Beat the Machine: Challenging Humans to Find a Predictive Model's "Unknown Unknowns"

Josh Attenberg
New York University
Brooklyn, NY
josh.attenberg@gmail.com

Panagiotis G. Ipeirotis
NYU Stern School of Business
New York, NY
panos@stern.nyu.edu

Foster Provost
NYU Stern School of Business
New York, NY
fprovost@stern.nyu.edu

ABSTRACT

We present techniques for gathering data that expose errors of automatic predictive models. In certain common settings, traditional methods for evaluating predictive models tend to miss rare-but-important errors—most importantly, rare cases for which the model is confident of its prediction (but wrong). In this paper we present a system that, in a game-like setting, asks humans to identify cases that will cause the predictive-model-based system to fail. Such techniques are valuable in discovering problematic cases that do not reveal themselves during the normal operation of the system, and may include cases that are rare but catastrophic. We describe the design of the system, including design iterations that did not quite work. In particular, the system incentivizes humans to provide examples that are difficult for the model to handle, by providing a reward proportional to the magnitude of the predictive model's error. The humans are asked to "*Beat the Machine*" and find cases where the automatic model ("*the Machine*") is wrong. Experiments show that the humans using Beat the Machine identify more errors than traditional techniques for discovering errors in from predictive models, and indeed, they identify many more errors where the machine is confident it is correct. Further, the cases the humans identify seem to be not simply outliers, but coherent areas missed completely by the model. Beat the machine identifies the "unknown unknowns."

> *"There are known knowns. These are things we know that we know. There are known unknowns. That is to say, there are things that we know we don't know. But there are also unknown unknowns. There are things we don't know we don't know."*

> – *Donald Rumsfeld*

1. INTRODUCTION

Many businesses and government organizations make decisions based on estimations made by explicit or implicit models of the world. Being based on models, the decisions are not perfect. Understanding the imperfections of the models is important (i) in order to improve the models (where possible), (ii) in order to prepare to deal with the decision-making errors, and (iii) in some cases in order to properly hedge the risks. However, a crucial challenge is that, for complicated decision-making scenarios, we often do not know where models of the world are imperfect and/or how the models' imperfections will impinge on decision making. *We don't know what we don't know.*

We see the results of such failures of omniscience in grand catastrophes, from terrorist attacks to unexpected nuclear disasters, in mid-range failures, like cybersecurity breaches, and in failures of operational models, such as predictive models for credit scoring, fraud detection, document classification, etc.

Unknown unknowns are related to the classic contrast between reasoning systems that make open- and closed-world assumptions [12]: are the only answers to a query Q those that are actually in the database? In the context of predictive modeling, in applications with limited labeled training data, small disjuncts[16], and possibly unknown selection biases, are we willing to make the assumption that regularities that have no or insufficient representation in the training data essentially do not exist?

In this paper we introduce and analyze a crowdsourcing system designed to help uncover the "unknown unknowns" for predictive models. The system is designed to apply to settings where assessing the performance of predictive models is particularly challenging. Later we will describe in detail the critical aspects of such settings, but first let us introduce a motivating example to make the discussion concrete.

Consider the following task: a firm has built a system for identifying web pages that contain instances of "hate speech" (e.g., racist content, antisemitism, and so on), based on a model that takes web pages as input and produces as output a "hate score." The firm would like to use this system to help protect advertisers, who (despite the best efforts of their advertising agents) sometimes see their ads appearing adjacent to such objectionable content. The advertisers do not want their brands to be associated with such content, and they definitely do not want to support such content, explicitly or implicitly, with their ad dollars.

How does this firm assess the strengths and weaknesses of its system and model? This scenario comprises a constellation of factors that are not uncommon in organizational decision making, but are quite problematic for conducting the assessment—particularly because of the problem of un-

known unknowns. Specifically, this paper considers applications where:

- Every decision-making case can be represented by a description and a target. We have a (predictive) model that can give us an estimate or score for the target for any case. For this paper, we assume for simplicity that the target is binary, and that the truth would not be in dispute if known.[1]

- We want to understand the inaccuracies of the model—specifically, the errors that it makes, and especially whether there are systematic patterns in the errors. For example, is there a particular sort of hate speech that the model builders did not consider, and therefore the model misses it?

- The process that is producing the data does not (necessarily) *reveal* the target for free. In our example, if we misclassify a hate speech page as being OK, we may never know. (Indeed, we usually never know.) This is in contrast to *self-revealing* processes; for example, in the case of credit-card fraud detection, we will eventually will be informed by the customer that there is fraud on her account. For targeted marketing, we often eventually know whether the consumer responded to an offer or not.

- Finally, there are important classes or subclasses of cases that are very rare, but nevertheless very important. The rarity often is the very reason these cases were overlooked in the design of the system. In our example, hate speech on the web itself is quite rare (thankfully). Within hate speech, different subclasses are more or less rare. Expressions of racial hatred are more common than expressions of hatred toward dwarves or data miners (both real cases).

These problem characteristics combine to make it extremely difficult to discover system/model imperfections. Just running the system, in vitro or in vivo, does not uncover problems; as we do not observe the true value of the target, we cannot compare the target to the model's estimation or to the system's decision.

We *can* invest in acquiring data to help us uncover inaccuracies. For example, we can task humans to score random or selected subsets of cases. Unfortunately, this has two major drawbacks. First, due to the rarity of the class of interest (e.g., hate speech) it can be very costly to find very few positive examples, especially via random sampling of pages. For example, hate speech represents far less that 0.1% of the population of web pages, with unusual or distinct forms of hate speech being far rarer still. Thus we would have to invest in labeling more than 1000 web pages just to get one hate speech example, and as has been pointed out recently, often you need more than one label per page to get high-quality labeling [14, 11].

