Detecting adversarial advertisements in the wild

Conference Paper · August 2011

DOI: 10.1145/2020408.2020455 · Source: DBLP

Matthew Eric Otey
Google Inc.
22 PUBLICATIONS **935** CITATIONS

SEE PROFILE

John Hainsworth
Cadence Design Systems, Inc.
13 PUBLICATIONS **342** CITATIONS

SEE PROFILE

Detecting Adversarial Advertisements in the Wild

D. Sculley
Google, Inc.
dsculley@google.com

Matthew Eric Otey
Google, Inc.
otey@google.com

Michael Pohl
Google, Inc.
mpohl@google.com

Bridget Spitznagel
Google, Inc.
drsprite@google.com

John Hainsworth
Google, Inc.
hainsworth@google.com

Yunkai Zhou
Google, Inc.
yunkaiz@google.com

ABSTRACT

In a large online advertising system, adversaries may attempt to profit from the creation of low quality or harmful advertisements. In this paper, we present a large scale data mining effort that detects and blocks such adversarial advertisements for the benefit and safety of our users. Because both false positives and false negatives have high cost, our deployed system uses a tiered strategy combining automated and semi-automated methods to ensure reliable classification. We also employ strategies to address the challenges of learning from highly skewed data at scale, allocating the effort of human experts, leveraging domain expert knowledge, and independently assessing the effectiveness of our system.

Categories and Subject Descriptors

I.5.4 [**Computing Methodologies**]: Pattern Recognition—*Applications*

General Terms

Experimentation

Keywords

online advertisement, data mining, adversarial learning

1. INTRODUCTION

The multi-billion dollar online advertising industry continues to grow [15]. This growth is fueled by users who find that online advertisements yield high quality, trustworthy content, as provided by millions of good-faith advertisers. However, in this favorable landscape a small number of *adversarial advertisers* may seek to profit by attempting to promote low quality or untrustworthy content via online advertising systems. Our goal is to detect and block these motivated adversaries, protecting users and ensuring that online advertisement remains a trustworthy source of commercial information.

The problem of detecting adversarial advertisements is complicated by scale. With millions of advertisers and billions of advertiser landing pages,[1] automated detection methods are clearly needed. However, unlike many data-mining tasks in which the cost of false positives (FP's) and false negatives (FN's) may be traded off, in this setting both false positives and false negatives carry extremely high misclassification cost. Thus, both FP and FN rates must be driven toward zero, even for difficult edge cases.

The need for extreme reliability at scale necessitates the use of both automated and semi-automated methods in a tiered system. Automated detection methods, based on high-precision, large-scale machine learning methods, are able to handle the bulk of the detection work. High-recall models are then used in semi-automated fashion to guide the effort of expert humans who can resolve hard edge cases. Together, these approaches form the basis of a system that quickly and reliably identifies adversarial advertisements and blocks them from serving.

1.1 Challenges

This paper presents the full anatomy of the multi-tiered data mining system currently deployed at Google for detecting and blocking adversarial advertisements, and is intended to serve as a detailed case study. This study is structured around the following key challenges:

- **High cost of both FP's and FN's.** In our setting, both FP's and FN's have high cost; we cannot trade off one against the other. Using a combination of automated and semi-automated effort helps drive both FP and FN rates towards zero.
- **Minority-class and multi-class issues.** The vast majority of ads are from good-faith advertisers; thus detecting adversarial advertisements presents a difficult class imbalance issue [6]. This challenge is compounded by the presence of many different classes of adversarial advertisements, described in Section 2.
- **Training many models at scale.** At a high level, our system may be viewed as an ensemble composed of many large-scale component models. Each of these models must be frequently trained, evaluated, calibrated, and monitored; an efficient paradigm for this effort is presented in Section 3.

KDD'11, August 21–24, 2011, San Diego, California, USA.
Copyright 2011 ACM 978-1-4503-0813-7/11/08 ...$10.00.

[1] The *ad landing page* is the web page to which a user is directed after clickling on an ad.

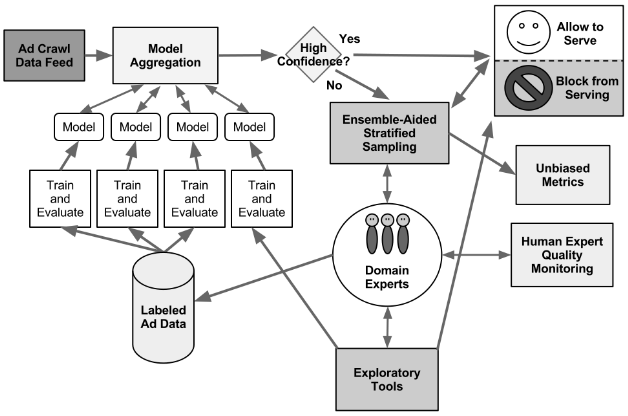

Figure 1: System-Level Architecture. Our system relies both on automated detection using large-scale learning and semi-automated detection in which learned models direct the effort of human experts.

- **Capturing expert knowledge.** To cope with constantly evolving adversarial tactics, our system needs to be able to capture and leverage expert knowledge quickly and efficiently. Using experts to label examples is one such method. Section 5 details additional approaches including the use of active learning, providing exploratory tools to experts, and enabling experts to develop rule-based models for fast response.

