A Hybrid Approach to Identifying Unknown Unknowns of Predictive Models

Colin Vandenhof
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Canada
cm5vande@uwaterloo.ca

Abstract

When predictive models are deployed in the real world, the confidence of a given prediction is often used as a signal of how much it should be trusted. It is therefore critical to identify instances for which the model is highly confident yet incorrect, i.e. the unknown unknowns. We describe a hybrid approach to identifying unknown unknowns that combines the previous crowdsourcing and algorithmic strategies, and addresses some of their weaknesses. In particular, we propose learning a set of interpretable decision rules to approximate how the model makes high confidence predictions. We devise a crowdsourcing task in which workers are presented with a rule, and challenged to generate an instance that "contradicts" it. A bandit algorithm is used to select the most promising rules to present to workers. Our method is evaluated by conducting a user study on Amazon Mechanical Turk. Experimental results on three datasets indicate that our approach discovers unknown unknowns more efficiently than the state-of-the-art.

Introduction

Predictive models are increasingly being deployed in the real world, with applications ranging from image annotation to autonomous driving. With more sophisticated learning algorithms, improvements to computational resources, and access to larger data sets, the accuracy of these models has increased. However, they can still make mistakes. As more reliance is placed on such systems, it becomes increasingly important to characterize its errors.

Confidence scores have traditionally been used to indicate the degree of certainty of a model in its prediction. This signal can be misleading when the model makes a high confidence prediction that is incorrect. Such instances have been termed the unknown unknowns (UUs). Attenberg et al. (2011; 2015) observed that UUs are often present as systematic errors in particular regions of the feature space, so-called blind spots of the model. Blind spots generally arise due to discrepancies between the distribution of training data and the target distribution, a problem known as dataset shift (Ben-David et al. 2006). These discrepancies can occur for various reasons: there may be systematic biases in the training data, the target distribution may shift over time, or a model may be applied to new domains (i.e. transfer learning).

UUs are especially important to identify in high-stakes settings like healthcare and law. Decision makers may use the model's confidence as a basis for how much to trust a given prediction; hence, any incorrect predictions that are made with high confidence can have serious consequences. There are many other cases in which identifying the UUs of a model is valuable. In adversarial settings, the identification of UUs may be used to highlight system vulnerabilities and prevent attacks. A classic example is spam filtering, in which an adversary tries to generate spam messages that will be misclassified with high confidence by the spam filter, and successfully pass into the recipient's main inbox. To defend against such attacks, UUs must be identified preemptively so that corrections can be made to the model.

Two general approaches currently exist for identifying UUs. The first is a crowdsourcing approach, in which candidates are proposed by workers. The second is an algorithmic approach, in which candidates are automatically selected from a fixed set of test instances. This paper introduces a hybrid method that combines both approaches and addresses some of their weaknesses. A key aspect of our method is learning a set of interpretable rules that explain how the classifier makes high confidence predictions. Then, we sequentially select a rule, choose a candidate instance from the test set that is covered by that rule, and present it to a crowd worker to label. If it is not already a UU, then the worker is challenged to modify the instance and turn it into a "contradictory" example - an instance that is still covered by the rule, yet whose ground truth label is now different. To decide which rule to next present to a worker, we devise a Bayesian bandit algorithm, in which the rules are the arms of the bandit, and rules that are more likely to yield UUs are queried more often.

We evaluate our approach on three datasets on Amazon Mechanical Turk. Results indicate that our hybrid approach substantially outperforms the purely algorithmic approach. Moreover, while the crowdsourcing task is challenging, it does not overburden workers in terms of time or workload.

Workers employ many creative and surprising techniques for generating contradictory examples, and the generated UUs help to characterize blind spots of the model.

Related Work

The two broad class of methods for identifying UUs—crowdsourcing and algorithms—differ in how candidates are chosen: proposed by workers versus selected from a fixed set of test instances.

In the crowdsourcing approach, labelled candidates are proposed by workers, and verified as UUs by obtaining the model's prediction and confidence score. Attenberg et al. (2011; 2015) present such a crowdsourcing system, which they call "Beat the Machine" . They examine a classifier for hate-speech in web pages. Workers receive a small base payment for submitting any valid URL, but if associated webpage is a hate-speech page that has been incorrectly classified as benign, they receive a bonus payment that is proportional to the confidence of the model. The authors find that this scheme incentivizes workers to discover UUs.

There are some limitations to this approach. First, the model is presented as a black-box to the workers. They are given no information about the model's inner workings, so it is naturally difficult to infer how to "beat" it. To infer anything about its behavior, they must submit many examples and learn through trial-and-error. This approach can be very costly in the crowdsourcing setting since workers are paid on a per-submission basis. Furthermore, it may be infeasible for humans to build an accurate mental model if the classifier is complex. In this work, we propose "opening" the black box. We learn an interpretable surrogate model of how high confidence predictions are made, and then present it to workers.

