Leveraging Machine Learning to Improve Unwanted Resource Filtering

Sruti Bhagavatula[*] Christopher Dunn[†]
Chris Kanich[*] Minaxi Gupta[†] Brian Ziebart[*]

[*]Department of Computer Science
University of Illinois at Chicago

[†]School of Informatics and Computing
Indiana University Bloomington

Abstract

Advertisements simultaneously provide both economic support for most free web content and one of the largest annoyances to end users. Furthermore, the modern advertisement ecosystem is rife with tracking methods which violate user privacy. A natural reaction is for users to install ad blockers which prevent advertisers from tracking users or displaying ads. Traditional ad blocking software relies upon hand-crafted filter expressions to generate large, unwieldy regular expressions matched against resources being included within web pages. This process requires a large amount of human overhead and is susceptible to inferior filter generation. We propose an alternate approach which leverages machine learning to bootstrap a superior classifier for ad blocking with less human intervention. We show that our classifier can simultaneously maintain an accuracy similar to the hand-crafted filters while also blocking new ads which would otherwise necessitate further human intervention in the form of additional handmade filter rules.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: On-line Information Services; J.m [**Computer Applications**]: Miscellaneous; K.4.4 [**Computers and Society**]: Electronic Commerce

Keywords

machine learning; web security; web privacy

General Terms

security; privacy

1. INTRODUCTION

For almost as long as the commercial world wide web has existed, annoying advertisements have competed with desired content for users' attention. To combat these annoyances, users have created methods for filtering out the unwanted ads. While the problem of identifying privacy-violating advertisements has received a large amount of attention from the research community (as outlined in Section 2), the current state of the art for end user ad blocking remains a process with several tedious manual steps: ad blocking engines provide a filter language and web page manipulation interface for removing ads, and then periodically update a list of filter expressions which are typically human generated. Both the creation of rules to match unblocked ads and prevention of false negatives must be done using human feedback which is typically slow and noisy. Today's world should leverage the many learning and intelligent algorithms in existence so that the labor put into manual ad blocking could be taken advantage of elsewhere. The already learned human knowledge of advertisement structures can be applied to the process of recognizing these unwanted web resources.

While advertisements are necessary to support the business model of most content providers, they have long been considered extremely annoying and intrusive to most users. Besides just serving ad content, ad networks use the advertisements to retrieve information about the user and their browsing habits. Additionally, many ads are being used as "malvertisements" that serve malware to unsuspecting victims [9]. Hence, advertisement blocking is definitely a non-trivial defense against the security risks present on the internet of today. There are various browser extensions that block advertisement resources and hide related HTML content. Currently, AdBlock Plus [26], AdBlock [14], Disconnect [3], Ghostery [28] and some proprietary software such as AdMuncher [15] are the most well-known and popular with effective results.

Modern ad blocker implementations have several technical shortcomings. Currently deployed solutions all make use of regular expressions written by contributing users to target specific ads or ad placement schemes. These regular expressions are then compiled into a list that is injected into the browser. Individual resources are matched against the list to test for inclusion in the page. In AdBlockPlus's EasyList [6], the number of filters is on the order of tens of thousands. The filter list can be extremely specific and thus a page element or URL that is clearly ad-related may not be classified by the list because a certain substring is not found. In fact, most blocking is done by keeping an additional blacklist of third-party advertisers. There is substantial human overhead as no automated process exists to generate these unwieldy blacklists. In suboptimal cases, RAM use due to ad blocker filters can balloon to hundreds of megabytes per page [23].

AISec'14, November 7, 2014, Scottsdale, Arizona, USA.
Copyright 2014 ACM 978-1-4503-3153-1/14/11 ...$15.00.
http://dx.doi.org/10.1145/2666652.2666662.

This paper presents an alternative technique for detecting and blocking advertisement resources (which we refer to as "ads" or "advertisements" throughout the paper, and not the images or HTML markup of an ad). We find that using the k-Nearest Neighbors approach to train a classifier that detects advertisements based on historical regular expression lists is the most successful, and show that this classifier is able to both classify current ads with high accuracy as well as detect new ads which might normally necessitate human intervention in the form of additional hand written filters.

2. RELATED WORK

The traditional ad blocking model using regular expression filters has been in use since 2002 with the introduction of McDonald's Adblock for Mozilla and Firebird [22]. Ad blocking is incredibly popular, with some sites experiencing as many as 50% of their visitors using ad blocking software [13].

