Flock: Hybrid Crowd-Machine Learning Classifiers

Justin Cheng and Michael S. Bernstein
Stanford University
{jcccf, msb}@cs.stanford.edu

ABSTRACT

We present hybrid crowd-machine learning classifiers: classification models that start with a written description of a learning goal, use the crowd to suggest predictive features and label data, and then weigh these features using machine learning to produce models that are accurate and use human-understandable features. These hybrid classifiers enable fast prototyping of machine learning models that can improve on both algorithm performance and human judgment, and accomplish tasks where automated feature extraction is not yet feasible. *Flock*, an interactive machine learning platform, instantiates this approach. To generate informative features, Flock asks the crowd to compare paired examples, an approach inspired by analogical encoding. The crowd's efforts can be focused on specific subsets of the input space where machine-extracted features are not predictive, or instead used to partition the input space and improve algorithm performance in subregions of the space. An evaluation on six prediction tasks, ranging from detecting deception to differentiating impressionist artists, demonstrated that aggregating crowd features improves upon both asking the crowd for a direct prediction and off-the-shelf machine learning features by over 10%. Further, hybrid systems that use both crowd-nominated and machine-extracted features can outperform those that use either in isolation.

Author Keywords

Crowdsourcing, interactive machine learning

ACM Classification Keywords

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

INTRODUCTION

Identifying predictive features is key to creating effective machine learning classifiers. Whether the task is link prediction or sentiment analysis, and no matter the underlying model, the "black art" of feature engineering plays a critical role in success [10]. Feature engineering is largely domain-specific, and users of machine learning systems spend untold hours experimenting. Often, the most predictive features only emerge after many iterations [36]. And though feature engineers may have deep domain expertise, they are only able to incorporate features that are extractable via code.

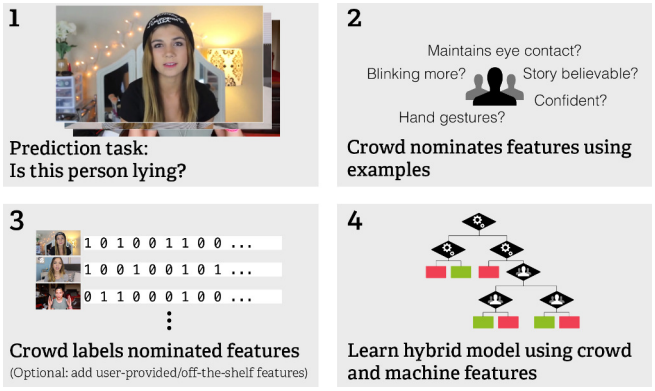

Figure 1. Flock is a hybrid crowd-machine learning platform that capitalizes on analogical encoding to guide crowds to nominate effective features, then uses machine learning techniques to aggregate their labels.

However, *embedding crowds inside of machine learning architectures* opens the door to hybrid learners that can explore feature spaces that are largely unreachable by automatic extraction, then train models that use human-understandable features (Figure 1). Doing so enables fast prototyping of classifiers that can exceed both machine and expert performance. In this paper, we demonstrate classifiers that identify people who are lying, perform quality assessment of Wikipedia articles, and differentiate impressionist artists who use similar styles. Previous work that bridges crowdsourcing and machine learning has focused on optimizing the crowd's efforts (e.g., [8, 21, 39]): we suggest that inverting the relationship and embedding crowd insight inside live classifiers enables machine learning to be deployed for new kinds of tasks.

We present *Flock*, an end-user machine learning platform that uses paid crowdsourcing to speed up the prototyping loop and augment the performance of machine learning systems. Flock contributes a model for creating hybrid classifiers that intelligently embed both crowd and machine features. The system allows users to rapidly author hybrid crowd-machine learners by structuring a feature nomination process using the crowd, aggregating the suggested features, then collecting labels on these new features. It loops and gathers more crowd features to improve performance on subsets of the space where the model is misclassifying many examples. For instance, given a decision tree that uses machine-readable features, Flock can dynamically grow subtrees from nodes that have high classification error, or even replace whole branches. In addition to

CSCW 2015, March 14–18, 2015, Vancouver, BC, Canada.

ACM 978-1-4503-2922-4/15/03 ...$15.00.
http://dx.doi.org/10.1145/2675133.2675214

improving performance, these constraints can help focus the crowd's brainstorming on a subset of the example space to generate more informative features.

Flock's success relies on crowds generating informative features. While crowds of people can excel at generating ideas [49, 22] and labeling subtle behavioral signals, they are generally poor at introspecting on the signals they use to make decisions [13], and even poorer at weighing evidence properly to make a decision [18]. In fact, there are many tasks where the crowd's predictions are significantly worse than even naïve algorithms — for instance, in identifying deceptive online reviews [31], or in categorizing businesses [46]. Nevertheless, crowds can identify useful attributes to classify images [25], suggesting that proper scaffolding of the process might lead to success. Comparing and contrasting examples highlights similarities and differences, and encourages deeper thinking [15]. To this end, we introduce an approach inspired by analogical encoding [14]: Flock asks crowd members to guess which of two examples is from a positive class and which is from the negative class, then write a reason why. These reasons become features: Flock automatically clusters the reasons, then recruits crowds to normalize each cluster and produce features. These features are then used by the crowd to annotate each example.

Rather than help people weigh evidence correctly, we recognize that this is a straightforward task for most learning algorithms. Thus, systems can harness the crowd's collective insight through repeated comparisons to identify potentially predictive features, including ones that require subtle human judgment, and then use statistical machinery to identify those that actually matter.

We demonstrate Flock's effectiveness through an evaluation of six broadly different prediction tasks, including discerning videos of people telling the truth or lying and differentiating between paintings by impressionist artists. We find that aggregating crowd features is more accurate than asking for a direct prediction from the crowd, and produce strong evidence that hybrid crowd-machine systems such as Flock can outperform systems that only use either. On these prediction tasks, these hybrid systems improve on both direct predictions and off-the-shelf classifiers by an average of over 10%.

Though crowdsourcing has generally either been portrayed as a stopgap solution for machine learning systems [43] or a goal for artificial intelligence to optimize [8], our work reinforces the benefits of a productive synthesis. Using crowds, classifiers can begin working minutes after a request, and gradually transition from a crowd-based to machine-based classifier. In this paper, we present a model for such a synthesis.

RELATED WORK

We start by reviewing systems that support the development of machine learning models. These include end-user systems designed to streamline the development process, as well as interactive machine learning systems. We then examine research at the intersection of crowdsourcing and machine learning. Prior work here has mainly focused on optimizing the monetary cost of crowdsourcing and on using crowds as part of the learning process, either through labeling or feature suggestion. This line of work suggests a trajectory toward integrating the entire machine learning pipeline with crowds, from feature generation to prediction.

Flock builds on these approaches by leveraging the strengths of both crowds and machines to learn hybrid models that can be iteratively improved. Extending the literature on integrating crowds and machines, Flock provides a generalized framework for augmenting machine learning models with crowd-nominated features. In particular, the crowd generates (and labels) features, doing so in ways that can integrate with machine learning algorithms to support weak areas of a machine-only classifier.

