Your Behavior Signals Your Reliability: Modeling Crowd Behavioral Traces to Ensure Quality Relevance Annotations

Tanya Goyal ♦, **Tyler McDonnell** ♦, **Mucahid Kutlu** ★, **Tamer Elsayed** ★, **and Matthew Lease** ♦
♦ University of Texas at Austin
{tanyagoyal, tmcdonnell, ml}@utexas.edu
★ Qatar University
{mucahidkutlu, telsayed}@qu.edu.qa

Abstract

While peer-agreement and gold checks are well-established methods for ensuring quality in crowdsourced data collection, we explore a relatively new direction for quality control: estimating work quality directly from workers' behavioral traces collected during annotation. We propose three behavior-based models to predict label correctness and worker accuracy, then further apply model predictions to label aggregation and optimization of label collection. As part of this work, we collect and share a new Mechanical Turk dataset of behavioral signals judging the relevance of search results. Results show that behavioral data can be effectively used to predict work quality, which could be especially useful with single labeling or in a *cold start* scenario in which individuals' prior work history is unavailable. We further show improvement in label aggregation and reducing labeling cost while ensuring data quality.

1 Introduction

Crowdsourcing marketplaces, e.g., Amazon Mechanical Turk (MTurk), enable *requesters* to post open calls for *workers* to perform a wide range of tasks, such as image labeling, audio transcription, and language translation. While workers freely choose which tasks to work on, ensuring the quality of the submitted work can be challenging for the requesters. Statistical quality control algorithms rely on either "gold" expert labels (Snow et al. 2008) or peer-agreement (Dawid and Skene 1979) to estimate or improve the quality of collected data. However, experts may be expensive or unavailable, work history for new workers may be unavailable, and methods relying on consensus amongst multiple workers necessitate redundant work performed in parallel.

In this work, we explore an alternative to these traditional quality control mechanisms: leveraging worker *behavioral* data (Rzeszotarski and Kittur 2011; Kazai and Zitouni 2016) to estimate labeling quality and accuracy of workers. This capability offers a wide range of potential applications, such as worker selection, weighted voting, and dynamically adjusting the number of labels collected per example (Dai et al. 2013). While many weighted aggregation algorithms already exist (Sheshadri and Lease 2013), we present the first work using behavioral data for label aggregation, which could also be used in conjunction with traditional gold and peer-based methods. Estimating label quality from behavioral data also opens an opportunity to reduce the need for redundant work, in part or entirely, depending on the level of data quality required. We explore a variety of such directions and investigate the following research questions:

RQ1 *How can worker behavioral data be effectively modeled to estimate label quality in a crowdsourcing task?*

We demonstrate the feasibility of using behavioral signals to predict label correctness and worker accuracy in a relevance judging task. Our approach achieves 70.3% accuracy and 0.0463 MSE for the two tasks, respectively.

RQ2 *How can we best leverage label or worker quality estimates to better devise crowdsourcing tasks?*

We propose a behavior-based label aggregation method which uses quality estimates for weighted voting and label filtering. Results show an increase in label accuracy by using the proposed approach compared to majority voting and EM (Dawid and Skene 1979) baselines. We also propose a dynamic labeling algorithm which automatically determines the number of labels to collect for each example based on estimated quality of the existing labels. Results show that we can decrease the annotation cost by 9% without decreasing the label quality.

Contributions. Contributions of our work are as follows.

- We collect and share[1] a new dataset of crowd workers' behavioral traces during relevance judging of search results, covering 106 unique workers and 3,984 HITs in total.
- We propose three prediction models based on only behavioral signal data, as well as a hybrid model which also uses non-behavioral data. While the hybrid model achieves slightly higher prediction accuracy, the models based on only behavioral data can be used when no prior work history about the crowd workers is available.
- We present the first use of behavioral modeling for label aggregation, with improvement over standard baselines.
- We propose a behavior-driven, dynamic labeling algorithm to detect the number of annotations needed for each document-topic pair to maximize the quality of the aggregated labels with the given fixed annotation budget.

[1] http://ir.ischool.utexas.edu/webcrowd25k/

Copyright © 2018, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

2 Related Work

2.1 Task Design

Since crowdsourcing is a human-centric enterprise, attention (or inattention) to human factors will clearly impact the quality of collected data. While poor quality of crowd data has been often blamed on contributors, task instructions and/or interfaces poorly designed for non-experts may be equally culpable. Moreover, if task instructions are unclear, or a task interface poorly designed, the inevitable low quality of crowd data can at best be ameliorated by statistical quality assurance methods. In general, approaches to quality assurance based on human factors, (Alonso 2015; McDonnell et al. 2016; Drapeau et al. 2016; Wu and Quinn 2017; Manam and Quinn 2018), vs. statistical methods (Dawid and Skene 1979; Snow et al. 2008; Whitehill et al. 2009; Sheshadri and Lease 2013) are largely complementary, allowing each to be independently studied. However, it is important to recognize that the quality of data collected from the crowd may vary significantly under different task designs. For example, behavioral data collected in this study reflects workers performing a relevance judgment task with rationales (Kutlu et al. 2018) based on our instructions.

2.2 Behavioral Data: Collection and Modeling

Rzeszotarski and Kittur (2011) propose *task fingerprinting* in which crowd worker interactions with a task interface (e.g., clicks, scrolls, key-presses, etc.) are logged and then correlated with worker accuracy. Each worker is assigned a binary "pass/fail" grade based on their overall accuracy, and a decision tree classifier is then used to predict this binary grade based on behavioral data. The authors demonstrate the feasibility of such prediction across three different tasks.

Kazai and Zitouni (2016) collect behavioral data from both crowd workers and experts as they perform search relevance judging. They then train a classifier to detect poor quality work on the basis of behavioral data. Note that whereas Rzeszotarski and Kittur (2011) utilize only expert gold labels, Kazai and Zitouni (2016) require expert behavioral data, which may be more difficult to obtain in practice. In regard to feature representations, we integrate ideas from both of these works to generate our set of features.