In practice, we often turn to particular heuristics to identify cases that can help to find the errors of our model. There has been a large amount of work studying "active learning" which attempts to find particularly informative examples [13]. A large number of these strategies (uncertainty sampling, sampling near the separating hyperplane,

query-by-committee, query-by-bagging, and others) essentially do the same thing: they choose the cases where the model is least certain, and invest in human labels for these. This strategy makes sense, as this is where we would think to find errors. Additionally, there has been a long history of understanding that "near misses" are the cases to use to best improve a model, both for machine learning [17] and for human learning [15].

Unfortunately, although helpful in understanding and improving modeling, for finding unknown unknowns, these strategies look exactly where we don't want to look. These strategies explicitly deal with the "known unknowns." The model is uncertain about these examples—we "know" that we don't know the answer for them (i.e., we have low confidence in the model's output). These strategies explicitly eschew, or in some cases probabilistically downweight, the cases that we are certain about, thereby *reducing* the chance that we are going to find the unknown unknowns.

With that substantial preamble, we can now state succinctly the goal and contributions of this paper. First, we describe the problem more formally, including relationships to prior work. We next discuss changes to how we need to view the evaluation of classifiers, if we want to move from a closed-world view of a predictive modeling problem to an open-world view. Then we introduce a technique and system to use human workers to help find the *unknown unknowns*. Our BeatTheMachine (BTM) system combines a game-like setup with incentives designed to elicit cases where the model is confident and wrong. Specifically, BTM rewards workers that discover cases that cause the system to fail. The reward increases with the magnitude of the failure. This setting makes the system to behave like a game, encouraging steady, accurate participation in the tasks. We describe our first experiences by the live deployment of this system, in a setting for identifying web pages with offensive content on the Internet. We show that this BTM setting discovers cases that are inherently different than the errors identified by a random sampling process. In fact, the two types of errors are very different. The BTM process identifies "big misses" and potential catastrophic failures, while traditional model-based example selection identifies "near misses" that are more appropriate for fine-tuning the system. The evidence shows that BTM does not just find individual "oddball" outlier cases, but it finds systematic big errors. In a sense, the BTM process indeed gives us the opportunity to learn our "unknown unknowns" and warn us about the failures that our current automatic model cannot (yet) identify by itself.

2. UNKNOWN UNKNOWNS

In order to provide a detailed discussion of unknown unknowns in the context of a predictive model, it is first necessary to formalize the concepts we will be discussing. Let x represent an example belonging to some problem space X. In classification settings, x has a "true" label, $\bar{y}$ from some set of possible labels Y. The task of a classification is to construct some predictive model, $f(x)$, that can estimate a label for each incoming example ($y = f(x)$) such that the estimated label y mirrors the (usually hidden) true label, $\bar{y}$, as closely as possible. In this work, we are concerned only with models that output a posterior probability estimate over the set of available labels, that is, $p(y|x) = f(x)$. Such probabilities can then be used to select a preferred label, for

---

[1]For our example, the description of the case would be the web page (its words, links, images, metadata, etc.). The target would be whether or not it contains hate speech.

instance by choosing the y with the highest probability, or in the cost-sensitive setting, choosing the example with the least expected cost [6]. Note that the focus on models that produce probability estimates is without loss of generality—there exists a variety of techniques for transforming "hard-labeling" models into probability estimators (see, eg. [5, 10]).

DEFINITION 1 (KNOWN UNKNOWN). *Let $M(x) = 1 - p(\hat{y}|x)$, where $\hat{y} = \arg\max_y p(y|x)$, the class with the greatest expected posterior. Let $\epsilon \in [0,1]$ be some confidence threshold denoting a radius from the decision boundary. Let x' be an example with $M(x') \leq \epsilon$. x' is said to be a "Known unknown" if for x', $\hat{y} \neq \bar{y}$.*

Known unknowns as described in Definition 1 corresponds to a commonly occurring notion in machine learning. The ϵ-radius around the decision boundary corresponds to an "uncertainty" region, an area where the predictive model is unsure of itself, and where mistakes are likely to be made. This concept has been exploited in a variety of contexts, for instance, when applied to the problem of gathering labels for the purpose of model training, selecting those examples within an ϵ-radius of the decision boundary corresponds to uncertainty sampling, perhaps the most well known active learning heuristic [9].

Used in prediction time, the "region of uncertainty" around the decision boundary has been used to identify those examples where prediction should be deferred. Classification with a "reject option" is an extension of traditional classification where in addition to labeling each example with some $y \in Y$, a predictive system may additionally defer prediction, either by ignoring an example entirely, or perhaps sending the example to a domain expert for manual evaluation [4, 3]. Given that such "rejection" likely comes at some non-trivial cost $q(x)$, the task of classification with a reject option is then to balance the expected misclassification costs with the costs of rejection.

Formally, let $\text{cost}(y^k|y^j)$ encode the penalty for predicting $f(x) = y^k$ when in fact the true label for x is y^j. In this case, based on the posterior probability estimates generated by $f(x)$, a model should ideally choose the y^k that minimizes the expected misclassification cost:

$$L(x, y^k) = \sum_{y' \in Y} p(y = y'|x)\text{cost}(y^k|y')$$

Imagine for simplicity, the setting with balanced misclassification costs, that is, w.l.o.g., $\text{cost}(y^k|y^j) = 1$ whenever $y^k \neq y^j$ with 0 costs for correct label attribution. In this case, it is straightforward to show [4, 3] that the optimal "reject" policy, $\mathcal{A}$ offering a minimum reject rate for a given expected error probability (or, equivalently, minimizing the expected error probability for a given reject rate) is given by:

$$\mathcal{A} = \left\{ x \| \min_c \hat{p}(y = c|x) > q(x) \right\} \tag{1}$$

As would be expected, larger query costs tend to prevent much of the efficacy of the reject option; indeed the reject option would never be exercised with $q(x) > \frac{1}{2}$. (It is optimal to always query the oracle when $q(x) = 0$, thereby yielding zero misclassification risk, assuming a perfect oracle.) Figure 1 presents the decision regions for varying query costs as a function of the posterior probability, $p(y = 1|x)$. It is never advantageous to query an oracle once the costs exceed
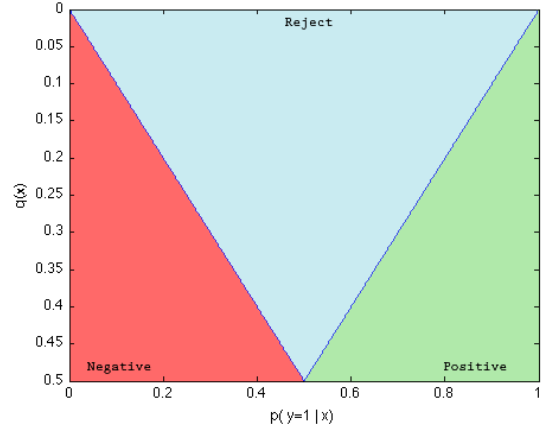
Figure 1: The reject and classification regions defined by Chow's original classification with a reject option criterion defined in Equation 1

0.5. Of course the uniform misclassification costs assumed by above are seldom realistic. Extending the reject rule of Chow to the case of asymmetric misclassification costs, Herbei and Wegkamp [8] show that the optimal $\mathcal{A}$ is given by:

$$\mathcal{A} = \left\{ x \| \min_{\hat{y}} L(x, \hat{y}) > q(x) \right\} \tag{2}$$

As in the case of symmetric cost classification with a reject option, there are certain regions of the posterior/cost space where predictions should be rejected, and where certain labels should be chosen. Figure 2 presents these regions for a binary classification as a function of the posterior, $p(y = 1|x)$ and of a (uniform) label cost, $q(x)$ for three different cost settings. In order to reduce the complexity of the problem being considered, we assume zero costs for correct classifications, and a false positive cost of 1, eg. $\text{cost}(1|0) = 1$, varying only $\text{cost}(0|1)$. This is equivalent to considering only the ratio of false negative to false positive costs, and re-normalizing.

In the context of classification with a reject option, "known unknowns" are those errors which are expected based on the confidence of the classification. These are cases where it may be less costly to "reject" than to make a risky label prediction. The uncertainty radius, ϵ, can be thought of as a constant $q(x)$ across all x. However, while it is important to understand the mistakes that your model is known to make and to react in an appropriate manner, models in production often make mistakes far from this area of predicted uncertainty. Consider the hate speech classification system discussed previously. While deployed in a production setting, this model is likely to encounter examples eliciting a high degree of predicted uncertainty.[2] Those managing the model can react to these borderline examples, and perhaps build some rough estimate of a model's overall exposure to misclassification risk. However, for a variety of reasons, the model may also encounter examples where it will assign a label with high confidence, and be wrong. Call all such examples "unknown unknowns."

[2]For instance, a encyclopedia entry discussing racial issues or a history of racism.
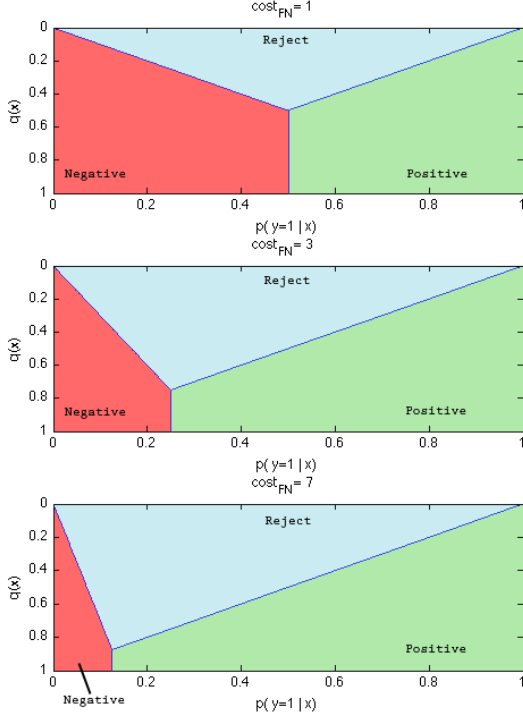
Figure 2: The reject and classification regions defined by Herbei and Wegkamp's misclassification cost-sensitive classification with a reject option criterion defined in Equation 2 for different false negative costs ($\mathbf{cost_{FN}} = \mathbf{cost}(0|1)$).

DEFINITION 2 (UNKNOWN UNKNOWN). *Following Definition 1, let $M(x) = 1 - p(\hat{y}|x)$ and let ϵ be some confidence threshold denoting a radius from the decision boundary, with $\epsilon \in [0, 1]$. In this setting, an example x' is said to be an "unknown unknown" if $M(x') > \epsilon$, that is, x' is outside the region of uncertainty, but $\hat{y} \neq \bar{y}$, i.e., the example is misclassified but the classifier is certain that it was correctly classified.*

Definition 2 codifies the notion of an "unknown unknown". Intuitively, these are examples that are distant from any decision boundary, examples that the model is quite certain a correct label can be assigned, yet are still labeled incorrectly. While in the strict sense, the above definition includes "random noise"—individual examples that for whatever reason do not have the expected label,[3]—the motivating case is disjunctive sub-regions of the problem space [16]. These are small, yet consistently labeled neighborhoods of examples isolated from the body of examples of the same class. These "islands" of examples may be sufficiently rare in the relative sense to avoid detection from random sampling processes used to generate training sets [1, 2]. However, their absolute size and prevalence in many real world problems makes them a genuine risk.

---

[3]For instance, due to erroneous labeling, signal degradation, or non-pathological difficulties in data collection.