Supporting Machine Learning

Interactive machine learning systems can speed up model evaluation and helping users quickly discover classifier deficiencies. Some systems help users choose between multiple machine learning models (e.g., [17]) and tune model parameters, for instance through visualizing confusion matrices [44]. Others enable rapid iteration by making model re-training instantaneous (e.g., [40]), or allowing for fast switching between programming and evaluating example attributes [35].

In interactive machine learning systems, end-users train classifiers by providing focused input. These systems may focus on example generation (e.g., [19]), labeling (e.g., [11]), re-ranking (e.g., [12]), or even feature selection (e.g., [40]). Nevertheless, the available features are often built into the system, and the user's goal is to efficiently explore that feature space and label examples. By identifying and training new features to extend the feature space using systems such as Flock, these tools could be made even more effective.

Crowdsourcing and Machine Learning

Artificial intelligence techniques have largely been applied to optimize crowdsourcing efforts. Algorithms such as expectation maximization enable better estimates of worker quality and improved aggregation of responses (e.g., [8], [21], [47]). Other approaches construct behavioral models to predict future performance [39] or estimate task appropriateness [38].

For instance, crowds can be directly integrated into the learning process by focusing on specific questions or features posed by an algorithm (i.e. active learning). Even if the crowd lacks the expertise to answer the high level prediction task, they can effectively label features, or answer questions that help algorithms answer it (e.g., [32, 34]). In computer vision, supervised learning approaches have been applied to object classification by asking crowd workers questions to reduce label uncertainty [6], or by highlighting portions of an image corresponding to a given annotation [45]. Where these active learning systems seek to optimize algorithm performance within a pre-specified feature space, Flock tries to actively learn the feature space itself, by exploring and generating new features that can help when it is performing poorly.

Alternatively, crowdsourcing can augment AI by performing tasks that are difficult for machines to handle alone. Large numbers of labeled examples can be generated on-demand

(e.g., [9, 24]); the crowd can even be asked to find difficult examples that tend to be misclassified (e.g., [4]). Apart from labeling examples, crowds can group sets of examples (with categories later automatically inferred [16]). They can even directly identify and label potential features [25]. In both cases, these approaches have been shown to be more effective than directly asking the crowd for the desired categories. Flock demonstrates how to integrate both human features and machine features intelligently into a joint classifier.

Drawing on research in machine learning and crowdsourcing, Flock is an attempt at designing an end-to-end system that generates models consisting of both human-generated, as well as machine-generated features. It utilizes the crowd to define and populate the feature space, then generates a hybrid human-machine model. By identifying weak areas and then learning new features, these models can then iteratively improve themselves.

FLOCK

By leveraging the complementary strengths of the crowd in exploring features and machine learning algorithms in aggregating these features, we can prototype and develop hybrid human-machine learning models that improve on existing methods. Flock, a platform that allows users to develop these hybrid models starting from a written description of the problem, instantiates this approach.

The user workflow has three main components (Figure 2):

- uploading training and test data, along with class labels and any pre-computed machine features
- launching crowdsourcing tasks to nominate features through paired example comparison and suggestion aggregation, then collecting crowd labels for the new features
- training a hybrid crowd-machine classifier, looping to focus new crowd features on areas where performance is poor

Creating a hybrid classifier

Flock is aimed at end users who have a basic understanding of machine learning. Users begin a new project by defining their prediction task in words (Figure 2a). For example: "Is this Wikipedia article a good article (GA-grade), or bad article (C-grade)?" The user then uploads a file containing training and test examples, as well as their classification labels. Users can add their own machine-generated features (e.g., the number of editors) as columns in this file. Flock can automatically generate some additional features, including n-grams for text or those of a randomized PCA in RGB space for images.

The user triggers the learning procedure by choosing the classifier type: Flock currently supports decision trees, logistic regression and random forests. If the feature vector is high-dimensional, as with automatically-generated n-gram features, Flock recommends a linear model such as logistic regression. When the user is ready, Flock asks the crowd on the CrowdFlower microtasking market [1] to generate and aggregate features that can help classify their dataset. For example, suggestions such as "broken down into organized sections" and "thorough and well-organized" are aggregated into a feature that asks, "Is this article well-organized or structured?" Simultaneously, Flock collects gold standard (ground truth) feature labels for each of these possible features. Once the crowd has completed this task, Flock shares the nominated features with the user (Figure 2b). The user chooses which features they want to keep, and can add ideas of their own. Flock then launches crowdsourced feature labeling tasks where workers assign a value to each training and test example across the new features.

Flock now trains the hybrid classifier using available machine features and crowd features on the training data (Figure 2c), using cross-validation to prevent overfitting. If the model is a decision tree, internal nodes may be either machine features or crowd features. Flock shows the tree to the user and suggests improvements by highlighting any leaf nodes that are impure and misclassify more than a threshold (2.5%) of all training examples. If a random forest model is used, Flock trains many hybrid decision trees on random subsets of the input data, then produces the averaged "forest" classifier and highlights commonly impure nodes in the trees inside the forest. If a logistic regression model is used, Flock first trains a decision tree on crowd features only to partition the input data into more coherent subsets, then trains parallel logistic regression classifiers for each leaf nodes at the bottom of the tree using machine features and that partition of training data. Flock highlights any leaf classifiers that perform poorly as possible candidates for additional features in the partition. While Flock currently only supports binary classification, this approach can also be extended to support multi-class or regression problems.

The user can then improve the initial learned models by adding targeted crowd features to weak parts of the classifier. The user can choose any node, including the ones that Flock highlighted earlier as impure, and expand that part of the classifier with new crowd features. When this happens, Flock's process loops and the system launches a new set of crowd tasks to nominate new features on the subset of the data that belongs to the selected node.

At each step, Flock trains multiple models with different feature subsets. It does so to show the user the performance of multiple prediction methods on the test set. These models include *Crowd prediction* (a baseline asking workers directly to guess the correct label for the example), *ML with off-the-shelf* (the chosen machine learning model using only out-of-the-box features such as n-grams), *ML with crowd* (the chosen machine learning model using only crowd features), or *Hybrid* (the full Flock model using both machine and crowd features). This also allows end-users to decide whether the performance improvement of a hybrid model is worth the additional time and cost associated with crowdsourcing.

To enable this workflow in Flock, we must (1) generate high-quality features, (2) gather feature labels for those features, and (3) train hybrid machine-crowd models. Next, we expand upon Flock's approach to solving each problem.