Because neither Rzeszotarski and Kittur (2011) nor Kazai and Zitouni (2016) shared their source code or collected data, reproducibility (Paritosh 2012) or follow-on studies have been limited. Recently, we released an open-source package for collecting such behavioral data (Dang, Hutson, and Lease 2016). We adopt this package in our work and further share our collected behavioral data.

Internet searching and browsing behaviors are known to signal user intent, user engagement, and search result relevance (Joachims 2002). Banerjee and Ghosh (2001) outline an algorithm that computes path similarity between two "clickstream" sequences (i.e., the sequence of pages a user visits on a website), also taking temporal similarities into account. In bioinformatics literature, a significant amount of work in analyzing sequences of genes, clustering gene sequences, and classification of gene sequences, e.g., (Wang and Zaïane 2002; Cardona-Escobar et al. 2015) uses ideas similar to clickstream data analysis. We similarly experiment with two models that leverage the clickstream event sequences to predict label accuracy.

2.3 Optimizing Label Collection

Sheng, Provost, and Ipeirotis (2008) investigate the relative benefit of collecting more labels for the same example. They show that the expected gain of additional labels is conditional on the quality of the labelers, with greater potential benefit from higher redundancy with less accurate workers.

Dai et al. (2013) propose a model that dynamically decides whether to ask for additional labels for a given data point, or infer the final label from the already collected data points. The authors model this problem as a partially-observable markov decision process (POMDP). They adopt Whitehill et al. (2009)'s method to estimate the difficulty and true labels for current annotations, using the MDP to decide on the next optimal action (e.g., whether or not to ask for more labels). Dai, Weld, and others (2011) later demonstrate the feasibility of this approach on a real world dataset. Ipeirotis and Gabrilovich (2014) also model the dynamic task design problem as a MDP. While these works estimate labeler accuracy via expectation-maximization (EM) (Dawid and Skene 1979) or worker history, we introduce use of worker behavioral data as the basis for prediction.

3 Dataset

We recently reported on a new WEBCROWD25K dataset of 25,099 search result relevance judgments collected on MTurk (Kutlu et al. 2018). Not mentioned in that work, let alone analyzed, was that for a subset of the judgments, we also collected behavioral data of the crowd workers. In this work, we introduce and analyze this crowd behavior dataset for the first time, and we share this data to support reproducibility (Paritosh 2012) and/or follow-on studies.

Crowd relevance judgments, along with rationales for judgments (McDonnell et al. 2016), were collected for the 2014 NIST TREC Web Track (Collins-Thompson et al. 2015). This test collection consists of: 1) collection of documents (specifically, web pages); 2) a set of 50 pre-defined search topics, comprised of a title and a one-sentence description of the searcher's *information need*; and 3) gold-standard "expert" relevance judgments, where each judgment assesses the relevance of one document to one search topic. These gold judgments were collected on a six-point relevance scale (Collins-Thompson et al. 2015). For the purpose of our analysis, we collapse these judgments to a simple binary relevance scale; see Kutlu et al. (2018) for details.

In our WEBCROWD25K dataset, for each of the 50 WebTrack'14 search topics, we selected 100 documents to rejudge (without reference to the original gold judgment) by 5 MTurk workers each (50 topics x 100 documents x 5 workers = 25K crowd judgments). We collect relevance judgments on a 4-point graded scale: {Definitely Not Relevant,

Probably Not Relevant, Probably Relevant, Definitely Relevant}, and then collapse these judgments to simple binary relevant vs. non-relevant for comparison to TREC gold.

To collect behavioral data, we use the open-source *Mmm*Turkey library (Dang, Hutson, and Lease 2016), which can capture a variety of worker interaction behaviors while completing MTurk Human Intelligence Tasks (HITs). Specifically, we record: clicks, scrolls, mouse movements, key presses, copy or paste actions, and change in window focus. The raw data from the framework contains the time stamp for each event (relative to the beginning of the task) and event specific characteristics of the action itself. For instance, for a *scroll* event, the log contains information of the total number of pixels scrolled. Similarly, for a *mouse movement* event, information about the movement in the horizontal and vertical directions is recorded. *Focus change* records the time and duration the worker was inactive on the HIT window. Finally, some overall HIT information is also captured, such as total task time, browser specifications, etc.

Higher-level events can be induced from combinations of raw events (e.g., *dwell time* can be induced to capture the time between successive events). We further discuss such higher-level events when defining the features we use with our predictive models (Section 4.2).

As noted above, behavioral data was only collected for a subset of the WEBCROWD25K dataset, excluding workers completing fewer than five HITs. The resultant behavioral data covers 3,984 unique HITs spanning 2,294 unique document-topic pairs, with 1 to 5 labels per document coming from 106 unique workers. In terms of labels (associated with behavioral data) per document-topic pair, 1,170 have 1 label, 656 have 2 labels, 374 have 3 labels, 90 have 4 labels, and 4 have 5 labels. Mean worker accuracy is 65.8%.

4 Predicting Label and Worker Accuracy

In this section, we describe our use of behavioral data for two tasks (Section 4.1): 1) predicting whether a given worker's label is correct or not (i.e., classification); and 2) predicting a given worker's accuracy (i.e., regression). Critically, note that these tasks do not involve label aggregation; neither task assumes nor requires that training or testing examples have been annotated by multiple workers. Section 4.2 presents three behavioral models for performing classification and regression tasks. We describe our experimental setup in Section 4.3, and our results in Section 4.4.

4.1 Task Definitions

Classification. In training data, we record whether each worker label is correct or not (i.e., does it match the expert gold label?). At testing time, the task is to predict whether a given worker labeled a given HIT correctly, on the basis of that worker's behavioral data in completing the HIT.

As an illustrative example, consider a toy nearest-neighbor (NN) classifier. Given a test example (represented by behavioral features) for which to predict whether or not a given worker label is correct, the NN classifier would find the training instance with the most similar behavioral features and simply output its correctness label.