Figure 3 presents a fairly typical classification scenario that might be impacted by "unknown unknowns". On the top, we see an inseparable two-class problem, with a linear decision boundary that minimizes the prediction errors on this space. Above and below this decision boundary, we see an example ϵ-radius, encapsulating the region where mistakes are known to occur. This represents a typical post-training understanding of the problem space: data is gathered by some random process, for instance via active learning. An imperfect model is trained, however, areas where mistakes occur are known and expectations can be managed. On the bottom we see a typical scenario when such models are deployed in the wild—rare disjunctive sub-regions of the problem space emerge. These are portions of the problem space that escaped the initial sampling process used to generate the training set. These unknown unknowns while small in terms of their proportion of the problem space may still exist in large absolute numbers. However, because they are unobserved during model construction, they have likely escaped any possible contingency planning for dealing with their associated mistakes.
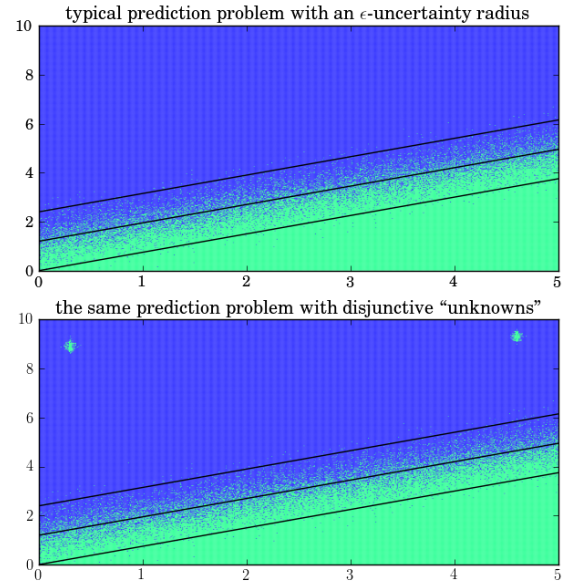
Figure 3: A typical classification setting. On the the top we see the decision boundary that minimizes the prediction prediction error of a inseparable training set. Additionally, we see the ϵ-radius around the classifier where mistakes are though to occur. The bottom, we see the same classifier with the inclusion of small, disjunctive "unknowns", presenting mistakes that occur well outside a model's region of uncertainty.

[**JOSH:***if i have time, go back and rephrase things in terms of cost bounds rather than probability bounds.*]

3. MEASURING UNKNOWN UNKOWNS

This section attempts to quantify the tradeoffs associated with unknown unknowns faced by predictive modeling sys-

tems. Prudence dictates that any predictive model should be evaluated thoroughly in a lab before deployment into the wild. However, because perfect predictive performance is rarely achieved in practice, a model's manager often needs to be concerned with the cases that the model deems to be uncertain. The uncertainty radius ϵ is a theoretical construct representing the cases of concern: those for which the model is too uncertain. A higher uncertainty radius reduces the chance of the model encountering unexpected surprises, but presumably there is an increased cost incurred to deal with these cases. The cases could be rejected from automatic processing, and sent to humans (cf., the setting of classification-with-a-reject option setting discussed above), or the cases may get extra-close monitoring, or customer-expectations may be lowered, etc. On the other hand, a lack of concern may result in unmitigated mistakes, which may induce cost beyond that enumerated in a problem's formal loss structure due to a lack of preparation and contingency planning.

Figure 4 presents example curves showing the tradeoffs between the percentage of cases lying inside a model's ϵ-radius uncertainty bounds and the number of unmitigated mistakes (unknown unknowns) made by that model, considering the same problem presented in Figure 3. Consider the black curve for the linear model. As ϵ increases, the number of points considered uncertain also increases (x-axis)—increasing the cost of dealing with the uncertain instances. So, moving to the right in the graph represents increasing the cost of dealing with the cases "of concern." On the other hand, increasing the ϵ radius also decreases the number of unknown-unknowns (y-axis)—thereby decreasing the corresponding unknown-unknown cost.

Viewing model performance in this way changes our perspective from simply looking at accuracy and confusion-matrix-based loss. Different models and different sampling regimes may have very different characteristic performance when comparing the relationship of the number of cases needed to be deemed uncertain in order to limit the number of unknown unknowns. To illustrate, in addition to the linear model displayed above, we also consider two additional models, a k-NN model comprising a set of training examples selected randomly from the problem space.[4] The intuition behind the k-NN is that by drawing random probes from across the problem space we may better cover the unknown unknowns. Is that the case? We see that the efficacy of such a regime depends both on the setting of ϵ and the coverage of the problem space, as quantified by the number of training examples.

We see from Figure 4(c) that given a very good deal of coverage over the problem space, there 3-NN model can reduce the cost of unknown unknowns substantially over the linear model while still offering only a slight uncertainty overhead. For the linear model, the percentage of unknown unknowns decreases rapidly as the uncertainty boundary expands from the decision threshold— this is coverage over the points that would be incorrectly classified in the "noisy" region around the decision boundary. This quick increase is in large part a product of the purely linear noise model used to generate this problem.[5] Beyond this "fuzzy" portion of the problem space, the uncertainty bound needs to be expanded greatly,

covering more than 95% of all examples before there is full coverage over the disjunctive sub-regions seen in the bottom of Figure 3.