While "Beat the Machine" relies on workers finding preexisting examples from the internet, we recognize that in many domains, it is impractical or impossible to find relevant examples via internet search. As an alternative, we propose a framework in which workers are provided with a template instance that they can modify to produce a new candidate.

The second avenue of research into identifying UUs is algorithms. Lakkaraju et al. (2017) propose an algorithm for identifying UUs, given a test set of examples with predicted labels and confidence scores, and an oracle to query for the true labels. Their approach has two steps. First, "Descriptive Space Partitioning" is performed to cluster all high confidence instances by both their features and confidence scores. Then, each cluster is treated as an arm in a multi-armed bandit, and candidates are queried from the most promising clusters based on their estimated mean utility. Bansal and Weld (2018) extend this work by proposing a coverage-based utility model that gives higher utility to the set of UUs which provide higher coverage of the test data. They outline a greedy algorithm for selecting candidates.

Although these algorithmic approaches are suitable for a fixed test set, we wish to identify UUs encountered by predictive models in the open world. Deployed models must be continually tested on new data, so the identification of UUs in a fixed set is often not adequate. We avoid this problem by employing workers to generate new instances. Like the crowdsourcing approach, current algorithmic approaches also fail to explicitly learn the model's behavior (i.e. *how* the model makes high confidence predictions). We show that learning a surrogate model and presenting it to workers can make the discovery of UUs markedly more efficient.

Problem Statement

We begin with the premise that a predictive model deployed in the wild is typically a black-box, for which neither the training data nor the details of the learned function are accessible. For any instance $\boldsymbol{x}$, the model M provides a predicted label, $\hat{y} \in C$, where C is the set of classes, and a confidence score $s \in [0,1]$. We define a UU as an instance that is predicted incorrectly, $\hat{y} \neq y$, and the model's confidence s is above some threshold τ, $s > \tau$. For simplicity, we target UUs of some critical class c for which false positives are particularly costly and need to be identified. In other words, we are interested in those UUs for which $\hat{y} = c \neq y$.

We also have access to a set of (unlabelled) test instances that is a subset of all possible instances, $X_{test} \subseteq X$. To obtain the label, the system can query a worker. In particular, the system queries only those instances in $X_{cand} \subseteq X_{test}$ which are valid UU candidates:

$$X_{cand} = \{\boldsymbol{x} | \boldsymbol{x} \in X_{test}, \hat{y} = c, s > \tau\} \tag{1}$$

Given a particular query instance $\boldsymbol{x_i} \in X_{cand}$, the worker knows the label y_i and has three actions available:

1. UU identification: if $y_i \neq c$, return `identify`, $(\boldsymbol{x_i}, y_i)$
2. modification: if $y_i = c$, modify $\boldsymbol{x_i}$ to produce some new instance $\boldsymbol{x_j} \in X$ such that $y_j \neq c$, and return `modify`, $(\boldsymbol{x_j}, y_j)$
3. rejection: if $y_i = c$, but the worker is unable to modify $\boldsymbol{x_i}$ to produce some new instance $\boldsymbol{x_j} \in X$ such that $y_j \neq c$, return `reject`, $(\boldsymbol{x_i}, y_i)$

We will provide details of the crowdsourcing interface in the Methodology section, and explain more specifically how each of these actions is performed. We assume that each of the three actions is associated with a fixed cost: $CostModify$, $CostIdentify$, and $CostReject$. In addition, we assume that the discovery of a UU has unit utility. The utility function can then defined as the utility of discovering a UU, minus the cost of the action performed:

$$u(q_t) = disc(q_t) - cost(q_t) \tag{2}$$

Here, $disc(q_t)$ is a function that returns 1 if the query q_t resulted in a UU discovery and 0 otherwise. $cost(q_t)$ is the cost of the action performed by the worker for that query. Note that the `identify` action always results in a UU discovery, while the `reject` action never results in a UU discovery. If the `modify` action is performed, the returned instance must be inputted into the model M to obtain its predicted label and confidence, to determine if it is a UU or not.

The objective is to find the sequence of queries that maximizes the total utility over n steps, $\sum_{t=1}^{n} u(q_t)$. This utility model is analogous to the one proposed by Lakkaraju et al., aside from costs for the `modify` and `reject` actions, which are non-existent in their setting.

Methodology

Our method for identifying UUs has two phases. In the first phase, we learn a surrogate model that explains how the black-box model M generates high confidence predictions of the critical class. This surrogate model is a set of interpretable decision rules. The second phase entails a crowdsourcing task in which workers are queried. Similar to Lakkaraju et al. (2017), we formulate the querying procedure as a multi-armed bandit, and develop a Bayesian bandit algorithm to sequentially choose rules to present to workers.