Regression. For this task, we aim to predict workers' time-varying task accuracy (Jung and Lease 2015). As ground truth, we estimate latent worker accuracy by a sliding-window over consecutive HITs. Specifically, for each HIT, we record the worker's accuracy at the time of that HIT as the percentage of correct labels in the worker's last t annotations (including the current HIT), where $t = 5$ in all our experiments. In addition to accounting for temporal variability, this sliding-window estimate mitigates against workers' chance agreement with binary relevance judgments. At testing time, the task requires predicting the worker's latent accuracy at the time of completing a given test HIT.

Extending our illustrative example above to regression, a toy NN model would similarly look up the most similar training point to a given test instance (i.e., the worker's behavioral features on a given test HIT) and simply output the worker's accuracy associated with that training point.

4.2 Predictive Models

We now describe three purely behavioral approaches (models and features) we pursue for predicting label correctness and worker accuracy. Since we report on fairly traditional learning models using hand-crafted features, one might wonder how a modern neural approach might compare. We investigated this, e.g., using a neural LSTM for sequence modeling (Hochreiter and Schmidhuber 1997), but did not observe competitive performance on the two tasks, likely due to data scale. Consequently, we do not further report our neural modeling results. We anticipate future work (especially from industry) could usefully exploit larger behavioral traces effectively with neural models.

4.2.1 Prediction with Aggregate Features (RF-AF)

The first behavioral approach we describe uses aggregate features (AF) from behavioral data within each HIT to induce higher-level features characterizing worker behavior on the overall HIT. We apply a *random forest* (RF) model (Breiman 2001) to both classification and regression tasks, though other models could also be used.

Aggregate features used are derived from those suggested in prior work. While Rzeszotarski and Kittur (2011) use only simple features derived from overall task statistics, such as task time, on focus time, and counts of individual events, Kazai and Zitouni (2016) include additional sophisticated features that capture the timing between successive events within a HIT. Our feature set combines both.

Table 1 describes the aggregate features we used, which are categorized into two classes: 1) *action-based features* that record information about individual types of actions; and 2) *time-based features* that capture temporal aspects of the data. Action-based features are derived from raw data by simple counting. The time-based features, largely derived from Kazai and Zitouni (2016), exploit time stamps in the raw data to capture temporal and sequential behaviors.

Table 1: Aggregate features for each HIT used with our RF-AF predictive model (Section 4.2.1).

Type	Name	Description
Action-Based	total_[X]_events	The number of logged events of type X where X $\in$ {clicks, mouse movements, scrolls, pastes, focus changes, key presses}
	mouse_movement_x/y	Total pixel movement in horizontal/vertical direction
	scroll_y	Total pixel scroll in vertical direction
Time-Based	total_time	Total time completing the HIT
	recorded_time_disparity	Difference between the total_time and the time spent outside the HIT
	on_focus_time	Fraction of the total_time that was spent completing the HIT
	dwell_time_[X]	Mean time between two successive logged events of type X, where X $\in$ {clicks, mouse movements, scrolls, pastes, focus changes, key presses}
	event_change_time_[X]	Average time between event X and any other event

4.2.2 Prediction with Sequence Features (RF-SF)

Prediction using only aggregate features (Section 4.2.1) does not capture ordering effects between events within a given HIT. For example, consider a simple HIT log listing three clicks over a 15 second period: [(t = 0, click), (t = 10, scroll), (t = 15, click)]. While aggregate features capture overall HIT properties of the events (e.g., total count, average dwell time, etc.), they do not capture the sequence between events: i.e., a click followed by a scroll, etc. Since each worker completes the same HIT (having the same set of instructions), workers are expected to perform actions in a similar order.

In this approach, we exploit such sequence features (SF) and only these features; we do not use the aforementioned aggregate features capturing total or average task statistics (future work might investigate combining both approaches). As with aggregate features, we adopt a RF model for prediction, although other models could also be used.

Similar to gene sequence classification (Cardona-Escobar et al. 2015), we extract sequence features from the event sequences in the behavior data. Given predetermined parameters k and m, we extract all possible event sequences of length $2k + m$ which is composed of two fixed event sequences of length k separated by m random events. As an example, for $k = 2$, $m = 1$, possible features include {Click, Click, <event>, Click, Scroll}, {Keypress, Click, <event>, Click, Keypress}, etc., where <event> corresponds to any possible event. We thus instantiate a set of feature templates covering all possible sequences of such events under the specified parameters. Then, given a worker event sequence for a HIT, we count how many times each feature is matched by the observed sequence (the two k length subsequence must match exactly, with m wildcard events in between), with this count defining the feature's value. Our experiments take the union of all sequence features constructed by the cross-product of k = $\{1, 2, 3\} \times$ m = $\{0, 1\}$. We also include single event features (i.e., sequences of length 1).

4.2.3 Prediction with Sequence Clusters (kmeans-SC)

Predicting with sequence features (Section 4.2.2) captures sequencing between events in a HIT but ignores the timing between those events. For example, one might expect more accurate workers proceed more slowly and carefully, while less attentive workers may rush through their work. To capture such distinctions, it is important to incorporate dwell-time features into the model. As an illustrative example, consider the following two event sequences: [(t = 0, click), (t = 2, scroll), (t = 5, click)] and [(t = 0, click), (t = 20, scroll), (t = 30, click)]. It would appear that the worker performing the first HIT rushed through the task, whereas the second worker took more time. The previous model (Section 4.2.2) would fail to differentiate between such behaviors, generating similar feature representations for both workers. Similarly, predicting with aggregate features (Section 4.2.1) would capture average and total dwell time between events in the HIT, but not the order of events.

