 2014. Ensemble: exploring complementary strengths of leaders and crowds in creative collaboration. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*. ACM, 745–755.

29. Aniket Kittur, Jeffrey V Nickerson, Michael Bernstein, Elizabeth Gerber, Aaron Shaw, John Zimmerman, Matt Lease, and John Horton. 2013. The future of crowd work. In *Proceedings of the 2013 conference on Computer supported cooperative work*. ACM, 1301–1318.

30. James Q Knowlton. 1966. On the definition of "picture". *AV Communication Review* 14, 2 (1966), 157–183.

31. Ranjay A. Krishna, Kenji Hata, Stephanie Chen, Joshua Kravitz, David A. Shamma, Li Fei-Fei, and Michael S. Bernstein. 2016. Embracing Error to Enable Rapid Crowdsourcing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 3167–3179. DOI: http://dx.doi.org/10.1145/2858036.2858115

32. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

33. Todd Kulesza, Saleema Amershi, Rich Caruana, Danyel Fisher, and Denis Charles. 2014. Structured labeling for facilitating concept evolution in machine learning. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 3075–3084.

34. Gierad Laput, Walter S Lasecki, Jason Wiese, Robert Xiao, Jeffrey P Bigham, and Chris Harrison. 2015. Zensors: Adaptive, rapidly deployable, human-intelligent sensor feeds. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 1935–1944.

35. Walter S Lasecki, Mitchell Gordon, Danai Koutra, Malte F Jung, Steven P Dow, and Jeffrey P Bigham. 2014. Glance: Rapidly coding behavioral video with the crowd. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM, 551–562.

36. Walter S Lasecki, Rachel Wesley, Jeffrey Nichols, Anand Kulkarni, James F Allen, and Jeffrey P Bigham. 2013. Chorus: a crowd-powered conversational assistant. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. ACM, 151–162.

37. Kathleen M MacQueen, Eleanor McLellan, Kelly Kay, and Bobby Milstein. 1998. Codebook development for team-based qualitative analysis. *Cultural anthropology methods* 10, 2 (1998), 31–36.

38. A. Mao, Y. Chen, K.Z. Gajos, D.C. Parkes, A.D. Procaccia, and H. Zhang. 2012. TurkServer: Enabling Synchronous and Longitudinal Online Experiments. In *Proceedings of HCOMP'12*.

39. Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics* 19, 2 (1993), 313–330.

40. Matt McGee. 2012. Yes, Bing Has Human Search Quality Raters and Here's How They Judge Web Pages. (2012). http://searchengineland.com/bing-search-quality-rating-guidelines-130592

41. Brian McInnis, Dan Cosley, Chaebong Nam, and Gilly Leshed. 2016. Taking a HIT: Designing around Rejection, Mistrust, Risk, and WorkersâĂŹ Experiences in Amazon Mechanical Turk. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 2271–2282.

42. George A Miller. 1995. WordNet: a lexical database for English. *Commun. ACM* 38, 11 (1995), 39–41.

43. Tanushree Mitra, Clayton J Hutto, and Eric Gilbert. 2015. Comparing person-and process-centric strategies for obtaining quality data on amazon mechanical turk. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 1345–1354.

44. Barbara Plank, Dirk Hovy, and Anders Søgaard. 2014. Linguistically debatable or just plain wrong?. In *ACL (2)*. 507–511.

45. Matt Post, Chris Callison-Burch, and Miles Osborne. 2012. Constructing parallel corpora for six indian languages via crowdsourcing. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*. Association for Computational Linguistics, 401–409.

46. Philip Resnik and Noah A. Smith. 2003. The Web As a Parallel Corpus. *Comput. Linguist.* 29, 3 (Sept. 2003), 349–380. DOI: http://dx.doi.org/10.1162/089120103322711578

47. Jakob Rogstadius, Vassilis Kostakos, Aniket Kittur, Boris Smus, Jim Laredo, and Maja Vukovic. 2011. An assessment of intrinsic and extrinsic motivation on task performance in crowdsourcing markets. *ICWSM* 11 (2011), 17–21.

48. Jeffrey Rzeszotarski and Aniket Kittur. 2012. CrowdScape: interactively visualizing user behavior and output. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. ACM, 55–62.

49. Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and Fast—but is It Good?: Evaluating Non-expert Annotations for Natural Language Tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '08)*. Association for Computational Linguistics, 254–263. http://dl.acm.org/citation.cfm?id=1613715.1613751

50. Anselm Strauss and Juliet Corbin. 1998. *Basics of qualitative research: Techniques and procedures for developing grounded theory* . Sage Publications, Inc.

51. Ann Taylor, Mitchell Marcus, and Beatrice Santorini. 2003. The Penn treebank: an overview. In *Treebanks*. Springer, 5–22.

52. Long Tran-Thanh, Trung Dong Huynh, Avi Rosenfeld, Sarvapali D. Ramchurn, and Nicholas R. Jennings. 2014. BudgetFix: Budget Limited Crowdsourcing for Interdependent Task Allocation with Quality Guarantees. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems (AAMAS '14)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 477–484. http://dl.acm.org/citation.cfm?id=2615731.2615809

53. Cynthia Weston, Terry Gandell, Jacinthe Beauchamp, Lynn McAlpine, Carol Wiseman, and Cathy Beauchamp. 2001. Analyzing interview data: The development and evolution of a coding system. *Qualitative sociology* 24, 3 (2001), 381–400.

54. Janyce M Wiebe, Rebecca F Bruce, and Thomas P O'Hara. 1999. Development and use of a gold-standard data set for subjectivity classifications. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*. Association for Computational Linguistics, 246–253.