Phase 1: Decision Rule Learning

First, we seek to learn a surrogate model S, which takes some instance $\boldsymbol{x_i} \in X$ as input, and outputs 1 if it predicts that instance will be given a high confidence prediction to the critical class by M, and 0 otherwise.

There are two desirable properties for the surrogate model S. First, it should be *interpretable*, in the sense that a human can readily determine whether S will output 1 or 0 for a particular instance. This notion of interpretability is treated as a design principle, rather than a property we attempt to prove experimentally. The second desirable property is for the model to be *decomposable*, meaning that S can be broken down into components that each make predictions on disjoint subsets of the possible inputs X. Given such a model, a human can readily determine the model output for any instance using the single component of the model that covers it.

These two properties can be fulfilled by generating a set of decision rules. The left-hand side of the rule is a conjunction of predicates. Each predicate is of the form (attribute = value), where the attribute is an interpretable feature, and the value is 1 or 0 to indicate the presence or absence of that feature. The right hand side of the rule is always 1, indicating a high confidence prediction to the critical class by M. For example, if the critical class is positive movie reviews, a rule might take the form `best=1 AND bad=0 => 1`. To encourage interpretability, we favor short rules and set L_{max} to be the maximum number of predicates in any rule. To ensure decomposability, the rules must be non-overlapping, such that each instance x is covered by at most one rule.

We can generate such a rule set by applying the Classification And Regression Tree (CART) algorithm (Breiman et al. 1984). We first obtain the prediction and confidence of the model M for instances in X_{test}. Then, we discretize the output into instances predicted (1) or not predicted (0) to the critical class with high confidence:

$$\hat{y_i}' = \begin{cases} 1, & \text{if } s_i > \tau, \hat{y}_i = c \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

The dataset $D = (X_{test}, \hat{Y}')$ is then used to construct a decision tree.

CART uses Gini impurity as the criteria for choosing the next split to perform while constructing the decision tree. Gini impurity gives the likelihood that an element from some node would be incorrectly labelled if it was randomly labelled according to the distribution of class labels at that node. Since we are only interested in rules that explain high confidence critical class predictions, we only compute Gini impurity with respect to instances with $\hat{y}' = 1$:

$$I_{G,1} = p_1 \cdot (1 - p_1) \tag{4}$$

where p_1 is the fraction of instances in the set with $\hat{y}' = 1$.

Finally, we convert the decision tree into a set of if-then rules for high confidence predictions to the critical class. Every path of the decision tree from root to leaf is traversed. The rules correspond to all paths that end on a leaf with a class 1 majority. Each node along the path is inserted as a predicate in the decision rule.

We perform a grid search over several decision tree hyperparameters (maximum tree depth, minimum samples at leaf node) and choose the values that maximize the F_1 score over 10-fold cross validation. We choose F_1 score because the rule set should ideally have both high recall and high precision. High recall ensures that a large proportion of the UU candidates in the dataset are covered by the rule set, while high precision ensures that a large proportion of the instances covered by the rule set are actually UU candidates. In our search over the optimal tree depth, we only search for tree depths under L_{max} to ensure that the rules are short enough to be interpretable. This constraint also helps to mitigate overfitting. Performance of the decision rules on three datasets is reported in Table 1.

	Precision	Recall	F_1 Score
Pang2005	60.7	59.5	60.1
McAuley2013	79.2	75.8	77.4
Almeida2011	84.6	76.7	80.5

Table 1: Performance of the decision rules for three datasets on a validation set.

Phase 2: Contradict the Machine

In the second phase, we use the surrogate model to search for UUs via a crowdsourcing task that we refer to as Contradict the Machine (CTM). For this task, the worker is given an instance from X_{cand} that is covered by a decision rule. If the label is not the critical class, it is confirmed to be a UU and the worker returns the instance (`identify` action). Otherwise, the worker is given a challenge – modify the instance such that its label changes, while still ensuring that it is covered by the rule (`modify` action). The result is a contradictory example - one that according to the decision rule, should be confidently predicted by M to the critical class, yet whose label is *not* the critical class. The worker then returns the modified instance. In the case that the worker is unable to generate such an example, a third `reject` action is available to the worker. To select which instance and rule to present to the worker, we follow a similar framework as Lakkaraju et al. (2017). However, instead of needing to explicitly cluster instances in X_{cand}, we find that instances can be be effectively clustered by the decision rule coverage alone. The instance-rule pair to be presented to workers is selected by a multi-armed bandit algorithm that treats the rules as arms of the bandit.

	KMB	DRC	DRC (w. uncovered)
Pang2005	0.020	**0.081**	0.066
McAuley2013	0.156	0.170	**0.229**
Almeida2011	0.101	**0.113**	0.107

Table 2: Normalized mutual information for k-means-both (KMB) and decision rule clustering (DRC). DRC (w. uncovered) includes an additional cluster for all uncovered instances. Higher scores suggest more informative clusters.