To model such behaviors, we adapt the time-weighted path similarity model outlined by Banerjee and Ghosh (2001). The algorithm computes a similarity metric between two sequences, by looking at their longest common subsequence (LCS) and the time overlap between the corresponding pairs of events in the two sequences. The similarity calculation is dependent on two factors:

1. *Time Similarity Component*: This similarity is computed between two sequences s^1 and s^2 in their overlap region (i.e., LCS). Consider a consecutive event pair (e_i, e_{i+1}) in the LCS (the starting index i may be different for each sequence). Let $t^j_{\Delta i} = t^j_{i+1} - t^j_i$ denote the time difference between the two events for each sequence s^j. Let $m_{i,i+1} = \frac{min_j(t_{\Delta i})}{max_j(t_{\Delta i})}$, and let $C'^{12} = avg_i(m_{i,i+1}), \forall_i \in \{1, ...|LCS|\}$.
2. *Importance Component*: For each sequence, we calculate: 1) the fraction of the time spent in the overlap (LCS) t^j_{overlap} compared to the total task time t^j_{total}. Given two sequences s^1 and s^2, we calculate the importance component $C''^{12} = \prod_j \frac{t^j_{\text{overlap}}}{t^j_{\text{total}}}$.

We use the final similarity metric $C''' = C' \cdot C''$ to cluster event sequences (training and test) via *k-means clustering* (Hartigan 1975). Then, to predict label correctness for a given test instance, we find the cluster it belongs to and output the majority label over all training instances of the cluster. Similarly, to predict worker accuracy, we average the worker accuracies over all training instances in the cluster.

While this aggregation across cluster instances may seem

Table 2: Results for Classification and Regression Tasks (Section 4.4). Best result per column is bold-faced.

Model Type	Model Name	Classification (Accuracy)		Regression (MSE*100)	
		Partition by HIT	Partition by Worker	Partition by HIT	Partition by Worker
Non-Behavioral	Constant BL	65.6	65.6	5.34	5.62
	SNB	69.6	N/A	4.69	N/A
Behavioral	DT-AF	61.5	60.4	6.99	6.71
	RF-AF	67.4	67.9	4.71	**4.73**
	RF-SF	68.4	**68.8**	**4.63**	4.82
	kmeans-SC	67.9	68.1	4.97	5.01
Hybrid	RF-SF+SNB	**70.3**	N/A	4.64	N/A

to resemble traditional label aggregation, there are several key differences. Firstly, whereas label aggregation methods aggregate multiple worker labels for the same HIT, here we aggregate over multiple labels in the same cluster, which could potentially all come from the same worker over time. This means the approach could be used even when only a single label is collected for each example. Secondly, note that we are aggregating over training instances, rather than over multiple worker labels for the test instance.

4.3 Experimental Setup

Metrics. To evaluate classification and regression tasks, we report accuracy in predicting label correctness, and *mean-squared error* (MSE) in predicting worker accuracy.

Baselines. To evaluate our models, we compare their performance against the following baselines:

- **Constant Baseline (Constant BL)**: This baseline always predicts the majority class (in case of label correctness classification), or the (oracle) mean worker accuracy (for worker accuracy regression). Behavioral data is not used.
- **Supervised Naive Bayes (SNB)** (Snow et al. 2008): We estimate worker confusion matrices and class priors from the training data. During testing, these worker confusion matrices are used to obtain the final label or worker accuracy value. For any test worker not previously observed in training data, we backoff from using an individual confusion matrix to a simple average matrix over all workers (Liu and Wang 2012). Behavioral data is not used. This model is the same as in Dawid and Skene (1979) but differs in how we estimate the confusion matrices.
- **Decision Tree with Aggregate Features (DT-AF)**: We compare our models' performance against a Decision Tree model built using the same aggregate features in Table 1 used by our RF model (a RF is in fact a collection of DTs, and thus a generalization of the DT which has been generally found to yield better accuracy). This DT baseline is comparable to what was reported by Rzeszotarski and Kittur (2011), albeit with a few extra features.

Hybrid Model. We also propose a hybrid model combining the RF-SF behavioral model and SNB non-behavioral baseline **(RF-SF + SNB)**. The SNB algorithm is used to obtain predicted labels (or worker accuracy scores) for the test data. These predicted accuracy values for the test data are included as pseudo-gold labels, and utilized in training the behavioral model. Thus, the behavioral model is trained using the gold labels for the training set, as well as the pseudo gold labels for the test set. We use appropriate sample weighting to avoid biasing the model towards the pseudo gold values. This model is a much simplified version of co-training (Blum and Mitchell 1998) in which two independent models, with different views of the data, label unlabeled examples for one another to generate psuedo-gold labels to supplement a small set of actual gold labels.

Parameters and Software. For RF models, we use an ensemble of 50 trees. Nodes are split based on information gain with a minimum requirement of 10 data points for splitting. We use scikit-learn (`http://scikit-learn.org`) throughout.

Division of Data. We use 10-fold cross validation, considering two alternative ways of partitioning the data: 1) by HIT, i.e., by unique document-topic pairs; or 2) by workers.

Partitioning by HIT (PbH) ensures that no topic-document pair in training data also appears in testing. However, this allows the same workers to appear in both training and testing data if they worked on multiple HITs. This is potentially problematic in that the MTurk workforce is quite transitory in practice, and requesters typically face a *cold start problem* in which they have no prior work history for the bulk of workers completing a given task.

In contrast, *partitioning by workers* (PbW) ensures that no worker seen in the training data also appears in testing. As such, this represents a potentially more realistic scenario, as well as a potentially harder prediction task which necessitates generalization across workers. Note that supervised approaches that rely on training data to estimate worker reliability by past performance are thus inapplicable because we observe different workers in testing phase. In contrast, our behavioral models do not depend on worker-specific history and hence do not suffer from the same restriction.

In comparison to prior work, Rzeszotarski and Kittur (2011) report results for the easier PbH scenario. Kazai and Zitouni (2016) consider a different task (predicting whether a worker is expert or not, based on the worker's full history), so their task formulation is not comparable.

While we do not down-sample data to simulate a strict single labeling scenario (i.e., where each example is annotated only once), note that baselines and proposed methods evaluated here can be used in a single labeling scenario in which

classic label aggregation methods (Dawid and Skene 1979; Snow et al. 2008) could not.