Nominating features with analogical encoding

Crowdsourcing can help discover and test a large number of potential features: more eyes on the prediction task can min-

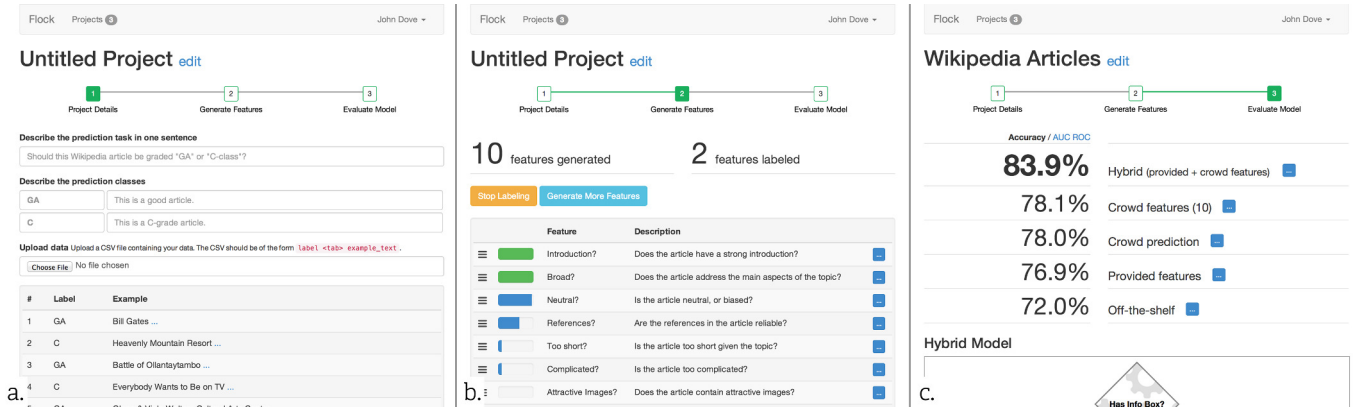

Figure 2. Workflow: (a) Users define the prediction task and provide example data. (b) Flock then automatically gathers features, combining them with machine-provided features if the user provided them. (c) Flock then allows users to visualize the features and results and compare the performance of the different models (e.g., human guessing, human features, machine features, a hybrid model).

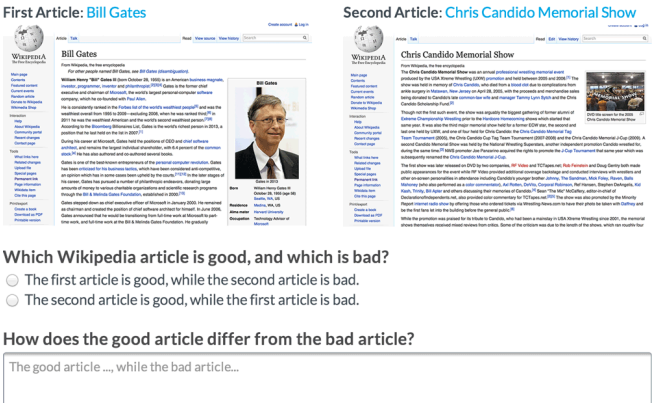

Figure 3. Crowd workers generate feature nominations in Flock by comparing two examples, predicting which example is in each class, and explaining how it is possible to tell them apart.

imize functional fixedness and surface counterintuitive features. Prior work suggests that the crowd may be able to generate features that can differentiate classes of input: previously, crowds have labeled training data for learning algorithms [24, 43], discovered classes of input that a classifier has never seen and tends to misclassify [4], brainstormed in creative settings (e.g., designing chairs [49]), extracted high-level concepts from unstructured data [3, 7, 23], and suggested features to classify images [25]. Querying the crowd also has the added benefit of nominated features being easily interpretable (e.g., "Does this painting tend to hide peoples' faces in shadow?"), in contrast to automatic but opaque features by computer vision algorithms such as SIFT [30].

In principle, any task that asks the crowd for predictive features could suffice here. Unfortunately, asking the crowd to generate features directly from examples often elicits poor responses — crowds are likely to be unfamiliar with the problem domain and generate surface-level features. For example, when presented with the task of deciding whether a joke will have popular appeal, the crowd gravitated towards responses such as "it makes me laugh" or "it depends on the person".

Previous research on analogical encoding has found that when considering single examples, people tend to focus on surface-level details, but when they compare examples, they focus on deeper structural characteristics [14]. So, instead of asking the crowd to *deduce* relevant features from a task description, Flock asks them to *induce* features from contrasting examples. Crowds can extract schemas from examples [48], and these suggestions (e.g., factors to consider when buying a camera) tend to exhibit long-tail distributions — while there is high overlap for a few dimensions, individual workers also suggest a large number of alternatives [23]. We hypothesize that eliciting features through contrasting examples will allow crowds to produce predictive features even when they have minimal domain expertise.

Flock first gathers feature suggestions from the crowd by generating crowdsourcing tasks on CrowdFlower. These tasks, which are automatically generated based the example type (text, image, or web page), show workers a random input from the set of positive training examples, and another random input from the set of negative training examples (Figure 3). However, the class labels are hidden from the worker. Workers are first asked to guess which of the two examples belongs to each class (e.g. matching paintings to Impressionist artists with similar styles). They are then asked to describe in free text how these examples differ. Typically, each explanation focuses on a single dimension along which the two examples differ. In early iterations, we showed the label to the workers and asked them for a reason directly, but hiding the label led them to engage in deeper reasoning about the differences between the examples [20]. Flock launches 100 comparison tasks with three workers each per dataset, resulting in 300 nominated features for about six cents per label ($20 total).

Next, Flock must aggregate this large number of free-text suggestions into a small set of potential features to show the user. The features have considerable overlap, and manual categorization suggested that the 300 nominations typically clustered into roughly 50 features. So, Flock clusters the suggestions and asks the crowd to generate an exemplar feature for each cluster (Table 3). First, the system splits each response into multiple suggestions (e.g. by sentence, newlines, and conjunctions such as "and"). The system then performs k-means clustering ($k = 50$) using tf-idf weighted bigram text

features. With these clusters in hand, Flock launches another task to CrowdFlower showing workers one cluster at a time and asking them to summarize each into a single representative question (or feature) that has a yes or no answer (e.g., "Is this person shifting their eyes a lot?"). Three candidate features are generated for each cluster.

A subsequent task then asks crowd workers to vote on the best of these three features, and label five examples using that feature in order to bootstrap the gold standard questions [28] for Flock's subsequent feature-labeling tasks. Question rewriting costs 5 cents per question, and each vote and five example labels costs 4 cents, for a total of about $15 to generate 50 features. CrowdFlower's adaptive majority voting algorithm (similar to Get Another Label [41]) aggregates the votes to decide on the correct feature label.

Gathering feature labels

After the crowd-generated features are returned, a user can edit them, filter them to remove ones they don't like or reduce cost, and add any new features that occur to them. They then launch the feature labeling process, which crowdsources labels for each of these user-validated features.

Flock spawns a CrowdFlower task for each feature to gather labels for that feature on the training and test examples. Each task shows workers one example input and asks them to pick a label for each feature (e.g., does this painting contain flowers?). Three workers vote on each feature label, for one cent each, and CrowdFlower's majority voting decides on the correct label.

Hybrid machine-crowd learners

When the feature labels are complete, Flock is ready to train a hybrid classifier (in addition to training other models such as *ML with off-the-shelf* so that they can be compared using the Flock user interface). The initial hybrid model begins by training using *k*-fold cross-validation on any pre-engineered machine features as well as any crowd-generated features. The exact form of hybridization depends on the user's choice of machine learning algorithm. Decision trees have access to both crowd and machine features, so nodes on the tree may be either a crowd feature or a machine feature. Random forests work similarly, aggregating many decision trees built on samples from the training data. A simple approach to hybridization with logistic regression would be to extend the crowd's features with the feature vector of machine features. However, we may be able to do better by using crowd features to partition the space so that previously ineffective machine features are now effective in one or more subregions of the space. So, the hybridized logistic regression first trains a decision tree at the top level using crowd features, then trains separate logistic regression classifiers as leaf nodes using only machine features. This approach tends to perform at least as well as simple logistic regression. Other linear classifiers (e.g., SVMs) could be added to Flock similarly.