- **Allocating expert effort for multiple concurrent goals.** Expert effort is required not only for handling edge-cases, but also for providing training data and unbiased evaluation metrics. Section 4 presents an ensemble-aided stratified sampling approach to achieve these multiple goals simultaneously.

- **Independent evaluation.** Because we rely on human experts for ground truth, regular independent evaluations are critical to ensure that our ground truth understanding is accurate and comprehensive (see Section 6).

1.2 System Architecture

A high-level overview of our system architecture is given in Figure 1. The main source of data for our system is provided by a feed of advertisement data, including a crawl of the ad landing pages themselves. Because the crawl is responsible for fetching the contents of billions of ad landing pages and is a massive system in its own right, a detailed description of the crawl system is outside the scope of this work.

Each ad is evaluated by a large number of deployed models. The decisions from the models are aggregated; if there is a high-confidence decision of `block` or `allow`, this decision is put into serving. If the automated models are unable to provide a high-confidence decision, the ad may be shown to human experts as part of our ensemble-aided stratified sampling process (see Section 4). Human experts may also develop models using automated assistance, or use exploratory tools to find adversarial cases and add this data to our system. Because ad content may change dynamically over time, we record a snapshot of all features for an ad at the moment it is labeled by a human so that our repository of labeled data is contextually accurate.

2. BACKGROUND: ADVERSARIAL ADVERTISEMENTS

To our knowledge, adversarial advertisements have not yet been widely studied in the literature (see Section 7). In this section, we clarify the problem area by describing some representative categories of adversarial advertisements that we have encountered in practice. Note that this section is a partial listing intended to guide the reader's intuition for this paper; official policies are available online via the Google AdWords Help Center [13].

- **Counterfeit goods.** Some adversaries attempt to sell counterfeit or otherwise fraudulent goods while representing the goods as authentic.

- **Misleading or inaccurate claims.** This class of adversarial advertisements attempt to make claims that are unrealistic or scientifically impossible, such as a weight-loss plan promising extreme weight loss without exercise or dieting.

- **User safety issues.** Some adversaries attempt to profit by causing the user some form of harm, such as with false financial or medical claims.

- **Phishing.** Adversaries may attempt to obtain sensitive personal information by disguising their site to look like another site.

- **Arbitrage.** Advertisements whose sole or primary purpose is to direct the user to additional advertisements add little or no value to the user experience, in contrast to ads that provide useful content.

- **Unclear or deceptive billing.** Advertisements that list inaccurate or deceptive pricing, or which obscure the pricing or billing method, can constitute adversarial attempts to profit from false pretenses.

- **Malware.** Some adversaries attempt to direct users to landing pages where they might unwittingly download malware, badware, or other malicious software.

3. LEARNING METHODS

We now turn to the details of our learning-based approaches to detecting adversarial advertisements, starting with an examination of the features available for learning. We then describe the approaches we use to cope with the particular form of hierarchical multi-class classification required in this setting, including methods for dealing with highly skewed class imbalance. We present a simple MapReduce framework for training such models at scale, and conclude this section with an examination of practical considerations that must be addressed in a live production setting.

3.1 Features

Feature engineering is a key component of effective data mining; the following is a listing of the features extracted from advertisements during training and classification.

- **Natural language features** are extracted from the text of advertisement keywords, creatives, and landing pages. These include term-level features, and semantically related terms and topic-level features using methods similar to [9] and [3].

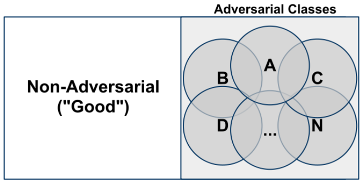

Figure 2: Class Structure. There is a clear distinction between `adversarial` and `non-adversarial`, but members of the `adversarial` classes overlap in places.

- **String-based features** are intended to avoid the possibility that adversaries may exploit alternate spellings or typographical manipulation to avoid detection. We incorporate features that allow inexact string matching, similar in spirit to [30].
- **Structural features** are extracted from the structural layout of the landing page.
- **Page-type features** are given by sub-classifiers that determine the general landing page type, such as a blog posting or a list of shopping results.
- **Crawl-based features** are extracted from the results of the http fetch of the landing page.
- **Link-based features** are based on links and redirects from the landing page.
- **Non-textual content-based features** yield information about the image, video, or other multimedia content on the page.
- **Advertiser account-level features** provide various information that may help identify suspicious or gaming behavior from adversaries.
- **Policy-specific features** include a variety of proprietary hand-crafted features that help to identify violations in particular policy areas.

3.2 Multi-Class and Minority-Class Issues

As described in Section 2, this problem area is inherently multi-class. For policy reasons, it is important to determine the exact category or categories for a given example, rather than a binary `adversarial` or `non-adversarial` classification. This problem is made more challenging by the fact that the vast majority of ads are `non-adversarial`, making each adversarial category an extreme minority class.

Our automated classification methods include a variety of inherently multi-class classifiers, including nearest neighbor approaches, naive-bayes variants, and semi-supervised graph-based algorithms. Because these methods are well known [2], we will not describe them further here.

Interestingly, the most effective methods we have found are based on sets of binary-class linear classifiers deployed as per-class classifiers, including linear support vector machines (SVM's) [16, 28], linear rank-based SVM's [17, 26], and linear models in cascades [33]. These methods will be the focus of this section.