4.4 Results

Table 2 presents classification and regression results. When we partition by HIT (PbH), the hybrid model achieves best classification accuracy (70.3%), followed by the non-behavioral SNB (69.6%) and behavioral RF-SF model (68.4%). Interestingly, the DT-AF baseline adapted from Rzeszotarski and Kittur (2011) performs worse than all other methods, including the constant baseline. When we partition by worker (PbW), the true strength of behavioral approach shines: even without any prior history for the workers in the test data, the behavioral approaches perform comparably to PbH partitioning, while the SNB baseline and hybrid RF-SF+SNB model can no longer be applied due to the aforementioned *cold-start problem*. With regard to the three proposed models, the sequential features model (RF-SF) performs best under both PbH and PbW settings.

For the regression task (i.e., predicting worker accuracy) with PbH partitioning, the best models perform roughly comparably: the SNB baseline, proposed RF-AF and RF-SF models, and the RF-SF+SNB hybrid. As with classification, we see behavioral models achieve roughly comparable performance in both PbW and PbH settings, while the supervised models requiring worker history cannot be applied. The DT-AF baseline continues to perform worst. Unlike classification, we see the RF-AF aggregate feature model slightly outperforms the RF-SF sequence feature model.

Overall, the RF-SF is typically the best performing behavioral model, while the kmeans-SC sequence clustering approach is always outperformed by the RF models.

5 Label Aggregation

In the previous section, we showed that behavioral data can be used to predict label correctness and worker accuracy, and that these models could be applied even in a single labeling scenario in which classic label aggregation methods (e.g., *majority voting* (MV), EM (Dawid and Skene 1979), or naive bayes (Snow et al. 2008)) could not. In this section, we now investigate use of these models for downstream tasks, such as label aggregation and label filtering.

While MV is the simplest strategy, since label quality often varies across workers and examples, it is prudent to weight votes by estimating each label's correctness. Whereas Dawid and Skene (1979) and Snow et al. (2008) estimate a confusion matrix for each worker, Demartini, Difallah, and Cudré-Mauroux (2012) instead use EM to estimate a single reliability parameter. Similarly, Whitehill et al. (2009) estimate a difficulty parameter for each example.

In this work, we propose to estimate parameters for weighted voting based on label correctness and worker quality predictions from behavioral models, instead of previous approaches based on peer-agreement or gold-agreement. We leave hybrid label aggregation using behavioral and non-behavioral evidence for future work.

Specifically, we pursue the following strategy. First, we use the behavioral models to predict: 1) the confidence of each label being correct; and 2) the accuracy of each worker (which can also be treated here as the confidence of the worker being correct). Secondly, we explore using each of these probabilities for weighted voting, weighting labels for a given example by the confidence that: 1) the label is correct; or 2) the worker providing the label is reliable. Thirdly, we measure a varying threshold for excluding low confident labels from voting. We begin by including all labels, then perform progressively stricter screening to exclude less confident labels. Whenever filtering becomes too strict for an example (such that all of its labels would be excluded), we fall back to the single label with highest confidence. This ensures full coverage for aggregation over all test examples.

5.1 Results

Section 4.4 findings showed that RF-SF was the most effective (pure) behavioral model, so we report its results (unreported RF-AF results were comparable). We compare to non-behavioral baselines: MV and EM (Dawid and Skene 1979)[2]. We use 5-fold cross validation for our experiments, partitioning by unique document-topic pairs. We build the behavioral models using the HITs for the document-topic pairs in the training data, and report label aggregation results on the document-topic pairs in the test set.

Figure 1 shows results. The left sub-figure reports the accuracy achieved by each approach across different threshold filtering levels. We see that weighted voting via behavioral models outperforms both MV and EM baselines. The label accuracy model achieves a maximum accuracy of 72.0% by filtering at a threshold of 10%, a comfortable gain over the MV (64.9%) and EM (66.7%) baselines. In regard to comparing the two behavioral models, the label accuracy model consistently performs best across filtering thresholds.

The right sub-figure shows the model-specific coverage of the behavioral models. For each filtering threshold, we see the fraction of topic-document examples for which labels can be aggregated without having to back-off to the single highest quality label. As expected, as the quality threshold becomes increasingly more demanding, fewer labels are predicted to meet the requirement. Coverage thus diminishes and back-off becomes increasingly needed.

One might expect that with higher thresholds, higher accuracy would result (since aggregation is performed using higher quality labels). It is somewhat surprising, therefore, that accuracy of the behavior models are seen to decrease with increasing quality threshold. This appears to be due to the aforementioned coverage issue, i.e., as the label is increasingly determined based on back-off to the single highest quality label, accuracy diminishes vs. weighted voting over all labels using behavioral weights.