Flock also enables the crowd to selectively supplement weak regions of the classifier. For example, crowds can be used to dynamically extend decision trees at nodes with high impurity, where the decision tree fails to correctly classify its examples. Flock can then ask the crowd to generate new features for examples at that node, providing a constrained space for brainstorming as these examples are already similar in some aspects. These features can then be used to grow a new subtree at that node, improving overall classification accuracy.

To do this, the model-building loops automatically as long as there exist impure nodes that misclassify large numbers of training examples. By default, one decision tree leaf node misclassifying 2.5% of all training examples triggers another round of targeted crowd feature generation. With random forests, we average the number of misclassified examples for each leaf node across all trees in the forest and use a similar filter. With logistic regression, Flock again triggers on leaf-node logistic regression models within the tree that are performing poorly. This process loops until either there exist no more impure nodes, or adding additional features does not improve the previous impure node.

This process can also be iterative and driven by user input. With automatic improvement turned off, Flock's interface highlights impure nodes that the user might consider extending with additional crowd features (Figure 4). When the user chooses a part of the model to improve, the crowd loops again through the process to perform a more targeted brainstorm on just training examples in the poorly-performing region.

Flock then allows any decision tree to add the new features as children of the selected subtree or node. (In the case of hybridized logistic regression, the new crowd features further partition the space and Flock trains new classifiers within the selected region.) Doing so ensures that the extra cost of generating these labels is constrained to inputs that need the help. For example, a user may start by identifying a few extractable features to differentiate good Wikipedia articles from mediocre ones (e.g. the page has missing citations, short sections, or no section headers). While a decision tree built on this algorithm classifies many of the examples well, it performs poorly on examples where none of the visual or structural red flags are triggered. Flock then provides only these difficult examples to the crowd, focusing the crowd's feature generation on a subset of pages where the heading structure is fine and the pages are of reasonable length to generate features like whether the article's introduction is strong.

EVALUATION

Flock's main thesis is that crowds can rapidly produce high-quality classifiers in tandem with machine learning techniques. In this section, we test this thesis by comparing Flock's predictions to those of machines across prediction tasks that require domain expertise, predict popularity and detect deceit. In particular, how effective are crowds at feature nomination? Does using crowd-nominated features perform better than other approaches such as direct prediction by the crowd? And finally, can crowd and machine features be combined to create effective hybrid crowd-machine learners?

Method

To understand the effectiveness of human-generated features and features generated by hybrid human-machine systems,

Dataset	# Examples	Purpose	Accuracy (ROC AUC)				
			Crowd Prediction	ML w/ off-the-shelf	ML w/ engineered	ML w/ crowd	Hybrid
Paintings	200	Predict expertise	0.64 (0.64)	0.69 (0.78)	-	0.74 (0.74)	**0.77 (0.83)**
Fake Reviews	200	Identify deception	0.65 (0.65)	0.87 (0.93)	-	0.72 (0.73)	**0.92 (0.96)**
Wikipedia	400	Evaluate quality	0.78 (0.78)	0.72 (0.72)	0.77 (0.77)	0.78 (0.78)	**0.84 (0.84)**
Jokes	200	Estimate popularity	0.58 (0.58)	0.56 (0.59)	-	0.63 (0.64)	**0.65 (0.64)**
StackExchange	200	Predict behavior	0.60 (0.60)	0.57 (0.57)	0.62 (0.62)	0.65 (0.65)	**0.71 (0.71)**
Lying	200	Identify deception	0.53 (0.53)	-	-	**0.61 (0.61)**	-

Table 1. A performance comparison of human and machine-learning-based approaches to different prediction tasks. Aggregating crowd-nominated features using machine learning consistently outperforms the crowd's direct predictions, and hybrid crowd-machine systems perform better than either crowd-only or machine-only models (unless either model was poor to begin with).

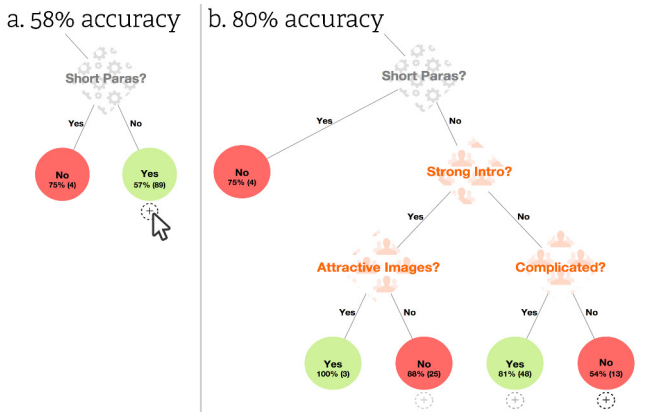

Figure 4. Users can augment weaker parts of the tree by requesting additional human features at specific nodes (a, highlighted). Flock will then automatically generate a new subtree based on these new features (b, highlighted).

we evaluated Flock's crowd feature generation against direct crowd classifications, off-the-shelf machine learning features, engineered features and a hybrid model that includes both crowd features and machine features (for a total of five conditions). For breadth, similar to related work (e.g. [7]), we used six different problem domains.

The problem domains encompass a large range of possible prediction tasks:

- PAINTINGS — differentiating between impressionist paintings by Claude Monet and Alfred Sisley. These painters are often confused due to similar styles. This task represents a learning goal that typically requires domain expertise.
- REVIEWS — telling honest from deceptive hotel reviews [31]. The goal is to identify subtle textual signals of honesty or deception.
- WIKIPEDIA — telling "good" and "C-grade" Wikipedia articles apart. This task requires domain expertise tantamount to that of an experienced Wikipedia editor.
- JOKES — identifying jokes that will have popular appeal. This task was chosen for its similarity to information cascade and popularity tasks in networks.
- STACKEXCHANGE — guessing which answer a question asker would pick as "best".
- LYING — discerning videos of people telling the truth or lying. Here, we try to identify subtle behavioral cues.

In each instance, we generated balanced datasets of at least 200 examples (e.g. for paintings, half were by Monet, and half by Sisley). In other words, random guessing would result in 50% accuracy. For PAINTINGS, images of Monet's and Sisley's paintings were obtained from claudemonet-gallery.org and alfredsisley.org respectively. REVIEWS consisted of a publicly available dataset of truthful and deceptive hotel reviews [31]. For WIKIPEDIA, articles were randomly sampled from a list of all Good Articles and all C-grade articles. JOKES were sampled from Reddit's /r/AskReddit joke threads — jokes with popular appeal were defined as those where the proportion of up-votes was in the upper quartile of the distribution, and those without had a proportion of up-votes in the lower quartile. All jokes received at least ten votes. Questions and answers for STACKEXCHANGE were randomly sampled from the set of all questions with at least two answers and a selected "best" answer on StackExchange's English Language & Usage community. Finally, for LYING, we sampled claims from YouTube videos of people playing "two truths and one lie tag", a game in which they tell truths and lies about themselves and the viewer must guess which is which.

