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

JOSHUA ATTENBERG, Etsy
PANOS IPEIROTIS, New York University
FOSTER PROVOST, New York University

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, 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 may 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 predictive models, and indeed, they identify many more errors where the machine is (wrongly) 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." Beat the Machine has been deployed at industrial scale by several companies. The main impact has been that firms are changing their perspective on and practice of evaluating predictive models.

"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

This paper discusses an important issue for researchers and practitioners to consider, that has received little if any attention in the literature, but becomes quite apparent when applying machine learning techniques in practice. How can we know about the vulnerabilities of a model that we have created, and that seems to perform well according to standardized evaluation measures?[1]

Let's consider an example that we will use throughout the paper. We worked with a firm to build systems for identifying web pages that contain instances of objectionable content, such as "hate speech" (e.g., racist content, antisemitism, and so on). The classi-

[1]For example, using area-under-the-ROC-curve, estimated using cross-validation.

This research has been supported by a George Kellner Faculty Fellowship, a Google Focused Award, an NEC Faculty Fellowship, and an Andre Meyer Faculty Fellowship. We also thank the Moore and Sloan Foundations for their support of the Moore-Sloan Data Science Environment at NYU.

DOI: http://dx.doi.org/10.1145/0000000.0000000

fication is based on models that take web pages as input and produce as output a score that can be used to block certain pages in the advertising ecosystem (or alternatively to produce reports on exposure). The firm would like to use the 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 predictive model and decision system? Unfortunately for applying machine learning, there exists no representative, benchmark, “gold standard” corpus of hate speech, a trait common to many real-world problems. Traditional evaluation and training methods make an implicit *closed-world assumption* [Reiter 1977]. In logical systems, the closed-world assumption is that the only answers to a query Q are those that are actually in the database. In the context of predictive modeling, the closed-world assumption is that our labeled data are sufficient to give us a satisfactory estimation of model performance. Effectively, machine learning methods make the assumption that regularities that have no or insufficient representation in the training data essentially do not exist. Unfortunately, such an assumption is dangerously naive in applications with limited labeled training data, small disjuncts [Weiss 2010], and/or possibly unknown selection biases. In these cases, we face the following problem: the model often cannot estimate properly its own performance on unseen data, and is vulnerable to the problem of *unknown unknowns*.

We will elaborate throughout the paper, but in a nutshell the problem is the following: when building predictive models, we assume that the available data are representative and therefore we can use them safely to train and to evaluate our algorithms. Furthermore, and crucially, we assume that the models can estimate properly their own level of performance and report back accurate confidence metrics for different parts of the space. Assuming that we can know how well our algorithms are performing, we can work to improve the models, and/or improve the performance in regions of lower confidence in our predictions, and/or act based on this confidence. We have faith in our training data, and we focus on what we can know—including *knowing* what we don’t know (e.g., regions of uncertainty in our learned model’s behavior).

The problem is that, for various reasons, processes that produce training data can completely miss important regions of the space. Consider our case of identifying hate speech online: The category “hate speech” is relatively rare on the Web (fortunately) and extremely varied—a “disjunctive concept” [Weiss 2010]. If a certain type of hate speech is not included in the training data, the resultant learned classifier is likely to (i) classify it as not-hate-speech, and worse, (ii) do so with high confidence. Such a case is an *unknown unknown*—the classifier does not “know” this sort of hate speech, and it doesn’t know that it doesn’t know it. So, the unknown unknowns belong to the regions in the space where the model estimates the misclassification cost to be low, while in reality the misclassification cost is high.[2]

Standard model evaluations may completely miss such problems: e.g., cross-validated error rates and area-under-the-curve values for such models may look nearly perfect. However, the scientist would be incorrect to report to the application stakeholders that the models are essentially perfect. It is critical to emphasize that such evaluations

[2]We will use the more general notion of misclassification cost rather than simply misclassification error, but the reader can interpret this as error without missing the main idea. The reason to generalize to cost is especially apparent for multiclass classification, where misclassification errors among different classes have very different costs. Then we are most interested in avoiding specific sorts of unknown-unknown errors—those that have high cost.

only consider the *known unknowns*. However, stakeholders of ML models need to also consider the possible impact of the unknown unknowns—for example, being blindsided by a client who discovers a completely unknown error, or suffering an embarassing PR disaster if the vulnerability of the model is exposed publicly by a third party or competitor. Finding the unknown unknowns, in a predictive modeling context, is the focus of our paper.

We present a novel framework for thinking about errors of predictive models that highlights the unknown unknowns (plenty of work has focused on the known unknowns). We then present a game-structured system, called *Beat the Machine*, that takes advantage of crowdsourcing to help us identify the unknown unknowns. The system is fully implemented at industrial scale; we discuss several design iterations that each presented hurdles to overcome and led to the current system. Finally we present experimental results demonstrating that on real problems (including finding hate speech pages), explicitly focusing on finding unknown unknowns (with Beat the Machine) indeed finds instances (e.g., hate speech pages) that were completely missed by the prior system, are systematic errors (rather than just isolated outliers), and are systematically different from the prior training data.