[2] https://github.com/dallascard/dawid_skene

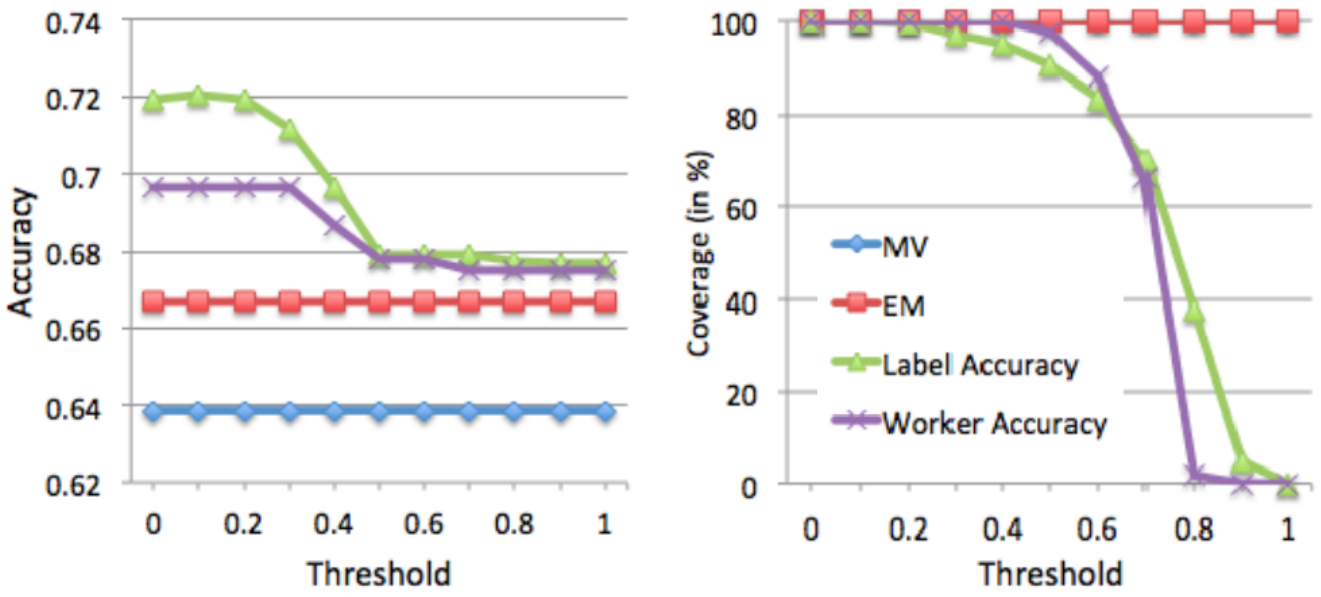

Figure 1: Label aggregation results for weighted voted via behavioral model confidence vs. classic MV/EM baselines. **Left**: accuracy achieved by the various algorithms at different threshold values. **Right:** coverage of the different algorithms.

6 Optimizing Label Collection

Prior work has investigated the relative benefit of collecting more labels for the same example and strategies for optimizing cost vs. aggregate label quality, e.g., for training a machine learning model (Sheng, Provost, and Ipeirotis 2008; Dai, Weld, and others 2011; Dai et al. 2013; Ipeirotis and Gabrilovich 2014). Similar to prior work, we model this data collection as a markov decision process (MDP), which determines how many labels to collect per example based on the quality and confidence of existing labels. Critically, the novelty of our MDP formulation lies in our use worker behavioral data to predict label accuracy, whereas prior work estimated this via expectation-maximization (EM) (Dawid and Skene 1979) or observed worker history.

Suppose a requester wishes to collect labels for a given example such that the probability that the crowd label agrees with the gold label is greater than k (i.e. the *target label quality*). We formulate this as follows.

1. **State Space**: $S = \{q \mid q \in [0,1]\}$, where q is the integrated or the overall quality of the aggregated label, i.e., the probability that the current aggregated label is equal to the gold label. For a single worker annotation, the integrated quality is the quality of that worker's annotation, obtained from the label quality prediction using behavioral traces, as outlined in Section 4. For more than one annotation, we use Dai et al. (2013)'s formulation:

$$q = Pr(\hat{y} = y) = \sum_{i=0}^{N} \binom{2N+1}{i} . p^{2N+1-i} . (1-p)^i$$

where i is the number of incorrect annotations. Here, the integrated crowd label ($\hat{y}$) corresponds to the majority label. Dai et al. (2013) proposed this for the scenario where all labelers have equal worker quality p. However, it is straightforward to extend the above formulation to cases where the worker accuracy scores differ, and we do so.

2. **Decision/Action Space**: At every stage, the model has to choose from amongst two decisions $A = \{\textit{get-new-label}, \textit{decide-true-label}\}$. The decision *get-new-label* signifies that the current quality of the integrated label is unsatisfactory and an additional label needs to be collected. In contrast, *decide-true-labels* means that the current label quality is satisfactory and the final label should be decided based on existing labels.

3. **Transition Matrix**: $T : S \times A \times S \rightarrow [0,1]$, which defines the probability of going from state s to state s' after taking a particular action a. Note that the label quality remains the same if the action *decide-true-label* is chosen. Hence, the corresponding transition matrix is diagonal. Transition values for the *get-new-label* action are conditioned on the quality of the labeler who performs the next annotation. This transition matrix is computed empirically from the data, sampling random decision sequences for all documents and filling the transition matrix by observing the change in integrated quality after adding each extra label.

4. **Reward Matrix**: $R : S \times A \rightarrow \mathbb{R}$ defines the reward function for each action in a given state. For the *get-new-label* action, the reward function is calculated as the summation of the expected increase in integrated quality and the cost c incurred due to additional labeling (c is set to -0.05, determined empirically). These values are determined using the transition matrix from the prior step.

$$\begin{aligned} R(q, get\_new\_label) &= E(q' - q) - cost\_labeling \\ &= \sum_{q} (q' - q) . P_{tran}(q, q') - c \end{aligned}$$

Table 3: Accuracy vs. cost achieved by MDP dynamic label collection vs. MV aggregation over fixed labeling.

	Accuracy	# Labels
Majority Voting	0.662	3984
Markov Decision Process	**0.674**	**3626**

For the *decide-true-label*, the reward is the difference between the current integrated quality and target quality (pre-specified parameter k). If the current quality is more than desired, the reward is set arbitrarily high.

Solving the above MDP provides us with an optimal action a for each state q in the state space. Thereafter, the MDP can be used to dynamically design a task in the following way: for each document-topic pair, we start by collecting a single label. We use the behavior models to compute its label quality estimate and use the above MDP formulation to decide on an action. We repeat this process iteratively until the algorithm outputs the *decide-true-label* action, or the number of collected labels for the given document-topic pair exhausts the number of available labels. At that point, we compute the aggregated label using majority vote (or return the highest quality annotation in case of a tie). We follow this process until we obtain the aggregated labels for all examples.

6.1 Results

We perform 10-fold cross validation on the behavioral data to obtain label quality estimates for all HITs. To test the proposed MDP formulation, we adopt the following approach. For each document-topic pair, we simulate a sequence of label annotations, obtained by randomly sampling from permutations of available crowd labels for that specific document-topic pair. Given such a sequence, we start with the initial label in the sequence and iteratively incorporate additional labels into the list of current labels until the MDP outputs the *decide-true-label* action (or the available labels are exhausted). We repeat this for all document-topic pairs and report the average agreement across 20 runs.