In one baseline condition, *Crowd Prediction*, the crowd is directly asked the prediction question. Three independent workers were directly asked the prediction problem for each example, with CrowdFlower's adaptive majority voting algorithm deciding on the eventual class label. These direct crowd predictions appear to be robust; for example, prediction performance for REVIEWS is slightly better than that described in previous work [31]. One machine learning condition, *ML with Off-the-shelf*, used features that did not require any programming. These consisted of logistic regression on text bigrams for the problems with text content (REVIEWS, WIKIPEDIA, JOKES, and STACKEXCHANGE), and using a randomized PCA model on RGB space for the image prediction task (PAINTINGS). A third condition, *ML with Engineered*, focused on features that a domain expert might extract. For WIKIPEDIA and STACKEXCHANGE, nine features were extracted by the authors, and random forests used for the prediction. For WIKIPEDIA, these features include the structural features extracted in previous work (e.g., infoboxes) [5]. For STACKEXCHANGE, these features are similar to those that were most predictive in a study of a question's long-lasting value (e.g., number of comments, user reputation) [2].

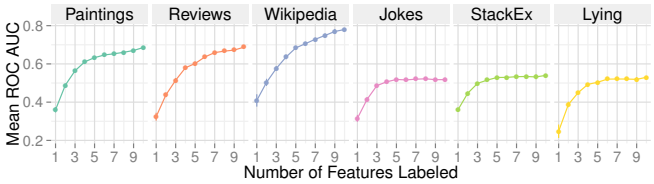

Figure 5. Mean classifier performance with random subsets of features gathered for each prediction task. As more features are collected, the overall performance of the classifier generally improves. Nevertheless, noisy features can impact performance.

In the *ML with Crowd* condition, Flock gathered and aggregated suggestions to generate crowd-nominated features. A final prediction was made by using machine learning to learn the best combination of these features. From the set of nominated features, ten features were chosen at random for labeling. Three different workers labeled each feature for each example, with CrowdFlower's adaptive majority voting algorithm determining the final feature value. Random forests ($n = 100$) were used to aggregate these nominated features, and backward feature selection performed afterward.

A final condition, *Hybrid*, combines crowd-nominated features (the same features as in the *ML with Crowd* condition) and machine features (user-engineered or off-the-shelf such as n-grams). By default, a decision tree model was used, but when the total number of features was large ($\geq$40), we tested both simple logistic regression, which combines both crowd-nominated and machine features into a single feature vector, and hybridized logistic regression, which trains a decision tree at the top level using crowd features, then trains separate logistic regression classifiers as leaf nodes using only machine features.

In all conditions, performance was evaluated using leave-one-out cross-validation.

Results

For each domain on average, 51 workers generated 334 feature nominations ($35 for nomination and aggregation in total), and 263 workers generated 1200 labels for each of 10 features during feature labeling ($12 per feature). Similar to prior work, we find that these feature nominations followed a long-tail distribution. A summary of results can be found in Table 1. While people performed essentially at chance in trying to detect lies on YouTube, *ML with Crowd* was able to detect lying 61% of the time. For other tasks, results were even stronger: for example, *Hybrid* was 92% accurate at detecting fake reviews, 84% accurate and judging Wikipedia page quality, and 77% accurate at differentiating Monet paintings from Sisley.

Aggregating crowd-nominated features outperforms direct predictions from the crowd, off-the-shelf features, and engineered features. In all datasets, aggregating crowd-nominated features (*ML with Crowd*) consistently outperforms directly asking the prediction task (*Crowd Prediction*), with an average improvement of 6% on an absolute scale. People effectively had access to the same features in both cases, but a machine learning architecture was more effective at combining the evidence to make a decision. Further, we find that this approach can exceed the performance of off-the-shelf classifiers, or even models with engineered features.

Dataset	Top Correlated Features
Paintings	A Monet (i) contains flowers (0.31), (ii) lilies (0.24), (iii) is abstract (0.24), (iv) does not contain people (0.23), and (v) uses broad brushstrokes (0.22).
Reviews	An honest review mentions (i) negative aspects of the stay (0.43), (ii) the prices (0.29), (iii) local activities (0.24), (iv) things only a guest would know about (0.20), and (v) getting something for free (0.17).
Wikipedia	A Good Article (GA) has (i) a strong introduction (0.52), (ii) attractive images (0.39), (iii) is well-structured (0.36), (iv) discusses the topic in detail (0.34), and (v) is not too short to sufficiently cover the topic (0.32).
Jokes	A popular joke (i) uses repetition (0.21), (ii) is long (0.15), (iii) is a "classic" joke (0.12), (iv) does not use toilet humor (0.10), and (v) is not offensive (0.09).
StackExchange	A selected answer (i) is well-written (0.20), (ii) has references (0.19), (iii) uses examples (0.18), (iv) makes logical sense (0.18), and (v) is detailed (0.16).
Lying	When lying, people (i) shift their eyes (0.14), (ii) keep sentences short (0.13), (iii) look confident (0.12), (iv) do not mention details (0.10), (v) and do not use gestures (0.09).

Table 2. For each dataset, the crowd-nominated features that were most correlated with the classification labels for each dataset (correlations in parentheses).

These nominated features were fairly informative and diverse: the mean correlation (Pearson's r) of nominated features with the actual labels was 0.15 (σ=0.09), suggesting that individual features are not strongly predictive but still carry some signal; a mean pairwise correlation between features of 0.19 (σ=0.12) suggests that most nominated features were not restatements of similar concepts. The median maximum correlation of any feature with the crowd predictions was 0.23, suggesting that crowds were not basing their decisions on any single feature. The only exception was WIKIPEDIA, where having a strong introduction was highly correlated (0.49).

Table 2 shows the most correlated features extracted by the crowd for each task. While some features appear subjective (e.g. abstractness for PAINTINGS or a strong introduction for WIKIPEDIA), they nonetheless carry important information. We coded these features to identify how many might be extractable through automated techniques, meaning that off-the-shelf classifiers exist (e.g., sentiment detection) or a simple algorithm can be written to do so (e.g., if an XPath selector could extract the number of headings in a web page). We find that a mean of 25% are easily extracted with machines, suggesting a straightforward approach to automating the collection of some of these features. Future work could study the effectiveness of using only machine-extractable features.

To get a sense of the number of features a user should gather, we plotted mean classifier performance on random subsets of features of different sizes. As Figure 5 shows, performance generally increases with the number of features, although substantial improvements tend to only be seen with four or more features gathered. Nevertheless, feature selec-

Suggestion cluster	Crowd-generated features
looks away from the camera several times · looks away from the camera · look away from the camera or use her hands · glances away from the camera slightly (+4 additional)	**Does she look away from the camera too much?** · Is there too much breaking of eye contact? · Why does she look away like shes trying to get someone elses opinion?
her eyes are shifty in the second video · Her eyes were too shifty in the second video · is shifty in the second video (+3 additional)	Do the eyes shift round causing suspicion of lying? · **Is this person shifting their eyes a lot?** · Is it unnatural?