Kushmerick first suggested using machine learning to block online advertisements, utilizing the C4.5 classification scheme to build an advertisement image blocker called AdEater [20]. The ecosystem of dynamically loaded third party resources has changed greatly since then, with advertisements having a completely different anatomy and feature set than the ones referred to in that work. AdEater only blocked advertisement images of static pages while many ads on the current Web are loaded dynamically via javascript or as flash objects. Thus it is unlikely that this strategy would be particularly successful at successfully blocking ads on today's web.

Nock and Esfandiari continued along the same lines to create an ad blocking extension with the purpose of identifying which requested resource URLs to block [24], which parallels our objective. In their work, they make use of the Weighted Majority Algorithm to make prediction accuracies converge to high levels after increased user feedback. However, their approach does not utilize on-page features and is trained entirely on human feedback, thus making it unsusceptible to more automatic feature-based training algorithms as it doesn't provide information about the mark-up of ad elements and URLs themselves.

In 2008, Krammer developed a rule-based classifier for classifying advertisement images and containers in a web page using features of the page, the ad image and features of the extraneous requests being made [18]. This study is very similar in motivation and approach to ours and captures the essence of online advertising, the business model and the categories into which advertisements fall into. Krammer's study attempts to create a more effective all-around ad blocking model through a large set of rules and features including those of the page, images, request traffic and ad-type by training on manually identified data. While we utilize a few of the features outlined in this paper for our classification, our goal differs in that we propose a lightweight classifier with few features to reduce the amount of human intervention in classification over long periods of time.

Orr et al. proposed a new technique for detecting advertisements being loaded via javascript code through static program analysis [25]. The EFF's "Privacy Badger" project aims to block non-consensual trackers on the web [10]. As many ads are provided alongside non-consensual trackers, Privacy Badger has the side effect of blocking many ads as well.

In this work, we aim to learn a one bit value from individual URLs and their context: specifically, whether they are advertisement-related or not. Ma et al. also leverage supervised machine learning to infer properties of URLs [21]. In their specific domain, they were attempting to determine whether URLs were malicious without access to page content; they use Logisitic Regression and SVMs to learn whether their URLs are malicious or not, which differs from our strategy of using k-Nearest Neighbors. Ma et al.'s approach has also been taken to identify phishing URLs by Garera et al. [12], wherein they determined Logistic Regression to be the most effective algorithm for their dataset using handpicked features.

3. APPROACH

Our approach to classifying ad URLs relies heavily on the AdBlockPlus [26] classification system. AdBlockPlus is widely regarded as the premier ad blocking extension [2] with EasyList [6] being its most common advertisement filter list. EasyList has well over 12 million subscribers today [1] and is often updated on a weekly basis [5]. EasyList can filter unwanted content via URL filters, DOM element filters, and third-party advertisement domain filters. When we refer to EasyList henceforth, we refer to only the general blocking rules for the URLs as our goal is to come up with alternative methods to domain name blacklists and hence do not make use of those particular filters. The list owners use constant user feedback as well as advertisement tracking to manually updates these lists [4].

Our goal is to overcome the problem of having to manually update the EasyList filter list. To achieve this goal, we must both achieve accuracy competitive with current manual filtering while simultaneously detecting ads which would be missed by contemporary manual filters. We use a supervised learning approach to bootstrap the process of learning whether a URL is ad-related by considering EasyList filters as oracles. By testing against EasyList filter lists from different points in time, we can evaluate how robust our classifier is to changes in how ads are served compared to an unmodified EasyList filter (i.e. one which has not been manually updated).

Our classification problem is a binary classification problem on URLs where positive examples are ad-related and negative examples are non-ad-related. Positive examples are resources requested to serve ads - typically requested from within an ad element - and negative examples are any other resource requested on a page that is not related to serving ads.

The basis of our methodology is in using the classification criteria of an older version of EasyList to train a classifier to accurately identify ads according to a much newer filter list. Hence, in this case, we treat the more recent version of EasyList as ground truth. We use 12 features for each URL, the majority of which are binary along with a few which are real-valued. Our feature space consists of features derived from the anatomy of the URL itself as well as some of the originating page properties from which the URL was requested. Using page properties can accommodate some of the caveats in EasyList filters where URLs should only be blocked if they are popups, images, or only on a certain domain.

In the following sections, we will describe in detail how we collected our datasets and features, the different machine learning classification methods we adopted, the evaluation metrics we used and the results of our evaluation.