3.2.1 *One-vs-Good Multi-Class Classification*

Typical strategies for performing k-class multi-class classification with binary classifiers include the *one-vs-all* method of training k individual models to distinguish each class from all other classes, and the *one-vs-one* method of training $\binom{k}{2}$ models to distinguish each class from each other class [14].

The class labels in our setting have a special structure. There is a single large class of `non-adversarial` (or "good") ads, and then a large number of possibly overlapping `adversarial` classes (see Figure 2). This setting naturally suggests a multi-class decomposition that we call *one-vs-good*, in which for each of $k-1$ adversarial classes, a model is trained to distinguish that class from members of the `non-adversarial` class only. This allows overlapping classes to be detected by examining the output of all models. In situations where several classes overlap significantly, we found it useful to train additional models to distinguish all members of the high-overlap set from the `non-adversarial` class.

3.2.2 *Learning-to-Rank Methods for Classification*

Linear SVM's have been found to be highly effective at high dimensional classification tasks similar to those encountered here, such as text classification [17]. Linear SVM's are trained by solving the following optimization problem:

$$\min_{\mathbf{w}} \frac{\lambda}{2}||\mathbf{w}||^2 + L(\mathbf{w}, D)$$

Here, $\mathbf{w} \in \mathbb{R}^d$ is a d-dimensional linear weight vector and λ is a regularization parameter controlling model complexity. $L(\mathbf{w}, D)$ is the total hinge-loss of $\mathbf{w}$ over the labeled training data $D = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m))$ given by $\sum_{i=1}^m \max(0, 1 - y_i\langle \mathbf{w}, \mathbf{x}_i\rangle)$. Each labeled example contains a feature vector $\mathbf{x} \in \mathbb{R}^d$, and a class label $y \in \{-1, +1\}$. Linear SVM's may be trained efficiently using stochastic gradient descent variants such as the Pegasos SVM algorithm [28].

However, in cases of extreme class imbalance, linear SVM's have been found to give less than ideal results [21]. One approach is to set per-class weights on the loss function to emphasize the importance of the minority class [21], but we have found (in accordance with prior work [18, 26]) that using a pairwise objective function both improves results and eliminates the need to tune special per-class weights.

In the binary-class case, we refer to this pairwise method as the ROC Area SVM, or ROC-SVM. An ROC-SVM is trained by solving the following optimization problem:

$$\min_{\mathbf{w}} \frac{\lambda}{2}||\mathbf{w}||^2 + L(\mathbf{w}, P)$$

Here, P is the set of all *candidate pairs* in the original data set D. A candidate pair contains one $\mathbf{x}_p$ member of the positive class and one $\mathbf{x}_n$ member of the negative class, and is used to construct a labeled pairwise example $((\mathbf{x}_p - \mathbf{x}_n), +1)$. The ROC-SVM may be trained efficiently (despite the quadratic number of candidate pairs) using stochastic gradient descent and an indexed sampling scheme; see [26].[2]

Results. Using ROC-SVM instead of a standard Pegasos SVM improves recall by as much as 15% at our high-precision threshold for automated blocking of adversarial advertisements.

[2] Open source code for (single-machine) ROC-SVM using SGD is freely available `http://code.google.com/p/sofia-ml`.

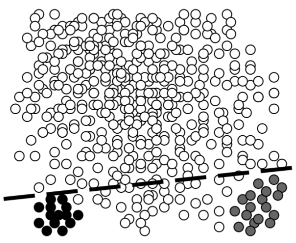

Figure 3: Visualizing Cascades. The vast majority of easy, good ads are filtered out by a high-recall coarse model (left). Finely-grained models then detect specific adversarial classes (right).

3.2.3 Cascade Models

The single model ROC-SVM approach works well for a number of adversarial classes, but other classes are more difficult to classify at the high-precision levels needed for automated blocking. For these cases, we use a more sophisticated methodology based on cascades.

The basic cascade framework uses a series of models, each of which rejects a portion of the data space, as illustrated in Figure 3. This approach has been particularly successful in the field of computer vision for tasks such as face recognition [33] and email spam filtering [37]. In theory, there is no limit to the number of stages that may be applied (and boosting approaches may result in dozens of stages). In practice, tuning a large number of cascade stages requires significant manual effort, creating a heavy maintenance burden.

After experimenting with a range of multi-stage configurations, we found a simple strategy that achieves good results with minimal system complexity. We use a single coarse model, common to all of our cascade models, trained to distinguish `adversarial` from `non-adversarial` with high recall. We then train a set of more finely-grained models to detect each of these more difficult classes with high precision, using the one-vs-good framework (see Figure 4).

Cascade models are particularly susceptible to problems of over-fitting. We have found tightly regularizing the coarse-level model to be effective. (Another approach uses cross-validation on the training data [37], but it is then non-trivial to combine the models in stages in a principled way.) The coarse model is tightly L1-regularized (see Section 3.3.2), inducing sparsity that keeps the memory footprint of this coarse model relatively small; this is an important consideration when dealing with billions of features. The sub-models are each trained on data sets that are much reduced in size due to the coarse-level filtering, reducing their size significantly as well.