Table 3. Crowds are able to generate features by summarizing clusters of suggestions into representative features (worded as yes/no questions), and then voting on them (feature with the most votes bolded).

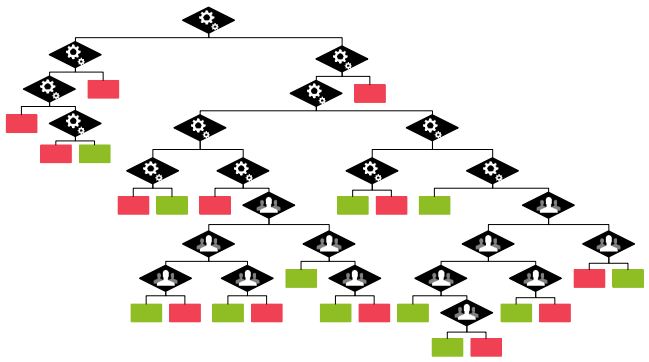

Figure 6. The decision tree used to differentiate "good" (green) and "C-grade" articles (red) on Wikipedia, where machine-extractable features (nodes with gears) were augmented with crowd-nominated features (nodes with people).

tion is important, especially for datasets where features may be noisier or uninformative, and using all nominated features can result in diminished performance (e.g., JOKES).

When comparing classifiers that use crowd-nominated features with those that use engineered features, we note that more time and effort could always have been spent to increase the number of available crowd-nominated, or engineered features. Nevertheless, prior work suggests that the engineered features we use are among the most predictive [5, 2], suggesting that these engineered features produce a classifier in the same performance neighborhood as one that would be more carefully optimized. Figure 5 also suggests that a relatively small number of crowd features is required to achieve reasonable performance as well.

In other words, Flock can help build classification systems that perform roughly as well or better than systems with engineered features with significantly less developer effort, and no programming involved. For example, Flock could be used to prototype and rapidly deploy classification systems that initially are entirely powered by the crowd, with machine-extracted features gradually replacing the crowd over time.

Crowds were able to generate predictive features, but were poor at estimating their features' predictiveness. The feature nomination pipeline generated an average of about 42 potential features (out of 334 unique suggestions) across the 100 examples. k-means clustering identified consistent clusters of suggestions, and crowds were able to extract representative features from these (see Table 3 for examples). Comparing the extracted features with a manually annotated set on the LYING dataset, we found that 80% of features suggested at least five times were identified by k-means and generated by the crowd. Accurately extracting rarer, but more informative features remains future work. Surprisingly, the crowd was also able to identify features that suggested specialized knowledge — for instance, correctly pointing out that the presence of flowers (or lilies) in a painting are indicative of a painting by Monet.

These crowd-nominated features can be gathered reasonably quickly — nomination tasks that asked for 3 independent feature suggestions on 100 examples were 90% complete in roughly nine hours, and finished entirely in 12.5 hours on average. Labeling these features is significantly quicker: the mean time to 90% completion was 1.5 hours, and labels for all 200 examples were complete in 2.8 hours on average (σ=1.6).

Our results showing that *ML with Crowd* outperforms *Crowd Prediction* suggests that crowds do not estimate the importance of the features they generate very well, underlining the importance of machine weighting. To better understand this, for each task, we asked ten workers to independently rank the importance of the ten crowd-nominated features, and compared this ranking with that generated by the absolute correlation of each feature with the provided labels. A Kendall's tau test reveals a coefficient of between -0.184 and 0.156 (a value of 1 corresponds to perfect agreement, 0 to independence in the rankings, and -1 to a completely reversed ranking). These results suggest that people's rankings of features are almost entirely uncorrelated with the true importance of these features with respect to the task. In fact, this coefficient was exactly zero in the case of STACKEXCHANGE. These ranking differences can be striking: for instance, repetitive structure was the strongest feature in JOKES but ranked on average as the least important feature. This experience reinforces the idea that Flock should balance popular suggestions with outlier but potentially very informative suggestions: a classic exploration vs. exploitation tradeoff.

Not only are workers individually poor at predicting feature importance, but the number of workers that suggest each feature is a weak indicator of relative importance. Here, τ ranges between -0.244 and 0.556. When τ is larger, semi-random exploration of the feature space weighted by the number of suggestions for each feature would reasonably discover important features; when τ is small, it would be a poor approach. These results demonstrate the strength of machine learning algorithms in revealing "surprising" features — features that people would not expect to be important.

Combining crowds and machines is better than using either in isolation. On average, hybrid systems that use both crowd-nominated and machine-extracted features improved upon the crowd-only approach by 7%, and on machine-only approaches by 8% (on an absolute scale). While crowd-nominated features are useful and potentially provide more information than machine-generated features, crowds are not always consistent in how they label features [6], and are thus less reliable than completely automatic approaches. By combining these two types of features, we can then take advan-

tage of the strengths of both. For example, a decision-tree approach improves overall accuracy in WIKIPEDIA and STACK-EXCHANGE by 7% and 9%, respectively, in comparison to using engineered features, and 6% in comparison to using crowd-nominated features. Figure 6 shows the tree generated by this approach.

The crowd helped most when a Wikipedia page does not have obvious visual or structural indications of needing additional work (e.g. missing citations or very short sections). In these cases, the decision tree often predicts incorrectly, but the crowd focusing on these examples nominates features related to the content of the article (e.g. level of detail) to help differentiate. If we examine the individual nodes where subtrees were added, we find the performance increases here are substantial — 12% in the case of WIKIPEDIA. For different parts of the tree, crowds nominated different features, and again, different features turn out to be important for each subtree. For example, covering a topic broadly and in detail was important when an article contained broken links, but article complexity and text relevance were important when it did not.

In cases when the total number of features is large (e.g., when using n-gram features), performance using either logistic regression or hybridized logistic regression was comparable: for REVIEWS, accuracy using the first approach is 0.92 (AUC=0.96) and 0.90 (AUC=0.90) using the second; for JOKES, the resulting accuracies were 0.65 (AUC=0.64) and 0.63 (AUC=0.64) respectively. Overall, improvements are substantial: 5% over crowd-nominated features for PAINTINGS and 5% over off-the-shelf features for REVIEWS, the next-best performing method for each.

Scale

It may not be realistic to pay workers to label several features for each item to be classified, especially as the number of inputs grows. We thus wanted to see whether we could learn individual classifiers for features that the crowd had nominated, using off-the-shelf features. Such an approach would enable Flock to scale to inputs of arbitrary size with no additional crowd input required, after gathering training data (or feature labels) for a small subset of examples. We expanded the WIKIPEDIA dataset to 1000 examples, instead of 400 previously. For each feature, we trained a bigram classifier on random subsets of 200 labeled examples, and predicted the feature value for the other 800 examples. Across all ten human-nominated features, we obtained a mean accuracy of 0.84 (mean AUC=0.61).

Using these predicted feature values, we again attempted the original prediction task. A model using only these predicted features had an accuracy of 0.74 (AUC=0.74), and a hybrid model that combined these predicted features with off-the-shelf features had an accuracy of 0.79 (AUC=0.79). Only the original *Hybrid* condition with crowd-provided feature values attained better performance. Future work could look at more expressive off-the-shelf classifiers to further improve performance; for some features (e.g., if an article contained reputable references), the bi-gram classifier performed at chance.