Our work builds on and extends our earlier paper [Attenberg et al. 2011]. Compared to the prior version, the present paper also includes the following material:

— We formally define the notion of unknown unknowns, using the concept of estimated and actual misclassification costs. We also examine typical prediction settings where our BTM approach is expected to be beneficial,
— We present a new design for the BTM system. This new design encourages workers to find diverse errors in the model, as opposed to being rewarded for submitting repeated instances of the same vulnerability.
— We describe the deployments of the BTM system in multiple industrial settings.

2. BACKGROUND AND SCOPE

Many businesses, government organizations, and NGOs 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 for 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.

Specifically, our 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. We assume, for simplicity and without loss of generality, that the target is binary; we also assume that the truth would not be in dispute if known.[3]
— 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 in regions of the space where the model is confident about its decisions and provides a very low

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

estimate for misclassification costs. For example, is there a particular sort of hate speech that the model builders did not consider, and therefore the model misses it, while at the same time being confident about the reported decision?

— Finally, there are important classes or subclasses of cases that are very rare, but nevertheless very important.[4] 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 [Provost and Fawcett 2013]. For example, we can task humans to score random or selected subsets of cases. Unfortunately, such a practice has two major drawbacks. First, due to the rarity of the class of interest (e.g., hate speech) it can be costly to find positive examples, especially via random sampling of pages. For example, hate speech represents far less than 0.0001% of the population of (ad supported) web pages, with unusual or distinct forms of hate speech being far rarer still. Thus we would have to invest in labeling more than a million randomly selected 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 [Sheng et al. 2008; Raykar et al. 2009].

In practice, we often turn to particular techniques 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 [Settles 2012]. A large number of these strategies (uncertainty sampling, sampling near the separating hyperplane, query-by-committee, query-by-bagging, and others) have one thing in common: they are biased *against* looking for cases where the model is certain[5] or for cases that are not expected to improve learning performance. The strategy makes sense, as these are not areas is where we would expect 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 [Winston 1970] and for human learning [VanLehn 1998].

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 cannot estimate them well (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.

In what follows, 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 that uses human workers to help find the unknown unknowns. Our Beat the Machine (BTM)

[4]The work also applies to applications where there are important classes or subclasses that are not rare in the population, but are rare or not represented at all in the training data, due to sampling processes.

[5]An exception is active learning work looking at improving class probability estimation [Saar-Tsechansky and Provost 2004], which could select cases where the model estimates a very high or low probability. However, even there, cases will only be selected if there is uncertainty in the estimation of the probabilities—those where the model or modeling procedure is certain of the probabilities will not be selected.

system combines a game-like setup with incentives designed to elicit cases where the model is confident but wrong. Specifically, BTM rewards workers who discover cases that cause the system to fail. The reward increases with the magnitude of the failure. The setting makes the system behave like a game, encouraging steady and accurate user participation. We describe our experiences with the use of the BTM system, in a setting for identifying web pages with offensive content on the Internet. (In Section 6, we also describe how we used the same system in a variety of other settings.) We show that the BTM setting discovers error cases that are inherently different from 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.

3. UNKNOWN UNKNOWNS

3.1. Expected and Actual Misclassification Cost

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 $\mathcal{X}$. In classification settings, x has a "true" label, $\bar{y}$ from some set of possible labels Y. The classification task is to construct some predictive model, $f(x)$, that can estimate a label for each incoming example such that the estimated label $\hat{y}$ mirrors the (usually hidden) true label, $\bar{y}$, as closely as possible. There is a cost c_{ij} for a misclassification decision, when we classify an example from the true category $\bar{y} = y_i$ into a category $\hat{y} = y_j$. In this work, we are concerned only with models that output a posterior probability estimate over the set of available labels, that is, $f(x) = p(y|x) = \langle p_1, \ldots, p_n \rangle$; such models, through the probability estimates, effectively also report the estimated misclassification cost of each example:

$$ExpCost(x) = \sum_{i,j} p_i \cdot p_j \cdot c_{ij} \qquad MinCost(x) = \min_j \sum_i p_i \cdot c_{ij} \tag{1}$$

Such probabilities and costs can then be used to select a preferred label, for instance by choosing the $\hat{y}$ with the highest probability, or in the cost-sensitive setting, choosing the label with the least expected cost [Elkan 2001]. 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, e.g., [Domingos 1999]).

3.2. Known Knowns, Known Unknowns, Unknown Unknowns, and Unknown Knowns

Notice that Equation 1 provides *estimates* of the misclassification cost. The *actual* misclassification cost can be different and depends on the accuracy of the model and the generated posterior probability estimates. The difference between the estimated and the actual misclassification costs gives rise to four different classification scenarios, as depicted in Figure 1.