Results. Representative results for three difficult per-class cascade models are given in Figure 5. (Note that the precision and recall values given in these graphs have been linearly transformed to obscure sensitive data; the relative performance trends remain unchanged.) In general, the cascade models give excellent improvement in recall at high precision levels (the upper-left corner of each graph), in comparison with a single ROC-SVM model for the same class problem. Note that in cases where the precision/recall curves cross (at lower precision levels, used for prioritizing human expert effort) we can always use the better of the two models in the different regions.

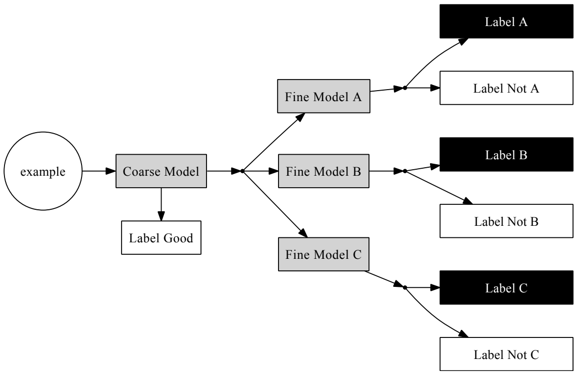

Figure 4: Multi-Class Cascade Framework. The coarse model filters out examples that are clearly `non-adversarial`. The remainder are passed to a set per-class models for fine-grained classification.

3.3 Large-Scale Training

By some standards, the data sets used to train our models may be seen as large (measured in terabytes); we need an efficient methodology for frequent training of dozens or hundreds of models. By enforcing sparsity during training, we ensure that the resulting models fit in memory on a single machine. This allows us to deploy an efficient stochastic gradient descent (SGD) training paradigm in a MapReduce[3] setting for fast model training.

3.3.1 MapReduce SGD

Solving optimization problems such as those presented in Section 3.2.2 may be done efficiently using SGD, which quickly converges to approximate solutions that are highly satisfactory from a machine learning perspective [4, 28]. SGD is an iterative solver, sequentially examining data points one at a time in a random order. The basic SGD training paradigm for linear SVM [38] is given in Algorithm 1.

Algorithm 1 Training Linear SVM using SGD.

```
 1: $\mathbf{w_0} \leftarrow 0$
 2: for $i = 1$ to $t$ do
 3:     $(\mathbf{x}, y) \leftarrow$ RandomExample($D$)
 4:     $\eta \leftarrow$ GetLearningRate($t$)
 5:     if $y\langle \mathbf{w_{i-1}}, \mathbf{x} \rangle < 1.0$ then
 6:         $\delta \leftarrow 1$
 7:     else
 8:         $\delta \leftarrow 0$
 9:     end if
10:     $\mathbf{w_i} \leftarrow (1 - \eta\lambda)\mathbf{w_{i-1}} + \delta\eta y\mathbf{x}$
11: end for
12: return $\mathbf{w}_t$
```

In general, each iteration is fast to compute when $\mathbf{x}$ is sparse, and can be completed in $O(s)$ time where s is the number of non-zero elements in $\mathbf{x}$. The Pegasos SVM variant includes a step that projects $\mathbf{w}$ into an L2-ball of fixed radius after step 10; this may be done in O(1) time with appropriate data structures [28]. The number of iterations t may be large, but is provably independent of the number of training examples, making the SGD framework ideal for

[3] MapReduce is a paradigm for “embarrassingly parallel” tasks, widely used in cluster-based computing [8].

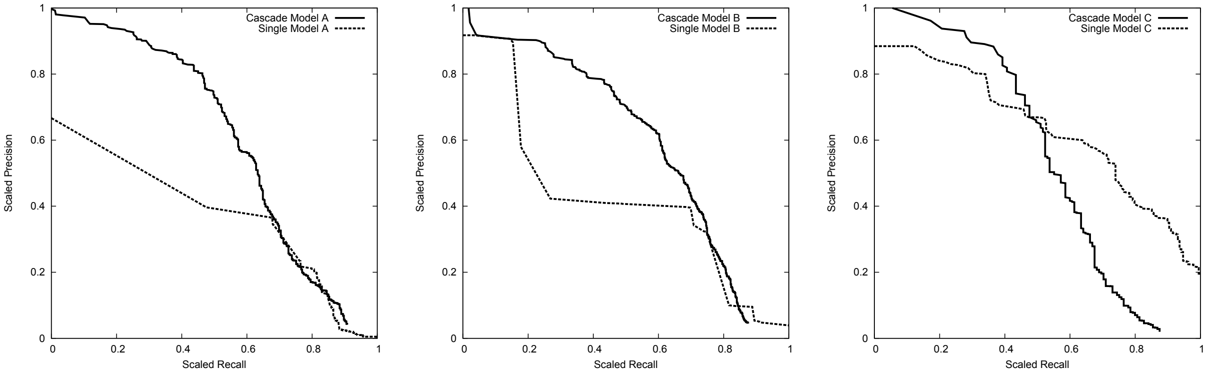

Figure 5: Cascade Models vs. Single Models. Values from three representative adversarial category identification tasks show that using cascade methodology significantly improves recall at low false positive rates. (Note that precision and recall values have been linearly transformed to protect sensitive data.)

large-scale learning [28, 4]. For example, only a few CPU seconds are required for training on data sets that are considered large in the academic literature, such as RCV1 [21]. But because SGD is a sequential online methodology, it is non-trivial to parallelize SGD training across multiple machines. Approaches that have been analyzed include message passing approaches [23], lazy distributed updates [39], training on multiple independent samples of the data [40], and performing multiple chained MapReduce steps [35]. However, these methods all add significant complexity to a relatively simple learning algorithm, and in some cases adversely impact model quality or give only limited incremental benefit as more machines are added.