Limitations

While our results suggest that our approach generally improves on existing methods, there are several limitations to the system. Like machine-only classifiers, performance suffers when class membership is potentially inconsistent (i.e. in cases where extrinsic factors strongly influence class membership, e.g., JOKES, where a joke may only become popular because it was the first to be posted).

And in some cases such as REVIEWS, a straightforward application of machine learning approaches already performs reasonably well. Although crowds can improve on the performance of these systems (e.g. 92% accuracy using a hybrid system vs. 87% using only off-the-shelf features), the tradeoff between improved performance and cost of crowdsourcing additional features should be considered. In the opposite case, where the machine-only model is poor to begin with (e.g. JOKES), using just crowd features alone can lead to performance comparable to a hybrid model, underlining the importance of comparing these different approaches. In this case, text models were unable to extract useful signals of popular appeal, possibly because such models work best with longer documents and a large number of examples.

Specific to Flock is its dependence on whether features can be easily perceived and identified by a crowd worker. Even if the crowd is poor at guessing, Flock can still perform reasonably well if the crowd can identify differentiating characteristics (e.g. "Does the joke use repetition?"). But while the crowd can generate useful features, both objective and subjective, they sometimes make suggestions that are restatements of the problem ("This seemed more genuine" for REVIEWS), or are too subjective ("It made me laugh" for JOKES) to be similarly interpreted by different workers. The phrasing of the prediction task is important in avoiding these degenerate features. Also, if features are hard to come up with, the features that do matter may not end up being captured in the first place.

We note that gathering more features does not always result in improved performance (i.e. the curse of dimensionality). In all cases, feature selection methods are useful in identifying the features that truly matter.

Finally, while Flock is designed as a tool with significant automation behind it, a laboratory user study which compares Flock with other workflows would provide insight into how it is used in practice.

DISCUSSION

Through Flock, we can build classifiers that are both effective and interpretable. As opposed to off-the-shelf models that use thousands of features and require significant effort to understand, models generated by Flock use explainable crowd-nominated features. We see several promising future directions: in improving the feature nomination process, in considering monetary, developer and computation costs in combination, and in devising methods to better automate such crowd-machine systems. Further, as Flock's design is fairly modular, each component (e.g., feature nomination, feature aggregation, or learning algorithms) can be easily replaced as more effective approaches are found.

Improving feature nomination

To start, we consider possible improvements to Flock's feature nomination process. Currently, examples are paired at random, and the crowd has little guidance as to the type of features to produce. By instead providing similar examples to users, alignable differences between these examples can become more salient (e.g., if one wanted users to focus on article layout when judging Wikipedia article quality by presenting two articles with similar content) [15]. Further, curriculum learning suggests that providing progressively more difficult-to-categorize training examples for training may further improve the crowd's ability to identify subtler features [42].

Different interfaces may also lead to beneficial outcomes: an input-agreement game, where users are incentivized to describe examples using keywords to each other to see if they were presented the same example, may uncover other features [27]. In cases where a larger number of examples can be presented simultaneously (e.g., images), complementary agreement games would allow users to rapidly identify and categorize examples with common characteristics [26]. For highly specialized prediction tasks, automatically targeting groups of workers, possibly by demographic [29], may also improve the quality of nominated features (e.g., identifying art lovers to build a classifier for Monet paintings).

Alternatively, exploring other approaches to feature aggregation such as tagged hierarchies [7] may lead to higher quality features, but at higher cost. More sophisticated algorithms could automatically identify high-level candidate features by clustering on subsets of confused examples when it is possible to automatically generate a large number of low-level features [33]. Pairing examples near the classifier's decision boundary, where the examples are currently most confused, is also a potential approach to improving performance at subnodes of a tree.

One downside to hybridized crowd-machine learning systems is monetary cost, especially with larger datasets. In some cases it may not be financially feasible to permanently wire crowd feature labeling into a system. We see several ways forward: to use Flock to prototype machine feature-driven systems, to more intelligently utilize crowd features, or to learn and then replace these features altogether.

In its current form, Flock could be used as a prototyping tool. While features that the crowd nominates may not be easy to code, our goal is to instead provide inspiration for related machine-extractable features. This approach allows the user to prototype features and understand their effectiveness before sinking hours into writing code to extract them.

We may also consider a system that directly integrates direct crowd guesses and machine features, as collecting a single prediction for each example is significantly cheaper than gathering multiple features. Our initial experiments are promising: on average, the resulting classifier performs better than either guessing or a machine-only classifier, but worse than the complete hybrid model. On ARTISTS, a hybrid system combining the crowd's direct predictions and off-the-shelf features had an accuracy of 0.72 (AUC=0.77). Future hybrid systems could weigh a crowd's direct predictions more heavily amongst other nominated features.

Alternatively, decision-theoretic models could step in to optimize the balance between crowd and machine features [8, 21]. Decision trees could then be optimized for cost-effectiveness and performance by using crowd-nominated features deeper in the hierarchy. Multi-armed bandit strategies could help balance the tradeoff between exploration (gathering more features and labels for these features) and exploitation (improving the quality of specific features or optimizing the existing set of features). Using an active learning approach, one could model the informativeness of each crowd feature, and selectively query the crowd for specific features on specific examples when needed, but also have the flexibility to simply generate more features on-demand.

And while crowd-nominated features improve the performance of machine learning systems, a natural goal would be to eventually replace each crowd feature with its own machine learning classifier. Our analysis of the WIKIPEDIA dataset shows promise in trying to learn classifiers that predicts crowd responses [37], assuming that crowd behavior can be effectively encoded in the classifier's feature space. Another approach could be to have the crowd recursively break down features into simpler sub-features (e.g. "whether a painting uses warm colors" could potentially be broken down into multiple instances of "whether a painting contains a particular color"), until each is simple enough to be automatically extracted. Alternatively, the crowd might be able to author that classifier themselves with appropriate end-user machine learning tools (e.g., [11]).

CONCLUSION

In this paper, we integrate crowdsourcing into machine learning to develop hybrid learners with crowd-nominated features, machine-driven aggregation of these features, and targeted hybrids that use both crowd and machine features. We present Flock, a machine learning platform that allows end users to inspect existing models, launch targeted, crowd-driven feature generation tasks, and guide the feature aggregation and selection process. In an evaluation across domains such as quality evaluation and deception detection, these hybrid systems offer significantly improved performance. Hybrid crowd-machine learning systems may offer a route to rapid prototyping and exploration of the feature space, even in traditionally difficult domains. Flock suggests a future where an end user could create a machine learning system just by explaining the prediction goal into a free-form textbox. A combination of crowds and algorithms could take over from there to collect examples, extract features, then tune and monitor a classifier that becomes useful to the end user within hours.

ACKNOWLEDGMENTS

We would like to thank Noah Goodman, Walter Lasecki, and our reviewers for their invaluable input, as well as CrowdFlower for their support. This work was supported by NSF Grant 1351131 and a Stanford Graduate Fellowship.