3.1 Datasets

To generate training and ground truth labels for a collection of URL data, we compared URLs against old and new EasyList filters to get two different sets of positive and negative examples. This resulted in two datasets each with the same URLs and features but with a small percentage of differing class labels for certain examples. For our non-recent or “old” EasyList, our most recent evaluations used the version from September 22nd, 2013 and the version from February 23rd, 2014 as our current or “new” filter list. We consider filters from February as “new”, relative to September, to analyze prediction accuracy over time with respect to the present. We further check for false positives from classification using a recent filter list from June 7th, 2014 as we will see in Section 6.1.

We crawled the Alexa top 500 US sites [7] from February 2014 up to depth 2 and checked each requested page resource URL against the new EasyList filters. Crawling all the links from a page up to depth 2 from the source page allows us to capture normal advertisement activity on pages other than the index pages and on several other sites beyond the Alexa top 500. For each requested URL, while crawling, we stored features that were computed within the context of the page as well as various URL-based and lexical features to create our feature vectors. We checked each URL against the old EasyList and new EasyList to obtain two sets of feature vectors for our “old” and “new” datasets respectively with differing class labels. We collected about 60,000 total URLs; roughly 30,000 of which were ad-related and 30,000 non-ad-related URLs according to the “new” filters. The collection of URLs and their features was done between February and April 2014.

3.2 Features

For the classifier to be effective, we attempted to identify features that may differentiate an ad-related URL from a non ad-related URL. These include characteristics of the structure of the URL, keywords present in the URL, the container it was requested from on a page, and other page properties. All features are either binary or real-valued between 0 and 1. Conveniently, this approach allows all features to be given equal weight at training time. Below is a summary of the key feature categories we used to train our classifier:

A. Ad-related keywords. A URL is more likely to be ad-related than not if it contains certain keywords. We check for: ‘ad’, ‘advert’, ‘popup’, ‘banner’, ‘sponsor’, ‘iframe’, ‘googlead’, ‘adsys’, and ‘adser’. To consider the case that these phrases might be part of a larger non-related keyword, one of the features indicates whether any of these substrings is present followed by a non-alphanumeric character such as ‘.’, ‘/’, ‘&’, ‘=’, ‘;’, ‘-‘ or ‘_’. Additionally to accommodate the exact opposite case where the phrase is present in an ad context but is followed by more alphanumeric characters, the second feature is whether any of these keywords is simply present without looking for a following non-alphanumeric character.

B. Lexical features. This feature set contains two features that come from the character arrangements in the URL itself. First, whether it contains semicolons to separate parameters. It could either be in the place of actual query parameters usually separated by ‘&’ or a string of semicolon parameters as part of the query parameters or even the path itself. For example, this segment of an ad-related URL: `sz=970x250;tile=1;dc_yt=1;kga=-1; kgg=-1;kmyd=ad_creative_1;ord=239235009066760?` contains several parameters separated by semicolons right before the ‘?’ for query parameters.

Binary feature	Ad	Non-Ad
Semicolon parameters	4,001	677
Keyword w/following char.	21,849	1,854
Keyword raw	26,818	6,456
Base domain in params	6,152	1,325
Valid query params	29,602	29,923
Screen/browser dimensions	81	136
Contains ad size	8,919	2,067
In an iframe	27,099	21,452
On base domain	3,086	14,350

Table 1: **True binary feature distribution for the ad and non-ad URLs respectively, each out of 30,000.**

The second feature in this set is whether the URL contains valid query parameters. Valid query parameters are formatted by being placed after the ‘?’ and then each parameter is separated by a ‘&’. Many URLs that were identified as ads violate this specification by placing all of the parameters without a preceding ’?’, so the feature evaluates to false for these examples.

C. Related to the original page. This feature set contains two features that contain information about relationships between the requested URL and its base page domain. The first feature represents whether the base domain name is present in the query parameters of the URL or in the part of the URL between the path and the query parameters (as sometimes semicolon parameters are present in this segment). Often when an ad resource is requested, the original domain name is included so that the ad server can decide what ad to render.

The second feature captures whether the requested URL is on the same domain or subdomain of the original page [18, 19] as URLs are less likely to be ad-related if they are requested on the same domain as the original page. This feature is not error-proof however, as it can backfire in certain situations like YouTube or Google ads where sometimes the ad is hosted on the base domain itself.