Interestingly, we have observed that the main cost of SGD training is actually *reading and parsing data from disk*; this can be orders of magnitude more expensive than the actual training. Langford *et al.* have found it effective to reduce this cost by using a specialized data format, which significantly reduces disk-read times [19]. This observation allows us to parallelize our training effort with a simpler approach than those described above. As shown in Figure 6 our approach is summarized as follows:

- **Do expensive work in parallel.** The expensive work of parsing, filtering, labeling, transforming, and encoding data can all be done independently in parallel. We use hundreds of machines in the Map phase. The output of the Map phase is labeled training data that is efficiently compressed.

- **Do cheap work sequentially.** Because our models are small enough to fit in memory, a single Reduce machine can perform the SGD training quickly once the data has been properly prepared and formatted. This eliminates the need for expensive message passing or synchronization.

This framework allows us to train models within minutes on large data sets, and is used for both ROC-SVM training and for training our cascade models. A similar framework is used for evaluating models on holdout test data.

3.3.2 Controlling Model Size

The learned models must be small enough to fit in memory on a single machine; we use two strategies for keeping model size suitably restricted.

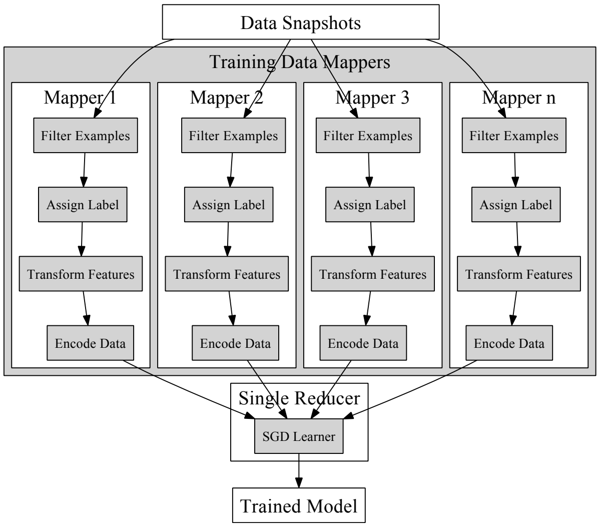

Figure 6: SGD learning via MapReduce. Preprocessing is parallelized; training is sequential.

The first of these is to use a feature-hashing approach similar in spirit to that of [36]. If we think of $\mathbf{w} \in \mathbb{R}^d$ as a set of key-value pairs where many values are exactly 0, then it is efficient to store the non-zero values in a hash map. Hashing the keys ensures that the model size will not grow beyond a certain bound. We have found that ignoring collisions does not degrade model performance, in line with results from [36] and keeps model size manageable.

The second strategy is to encourage sparsity in the learned model, so that many weight values are indeed exactly 0. We follow a projected-gradient methodology similar to that of [10], projecting $\mathbf{w}$ to an L1-ball of a specified radius after updates. This is done every k steps, after step 10 in Algorithm 1. The exact L1-projection of [10] was somewhat slow, so we use a simpler and faster approximate projection given in Algorithm 2. The method of Duchi *et al.* uses an approach similar to randomized median finding to find the exact value of τ that is used to project a given vector $\mathbf{w}$ onto an L1-ball of radius at most λ. We make do with using a value of τ that is guaranteed to cause the $||\mathbf{w}||_1$ to converge to radius at most λ after repeated calls. In practice, we find this work well, is fast to compute, and is easier to tune than the truncated gradient approach of [20].

Algorithm 2 Approximate projection to L1-ball of radius at most λ. Repeated calls to this projection will converge to radius λ.

```
$c \leftarrow \max(\lambda - ||\mathbf{w}||_1, 0)$
$d \leftarrow ||\mathbf{w}||_0$
$\tau \leftarrow \frac{c}{d}$
for each non-zero element $i$ of $\mathbf{w}$ do
    $s \leftarrow sign(\mathbf{w}_i)$
    $\mathbf{w}_i \leftarrow s * \max(|\mathbf{w}_i - \tau|, 0)$
end for
```

3.4 Model Management

It is worth briefly looking at some of the engineering issues involved in maintaining a large-scale data mining system with many component models. Our management strategies include performing automated model calibration, establishing effective automated monitoring of live models, and bundling useful information into models.

3.4.1 Calibration

As models are re-trained over time, the semantic meaning of their output scores may drift or vary, resulting in constant adjustment of decision thresholds. To avoid the need for manual threshold adjustment, we automatically calibrate each model so that its final output is an actual probability estimate rather than an arbitrary score.

Recall that each linear model $\mathbf{w}$ scores a given example $\mathbf{x}$ using a scoring function $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$. We learn a calibration function $c(\cdot)$ using holdout validation data to ensure that $c(f(\mathbf{x})) = Pr[y_{\mathbf{x}} = 1]$. This ensures that scores from different model versions may be interpreted along the same natural scale.