Decision Rule Clustering Lakkaraju et al. (2017) describe an algorithm called k-means-both for generating informative clusters in X_{cand} with respect to UUs. This method works by clustering instances twice with the k-means algorithm, first by the confidence scores given by M, then by their features. The optimal number of clusters is selected via the elbow method.

In contrast, we show that instances in X_{cand} can be effectively clustered by decision rule coverage. This simple method, which we call decision rule clustering, entails grouping instances together that are covered by the same rule.

To compare these approaches, we determine the true distribution of UUs within each clustering, and then compute normalized mutual information (NMI), which quantifies the amount of information gained about whether an instance is a UU by knowing the cluster it belongs to. Normalization allows for a fair comparison between methods that produce different numbers of clusters. Our results show that decision rule clustering, with and without a cluster for orphaned instances, attains a higher NMI score than k-means-both (Table 2). This is a somewhat surprising result, since the main purpose of the decision rules is as a surrogate model to present to workers. The fact that UUs can be effectively clustered by decision rule coverage is an added benefit.

Bayesian bandit Next, we present a multi-armed bandit algorithm to intelligently select decision rules to present to workers in the CTM task. We compute statistics about each rule by the set of instances that it covers (i.e. decision rule clustering). In particular, we estimate the expected utility of each rule, as described below (Equation 5). Then, we choose the rule with the highest expected utility, randomly sample an instance from X_{cand} that is covered by that rule, and present that rule-instance pair to the worker. This process repeats for n steps.

The expected utility of each rule is computed as follows:

$$\begin{aligned}\mathbf{E}[u(q_t)] = P(A_t = \texttt{identify}) \cdot (1 - CostIdentify) \\ + P(A_t = \texttt{reject}) \cdot (-CostReject) \\ + P(A_t = \texttt{modify} \wedge UU_t) \cdot (1 - CostModify) \\ + P(A_t = \texttt{modify} \wedge \neg UU_t) \cdot (-CostModify)\end{aligned} \quad (5)$$

Here, A_t is the action taken by the worker at time t, and UU_t is a boolean variable indicating whether a UU was discovered or not.

In essence, Equation 5 specifies the expected utility by considering all possible worker actions. If the instance is already a UU (i.e. not from the critical class), the worker would perform the `identify` action. The probability of the `identify` action is the same as the probability that the instance x_t is not from the critical class, given it is covered by rule R (denoted as $x_t \in R$), i.e.,

$$P(A_t = \texttt{identify}) = P(y_t \neq c | x_t \in R) \quad (6)$$

Otherwise, either the `modify` or `reject` action is performed. We maintain some probability $P(rej)$ that the worker will *reject* a critical class instance instead of modifying it. Thus,

$$P(A_t = \texttt{reject}) = (1 - P(y_t \neq c | x_t \in R)) \cdot P(rej) \quad (7)$$

If the critical class instance is not rejected, the only remaining possibility is that the instance is *modified*. For the `modify` action, the probability that a UU is discovered depends on the precision of the rule over the set of all possible modified instances. Thus, we obtain:

$$\begin{aligned}P(A_t = modify \wedge UU_t) = (1 - P(y_t \neq c | x_t \in R)) \\ \cdot (1 - P(rej)) \\ \cdot P(\hat{y}_{mod,t} = c \wedge s_{mod,t} > \tau | x_{mod,t} \in R)\end{aligned} \quad (8)$$

$$\begin{aligned}P(A_t = modify \wedge \neg UU_t) = (1 - P(y_t \neq c | x_t \in R)) \\ \cdot (1 - P(rej)) \\ \cdot (1 - P(\hat{y}_{mod,t} = c \wedge s_{mod,t} > \tau | x_{mod,t} \in R))\end{aligned} \quad (9)$$

Hence, there are three probabilities that need to be estimated: $P(rej)$, $P(y_t \neq c | x_t \in R)$, and $P(\hat{y}_{mod,t} = c \wedge s_{mod,t} > \tau | x_{mod,t} \in R)$. For $P(rej)$, we assume a fixed probability that any given worker will reject a critical class instance instead of trying to modify it. This probability is estimated by the observed frequency of rejections.

The remaining two probabilities are rule-specific. $P(y_t \neq c | x_t \in R)$ represents the probability that an instance covered by rule R will already be a UU. We begin with a uniform prior, $Beta(1, 1)$ and update our prior according to the observed counts of input instances covered by R that are UUs/non-UUs. $P(\hat{y}_{mod,t} = c \wedge s_{mod,t} > \tau | x_{mod,t} \in R)$ represents the probability that the modified example will be predicted to the critical class with high confidence. We begin with the prior $Beta(a_{val}, b_{val})$, where a_{val} and b_{val} represent the counts of covered instances in the validation set that are predicted, and not predicted, to the critical class with high confidence. We update the priors according to the observed counts of modified instances covered by rule R that are UUs/non-UUs. To trade off the exploitation of the most promising rules with exploration, we employ Thompson sampling (Thompson 1933). At each step, we draw a single sample from each of the posteriors to obtain the probability estimates.

