Beat the Machine: Challenging Workers to Find the Unknown Unknowns

Josh Attenberg
Polytechnic Institute of NYU
Brooklyn, NY
josh@cis.poly.edu

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."

Introduction

"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***

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.

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 unknown 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 (Sheng, Provost, and Ipeirotis 2008; Raykar et al. 2009).

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 (Settles 2010). 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 (Winston 1970) and for human learning (VanLehn 1998).

Unfortunately, although helpful in understanding and improving modeling, 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. 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.

The Design of "Beat the Machine"

Assessing and improving the quality of an automatic classification system is challenging in environments with the characteristics listed above. Tra-

[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.

ditionally, 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 will be accurate, and without intelligent sampling, humans will encounter errors very infrequently. Second, if there is class imbalance, ceteris paribus, most of the encountered errors would be misclassifications of examples of the majority class into the minority. If both of these conditions hold, then it becomes quite difficult to identify misclassifications of 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).* □

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.

Instead of passively waiting for such 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.

The selective acquisition of example labels with the intent of building robust performance estimators at minimal cost is a topic getting recent attention in the research literature (Sawade, Christoph, Bickel, Steffen, and Scheffer, Tobias 2010; Bennett and Carvalho 2010). However, while promising and potentially useful in practice, such acquisition strategies are focused on minimizing the cost required to compute a robust estimator for precision or total loss. In order to construct such an estimator, existing selective acquisition strategies sample from the problem space in accordance to some function of the output score of the model being considered. However, given a capable model deployed in a production system, it may take millions of samples from high-confidence positive predictions to reveal a single example that "beats the machine." Incorporating performance bounds such as those presented in the referenced research with our proposed selection strategy is an interesting direction for future work.

Task Design Iterations

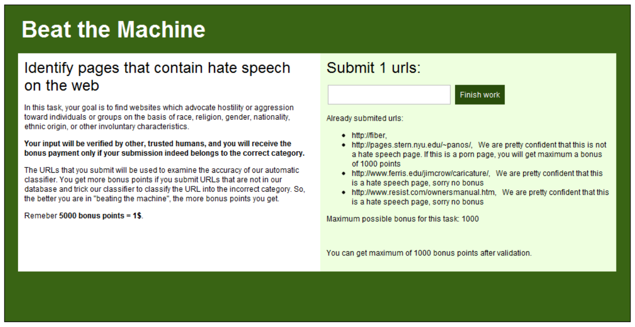
For the purpose of this workshop, let's now walk through several design interations, focusing on the ideas, challenges, and subsequent redesigns.

Initial design: The initial idea was straightforward: 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 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



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.

Figure 1: A screen-shot of the BTM interface on Mechanical Turk.

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 1 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 misclassifications proportionally to the confidence of the classifier. If the model is not very confident of its classification of a submitted URL, 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.[2] 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.

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 were even better: workers identified errors only in 9% of the inspections for stratified random sampling, but they identified errors in 27% of the URLs with BTM. These results indicate that the BTM process is indeed more efficient than the standard evaluation procedure in identifying problematic cases. It

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

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.

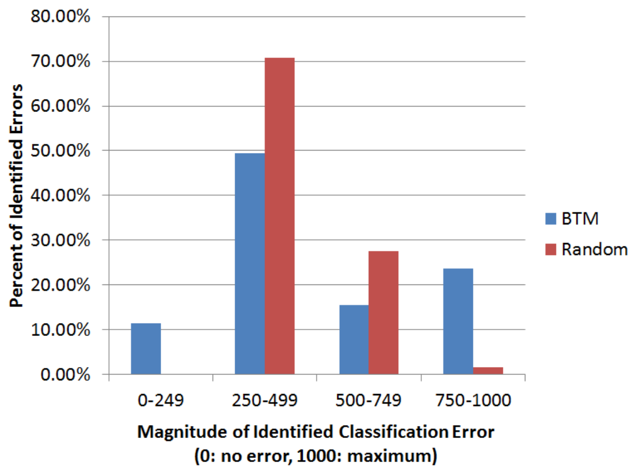
Comparing the severity of errors: Figure 2(a) and 2(b) show the distribution of errors for hate speech and adult content, respectively. A consistent behavior is observed for both categories: BTM identifies a significantly larger number of severe misses—the unknown unknowns. Within the errors identified 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: The next, natural question is whether the identified erroneous decisions could be used to improve the decision models. This actually is a very complicated problem, and a thorough treatment is beyond the scope of this short paper. 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 (Freund and Schapire 1999) have gotten tremendous advantage by overweighting cases where the current model is incorrect.