3.4.2 Monitoring

In a live production setting, it is critical to monitor the quality and output of models. Our first level of monitoring involves a set of precision/recall tests that each model must pass, based on holdout test data, whenever the model is re-trained. If the model fails these tests, the new version is not pushed to production. Second, we monitor the input features, to make sure that in aggregate the distribution of values is relatively stable. We need to be alerted if our input features or signals were to vary significantly, as this would cause changed behavior from our models. Third, we monitor model output scores to detect drift in distribution of values. A sudden drift would be a warning of a change somewhere in our system. We also monitor the actual decision rates in addition to the score distributions, to ensure we are aware of any sudden changes there. Finally, we monitor the overall system quality, based on data from our ensemble-aided stratified sampling pipeline, as described in Section 4.

3.4.3 Bundling Model Data

It is clear that a model needs to know how to classify examples. But what else should a model know how to do? Over time, we have found that it is useful to bundle a surprisingly large amount of information together into a model object. In particular, a model should know how to do the following:

- **Filter** out examples that should be ignored, for example if they are in the wrong language.

- **Transform** features as needed, including scaling, discretizing, etc.

- **Label** training data as a positive or a negative, and distinguish test data from training data.

- **Report** the parameters that were used to train it, so that the model may be re-trained if needed.

- **Score** an example using a feature vector $\mathbf{w}$.

- **Calibrate** its output scores onto a consistent scale.

Together, these requirements define a somewhat broader view of a "model" than is generally considered in academic literature, which often only discuss the weight vector $\mathbf{w}$. We have found bundling this data together reduces system complexity and eases the burden of managing and maintaining a large number of models in production.

4. ENSEMBLE-AIDED STRATIFIED SAMPLING

Acquiring hand-labeled data represents a significant cost, requiring expert judgement to navigate intricate policies and to recognize a wide variety of clever adversarial attacks. Our pilot experiments testing low-cost, crowd-sourced rater pools showed that crowd-sourcing was not a viable option to achieve labels of the needed quality for production use. Thus, we rely on more expensive, specially trained expert raters. It is therefore critical that we make the most efficient use of this limited resource.

In this section, we detail an ensemble-aided approach to stratified sampling that helps allocate rater effort efficiently to achieve multiple distinct goals.

4.1 Multiple Needs for Hand-Labeled Data

In machine learning literature, it is common to focus on gathering hand-labeled data only for the purpose of model training. But in our real-world setting, there are actually several important and distinct areas in our system that require hand-labeled data from expert raters. These are:

- **Catching hard adversaries.** Human judgement is needed to provide a final verdict on cases which are too difficult for automated methods to classify with high precision.

- **Improving learned models.** Hand-labeled data is needed to train models, and to keep the models current as adversarial methods change over time. Examples which are most beneficial for improved models (such as may be selected by various active learning strategies [32]) may be different from hard adversaries, above.

- **Detecting new trends.** It is helpful to prioritize the review of new ads, so that new trends from adversaries are quickly detected both by our automated systems and by our human domain experts.

- **Maximizing impact.** Because expert human-rater capacity is expensive and limited, it is desirable to maximize the impact by focusing on advertisements that have high impression counts.

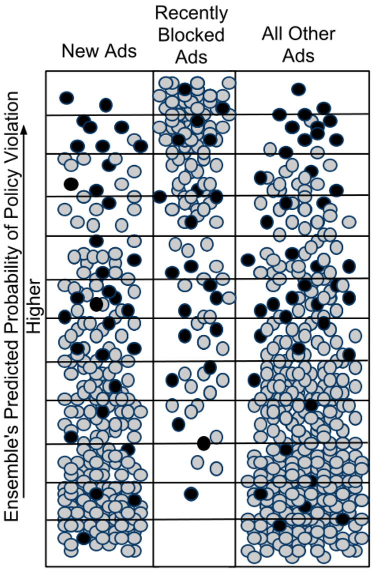

Figure 7: Ensemble-Aided Stratified Sampling. Ads in a given language are binned based on probability estimates from an ensemble. Grey and black circles represent un-sampled and sampled ads, respectively.

- **Providing unbiased metrics.** Hand rated data should be able to provide unbiased estimates for model-level and system-level precision and recall. The naive approach to gathering unbiased evaluation data would be to hand rate a uniform sample of ads. However, this would waste effort because the vast majority of ads are not adversarial. Sampling ads to achieve the other goals above results in biased evaluation data; by constructing the sample in a careful manner we can later remove the sample bias when computing metrics.

Not surprisingly, these different goals are largely disjoint, making it non-trivial to determine which ads should be selected for human rating.

4.2 Ensemble-Aided Stratification

We considered various forms of model-aided sampling [25], using learned models to induce a sampling bias towards more useful ads for human rating. We first tried to define an aggregate "utility" score based on these different factors, but it was unclear how to combine (or in some cases even how to measure) these different quantities. We also considered framing this problem in a bandit setting; however, McMahan and Streeter show that the use of bandit algorithms is problematic for selecting data for model-training due to the non-stationarity of the underlying models [24].

Our approach is to stratify the data across several different dimensions, as shown in Figure 7. First, ads in each language are considered separately. Within a given language, ads are divided into three categories, `new ads` that are less than t hours old, `recently blocked ads` that have been caught and turned off within the last t hours, and `all other ads` that have been actively served over the last t hours.