D. Size and dimensions in URL. Ad URLs very often contain information about the size of the ad it should display or the dimensions of the screen or browser it is going to be displayed on. One feature is if the URL contains an ad size. The format of the ad size we checked for is 2-4 numeric digits followed by the character `x` and then again followed by 2-4 numeric digits. Some ads contain the screen or browser dimensions so this second feature checks for the presence of one or more keywords from: `screenheight`, `screenwidth`, `browserheight`, `browserwidth`, `screendensity`, `screenresolution` and `browsertimeoffset`.

E. In an iframe container. This feature is binary and indicates whether the incriminating URL was requested from within an iframe either in the context of the page or in the context of nested iframes.

F. Proportion of external requested resources. The features in this set are based on the intuition that in a given page, given all the above features, a URL is more likely to be ad related if the original page requests a higher number of external iframes, scripts or external resources overall. The 3 features are the proportions of external iframe, script and resource requests to all the iframe, script and resource re-

% External	Scripts	Iframes	Resources
0-10	31,703	48,761	26,641
10-20	5,569	1,433	6,411
20-30	2,858	718	4,395
30-40	4,252	2,797	3,830
40-50	3,339	1,503	2,549
50-60	1,839	200	2,992
60-70	3,509	464	2,947
70-80	2,142	896	3,243
80-90	2,069	188	3,936
90-100	2,720	3,040	3,056

Table 2: **Value distribution for the 3 real-valued features. The right 3 columns represent the number of URLs out of all 60,000 examples whose pages request that percentage of external scripts, frames and domains.**

quests from a page respectively. These are computed for each category by the ratio between the number of URLs which point to external domains and the total number of URLs.

All of the above features sum up to a total of 12 features for each example. Table 1 shows the true binary feature distribution among the ad and non-ad URLs separately. Table 2 shows the value distribution of the 3 real-valued features between 0 and 1, which are mapped to 0% and 100% in the table.

3.3 Classification Models

To determine what classification scheme fit our data the best, we implemented several of the most common classifiers used for supervised learning. To implement the classification, we made use of the machine learning development kit in Python called Scikit-Learn [27]. Each example was encoded as a 12-dimensional feature vector and all of our datasets were in the svmlight format [16].

The following are the classification models we employed due to their suitability for binary classification problems.

Naïve Bayes: Naïve Bayes is a simple probabilistic classifier which relies on Bayes Theorem with the assumption of independence between individual features of an example. Naïve Bayes has been shown to be effective on low-dimensional data in certain situations [8]. In our evaluation, the likelihood of each feature $P(x_i|y)$ is a Gaussian distribution because of the presence of continuous features. The class is computed by the following equation:

$$y = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

Support Vector Machines: Support Vector Machines or SVMs are non-probabilistic classifiers that typically work in high-dimensional feature spaces. SVMs for binary classification are trained to construct hyperplanes with the widest possible margins to the nearest training examples. In order to not have the margins too sensitive to outliers, we used a moderate value of the penalty parameter C at training time to determine the appropriate hyperplanes. The decision rule during testing is made by determining which side of the hyperplane the point lies on and is given by the sign of the result of the following formula:

$$\sum_{i=1}^{n} y_i \alpha_i K(x_i, x)$$

Here $K(x_i, x)$ is the kernel function that implicitly maps the feature vectors into higher-dimensional space. x_i refers to each example in the training set, y_i to each corresponding class and α_i to the weight of each training example.

To increase the dimensionality of our feature vectors to be better suited to SVMs, we expanded the feature space by taking the polynomial combinations of degree less than or equal to 2, of all features. This increased the number of features from 12 to 91. Taking polynomial combinations up to higher dimensions or further polynomial combinations of the original combination results resulted in a memory error on our machines, hence we kept the dimensionality at 91. To further map the features into high-dimensional space, we tested with polynomial and radial basis function (rbf) kernels in addition to linear kernel functions.