Because the 3-NN model was generated with points selected completely at random, a large portion of coverage is wasted on uninteresting parts of the problem space. Covering the "noisy" region near the linear decision boundary requires covering nearly the entire problem space. On the other hand, because the points were drawn initially at random, the incorrectly classified disjuncts are more likely to be close to or within a given ϵ uncertainty radius for 3-NN than for the linear model. This is particularly troubling for smaller training set sizes: a great deal of uncertainty is wasted on uninteresting parts of the problem space. It is only at substantial training set sizes that the uncertainty tradeoffs benefit 3-NN, however, such datasets doubtless come at a substantial cost of data acquisition. Note that even with the smallest training sizes as in Figure 4(a), 3-NN and the linear model have comparable accuracy, with roughly 0.95% of examples labeled correctly.
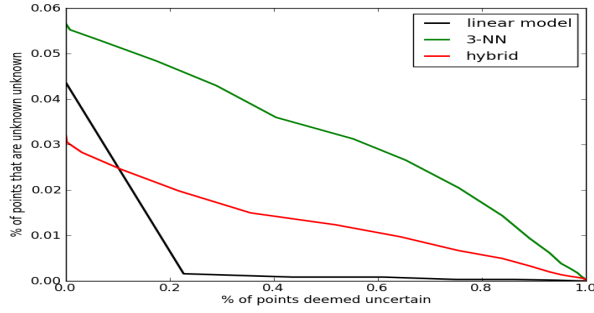
By concentrating the set of training points around the decision boundary[6] would likely result in a much steeper coverage of the incorrectly points, this would be at the consequence of reduced coverage over the incorrectly incorrectly classified disjuncts as the 3-NN coverage would be much sparser in these regions.

As an attempt to combine the fast coverage over the incorrectly classified examples around the decision boundary and the coverage over areas around the misclassified disjuncts, Figure 4 also presents a hybrid model combining both the k-NN model and the linear predictor shown in Figure 3. In this hybrid model, the component with the greatest degree of label certainty is used for making a label prediction. This hybrid approach illustrates yet another characteristic performance- fast initial coverage of mistakes near the decision boundary, yet an overly broad uncertainty region as ϵ increases. Increasing the number of 3-NN training points clearly improves the cost tradeoffs, however, the amount of data acquisition required may be prohibitive. What is needed is a way to improve the random selection of a nearest neighbors model. By choosing only a few examples from in or near the disjunctive subregions that are being misclassified, data acquisition costs could be kept low, while still offering very good unknown coverage. Additionally, because so many probes aren't wasted in areas of the problem space without any problems, a model's manager need not be worried about uncertain regions that won't cause problems.

4. BEAT THE MACHINE

[**JOSH:***ensure that the language matches the earlier motivation in secs 2 and 3*] Assessing the in-the-wild performance of any automated classification system can be challenge. Situations with class imbalance and rare disjunctive sub-concepts such as the hate speech classifier presented in Section 1 makes accurate assessment particularly difficult. Traditionally, we would sample from the output decisions and employ humans to verify the correctness of the classifications. Using these judgments we can estimate the error rate. Unfortunately, given our problem characteristics, this process can be woefully inefficient. First, if the classification decisions are relatively accurate, then most of the results
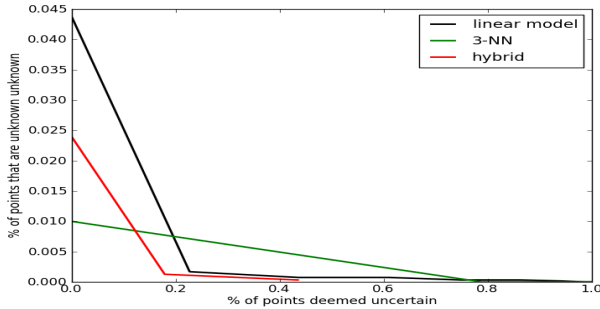
---

[4]Here, we set $k = 3$.

[5]Here, the data was generated according to $y = mx + b + \eta$, where $\eta \sim \mathcal{N}(0, \sigma)$ for some σ.

[6]Such as a dataset likely gathered through active learning.

(a) 150 k-NN Training Examples
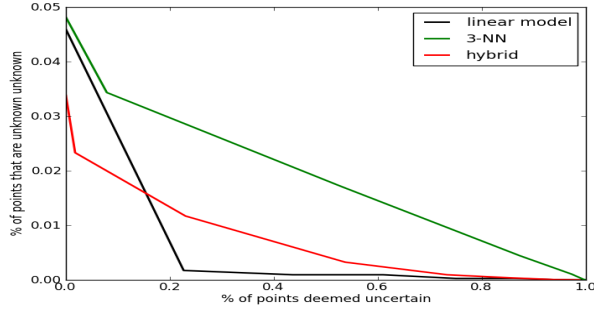
(b) 1,500 k-NN Training Examples

(c) 15,000 k-NN Training Examples

Figure 4: The tradeoffs between UUs and problem uncertainty for a linear model, KNN, and a hybrid model for differing KNN training sizes

will be accurate, and without intelligent sampling, humans will encounter errors very infrequently. Second, if there is substantial class imbalance, most of the encountered errors would be misclassifications of examples truly of the majority class into the minority. This is problematic since in significantly imbalanced classification problems, the minority class must necessarily incur a far greater mistake cost— as in the case of hate speech, this is what is being predicted. Third, if the problem space has rare disjunctive sub-concepts, identification may be particularly tricky— chances of occurrence may be $1 : 1,000,000$ or less. In these situations, it can become quite difficult to identify misclassifications of examples truly in the minority class.

EXAMPLE 1. *Consider the case of identifying pages with hate speech content. In reality, less than 0.1% of the pages on the Internet contain such content. If we have a relatively accurate classifier, with 95% error rate on each class, it becomes very difficult to identify misclassified pages that contain hate speech. In a random sample, most of the pages are correctly classified as benign. To find one "false negative" (the severe error: hate speech passing as benign) we will have to inspect approximately $20,000$ pages (and in the process would find around $1,000$ false positives). This is echoed in the performance of the "random labeling" used to generate the k-NN model presented in Figure 4.* □

It is tempting to consider such problems inconsequential. However, when such a system is used to filter billions of pages, such "relatively infrequent" errors become frequent in absolute numbers. Furthermore, even isolated, "outlier" cases can cause significant damage, for example, to the public image of a company that accidentally supports a site containing such content through advertising. Unknown unknowns may be particularly damaging; client's expectations haven't been properly managed, and detailed contingencies are unlikely to exist.