Along the diagonal, we have examples for which the estimates of the model in terms of misclassification cost are in line with the actual misclassification costs. In the lower-left corner we have the examples that the model can classify correctly *and* is confident about the reported decisions. These are the *known knowns*. In the upper-right corner,

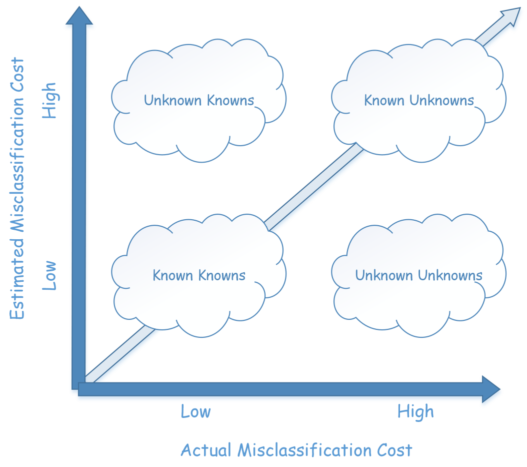


Fig. 1. The decisions made by a predictive model can be broadly separated into four conceptual regions: (i) The *known knowns*, which are the examples for which the model is mostly correct and also returns a low expected misclassification cost (i.e., is also confident of being correct); (ii) The *known unknowns*, which are the examples for which the model is often wrong but also anticipates these errors by returning a high expected misclassification cost for these decisions; (iii) The *unknown knowns*, which are the examples for which the model is often correct but returns high expected misclassification costs; and (iv) The *unknown uknowns*, which are the examples for which the model is wrong but is wrongly confident that it is correct.

we have the cases of *known unknowns*: these are the cases where the model fails often but is also aware of the problem.

Definition 3.1 (*Known Unknown*). Let $X \subset \mathcal{X}$ be a region of the problem space and x be a randomly chosen example from X. Let $ExpCost(x)$ be the expected misclassification cost of an example from X, and $Cost(x)$ be the actual misclassification cost of an example from X. A *known unknown* is an example for which the actual misclassification cost $Cost(x)$ is high *and* the estimated misclassification cost $ExpCost(x)$ is also high. □

Known unknowns as described in Definition 3.1 correspond to a commonly occurring notion in machine learning. The X region corresponds to a region of high expected misclassification cost, an area where the predictive model is unsure of itself, and where mistakes are likely. 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 [Lewis and Gale 1994].

In the context of classification, known unknowns are those cases for which errors are expected based on the probability estimates of the classification. These are cases where it may be less costly to "reject" than to make a risky label prediction. Classification with a "reject option" is an extension of traditional classification where in addition to labeling each example with some $\hat{y} \in Y$, 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 [Chow 1957; Chow 1970]. 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.

While it is important to understand the mistakes that a 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 above. While deployed in a production setting, this model is likely to encounter examples with a high estimated misclassification cost.[6] 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 correspondingly low estimated cost, and be wrong. Such examples are the unknown unknowns.

Definition 3.2 (*Unknown Unknown*). Following Definition 3.1, an example x' is an *unknown unknown* if x' is outside the region of model uncertainty and therefore has *low* estimated misclassification cost $ExpCost(x')$, but the actual misclassification cost $Cost(x')$ is in fact *high*. In other words, the example is misclassified, but the classifier (wrongly) has a high certainty that it was correctly classified. □

Definition 3.2 codifies the notion of an unknown unknown. Intuitively, these are examples for which the model is certain that it assigns the correct label—for instance, examples that are distant from any decision boundary—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[7]—the motivating case is disjunctive sub-regions of the problem space [Weiss 2010]. 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 [Attenberg and Provost 2010]. However, their non-negligible absolute size and prevalence in many real-world problems makes them a genuine risk. Furthermore, very often the processes used to generate training sets in real applications are not random sampling at all; they are biased based on pragmatic constraints [Perlich et al. 2014]. Such biased selection can then miss such islands completely.

Figure 2 presents a fairly typical classification scenario that might be impacted by unknown unknowns. On the left, 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 a region of uncertainty, ϵ-wide, where mistakes are *known* to occur. This represents a typical post-training understanding of the problem space: data is gathered by some 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 right 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.

Finally, from Figure 1, we can see that there is one more type of example: the "*unknown knowns*." These are the cases where the model is incorrectly pessimistic: the model reports low confidence and high estimated cost for the examples in these region(s). However, in reality the model generates mostly correct predictions. Such cases are typically easier to manage but may still cause problems in the stakeholders: if the region generates many rejects (with an associated inspection and intervention costs), then the model will be perceived as being overly cautious and potentially inefficient. Despite the fact that it is a promising direction, identifying and dealing with cases of

[6]For instance, a encyclopedia entry discussing racial issues or a history of racism.
[7]For instance, due to erroneous labeling, signal degradation, or non-pathological difficulties in data collection.

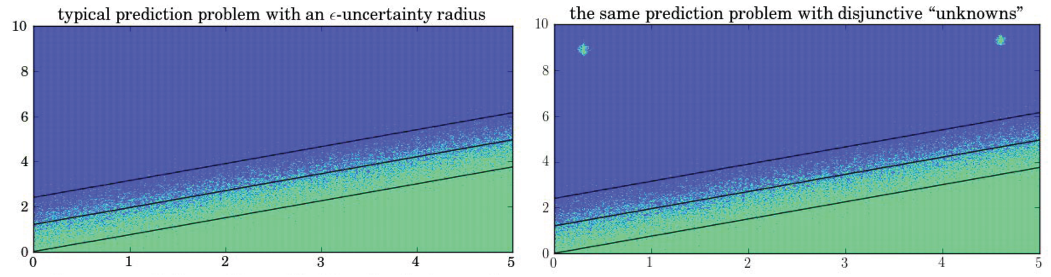


Fig. 2. A typical classification setting. On the the left we see the decision boundary that minimizes the prediction error of a inseparable training set. Additionally, we see the ϵ-radius around the classifier where mistakes are thought to occur. On the right, we see the same classifier with the inclusion of small, disjunctive "unknowns," presenting mistakes that occur well outside a model's region of uncertainty.

unknown knowns is beyond the scope of the current paper, and we leave their study for future work.