Table 4: Accuracy vs. labeling cost of MDP label collection vs. baseline MV fixed labeling for different label pluralities.

	Accuracy		Δ	**# Labels**		Δ
L	MV	MDP		MV	MDP	
2	0.665	**0.702**	5.56%	2248	**2189**	-2.62%
3	0.701	**0.709**	1.12%	1404	**1156**	-17.66%
4	**0.680**	0.645	-5.15%	376	**298**	-20.74%
5	0.750	**0.840**	12.00%	20	**12**	-40.00%

We use the MDP toolbox[3] to solve the MDP with the following parameters: target accuracy: 0.7, cost of labeling: 0.05, discount factor: 0.1.

Table 3 reports the accuracy vs. the cost (i.e., the number of collected labels) of the MDP approach vs. a baseline majority voting (MV) approach. The MDP achieves accuracy comparable to the MV baseline using 9% fewer labels.

[3] http://pymdptoolbox.readthedocs.io/en/latest/api/mdptoolbox.html

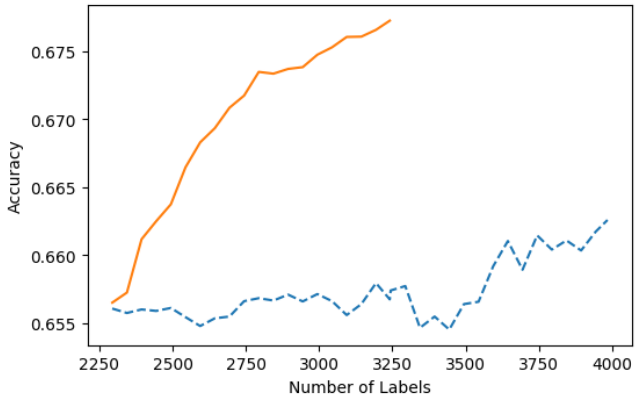

Figure 2: Aggregated label accuracy vs. the number of labels when intelligently selecting which example to label next via MDP (orange) vs. a simple round-robin visit order (blue).

Next, we further analyze the accuracy vs. labeling cost as a function of the number of available labels. For each value of label plurality $l \in [2{:}5]$, we identify all document-topic pairs that have at least l labels, randomly down-sampling to l for those having more labels. We then run the MDP with the same parameters on this down-sampled dataset and compare the accuracy vs. cost of MV over all down-sampled labels. We repeat this experiment 20 times and report the average results in **Table 4**. The MDP always collects fewer labels and almost always achieves greater accuracy.

Finally, we consider another use scenario. Thus far, we collected labels for each example until a satisfactory aggregated label was obtained. However, since the MDP provides aggregated label estimates for all document-topic examples, we might instead use the MDP to intelligently determine which example to label next (e.g., the document-topic pair with the lowest estimated quality). This is especially useful in low budget settings, where we cannot afford to collect multiple labels for every instance and want the MDP to intelligently allocate the budget across the examples.

We simulate this process by first randomly selecting one label for each example from the set of all earlier collected labels. Thereafter, we select the document-topic pair to label next which has the lowest estimated aggregated label quality. We iteratively repeat this process until we either achieve the desired quality target or exhaust the labeling budget. We compare this approach with a round-robin baseline (Sheng, Provost, and Ipeirotis 2008), where we iterate through topic-document examples in a fixed order in soliciting additional labels. We average results across 20 runs.

Figure 2 presents results. The x-axis indicates the number of labels collected and the y-axis denotes the corresponding aggregated label accuracy achieved. As we can see, the MDP results in a steadily increasing accuracy, which demonstrates the MDP's ability to effectively identify those examples in greatest need of additional labels, thus yielding the greatest benefit from additional label collection. In contrast, the ac-

curacy of the round-robin strategy tends to fluctuate though eventually trends upward. As the earlier Table 3 showed, the MDP also requires fewer annotations (compared to random example selection) in order to obtain a conclusive aggregate label, so the MDP plot terminates earlier than round-robin's.

7 Conclusion

Whereas peer-aggregation (Dawid and Skene 1979) and gold-checks (Snow et al. 2008) are the typical hallmarks of statistical quality assurance in crowdsourced labeling today, behavioral models offer an intriguing alternative approach which, once trained, can predict future work quality without requiring redundant labeling or further gold label creation.

We investigated the feasibility of using behavioral traces of crowd workers for two tasks: predicting correctness of labels (classification) and predicting the labeling accuracy of workers (regression). We proposed three prediction models that use only behavioral data, as well as a hybrid model which also uses non-behavioral data. We further described how these behavioral models could be used to improve label aggregation. Finally, we showed how behavioral modeling could drive a markov decision process (MDP) model for dynamic label collection in order to optimize aggregate label quality vs. labeling costs, e.g., using 9% fewer labels.

Results show that behavioral data can be effectively modeled to predict label correctness and workers' accuracies, further bolstering the limited prior work in this area (Rzeszotarski and Kittur 2011; Kazai and Zitouni 2016). In particular, results showed that behavioral models can be especially useful for the *cold start problem* in which individuals' prior work history is unavailable. Results also showed success for downstream label aggregation, label filtering, and optimized collection of labels via our behavior-driven MDP model.

Many directions for future work can be envisioned, such as developing more sophisticated hybrid models integrating behavioral and non-behavioral cues. We might train behavioral model via peer-agreement pseudo-gold. Future work (especially from industry) might exploit larger (proprietary) behavioral traces, and at this scale, likely benefit from neural modeling (Hochreiter and Schmidhuber 1997). Modeling longer sequential behaviors, e.g., across multiple HITs, would also be interesting, as would varied types of work on different platforms having diverse demographics.

Acknowledgments

We thank the many talented crowd members who contributed to our study, and the reviewers for their valuable feedback. This work was made possible by NPRP grant# NPRP 7-1313-1-245 from the Qatar National Research Fund (a member of Qatar Foundation), and supported in part by by National Science Foundation grant No. 1253413. Statements made herein are solely the responsibility of the authors.