Instead of passively waiting for such unknown errors to "emerge" we can instead actively seek to find them. In a sense, this is similar to "white hat" hackers that are hired by companies to find vulnerabilities and break into their own security systems. In our case, human workers are asked to submit pages that will "beat" our classifier.

4.1 BTM Task Design

To describe the design of Beat the Machine, we now will walk through several designs of increasing sophistication, building up the ideas by focusing on challenges and subsequently more sophisticated designs.

Design 1: Let's start with a straightforward idea: Ask humans to find cases that "beat the machine"—the users would submit URLs that they believed would be incorrectly classified by the current classification model. To spur engagement, a user would receive a nominal payment for just submitting the URLs, and then she would receive a significant bonus payment for every URL that was misclassified. (In the implementation, the nominal payment was 1 cent per 5 URLs, and the payment per misclassified URL was a maximum of 50 cents.) To judge the misclassification, we asked other (trusted) humans to classify these URLs, and then to determine whether the URL beat the machine, we compared the outcome of the trusted human classification with the outcome of the machine model. To avoid certain issues of gaming, the BTM workers were recruited through Amazon Mechanical Turk, and the trusted human judges were recruited and trained through oDesk for the fully automated system, and were student interns using a separate system for the experimental evaluation below. Unfortunately, this simple design was not as effective as we would have liked, for a variety of reasons.

The first, and most obvious, problem that we encountered was the lack of interactivity. The workers could easily submit URLs that would break the model, but then they had to wait for other humans to inspect the results, in order to assess whether they had succeeded. This process would take from a few minutes to a few hours. The delay made the task opaque to the players of the BTM game, as they did not know if they were "playing the game" well or not.

Adding immediate classification feedback: To resolve (partially) the lack of interactivity, we augmented the
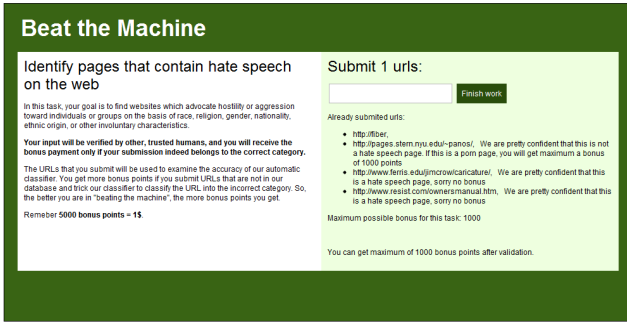
Figure 5: A screen-shot of the BTM interface on Mechanical Turk.

system to classify URLs on the fly, and give immediate feedback to the humans about the classifier outcome. (For example "The machine believes that this URL contains hate speech. Do you believe that this is correct?") The BTM player could then decide whether the URL was indeed a misclassification case and submit it for further consideration. Upon submission, the user received provisional bonus points that correspond to a cash reward. The bonus points became permanent and the worker was paid immediately after inspection and verification of the submitted content by the human judges.

Unfortunately, this design did not provide the proper incentives. Players found it much easier to locate pages from the majority class (e.g., pages without any hate speech content) that would be misclassified as containing hate speech. So, instead of locating the desired, severe infrequent errors, we received the type of errors that we could find more easily by observing the positive classifications. (Recall that due to the class imbalance, most of the observed errors would be good pages being classified as containing hate speech.) As described above, we are particularly interested in finding pages that contain hate speech but are incorrectly classified as benign. (And especially, among these, the "unknown unknowns.") Furthermore, we experienced a significant number of cheating attempts where users were submitting random URLs and always insisting that the content is different than the classification decisions, even though the classifier was correct.

Segmenting the task by class: To deal with these problems, we split the task into two subtasks: (1) Seek pages in the minority class that are misclassified in the majority class (i.e., pages that contain offensive content but are classified as benign), and (2) seek pages with benign content that would be classified as offensive. This segmentation simplified the overall design and made the task easier for participants to understand. Moreover, it allowed us to quickly reject submissions that were of no interest. For example, if we are asking for misclassified hate speech pages, we can quickly reject pages that our classifier unambiguously classifies as hate speech. (In the original design, users had the incentive to mark these as "non-hate-speech" hoping that the human judge would accept their judgments.) Figure 5 shows the (simple) task interface.

Expanding the incentives: In the final design (for this paper) we also improved the incentive structure by rewarding differently users that discover "big mistakes" (the "unknown unknowns") and those that discover the "small mistakes" (the "known unknowns"). Instead of giving a constant bonus to the player for a misclassified URL, we reward mis-

classifications proportionally to the confidence of the classifier. Examples submitted withing the ϵ uncertainty boundary of a model are given a the reward is small. This was a known unknown. On the other hand, if the model is very confident in its decision (i.e., a classification confidence close to 100%), but the decision is incorrect, then the BTM system gives the highest possible bonus to the worker.[7] If the confidence was lower, say 75%, then the reward was proportionally smaller. We also reward players that provide examples for which the model was correct but uncertain: if the model predicted that the page is 60% likely to contain hate speech, and the page indeed contained hate speech, the user received a small bonus.

[**JOSH:***reword the last part to better incorporate the strict definition of UU?*]

5. EXPERIMENTAL STUDIES

To provide a first experimental evaluation of BTM, we asked two questions:

- Does BTM identify errors efficiently?

- Can we use the discovered errors to improve the models?

For our experiments, we used the BTM system to challenge two classification systems. One for detecting pages with hate speech, and one for detecting pages with adult content. We ran the systems with the configuration details described in the previous section (1 cent for the base task, 50 cents maximum payment for a URL that generates an error).

Comparison with stratified random testing: For the two systems, we compared BTM with the usual quality assurance process of examining the output of the classifier to identify errors. Examining a uniform random sample of the output is particularly uninformative, as the classifiers are quite accurate and the distributions are quite unbalanced, and so the vast majority of cases are correctly classified and not objectionable. Therefore, standard procedure is to examine a random sample, stratified by the model's confidence score. Specifically, the range of confidence scores [0,1] was divided into k equal-width bins. A set of N URLs for testing was sampled randomly, with $\frac{N}{k}$ from each bin. This stratification is used because it generally finds more errors, because it over-samples the URLs for which the models have low confidence (and are likely to be wrong). However, the discovered errors are likely to be "known unknowns."