Experiments

We evaluated our method on three datasets:

Pang2005 (Pang and Lee 2005): This dataset is comprised of 10k movie review snippets from Rotten Tomatoes. Each sentence is classified by sentiment as negative or positive (the critical class).

McAuley2013 (McAuley and Leskovec 2013): This dataset contains ∼500k reviews from the Amazon Fine Food Store, labeled as negative (1 or 2 star ratings) or positive (the critical class – 4 or 5 star ratings). Only short ($\leq$ 280 characters) reviews are used.

Almeida2011 (Almeida, Hidalgo, and Yamakami 2011): This dataset is comprised of ∼5k SMS text messages which are classified as non-spam or spam (the critical class).

A logistic regression classifier was used as the black-box model, trained on a bag-of-words representation of the text. Following prior work, we induced bias in the training data to ensure that there were sufficient UUs to be discovered (Lakkaraju et al. 2017). We follow the same procedure for inducing bias – a decision tree is learned on the training data, and all examples from a randomly chosen leaf are deleted. For Almeida2011, we additionally bias the training data by making the distribution of spam and non-spam balanced (the dataset otherwise skews heavily towards non-spam).

Baselines

The Contradict the Machine (CTM) method was evaluated against several baselines.

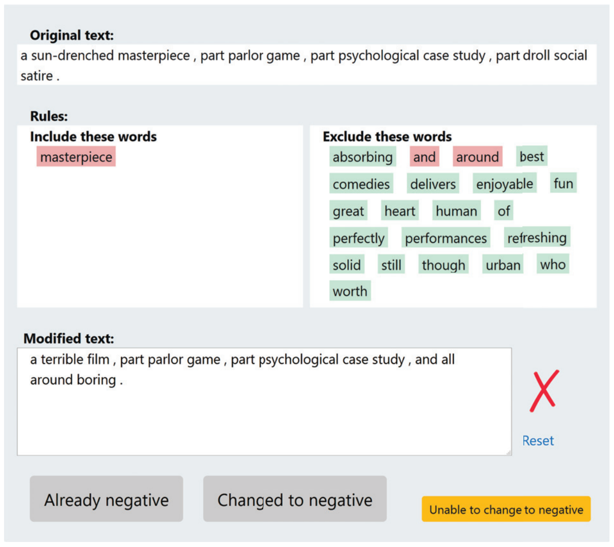
- **UUB**: A re-implementation of the bandit algorithm proposed by Lakkaraju et al. We test versions using k-means-both (UUB-KMB) and decision rule clustering (UUB-DRC).
- **CTM-NoRule**: A variant of CTM that does not present the worker with any rule that the modified instance must satisfy.
- **CTM-Random**: A variant of CTM that randomly selects instance-rule pairs to present to workers instead of the Bayesian bandit algorithm.

Crowdsourcing Interface

The task interface is shown in Figure 1. At the top of the page is the text of the original instance. The rule is represented by a set of words that must be included, and a set of words that must be excluded. There are three buttons at the bottom of the page that correspond to the three possible actions that workers can take. The worker can select the `identify` button if the text is already not the critical class. Otherwise, the worker can attempt to modify the text so that it is no longer the critical class, and presses the `modify` button if successful, and the `reject` button otherwise. When modifying text, the modified text is validated in real-time, with a graphical indicator showing whether the rule is satisfied and, if not, which conjunct(s) are unsatisfied. The CTM-NoRule interface is similar, except that no rule is displayed, and the user is free to make any modifications to the text that they wish.

User Study

To evaluate our Contradict the Machine approach, we conducted a user study on Amazon Mechanical Turk. The HIT was comprised of three sections: pre-study questionnaire, CTM tasks, and post-study questionnaire. We required workers to have at least 1000 approved HITs with a 97% or higher approval rate.


Original text:
a sun-drenched masterpiece , part parlor game , part psychological case study , part droll social satire .

Rules:

Include these words
masterpiece

Exclude these words
absorbing and around best comedies delivers enjoyable fun great heart human of perfectly performances refreshing solid still though urban who worth

Modified text:
a terrible film , part parlor game , part psychological case study , and all around boring .

Figure 1: Crowdsourcing interface for a sample text from Pang2005. In this case, the label has been successfully modified from positive (the critical class) to negative. However, the rule is not satisfied. Words highlighted in green represent satisfied conjuncts, and in red, unsatisfied conjuncts.