3.3. Active Learning and Beat-the-Machine

A natural question that arises in this setting is the connection of the "known unknowns" and "unknown unknowns" with active learning. From a 10,000 feet view, we could say that approaches for active learning focus on the cases of "known unknowns." Although this is a well-known and established heuristic for active learning, it would be wrong to pigeonhole the whole field of active learning using this characterization. There are many active learning techniques that do not use this uncertainty heuristic, and instead examine regions that are likely to modify the decisions of the machine learning model, prefer to focus on the dense regions of the space, and so on [Settles 2012]. In general these other active learning methods still tend to focus, albeit indirectly and not exclusively, on the known unknowns.

A better way to describe the difference of "Beat the Machine" (described in detail next) with active learning is to focus on who is the agent directing the selection of the data for evaluation:

— In active learning, the choice of the data points that will be evaluated by human "oracles" is directed by an algorithm. The algorithm, by definition, picks the data points from a "closed world" sample of potential training data. Hence active learning adopts a "closed world, machine-pushes-to-humans" model, where the algorithm chooses the data points that will be evaluated by humans.
— In Beat the Machine, the choice of the data points to be evaluated by human oracles is done by humans. The humans are not restricted in their search space, and hence pick data points from an "open world" setting. Then, the algorithm decides whether the submitted data points need to be investigated further. Hence, BTM adopts an "open world, human-pushes-to-machine" model, where humans discover the data points and then a machine filters the submissions and decides which data points need to be further evaluated by humans.

4. BEAT THE MACHINE

Assessing the in-the-wild performance of any automated classification system can be a challenge. Situations with class imbalance and rare disjunctive sub-concepts such as the hate speech classifier introduced in Section 1 make accurate assessment particularly difficult and lead to the existence of unknown unknowns. 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 in different parts of the classification space. Unfortunately, given our problem characteristics, the process can be woefully inefficient. First, if the classification decisions are relatively accurate, then most of the results 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 majority-class examples into the minority. This is problematic, since in significantly imbalanced classification problems, errors on minority-class examples generally incur a far greater misclassification cost—as in the case of hate speech. 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 even much less. In these situations, it can become quite difficult to identify misclassifications of examples whose true class is the minority class.

Example 4.1. *C*onsider the case of identifying pages with hate speech content. If we have a relatively accurate classifier, with 95% accuracy 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. Even in the unrealistically generous case that 0.1% of the pages on the Internet contain such objectionable content, 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 we would incur around $1,000$ false positives). □

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 one-off 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 careful contingency plans are unlikely to exist.

Instead of passively waiting for such unknown errors to "emerge" we can instead actively seek to find them. Beat the Machine engages human intelligence, accessed through crowdsourcing, to identify these unknown unknown cases. In a sense, this is similar to "white hat" hackers who 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. Beat the Machine Task Design

The Beat the Machine (BTM) system has evolved over time. We will walk through several design iterations, showing the challenges that emerged with each new design, and how these challenges were addressed.

Design 1, Initial version: Let's start with a straightforward idea: Ask humans to find the unknown unknowns, i.e., the cases that "beat the machine." Users submit URLs that they believe will be incorrectly classified by the current classification model. To spur engagement, a user receives a nominal payment for just submitting the URLs, and then she receives 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 20 cents.)

Of course there is an obvious problem: how could such a system tell that the URL indeed beats the machine? The whole point is to find cases that the system does not know that it gets wrong! To judge misclassification, we task another set of (trusted) humans to classify these URLs. Then, to determine whether the URL beats the machine, we compare the classification of the trusted set of humans with the outcome of the machine model.[8]

Unfortunately, this simple design was not as effective as we would have liked, for a variety of reasons. The first, and most obvious, problem 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 can 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 good at "playing the game" or not. This led to short user engagement, and high rates of abandonment.