For the adult classifier, the human workers identified errors in 16% of the inspected cases (*much* higher than the natural error rate of the classifier). In contrast, using BTM, more than 25% of the submitted cases generated an error (a 56% increase). The corresponding statistics for hate speech favored BTM even more strongly: workers identified errors only in 9% of the inspections for stratified random sampling, but they identified errors in 27% of the URLs with BTM (three times as many). [**FOSTER:***Insert a projection of the rate of finding UUs for (1) random sampling, and (2) if the stratification were all on the top bucket. Add these estimations to the bar charts.*] These results indicate that the BTM process is indeed more efficient than

---

[7]In our particular implementation, the highest bonus is worth 1000 points, or 50 cents.

the standard evaluation procedure in identifying problematic cases. It should be noted that we could increase the "efficiency" of the non-BTM procedure by simply sampling more from the low-confidence cases. However, this would directly reduce the number of "unknown unknowns" discovered. At the extreme, the largest number of errors would be found by sampling only in the low-confidence region. All the errors found would then be known unknowns. So, let's now consider the effect of BTM on the severity of the errors found.

Comparing the severity of errors: Figure 6(a) and 6(b) show the distribution of modeling mistakes identified for hate speech and adult content tasks, respectively. A consistent behavior is observed for both categories: BTM identifies a significantly larger number of severe misses—the unknown unknowns. Within the errors identified by BTM, 25% were cases of high severity; the model was confident that it was making the correct decision (classifying the content as benign, with 100% confidence), but in reality the decision was incorrect. So, not only does BTM identify a larger number problematic cases than the stratified testing, but also a significant number of these cases were unknown unknowns: cases that would be missed and without a very unpleasant event (possibly a catastrophe), we never would know that we missed them. In contrast, and by now as expected, most of the identified errors for the stratified random sampling were near misses that occur near the decision boundary.

Learning from identified errors: A natural question to ask is if the cases found by BTM seem to be isolated outliers, or whether they seem to be regularities that can be modeled. To this end we ran the following experiment: We attempted to learn a model that would classify positive and negative examples from amongst the BTM-identified cases.[8] Internal consistency in the identified errors would suggest that these cases are not outliers, but rather constitute parts of the space where the model fails systematically (potentially without being aware of the failures).

Figure 7 shows the results of this process. The "btm only" line shows the quality of the model built and tested using the error cases identified by the BTM process. The "student only" line shows the quality of the model built and tested using examples gathered through stratified random sampling (the pages selected through random sampling were inspected by students, hence the name). Both the btm-only and student-only lines show quality measurements computed via cross-validation. Importantly, the results show that the quality of both models is fairly high. This illustrates that there is consistency and internal coherence in these sets pages. The fact that the BTM model can reach high levels of accuracy indicates that BTM indeed identifies systematic errors, and not just disparate outliers. However, note the difference between the quality that is achievable by training with the two different data sets. The comparatively lower quality of the random sampling model illustrates that these pages are inherently more difficult to learn from; this is consistent with our discussion above that the discovery via stratified random sampling (DVSRS) focuses on the ambiguous cases (those that the current model is uncertain about), while BTM discovers incorrectly classified areas of the space that have been systematically ignored.

We also can examine whether the two approaches (DVSRS

---

[8]That is, false negatives and false positives from model being considered, respectively

(a) Hate Speech


(b) Adult Content
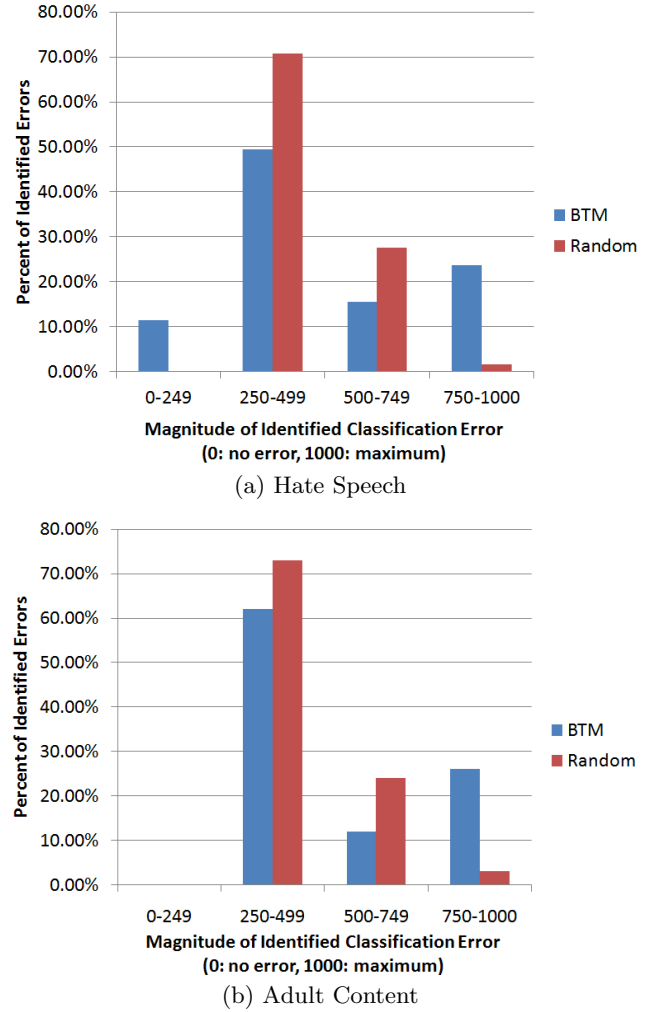
Figure 6: Distributions of the magnitude of the identified mistakes in the predictive model's output by BTM and by random sampling for two ad safety tasks. Each bar indicates the percentage of successfully identified mistakes that reside in the associated score range.

and BTM) identify sets of similar examples, or whether each of them identifies something completely different. For that, we tested the performance of BTM using the examples from DVSRS ("student") and vice versa. The results indicate that there is little cross-consistency between the models. What we discover using BTM has little effectiveness for classifying the error cases identified through DVSRS, and vice versa. This finding indicates that BTM reveals errors in parts of the space unexplored by DVSRS.