To aggregate the scores from the many different models in production, we train an ensemble model [2] for the given language, using the output scores of each of the m models as features for the ensemble. We use a binary-class (`adversarial` vs. `non-adversarial`) linear ROC-Area SVM to train the ensemble, and calibrate its output scores as described in Section 3.4.1. The score from the ensemble-model is used to divide the ads in each category into uniformly spaced score-bins containing different numbers of ads.

This coarse binning allows us to explicitly decide how many ads to select from each bin in order to balance the differing goals listed in Section 4.1. Selecting sites from mid-probability bins is akin to uncertainty sampling [32], and provides benefit for model training. Selecting ads from higher probability bins in the `new ads` or `all other ads` categories gives priority to catching adversaries that have not yet been detected automatically. Selecting some ads from every bin ensures coverage of the entire data space.

Algorithm 3 Priority Sampling Ads from a Bin. We use the following priority-sampling algorithm from [11] to select ads from a given bin for near-optimally low variance.

```
for each advertisement $i$ with impressions $w_i$ do
    pick $p_i$ uniformly at random from $(0, 1]$
    let priority $q_i = w_i/p_i$
end for
sort all ads by their priority $q$
let $\tau$ equal the $(k+1)$-th highest priority
for each of the $k$ highest priority ads do
    let effective weight $w'_i = \max(w_i, \tau)$
end for
exclude all other ads from the sample for this bin
```

4.3 Priority Sampling from Bins

Assuming we have decided to select k ads from a given bin, how should we choose which k to pick from that bin? Ideally, we would like a low-variance estimate of the impression-weighted total of each class of adversarial advertisement from the given bin; however, impression counts vary dramatically across ads, following a heavy-tail distribution.

In this heavy-tail setting, using uniform sampling to select ads within a bin is a poor strategy, resulting in high variance estimates [11]. Using an intuitive sampling proportional to impression-count strategy is better, but far from optimal [11]. Instead, we use the Priority Sampling strategy of Duffield *et al.* which has been proven to yield near-optimally low variance estimates for arbitrary subset sums [11]. This strategy is reviewed in Algorithm 3.

The variance from an estimate based on k samples selected with this strategy is provably at least as low as the variance of $k+1$ samples selected with the (computationally infeasible) optimal strategy [31]. It also has the attractive quality of tending to prioritize very-high impression ads for review to maximize impact of human effort while maintaining good coverage of the long tail.

Results. Using our ensemble-aided stratified sampling approach instead of the naive approach of separate samplings increased the effective impact of our human experts by 50%, and reduced the latency required to compute unbiased metrics by an order of magnitude with no added cost.

5. LEVERAGING EXPERT KNOWLEDGE

We have found it critical to leverage the knowledge of human experts to help detect evolving adversarial advertisements. Here, data mining methods provide automated guidance, ensuring the most effective use of human effort.

5.1 Active Learning

Experts periodically detect new categories of bad ads, or particular emerging trends, for which it is useful to develop a new model. Lacking initial training data, we have found that margin-based uncertainty sampling (akin to the `simple` strategy of Tong and Koller [32]) has been an effective methodology for rapid development of new models, often requiring only a few dozen hand-labeled examples.

5.2 Exploring for Adversaries

Attenberg *et al.* recently reported that in cases of extreme class imbalance, traditional active learning strategies may fail from difficulty in locating any members of the proposed positive class. They suggested using information retrieval systems in such cases, allowing expert users to search for positive examples guided by their intuition.

Independently, we have also found that providing a search-based interface for expert users provides valuable automated assistance for finding new examples of adversarial advertisements. Because this search-based tool is used by experts, it has been practical to augment standard keyword-based search with a variety of feature-based filters (using many of the features listed in Section 3.1). This allows experts to make guided searches in real time, based on their intuition and a large store of informative data.

5.3 Rule-Based Models

Coming from a machine-learning background, it has surprised us that our experts have proven capable of developing hand-crafted, rule-based models with extremely high precision. Enabling such models to be served in production provides a rapid response mechanism to new adversarial attacks, and gives an effective means of injecting expert knowledge directly into our system.

Because such models do not adapt over time, we have developed automated monitoring of the effectiveness of each rule-based model; models that cease to be effective are removed. Although rule-based models only account for less than 4% of the overall system impact, they provide an important capability to respond to new classes of adversarial attacks within minutes of discovery.

6. INDEPENDENT EVALUATION

Finally, we examine the data challenge of evaluating the human components of our system.

6.1 Monitoring Human Rater Quality

Because human ratings are used as our ground truth, it is critical to measure how reliable these ratings are. To establish this, we regularly evaluate the precision and recall of our base-level raters, using higher-level experts to re-rate a sample of ratings from each lower-level rater. We also regularly double-check these results using an independent set of higher-level experts. This allows us both to assess the performance of the base-level raters and to measure our confidence in those assessments.

6.2 Monitoring User Experience

The different levels of human experts described above are all paid and carefully vetted experts, and as such may have a viewpoint that does not always align with the perception of common users. To ensure that we get an accurate reading of real user perception, we additionally perform regular large-scale evaluations using an approach similar to crowdsourcing [29]. These evaluations are used to calibrate our understanding of real user perception and ensure that our system continues to protect the interests of actual users. The aggregate results from these independent evaluations have consistently shown strong agreement with our human expert opinon.