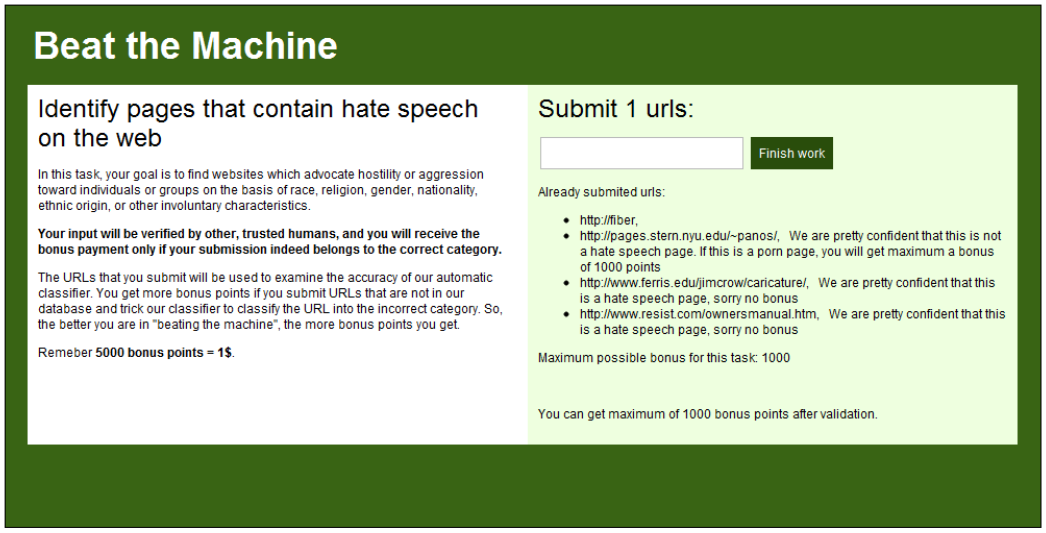
Design 2, Adding immediate classification feedback: To resolve (partially) the lack of interactivity, we augmented the system to classify submitted URLs on the fly, and give immediate feedback to the humans about the classifier outcome. For example "The machine believes that this URL contains hate speech. Do you believe that this is correct?" The BTM player can then decide whether the URL is indeed a misclassification case and submit it for further consideration. Upon submission, the user receives provisional bonus points that correspond to a cash reward. The bonus points become permanent, and the worker is paid, immediately after the inspection and verification of the submitted content by the human judges.

Unfortunately, this design still did not provide the proper incentives. Players typically found it much easier to locate pages from the majority class misclassified into the minority (e.g., pages without any hate speech content that are misclassified as containing hate speech). So, instead of locating the desired, high-cost errors, we received the type of errors that can be found more easily by observing the positive classifications. Recall that due to the class imbalance, most of the observed errors are 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 submitted random URLs and insisted that the classification of the content should be different from the model's classification, even though the classifier was indeed correct.

Design 3, Segmenting the task by class: To deal with these problems, we split the task into two subtasks: (a) Seek pages that contain offensive content but are classified as benign and (b) seek pages with benign content that are classified as offensive. This segmentation simplifies the overall design and makes the task easier for participants to understand. Moreover, it allows us to quickly reject submissions that are 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 Designs 1 and 2, users had the incentive to mark these as "non-hate-speech" hoping that the human judge would accept their judgments.) Figure 3 shows the, intentionally simple, task interface.

Design 4, Expanding the incentives: In this design, we improve 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 misclassifications proportionally to the estimated cost, which we infer through the returned probability distribution for the example label. For examples that have high estimated cost, the reward is small: this was a known unknown. On the other hand, if

[8]To avoid certain issues of gaming, the BTM workers are recruited through Amazon Mechanical Turk, and the trusted human judges are recruited and trained through oDesk for the fully automated system. For the experimental evaluation, we used student interns using a separate system.


Beat the Machine

Identify pages that contain hate speech on the web

In this task, your goal is to find websites which advocate hostility or aggression toward individuals or groups on the basis of race, religion, gender, nationality, ethnic origin, or other involuntary characteristics.

Your input will be verified by other, trusted humans, and you will receive the bonus payment only if your submission indeed belongs to the correct category.

The URLs that you submit will be used to examine the accuracy of our automatic classifier. You get more bonus points if you submit URLs that are not in our database and trick our classifier to classify the URL into the incorrect category. So, the better you are in "beating the machine", the more bonus points you get.

Remeber **5000 bonus points = 1$**.

Submit 1 urls:

Already submited urls:

- http://fiber,
- http://pages.stern.nyu.edu/~panos/, We are pretty confident that this is not a hate speech page. If this is a porn page, you will get maximum a bonus of 1000 points
- http://www.ferris.edu/jimcrow/caricature/, We are pretty confident that this is a hate speech page, sorry no bonus
- http://www.resist.com/ownersmanual.htm, We are pretty confident that this is a hate speech page, sorry no bonus

Maximum possible bonus for this task: 1000

You can get maximum of 1000 bonus points after validation.

Fig. 3. A screen-shot of the BTM interface on Mechanical Turk.

the model is very confident in its decision (i.e., estimated cost close to 0), but the decision is incorrect, then the BTM system gives the highest possible bonus to the worker.[9] If the the estimated misclassification cost is higher, then the reward is proportionally smaller. We also reward players who provide examples for which the model is 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 receives a small bonus.

Design 5, Encouraging diversity: Design 4 has most of the incentives in place for users to try to discover unknown uknowns. The key problem is what happens after the discovery of the first unknown uknowns. Users are typically able to understand what is the cause of the failure, and then keep submitting cases that are similar to the previously submitted successes. For example, for the web classification task, users realized that the classifier was trained with English web sites, and therefore adult or hate speech sites written in languages other than English (Japanese, German, Arabic, etc.) are effectively unknown unknowns.[10] Users realize the vulnerability and then keep submitting similar sites, retrieving high rewards for very little effort. However, once we receive the first few submissions of such unknown unknowns, the similar submissions are only technically unknown unknowns. In reality they become effectively "known" unknowns *to the stakeholders*, although for the classification system and BTM they remain unknown uknowns.