REFERENCES

1. Crowdflower. `http://www.crowdflower.com`.
2. Anderson, A., Huttenlocher, D., Kleinberg, J., and Leskovec, J. Discovering value from community activity on focused question answering sites: a case study of stack overflow. In *Proc. KDD* (2012).
3. André, P., Kittur, A., and Dow, S. P. Crowd synthesis: Extracting categories and clusters from complex data. In *Proc. CSCW* (2014).
4. Attenberg, J., Ipeirotis, P. G., and Provost, F. J. Beat the machine: Challenging workers to find the unknown unknowns. In *Proc. AAAIW* (2011).
5. Blumenstock, J. E. Automatically assessing the quality of wikipedia articles. *UCB School of Information* (2008).
6. Branson, S., Wah, C., Schroff, F., Babenko, B., Welinder, P., Perona, P., and Belongie, S. Visual recognition with humans in the loop. In *Proc. ECCV*. 2010.
7. Chilton, L. B., Little, G., Edge, D., Weld, D. S., and Landay, J. A. Cascade: Crowdsourcing taxonomy creation. In *Proc. CHI* (2013).
8. Dai, P., Weld, D. S., et al. Decision-theoretic control of crowd-sourced workflows. In *Proc. AAAI* (2010).
9. Deng, J., Krause, J., and Fei-Fei, L. Fine-grained crowdsourcing for fine-grained recognition. In *Proc. CVPR* (2013).
10. Domingos, P. A few useful things to know about machine learning. *Communications of the ACM* (2012).
11. Fails, J. A., and Olsen Jr, D. R. Interactive machine learning. In *Proc. IUI* (2003).
12. Fogarty, J., Tan, D., Kapoor, A., and Winder, S. Cueflik: interactive concept learning in image search. In *Proc. CHI* (2008).
13. Gazzaniga, M. S., LeDoux, J. E., Gazzaniga, M., and LeDoux, J. *The integrated mind*. 1978.
14. Gentner, D., Loewenstein, J., and Thompson, L. Learning and transfer: A general role for analogical encoding. *Journal of Educational Psychology* (2003).
15. Gentner, D., and Markman, A. B. Structure mapping in analogy and similarity. *American Psychologist* (1997).
16. Gomes, R. G., Welinder, P., Krause, A., and Perona, P. Crowdclustering. In *Advances in Neural Information Processing Systems* (2011), 558–566.
17. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. The weka data mining software: an update. *ACM SIGKDD Explorations* (2009).
18. Hammond, K. R., Hursch, C. J., and Todd, F. J. Analyzing the components of clinical inference. *Psychological review* (1964).
19. Hartmann, B., Abdulla, L., Mittal, M., and Klemmer, S. R. Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In *Proc. CHI* (2007).
20. Kahneman, D. A perspective on judgment and choice: mapping bounded rationality. *American psychologist* (2003).
21. Kamar, E., Kapoor, A., and Horvitz, E. Lifelong learning for acquiring the wisdom of the crowd. In *Proc. JCAI* (2013).
22. Kim, J., Cheng, J., and Bernstein, M. S. Ensemble: Exploring complementary strengths of leaders and crowds in creative collaboration. In *Proc. CSCW* (2014).
23. Kittur, A., Peters, A. M., Diriye, A., and Bove, M. Standing on the schemas of giants: socially augmented information foraging. In *Proc. CSCW* (2014).
24. Lasecki, W. S., Song, Y. C., Kautz, H., and Bigham, J. P. Real-time crowd labeling for deployable activity recognition. In *Proc. CSCW* (2013).
25. Law, E. *Attribute Learning using Joint Human and Machine Computation*. PhD thesis, 2012.
26. Law, E., Settles, B., Snook, A., Surana, H., Von Ahn, L., and Mitchell, T. Human computation for attribute and attribute value acquisition. In *FPGV* (2011).
27. Law, E., and Von Ahn, L. Input-agreement: a new mechanism for collecting data using human computation games. In *Proc. CHI*, ACM (2009).
28. Le, J., Edmonds, A., Hester, V., and Biewald, L. Ensuring quality in crowdsourced search relevance evaluation: The effects of training question distribution. In *Proc. SIGIR* (2010).
29. Li, H., Zhao, B., and Fuxman, A. The wisdom of minority: Discovering and targeting the right group of workers for crowdsourcing. In *Proc. WWW* (2014).
30. Lowe, D. G. Object recognition from local scale-invariant features. In *ICCV* (1999).
31. Ott, M., Choi, Y., Cardie, C., and Hancock, J. T. Finding deceptive opinion spam by any stretch of the imagination. In *Proc. ACL* (2011).
32. Parameswaran, A., Sarma, A. D., Garcia-Molina, H., Polyzotis, N., and Widom, J. Human-assisted graph search: it's okay to ask questions. *Proc. VLDB* (2011).
33. Parikh, D., and Grauman, K. Interactively building a discriminative vocabulary of nameable attributes. In *CVPR* (2011).
34. Parkash, A., and Parikh, D. Attributes for classifier feedback. In *Proc. ECCV*. 2012.
35. Patel, K., Bancroft, N., Drucker, S. M., Fogarty, J., Ko, A. J., and Landay, J. Gestalt: integrated support for implementation and analysis in machine learning. In *Proc. UIST* (2010).

36. Patel, K., Fogarty, J., Landay, J. A., and Harrison, B. Investigating statistical machine learning as a tool for software development. In *Proc. CHI* (2008).

37. Raykar, V. C., Yu, S., Zhao, L. H., Valadez, G. H., Florin, C., Bogoni, L., and Moy, L. Learning from crowds. *JMLR* (2010).

38. Rzeszotarski, J., and Kittur, A. Crowdscape: interactively visualizing user behavior and output. In *Proc. UIST* (2012).

39. Rzeszotarski, J. M., and Kittur, A. Instrumenting the crowd: using implicit behavioral measures to predict task performance. In *Proc. UIST* (2011).

40. Settles, B. Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In *Proc. EMNLP* (2011).

41. Sheng, V. S., Provost, F., and Ipeirotis, P. G. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proc. KDD* (2008).

42. Skinner, B. F. Reinforcement today. *American Psychologist* (1958).

43. Snow, R., O'Connor, B., Jurafsky, D., and Ng, A. Y. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proc. EMNLP* (2008).

44. Talbot, J., Lee, B., Kapoor, A., and Tan, D. S. Ensemblematrix: interactive visualization to support machine learning with multiple classifiers. In *Proc. CHI* (2009).

45. Vijayanarasimhan, S., and Grauman, K. Large-scale live active learning: Training object detectors with crawled data and crowds. In *CVPR* (2011).

46. Wais, P., Lingamneni, S., Cook, D., Fennell, J., Goldenberg, B., Lubarov, D., Marin, D., and Simons, H. Towards building a high-quality workforce with mechanical turk. In *Proc. NIPSCSS* (2010).

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

48. Yu, L., Kittur, A., and Kraut, R. E. Distributed analogical idea generation: Inventing with crowds. In *Proc. CHI* (2014).

49. Yu, L., and Nickerson, J. V. Cooks or cobblers?: crowd creativity through combination. In *Proc. CHI* (2011).