All workers were required to fill out a consent form before starting the HIT. Then, demographics information was collected in the pre-study questionnaire. Next, a block of 10 CTM tasks was given to each participant. Finally, a post-study questionnaire was administered. Workers were asked to rate the difficulty of labelling and modifying the text, 1 being 'not difficult at all' to 7 being 'very difficult'. The Raw-TLX Scale (RTLX) was also administered to measure workload (Hart 2006).

Workers were given a base payment of \$0.50, and bonus payments for each action that corresponded to their relative cost. The $CostIdentify$ and $CostReject$ were both set to \$0.02, while $CostModify$ was set to \$0.20. The costs reflected the estimated time and effort required to perform each action in a crowdsourcing setting.

We evaluated CTM for 500 steps (corresponding to ∼50 workers) across all conditions and datasets, with the exception of Almeida2011, which we evaluated for 300 steps (∼30 workers) due to a smaller number of candidate instances available to query.

Results

We report on both the cumulative utility of our method and the experience of workers performing the CTM task. Our baseline for assessing cumulative utility is the previous algorithmic approach, UUB. Since UUB queries consist of only labelling, and all of the datasets were already labelled, this algorithm could be evaluated entirely in simulation. Labelling is 100% accurate in simulation, so to ensure a fair comparison, we also assume perfect labelling when evaluating CTM.

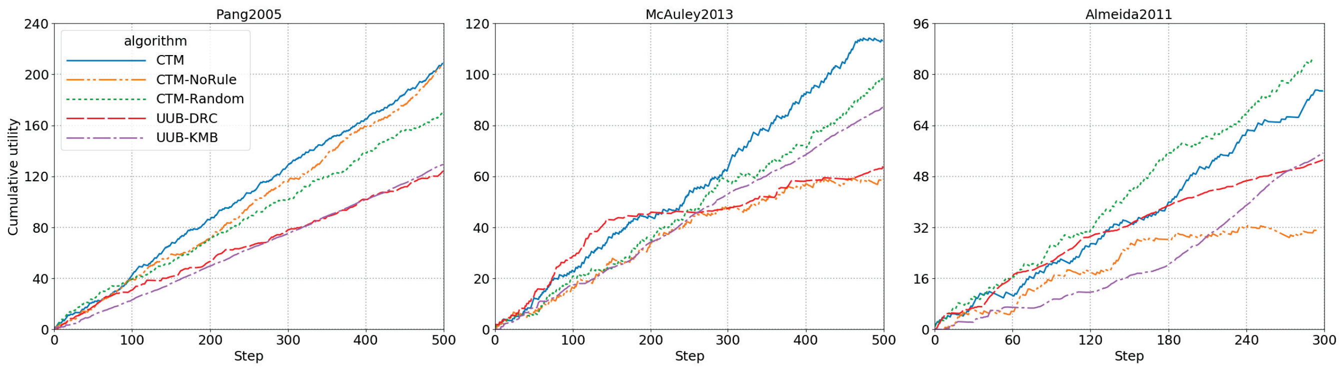


Figure 2: Performance of the CTM and baselines in terms of cumulative utility. The y-axis is the utility, and the x-axis is the number of worker queries. Results for CTM are obtained from a user study on Amazon Mechanical Turk (1 run), while results for UUB are obtained in simulation (average of 100 runs).

	% improvement
Pang2005	67.5
McAuley2013	32.1
Almeida2011	68.5

Table 3: Percentage increase in cumulative utility of CTM over UUB-KMB, found by comparing areas under the cumulative utility curves.

Utility

The cumulative utility achieved by CTM and baselines is shown in Figure 2. On all three datasets, CTM and its variants tended to perform better than UUB. Table 3 shows the percentage increase in utility of CTM. These results suggest that CTM is able to successfully elicit UUs from workers, and that the added costs of modifying instances is outweighed by the number of UUs that the workers are able to successfully generate.

Figure 3 shows the breakdown of UUs discovered from the test set (i.e. algorithm proposed) and UUs generated by the worker (i.e. worker proposed). Both contributions are substantial, indicating the value of a hybrid approach.

Comparison of CTM with CTM-NoRule suggests that the rules are important, but their importance may vary between datasets. Performance is vastly improved with rules on the McAuley2013 and Almeida2011 datasets. On the other hand, performance is comparable on Pang2005. This may be attributed to the fact that the Pang2005 rule set has the poorest precision (Table 1), and therefore is not as helpful in the task.