To avoid this issue, we altered the setting to include one additional binary classifier: This extra classifier receives the proposed URL from the user (a "probe") and decides whether the submitted probe is similar to a previously submitted probe. This classifier is trained using the previously submitted probes. If the classifier determines that the probe is similar to a previously submitted succesful probe, the user receives a smaller reward compared to the case of submitting a "new" unknown unknown.

[9]For the experiments, the highest bonus per URL was worth 1000 points, or 20 cents.
[10]Since most of the terms in non-English websites were not in the training data, the classifier had very few features to work with, and effectively classified these sites into the majority class.

5. EXPERIMENTAL STUDIES

In Section 3 we defined unknown unknowns and in Section 4 we described a gamified structure that incentivizes humans to identify such unknown unknowns in a predictive model.

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

— Does BTM identify errors efficiently?
— Does BTM identify isolated examples of unknown unknowns, or does it identify systematic unknown-unknown regions in the space?

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 of Design 4, described in the previous section (0.2 cents for the base task, 20 cents maximum payment for a URL that generates an error). In both cases, we challenged the workers to find errors in the minority class (the high-severity errors).

Comparison with stratified random examination: In application domains with highly unbalanced data sets, the standard procedure for quality assurance of models such as these is stratified random examination of model predictions. 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 have human workers examine a random sample, stratified into bins by the model's confidence score. Following the standard procedure, in our evaluation each bin contained the same number of URLs; specifically, the range of expected misclassification costs 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—it over-samples the URLs for which the models have low confidence (and are likely to be wrong).[11] However, the discovered errors are likely to be "known unknowns," in contrast to the errors that we expect BTM to discover.

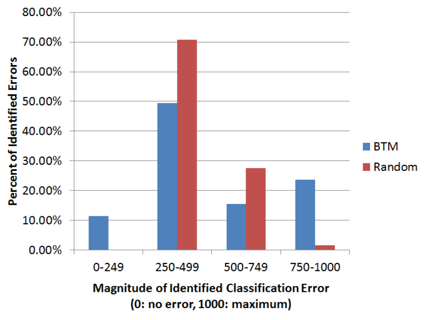
We compared the stratified sampling quality assurance procedure with BTM to examine whether the hypothesis that they discover different types of errors is true. It is important not to see this as a bake-off. Although we will compare identified error rates, it is important to emphasize that the two procedures are complementary and not competing: they are designed to assess different quantities as we explain below.

In our experiment with BTM, we asked users to find pages that contain objectionable content, but would be classified as benign by the classifier.[12] We collected 500 URLs for each of the tasks, and each URL had a non-zero probability assigned by the classifier of being benign. For the adult content task, a total of 25 BTM workers participated, and for the hate speech task, a total of 28 BTM workers participated.

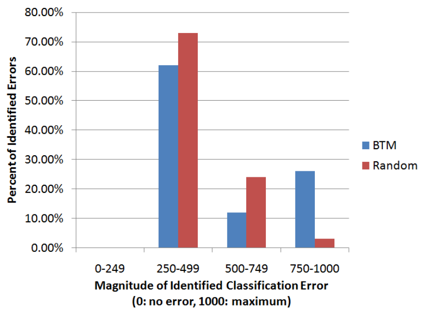
For the adult classifier, the human workers in the stratified examination identified errors in 16% of the inspected cases—*orders of magnitude* higher than the error rate of the classifier when evaluated with examples selected via non-stratified random sampling. In contrast, using BTM, more than 25% of the submitted cases exhibited an error. The corresponding statistics for hate speech favored BTM even more strongly: workers identified errors in only 9% of the inspections for stratified examination.

[11]It also allows the management and data science teams to estimate the calibration of the scoring system, by examining the percentages of positive and negative instances in each bin.

[12]The underlying classifiers were models built using logistic regression, trained in each case with a dataset of approximately 20,000 manually classified URLs. For these experiments, the pages were featurized based on the text and metadata of the page, using standard text-classification methods, with different name spaces for different page segments (e.g., title, body, metadata).



(a) Hate Speech



(b) Adult Content

Fig. 4. Distributions of the magnitude of the identified errors by BTM and by stratified random sampling. Each bar indicates the percentage of successfully identified errors that reside in the associated score range.

Workers identified errors in 27% of the inspected URLs with BTM. On the surface, these results seem to indicate that the BTM process is more efficient than the standard quality assurance procedure in identifying problematic cases. However, note that we could increase the "efficiency" of the non-BTM procedure by simply sampling a larger proportion from the low-confidence predictions. Unfortunately, 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, and none of the errors found would be unknown unknowns.

Comparing the severity of errors: Figures 4(a) and 4(b) show the distribution of errors identified for the hate speech and adult content tasks, respectively. The blue bars show the mistakes identified by BTM; the red bars show the mistakes identified by the stratified evaluation. The number ranges on the horizontal axis show the severity of the errors, in four buckets—how badly the classifier misjudged its classification. The severities range from the least severe on the left (zero is no error), to maximum severity on the right: 1000 means that the classifier was certain of one class and the actual class was the other. The unknown unknowns are on the far right of each graph.