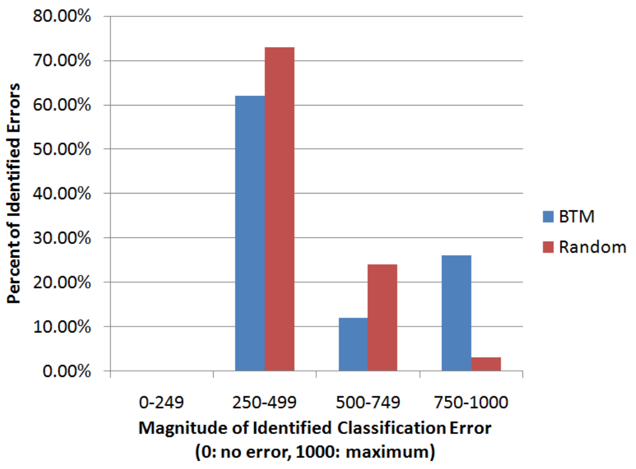
Nevertheless, we can offer some initial insights. We can examine whether the cases found by BTM seem to be isolated outliers, or whether they seem to be regularities that can be modeled. To this end we ran the following experiment: We attempted to learn a model that would classify positive and negative examples from amongst the BTM-identified cases.[3] 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 3 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

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



(a) Hate Speech



(b) Adult Content

Figure 2: Distributions of the magnitude of the identified errors by BTM and by random sampling for two ad safety tasks

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. The results show that the quality of the models is fairly high, illustrating 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. The comparatively lower quality of the random sampling model also 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

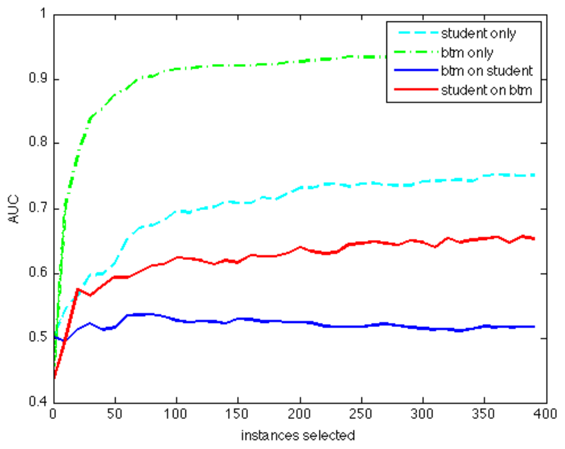


Figure 3: Learning curves generated by the models using cross-validation (BTM and student lines), and then use as test case for BTM the errors identified by random sampling (BTM on students), and vice versa (students on BTM).

ignored.

We also can examine whether the two approaches (DVSRS 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 on 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 cases where the model already knows that it is not confident. The BTM process, through its game-like structure and probing nature, encourages the 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.

Current and Future Research

We discussed 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."

Acknowledgements

The authors thank George A. Kellner and NEC for faculty fellowships, and AdSafe Media for expertise, support, and data. The models used in this paper are not necessarily models used in production by any company. This work was partially supported by the National Science Foundation under Grant No. IIS0643846.

References

Bennett, P. N., and Carvalho, V. R. 2010. Online stratified sampling: evaluating classifiers at web-scale. In *CIKM'10*.

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

Raykar, V.; Yu, S.; Zhao, L.; Jerebko, A.; Florin, C.; Valadez, G.; Bogoni, L.; and Moy, L. 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. ACM.

Sawade, Christoph; Bickel, Steffen; and Scheffer, Tobias. 2010. Active Risk Estimation. In *ICML*.

Settles, B. 2010. Active learning literature survey.

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

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

Winston, P. 1970. Learning structural descriptions from examples.