BTM and DVSRS seem to be different processes, capable of identifying different types of errors. Each of these has its place in the evaluation and improvement of automatic models. DVSRS identifies primarily cases where the model already knows that it is not confident. The results show that even if the DVSRS were stratified only on the "unknown" region, it still would not identify nearly as many unknown unknowns as Beat the Machine. The BTM process, through its game-like structure and probing nature, encourages the
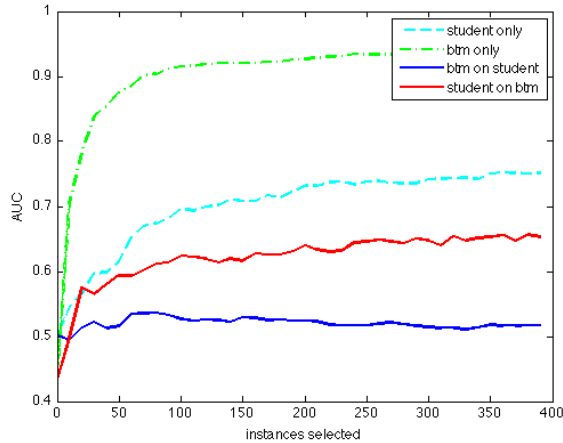
Figure 7: Learning curves generated by the models using cross-validation (BTM and student lines), and then use as test case for BTM the errors identified by random sampling (BTM on students), and vice versa (students on BTM).

discovery of unknown problems in the model. The fact that humans can easily find challenging cases for the automatic models, when being themselves confronted with this challenge, also indicates that human expertise and curiosity can improve even very accurate automatic models.

6. CURRENT AND FUTURE RESEARCH

We presented the problem of "unknown unknowns" in the setting of predictive modeling and explored the design of the *Beat the Machine* process for directly integrating humans into testing automatic decision models for vulnerabilities. Our results suggest that BTM is especially good in identifying cases where the model fails, while being confident that it is correct. It is naturally interesting to examine how to best use knowledge of such vulnerabilities to improve the automatic decisions models.

Vulnerability testing is common in areas of computer security, where "white hat" hackers with the appropriate expertise try to expose vulnerabilities in the security infrastructure of a firm. In our setting, we see that even lay users can easily find unknown holes in automatic decision models that test very well in "standard" tests, and show high classification performance when measured with the traditional, usual metrics (accuracy, AUC, etc). Thus, builders of automatic decision models should take extra care when using these traditional metrics for evaluations.

In our live deployment, untrained humans, with the appropriate incentives, were able to "beat the machine" seemingly easily, and discover a large number of vulnerabilities. This is, of course, useful by itself: the "unknown unknowns" become "known unknowns" and we can prepare to deal with these cases. But the key question for future research is also: how can we best incorporate such knowledge so that both "unknown unknowns" and "known unknowns" become "known knowns." It turns out that building predictive models in the BTM setting is a very complicated problem. For example, oversampling cases where a model makes big mistakes can be catastrophic for learning (think simply about oversampling

outliers in a linear regression). On the other hand, techniques like boosting [7] have gotten tremendous advantage by overweighting cases where the current model is incorrect, and may aid future work. However, the benefit of being able to simultaneously explore a model's unknowns and offer robust model improvement make taking on such challenges worth while.

Acknowledgements

7. REFERENCES

[1] J. Attenberg and F. Provost. Inactive Learning? Difficulties Employing Active Learning in Practice. *SIGKDD Explorations*, 12(2):36–41, 2010.

[2] J. Attenberg and F. Provost. Why label when you can search? strategies for applying human resources to build classification models under extreme class imbalance. In *KDD*, 2010.

[3] C. Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16(1):41–46, January 1970.

[4] C. K. Chow. An Optimum Character Recognition System Using Decision Functions. *IEEE Transactions on Electronic Computers*, (4):247–254, December 1957.

[5] P. Domingos. Metacost: A general method for making classifiers cost-sensitive. In *KDD '99*, 1999.

[6] C. Elkan. The Foundations of Cost-Sensitive Learning. In *IJCAI*, pages 973–978, 2001.

[7] Y. Freund and R. E. Schapire. A short introduction to boosting, 1999.

[8] R. Herbei and M. H. Wegkamp. Classification with reject option. *Can J Statistics*, 34(4):709–721, 2006.

[9] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *SIGIR '94*, pages 3–12, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[10] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

[11] V. Raykar, S. Yu, L. Zhao, A. Jerebko, C. Florin, G. Valadez, L. Bogoni, and L. Moy. Supervised Learning from Multiple Experts: Whom to trust when everyone lies a bit. In *ICML*, pages 889–896. ACM, 2009.

[12] R. Reiter. On closed world data bases. In *Logic and Data Bases*, pages 55–76, 1977.

[13] B. Settles. Active learning literature survey, 2010.

[14] V. S. Sheng, F. Provost, and P. G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *KDD '08*, 2008.

[15] K. VanLehn. Analogy events: How examples are used during problem solving. *Cognitive Science*, 22(3):347–388, 1998.

[16] G. M. Weiss. The impact of small disjuncts on classifier learning. In R. Stahlbock, S. F. Crone, and S. Lessmann, editors, *Data Mining*, volume 8, chapter 9, pages 193–226. Springer US, Boston, MA, 2010.

[17] P. Winston. Learning structural descriptions from examples. 1970.