7. RELATED WORK

To our knowledge, the general problem of detecting adversarial advertisements has not previously been studied. In the most closely related work, Attenberg *et al.* detected *unsafe* advertisements, such as those containing adult content or hate speech, and used a search-based methodology over active-learning for model training [1]. We consider a broader range of adversarial advertisements, including many that are often difficult for non-experts to distinguish from good-faith advertisements without aid.

The field of email spam filtering has a large body of literature on the use of data mining for blocking adversarial messages (see [12] for an informative survey). The problem of adversarial advertisement detection differs in several ways, including the multi-class nature of the problem, the minority class difficulties, the presence of dynamically changing content, and the need for trained expert human raters.

Dalvi *et al.* attempt to learn classifiers in adversarial situations by modeling the adversaries [7], but accurately modeling motivated adversaries is problematic in real-world settings. Lowd and Meek point out that in a publicly-facing system, adversaries may attempt to reverse-engineer the model via membership queries [22]. Our inclusion of semi-automated methods involving human effort helps to minimize the effectiveness of such strategies.

Crowd-sourcing efforts like reCAPTCHA [34] attempt to use the effort of anonymous users to block abuse on the web. This approach is difficult in the case of advertisements because it would be problematic to keep motivated adversaries from poisoning the signal. Sculley *et al.* explored the use of aggregate user-based signals such as *bounce-rate* for determining user satisfaction [27], but this approach is unsuitable for making per-advertisement decisions with high precision due to signal noise. Various approaches such as that of Chakrabarti *et al.* have used click-feedback to determine ad *relevance* [5], but for adversarial advertisements relevance is a secondary factor compared to user safety.

8. REFERENCES

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

[2] C. M. Bishop. *Pattern Recognition and Machine Learning.* Springer-Verlag New York, Inc., 2006.

[3] D. M. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *JMLR*, 3, 2003.

[4] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems 20.* 2008.

[5] D. Chakrabarti, D. Agarwal, and V. Josifovski. Contextual advertising by combining relevance with click feedback. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, 2008.

[6] N. V. Chawla, N. Japkowicz, and A. Kotcz. Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explor. Newsl.*, 6, June 2004.

[7] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, 2004.

[8] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51, January 2008.

[9] S. Deerwester, S. Dumais, T. Landuaer, G. Furnas, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 1990.

[10] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *ICML '08: Proceedings of the 25th international conference on Mach ine learning*, 2008.

[11] N. Duffield, C. Lund, and M. Thorup. Priority sampling for estimation of arbitrary subset sums. *J. ACM*, 54, December 2007.

[12] J. Goodman, G. V. Cormack, and D. Heckerman. Spam and the ongoing battle for the inbox. *Commun. ACM*, 50(2), 2007.

[13] Landing page and site policies. Google AdWords Help Center, 2011. `http://goo.gl/XcbPO`.

[14] C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, 13(2), Mar. 2002.

[15] IAB internet advertising revenue report, 2010. `http://www.iab.net/media/file/IAB_report_1H_2010_Final.pdf`.

[16] T. Joachims. Making large-scale support vector machine learning practical. 1999.

[17] T. Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002.

[18] T. Joachims. A support vector method for multivariate performance measures. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, 2005.

[19] J. Langford. Vowpal wabbit. Open source release, 2007. `http://hunch.net/~vw/`.

[20] J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *J. Mach. Learn. Res.*, 10, 2009.

[21] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5, 2004.

[22] D. Lowd and C. Meek. Adversarial learning. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, KDD '05, 2005.

[23] G. Mann, R. McDonald, M. Mohri, N. Silberman, and D. Walker. Efficient large-scale distributed training of conditional maximum entropy models. In *Advances in Neural Information Processing Systems 22*. 2009.

[24] H. B. McMahan and M. Streeter. Tighter bounds for multi-armed bandits with expert advice. In *COLT '09: 22nd Annual Conference on Learning Theory*, 2009.

[25] C.-E. Särndal, B. Swensson, and J. Wretman. *Model Assisted Survey Sampling*. Springer, 2003.

[26] D. Sculley. Large scale learning to rank. In *NIPS 2009 Workshop on Advances in Ranking*, 2009.

[27] D. Sculley, R. G. Malkin, S. Basu, and R. J. Bayardo. Predicting bounce rates in sponsored search advertisements. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009.

[28] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, 2007.

[29] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, 2008.

[30] S. Sonnenburg, G. Rätsch, and B. Schölkopf. Large scale genomic sequence svm classifiers. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, 2005.

[31] M. Szegedy. The DLT priority sampling is essentially optimal. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, STOC '06, 2006.

[32] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2, March 2002.

[33] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *CVPR: IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:511, 2001.

[34] L. von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum. reCAPTCHA: Human-based character recognition via web security measure. September 2008.

[35] M. Weimer, S. Rao, and M. Zinkevich. A convenient framework for efficient parallel multipass algorithms. In *NIPS 2010 Workshop on Learning on Cores, Clusters and Clouds*, 2010.

[36] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, 2009.

[37] W. Yih, J. Goodman, and G. Hulten. Learning at low false positive rates. In *Proceedings of the Third Conference on Email and Anti-Spam (CEAS)*, 2006.

[38] T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, 2004.

[39] M. Zinkevich, A. Smola, and J. Langford. Slow learners are fast. In *Advances in Neural Information Processing Systems 22*. 2009.

[40] M. Zinkevich, M. Weimer, A. Smola, and L. Li. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems 23*. 2010.