A consistent behavior is observed for both categories: a significantly larger proportion of the BTM errors found are severe misses—the unknown unknowns. Within the errors identified by BTM, 25% were cases of high severity; here the model estimated that it was making the correct decision (classifying the objectionable content as benign, with low expected misclassification cost), but in reality the decision was incorrect.

In sum, BTM identifies a larger number of problematic predictions than the stratified testing. Out of these predictions, BTM also identifies a much larger number of unknown unknowns. These cases would be missed in practice and without an unpleasant event (possibly a catastrophe), the model users would never would know that they had missed them. In contrast, and by now as expected, most of the identified errors for the stratified examination were misses that occur near the decision boundary.

Isolated outliers or systematic regularities? A natural question to ask is if the cases found by BTM seem to be isolated outliers, or whether they seem to be regularities. We framed this question pragmatically, based on the assumption that regularities can be modeled.[13]

Following this reasoning, we ran the following experiment. We attempted to learn a model that would classify positive and negative examples from amongst the BTM-

[13]So therefore we may miss regularities that fall outside the limits of the inductive bias of our modeling procedures.

identified cases. Internal consistency in the identified errors would suggest that these cases are not outliers, but rather that they constitute parts of the space where the model fails systematically (potentially without being aware of the failures). Figure 5 shows the results of this process.

First consider the “btm only” learning curve and the “student only” learning curve, showing 10-fold cross-validated areas under the ROC curve (AUC). The “btm only” curve shows the performance of models built and tested using the error cases identified by the BTM process. The “student only” curve shows the performance of models built and tested using examples gathered through stratified examination (the pages selected by stratified examination were inspected by students for this experiment, hence the name). 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 of identifed errors. The fact that the BTM model can reach high levels of accuracy indicates that BTM indeed identifies systematic regions that contain unknown unknowns, 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 stratified examination 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 examination (DVSRE) 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 (DVSRE and BTM) identify sets of similar examples, or whether each of them identifies something completely different. For that, we tested the performance of BTM-trained models using the examples from DVSRE (“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 DVSRE, and vice versa. This finding indicates that BTM and DVSRE reveal errors in different parts of the space; importantly, BTM finds errors that are systematically different from those found by DVSRE. BTM and DVSRE are different processes, capable of identifying different types of errors. Each of these has its place in the evaluation and improvement of automatic models. DVSRE identifies primarily cases where the model already knows that it is not confident. The results show that even if the DVSRE were stratified only on the “unknown” region, it still would not identify nearly as many unknown unknowns as Beat the Machine.

6. IMPACT IN INDUSTRIAL DEPLOYMENTS

The Beat the Machine design has directly changed the way several companies view and practice the evaluation of predictive systems. In our example domain for this paper, Integral Ad Science,[14] a leading analytics company for online advertisers that does massive-scale webpage classification in the online advertising space, has decided to move beyond traditional system evaluation methods. Traditionally, deployment models had been evaluated using cross-validation, expert model examination, and stratified case examination. Once a prediction system was deployed in practice, evaluation was based on continual stratified examination from the production classification stream. Let’s call this practice “passive testing.” According to the firm’s founding data science team, this work convinced Integral Ad Science that Beat the Machine and other “active testing” practices are vital to understand their predictive models’ performance and alert stakeholders about areas of concern. The most convincing impact is that Integral Ad Science has invested in the industrial development of Beat the Machine, and is pursuing its use across classification tasks (not just objectionable content).

[14]http://integralads.com/

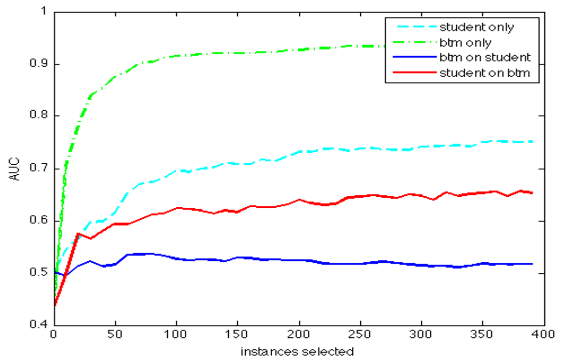


Fig. 5. Learning curves generated by the models first using cross-validation (BTM only and student only), and then each trained on one data set (BTM, student) and tested on the other. Specifically, in the latter case, the student-classified stratified random sampling data were used as test cases for models trained using the BTM-identified cases (BTM on students), and vice versa (students on BTM).

Beat the Machine also has influenced the practice of deploying new machine learning algorithms within oDesk,[15] a firm that runs one of the most popular online labor marketplaces. Before deploying a machine learning algorithm, the algorithm is first tested by asking users to identify cases that are likely to "break" the algorithm. Consider the example of testing an algorithm that automatically classifies posted jobs into categories. To test the system, a set of contractors were hired and asked to submit job descriptions that a human would legitimately and unambiguously classify into one category, but the automatic system would classify into another. One of the interesting side-effects of this practice is that the contractors are typically capable of identifying earlier than the machine learning algorithm shifts in the content of the typical job posting. For example, when a new technology appears, the machine learning system may not have sufficient training data to identify that "Bootstrap" is a Web Design term/skill, and not a statistical sampling technique; contractors are able to identify such cases without the need for a large sample sizes of training data. So, when a new type of task starts emerging in the market (and it cannot be classified properly by the current automated engine) the BTM system is likely to catch this trend early, before it becomes a major issue of user dissatisfaction.