The difference in performance between CTM and CTM-Random is small across all datasets, with CTM slightly outperforming CTM-Random in the Pang2005 and McAuley2013, and vice versa on Almeida2011. The only difference between CTM and CTM-Random is the querying strategy — the crowdsourcing task facing workers is identical. However, we find that the number of `reject` actions is quite variable, with a small number of workers accounting for a large number of the `reject` actions. Indeed, a higher

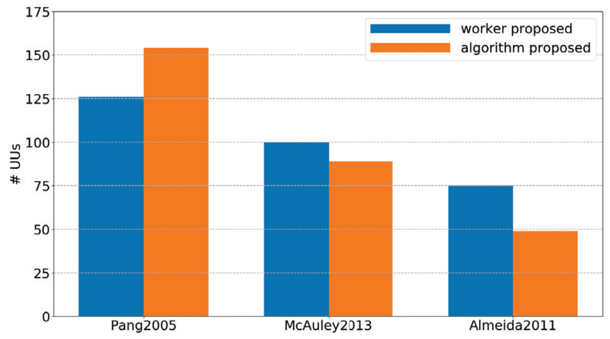
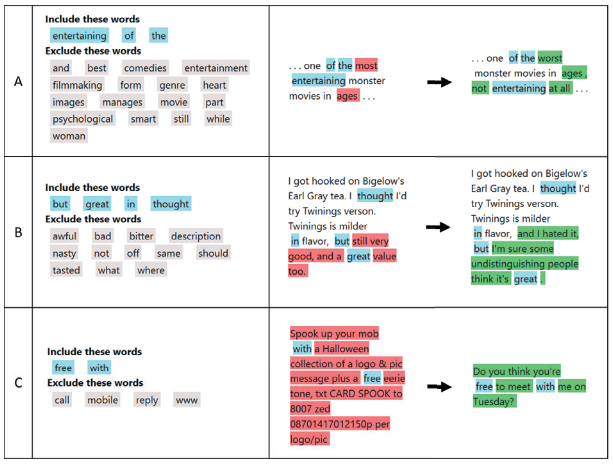


Figure 3: UUs identified from the test set (algorithm proposed) and generated by the worker (worker proposed) using CTM.

rejection rate is associated with lower performance when comparing CTM and CTM-Random across all datasets. It appears that the large variation in rejection rate may obscure the effect of the querying strategy.

Worker Experience

We used RTLX responses to assess the workload of users in each task variant (CTM, CTM-NoRule) and dataset. A two-factor ANOVA using Type-3 SS was performed to test the influence of task and dataset on the overall workload. Neither factor was found to have a statistically significant interaction with overall workload. To probe further into workload differences, two-factor ANOVAs were performed on each RTLX subscale: mental demand, physical demand, temporal demand, overall performance, effort, and frustration level. Statistical significance was only found between task and frustration level ($F(2, 363) = 11.641, p < 0.001$). A T-test revealed that the frustration level of CTM-NoRule ($M = 32.6, SD = 26.9$) was statistically significantly lower ($t(244) = -3.088, p < 0.01$) than CTM ($M = 43.7, SD = 29.5$). To evaluate the difficulty of labelling and modifying the text across tasks and datasets, two-factor ANOVAs were performed as above. There was no statis-



A	**Include these words** entertaining of the **Exclude these words** and best comedies entertainment filmmaking form genre heart images manages movie part psychological smart still while woman	. . . one of the most entertaining monster movies in ages . . .	→	. . . one of the worst monster movies in ages , not entertaining at all . . .
B	**Include these words** but great in thought **Exclude these words** awful bad bitter description nasty not off same should tasted what where	I got hooked on Bigelow's Earl Gray tea. I thought I'd try Twinings verson. Twinings is milder in flavor, but still very good, and a great value too.	→	I got hooked on Bigelow's Earl Gray tea. I thought I'd try Twinings verson. Twinings is milder in flavor, and I hated it, but I'm sure some undistinguishing people think it's great .
C	**Include these words** free with **Exclude these words** call mobile reply www	Spook up your mob with a Halloween collection of a logo & pic message plus a free eerie tone, txt CARD SPOOK to 8007 zed 08701417012150p per logo/pic	→	Do you think you're free to meet with me on Tuesday?

Figure 4: Sample UUs proposed by workers for (A) Pang2005, (B) McAuley2013, and (C) Almeida2011. On the left is the rule, indicating which words must be included and excluded. In the center is the original text, and on the right is the modified text. Deletions are marked in red, insertions in green, and mandatory words to include in blue.

tically significant interaction between these factors and labelling difficulty or modifying difficulty.

Taken together, these results indicate that the task is challenging, but does not overburden workers in terms of workload. These findings are consistent across all datasets. The addition of rules does increase worker frustration, which is not surprising since they constrain what the workers are allowed to submit. However, the overall workload is not significantly different whether workers are given rules or not.

UUs generated

We noticed some commonalities in how workers generated UUs. Since workers were provided with a instance that satisfied the rule, it was common for workers to keep much of the text the same, and make only the minimal modifications required to change the ground truth. This behavior is advantageous for the purposes of discovering UUs: if the original instance is predicted with high confidence to the critical class, a minimal modification is unlikely to change this prediction, so the change of ground truth will likely turn it into a UU. An exception to this behavior is found with Almeida2011 — in this case, less of the original text was preserved by workers, since changing a SMS text from spam to non-spam generally required much of the content of the message to be changed. This can be observed in Figure 4C.