References

Alonso, O. 2015. Practical lessons for gathering quality labels at scale. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1089–1092. ACM.

Banerjee, A., and Ghosh, J. 2001. Clickstream Clustering using Weighted Longest Common Subsequences. In *Proceedings of the web mining workshop at the 1st SIAM conference on data mining*, volume 143, 144.

Blum, A., and Mitchell, T. 1998. Combining labeled and unlabeled data with co-training. In *Proc. of the eleventh annual conference on Computational learning theory*, 92–100.

Breiman, L. 2001. Random Forests. *Machine learning* 45(1):5–32.

Cardona-Escobar, A.; Pineda-Iral, J.; Guarnizo-Cutiva, N.; and Jaramillo-Garzón, J. 2015. A methodology for the prediction of embryophyta protein functions using mismatch kernels. In *Signal Processing, Images and Computer Vision (STSIVA), 2015 20th Symposium on*, 1–7. IEEE.

Collins-Thompson, K.; Macdonald, C.; Bennett, P.; Diaz, F.; and Voorhees, E. M. 2015. TREC 2014 Web Track Overview. In *Proceedings of the Twenty-Third NIST Text REtrieval Conference (TREC)*.

Dai, P.; Lin, C. H.; Mausam; and Weld, D. S. 2013. POMDP-based Control of Workflows for Crowdsourcing. *Artif. Intell.* 202(1):52–85.

Dai, P.; Weld, D. S.; et al. 2011. Artificial Intelligence for Artificial Artificial Intelligence. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Dang, B.; Hutson, M.; and Lease, M. 2016. MmmTurkey: A Crowdsourcing Framework for Deploying Tasks and Recording Worker Behavior on Amazon Mechanical Turk. In *Proc. of the 4th AAAI Conference on Human Computation and Crowdsourcing (HCOMP): Works-in-Progress*. arXiv:1609.00945.

Dawid, A. P., and Skene, A. M. 1979. Maximum Likelihood Estimation of Observer Error-rates using the EM Algorithm. *Applied statistics* 20–28.

Demartini, G.; Difallah, D. E.; and Cudré-Mauroux, P. 2012. ZenCrowd: Leveraging Probabilistic Reasoning and Crowdsourcing Techniques for Large-scale Entity Linking. In *Proc. of the 21st World Wide Web Conference*, 469–478.

Drapeau, R.; Chilton, L. B.; Bragg, J.; and Weld, D. S. 2016. Microtalk: Using argumentation to improve crowdsourcing accuracy. In *Fourth AAAI Conference on Human Computation and Crowdsourcing*.

Hartigan, J. A. 1975. *Clustering Algorithms*. John Wiley & Sons, Inc.

Hochreiter, S., and Schmidhuber, J. 1997. LSTM Can Solve Hard Long Time Lag Problems. In *Advances in neural information processing systems*, 473–479.

Ipeirotis, P. G., and Gabrilovich, E. 2014. Quizz: targeted crowdsourcing with a billion (potential) users. In *Proceedings of the 23rd World Wide Web Conference*, 143–154.

Joachims, T. 2002. Optimizing Search Engines Using Clickthrough Data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 133–142. ACM.

Jung, H. J., and Lease, M. 2015. Modeling Temporal Crowd Work Quality with Limited Supervision. In *Proceedings of the 3rd AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*, 83–91.

Kazai, G., and Zitouni, I. 2016. Quality management in crowdsourcing using gold judges behavior. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, 267–276. ACM.

Kutlu, M.; McDonnell, T.; Barkallah, Y.; Elsayed, T.; and Lease, M. 2018. Crowd vs. Expert: What Can Relevance Judgment Rationales Teach Us About Assessor Disagreement? In *Proceedings of the 41st ACM SIGIR conference on Research and development in Information Retrieval*.

Liu, C., and Wang, Y.-M. 2012. TrueLabel+ Confusions: a Spectrum of Probabilistic Models in Analyzing Multiple Ratings. In *Proceedings of the 29th International Conference on on Machine Learning (ICML)*, 17–24.

Manam, C. V. K., and Quinn, A. J. 2018. WingIt: Efficient Refinement of Unclear Task Instructions. In *Proceedings of the Sixth AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*.

McDonnell, T.; Lease, M.; Kutlu, M.; and Elsayed, T. 2016. Why Is That Relevant? Collecting Annotator Rationales for Relevance Judgments. In *Proceedings of the 4th AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*, 139–148.

Paritosh, P. 2012. Human Computation Must Be Reproducible. In *Proceedings of the Workshop on Crowdsourcing Web Search (CrowdSearch), held at the 25th International World Wide Web (WWW) Conference*, 20–25.

Rzeszotarski, J. M., and Kittur, A. 2011. Instrumenting the Crowd: Using Implicit Behavioral Measures to Predict Task Performance. In *Proceedings of the 24th ACM symposium on User interface software and technology*, 13–22.

Sheng, V. S.; Provost, F.; and Ipeirotis, P. G. 2008. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 614–622.

Sheshadri, A., and Lease, M. 2013. SQUARE: A Benchmark for Research on Computing Crowd Consensus. In *Proceedings of the 1st AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*, 156–164.

Snow, R.; O'Connor, B.; Jurafsky, D.; and Ng, A. Y. 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*, 254–263. Assoc. for Computational Linguistics.

Wang, W., and Zaïane, O. R. 2002. Clustering Web Sessions by Sequence Alignment. In *13th International Workshop on Database and Expert Systems Applications*, 394–398.

Whitehill, J.; Wu, T.-f.; Bergsma, J.; Movellan, J. R.; and Ruvolo, P. L. 2009. Whose Vote Should Count More: Optimal Integration of Labels from Labelers of Unknown Expertise. In *NIPS*, 2035–2043.

Wu, M.-H., and Quinn, A. J. 2017. Confusing the crowd: Task instruction quality on amazon mechanical turk. In *Proceedings of the Fifth AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*, 206–215.