Finally, a BTM-like system has been deployed as part of an image tagging service for Tagasauris.[16] Before BTM, automatic systems together with humans were used to tag images with keywords. Under the new BMT-style design, there is an extra phase where humans examine an image to "challenge" existing tags, with the goal that the newly provided tag will be better and more relevant than the one currently assigned. This design allows for higher quality keywords to be assigned to the images, avoiding cases where only a set of generic, uninteresting keywords are assigned to an image (either by algorithms or by humans).

In general, the main practical impact of the BTM system is the new approach for testing and debugging automatic machine learning models. The technique of rewarding users for locating vulnerabilities and bugs is common in security. However, there is a considerable difference: When dealing with statistical models, merely locating an incorrect classification decision is hardly an event worth rewarding, or even recording.

[15]http://www.odesk.com/
[16]http://www.tagasauris.com/

The BTM system is designed to reward cases that help to identify and avoid failures that would blindside the users of the predictive model.

7. CONCLUSION AND FUTURE WORK

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 severe, unknown vulnerabilities. Our results suggest that BTM is especially good in identifying cases where the model fails, while being confident that it is correct. Several companies have implemented systems based on the ideas of Beat the Machine, based directly on preliminary reports of this research.

Our experience suggests that researchers should devote more study to this sort of system. Presumably, even though we have gone through several design iterations, there is a lot to learn about how to design such systems to work well. As we discussed in the deployment section, in addition to using BTM proper, companies already have been using the ideas in ways slightly different from the exact design we've presented. Further, it is naturally interesting to ask how to best use knowledge of such vulnerabilities to improve the automatic decisions models. To our knowledge, no work has yet studied this. Our preliminary experiments indicate 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 [Schapire 1999] have gotten tremendous advantage by overweighting cases where the current model is incorrect. The potential benefit of being able to simultaneously explore a model's unknowns and offer robust model improvement would be an exciting advance.

REFERENCES

Josh Attenberg, Panagiotis G. Ipeirotis, and Foster J Provost. 2011. Beat the Machine: Challenging Workers to Find the Unknown Unknowns. In *The 3rd Human Computation Workshop (HCOMP 2011)*.

Josh Attenberg and Foster Provost. 2010. Inactive Learning? Difficulties Employing Active Learning in Practice. *SIGKDD Explorations* 12, 2 (2010), 36–41.

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

C. K. Chow. 1957. An optimum character recognition system using decision functions. *IRE Transactions on Electronic Computers* EC-6, 4 (Dec. 1957), 247 –254.

Pedro Domingos. 1999. MetaCost: A General Method for Making Classifiers Cost-sensitive. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 155–164.

Charles Elkan. 2001. The Foundations of Cost-sensitive Learning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 2*. 973–978.

David D. Lewis and William A. Gale. 1994. A Sequential Algorithm for Training Text Classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 3–12.

C. Perlich, B. Dalessandro, T. Raeder, O. Stitelman, and F. Provost. 2014. Machine learning for targeted display advertising: Transfer learning in action. *Machine Learning* 95, 1 (2014), 103–127.

Foster Provost and Tom Fawcett. 2013. *Data Science for Business: What you need to know about data mining and data-analytic thinking*. O'Reilly Media.

Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Anna Jerebko, Charles Florin, Gerardo Hermosillo Valadez, Luca Bogoni, and Linda Moy. 2009. Supervised Learning from Multiple Experts: Whom to Trust when Everyone Lies a Bit. In *Proceedings of the 26th Annual International Conference on Machine Learning*. 889–896.

R. Reiter. 1977. *On Closed World Data Bases*. Technical Report. University of British Columbia, Vancouver, BC, Canada, Canada.

Maytal Saar-Tsechansky and Foster Provost. 2004. Active Sampling for Class Probability Estimation and Ranking. *Machine Learning* 54, 2 (2004), 153–178.

Robert E. Schapire. 1999. A Brief Introduction to Boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*.

Burr Settles. 2012. *Active Learning*. Vol. 6. Morgan & Claypool Publishers.

Victor S. Sheng, Foster Provost, and Panagiotis G. Ipeirotis. 2008. Get Another Label? Improving Data Quality and Data Mining Using Multiple, Noisy Labelers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 614–622.

Kurt VanLehn. 1998. Analogy Events: How Examples are Used During Problem Solving. *Cognitive Science* 22, 3 (1998), 347–388.

Gary M. Weiss. 2010. The Impact of Small Disjuncts on Classifier Learning. In *Data Mining*, Robert Stahlbock, Sven F. Crone, and Stefan Lessmann (Eds.). Annals of Information Systems, Vol. 8. Springer US, 193–226.

Patrick H. Winston. 1970. *Learning Structural Descriptions From Examples*. Technical Report. Massachusetts Institute of Technology.