Workers devised creative and surprising strategies for modifying examples under the rule constraints. Often, workers exploited the fact that a word feature could have multiple meanings. The example in Figure 4C shows such a modification. The rule states that the classifier will predict any instance that includes "free" and "with", and excludes "call", "mobile", "reply" and "www" as spam with high confidence. Clearly, this rule is sensible if "free" is used in the sense of cost, since SMS spam often advertises free products or services. However, when modifying the instance, the worker changes "free" to mean "available", and in doing so, changes the instance to a plausible non-spam message.

Another strategy was to not change the meaning of the word features, but instead to use them in a different context. The example shown in Figure 4B illustrates this idea. The rule requires that the word "great" be included in the negative review. Rather than calling product itself great, the worker modifies the context so that it is "indistinguishing people" who *think* the product is great. In extreme cases, almost the entire modified instance was identical to the original, with a slight shift of context to change the ground truth. For example, when modifying a spam text from Almeida2011, one worker put the entire text in quotes, and added a short message that explains to their friend how much they dislike receiving spam like the one quoted.

Conclusion

Two avenues of research have emerged for discovering the UUs of a predictive model. In the crowdsourcing approach, it is the humans who suggest candidate UUs. In the algorithmic approach, it is an algorithm that suggests candidate UUs. This work proposes a hybrid approach, in which candidates are suggested by both the algorithm and human workers. To combine these approaches, we propose learning a set of interpretable decision rules that explain how high confidence predictions are made. We design a crowdsourcing task called Contradict the Machine, in which these decision rules can augment the ability of workers to generate UUs. A Bayesian bandit algorithm is used to intelligently choose which rules and instances to present to workers.

We show that this hybrid strategy outperforms the algorithmic approach proposed by Lakkaraju et al. in terms of cumulative utility. Both the worker-proposed and algorithm-proposed UUs make substantial contributions to the overall utility. Across all datasets, workers are able to complete the task effectively, without being overburdened by workload.

There are some limitations to our implementation. The crowdsourcing task was implemented for text datasets, but would likely need to be redesigned to accommodate other kinds of datasets (e.g. tabular datasets). In addition, the length of text in the datasets we tested were fairly short. This made it amenable to crowdsourcing since each task was fairly simple and quick to complete. If our approach is to be applied to longer and more complex instances (e.g. news articles), strategies would need to be developed to make the task more manageable. For example, it could be designed as a collaborative task wherein multiple people work together to modify the full text, or decomposed into smaller tasks that can be completed individually.

The ideas presented in this paper represent a viable new direction of research into identifying blind spots of predictive models. There are various possible avenues for future work. In addition to adapting this approach to other types of data, it might be adjusted to suit other utility models, like coverage-based utility (Bansal and Weld 2018). We could also incorporate mechanisms for repeated labelling to take account of human error. Finally, we could explore strategies to make use of the specific domain knowledge and expertise of workers.

References

Almeida, T. A.; Hidalgo, J. M. G.; and Yamakami, A. 2011. Contributions to the study of SMS spam filtering: new collection and results. In *Proceedings of the 2011 ACM Symposium on Document Engineering, Mountain View, CA, USA, September 19-22, 2011*, 259–262.

Attenberg, J.; Ipeirotis, P. G.; and Provost, F. J. 2011. Beat the machine: Challenging workers to find the unknown unknowns. In *Human Computation, Papers from the 2011 AAAI Workshop*.

Attenberg, J.; Ipeirotis, P.; and Provost, F. J. 2015. Beat the machine: Challenging humans to find a predictive model's "unknown unknowns". *J. Data and Information Quality* 6(1):1:1–1:17.

Bansal, G., and Weld, D. S. 2018. A coverage-based utility model for identifying unknown unknowns.

Ben-David, S.; Blitzer, J.; Crammer, K.; and Pereira, F. 2006. Analysis of representations for domain adaptation. 137–144.

Breiman, L.; Friedman, J. H.; Olshen, R. A.; and Stone, C. J. 1984. *Classification and Regression Trees*. Wadsworth.

Hart, S. G. 2006. Nasa-task load index (nasa-tlx); 20 years later. In *Proceedings of the human factors and ergonomics society annual meeting*, volume 50, 904–908. Sage publications Sage CA: Los Angeles, CA.

Lakkaraju, H.; Kamar, E.; Caruana, R.; and Horvitz, E. 2017. Identifying unknown unknowns in the open world: Representations and policies for guided exploration. 2124–2132.

McAuley, J. J., and Leskovec, J. 2013. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, 897–908.

Pang, B., and Lee, L. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. 115–124.

Thompson, W. R. 1933. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika* 25(3/4):285–294.