Logistic Regression: Logistic Regression is a type of probabilistic classification model that is modeled by the relationship between a dependent variable and one or more independent variables. The decision for a point's label is determined by it's distance from a hyperplane decision boundary that is estimated at training time. Because the feature vectors are low dimensional and not sparse, we used $l2$-regularization with a moderate regularization parameter to avoid overfitting. With varying thresholds on the output, the accuracy decreased rapidly as the parameter increased above 0.5, with precision and false positive rate increasing and recall decreasing significantly at the cost of accuracy.

k-Nearest Neighbors: k-Nearest Neighbors is a popular non-parametric estimator for classification problems and can be efficient on low-dimensional data. The decision function is computed implicitly by taking the majority vote of a data point's k nearest neighbors' classes. In our model, the kNN classifier uses the distance-weighted k-nearest neighbors rule and assigns weights to the k nearest points proportional to the inverse of the Euclidian distance between each neighbor and the point in question. The weight w_i of each point x_i in the vote is given by:

$$w_i \propto \frac{1}{d(x, x_i)}$$

For this classification, we kept our modest dimensionality of 91 features and compared training and testing times using different neighbor search algorithms as seen in Table 5.

4. EVALUATION METHODOLOGY

To evaluate the fundamental accuracy of our techniques, we first use a straightforward methodology. We train the classifier on the labels determined by the older EasyList and then test against a different set of URLs with labels determined by the current EasyList filters. This method is meant to simulate a real-world use of machine learning to improve upon the traditional ad blocking method by showing how successful the classifier would be at detecting ads that may or may not be caught by current manually generated filters.

We make the simplifying assumption that the current or "new" EasyList is "ground truth": all URLs which match are true positives, and all URLs which do not match are true negatives in our results.

To evaluate the performance, we show two different types of evaluation metrics we aim to optimize. The first is a simple accuracy measure of how well the classifier trained on old EasyList data can accurately identify ads matched by a recent EasyList filter list. We use this metric, termed "baseline accuracy," to determine general effectiveness of a classifier in our experiments.

Classification Method	Avg. Accuracy	Precision	FP Rate
Naïve Bayes	89.50%	89.09%	14.3%
SVM (Linear)	92.10%	92.36%	7.4%
SVM (Poly Kernel)	90.51%	90.56%	7.34%
SVM (RBF Kernel)	92.18%	92.43%	7.7%
Logistic Regression	92.44%	92.43%	7.5%
k-Nearest Neighbors	97.55%	98.6%	1.3%

Table 3: **A comparison of the average accuracy, precision and false positive rate over 6-fold CV of various machine learning approaches to ad classification. The logistic regression was performed with a threshold of 0.5.**

Beyond recreating the functionality of EasyList, we wish to improve utility by being able to detect ads which evade EasyList, either purposefully or incidentally. Thus, the second evaluation measure is the average positive classification accuracy of ads that were caught by the new EasyList but not the old which we call "new-ad accuracy". The baseline accuracy is an indicator of whether or not the classifier can learn the basic rules of ad identification.

We computed these metrics over the entire dataset of 60,000 elements. There were 123 URLs that were caught by the new EasyList but not by the old. Assuming two sets of URLs: ad URLs caught by both old and new EasyList (A) and ad URLs caught by the new EasyList but not the old one (B), we give a succinct definition for baseline accuracy in Expression 1 and one for new-ad accuracy in Expression 2.

$$\frac{\text{Number of positively classified URLs in set A}}{\text{Number of URLs in set A}} \tag{1}$$

$$\frac{\text{Number of positively classified URLs in set B}}{\text{Number of URLs in set B}} \tag{2}$$

To evaluate the average classification accuracy of each classifier, we used k-fold cross validation (CV) with k=6. In each fold, the training set uses the labels from the "old" dataset and the testing set uses labels from the "new" dataset. By using k-fold CV, we ensure that the classifier is trained on every example at one point as opposed to random splits where some examples might be excluded all together. In comparing the overall effectiveness of the classifiers, we looked at the overall average accuracy on the testing data as well as the precision (true positives/(true positives + false positives)) and false positive rate (false positives/(false positives + true negatives)), as we would like to minimize false positives over false negatives and maximize the overall positive classification accuracy.

Figure 1 shows the baseline accuracy and new-ad accuracy for the six different classification methods discussed in the previous sections.

In the following section we will discuss the results of different classifiers on the datasets to determine the most effective algorithm. Then, we test the classifier on the different feature sets and compare the accuracies and relevant scores.

5. RESULTS

Table 3 shows a comparison of accuracy measures among different classifiers. The training was done on the URLs matching old EasyList data and tested on new EasyList-matched URLs for scoring. For the comparison on classifiers, the original feature space was 12-dimensional, i.e. using all the features.

Naïve Bayes did not perform as well as expected with a rather high false positive rate and low baseline accuracy. Despite its low baseline accuracy, Naïve Bayes showed substantial promise at classifying new positive ads as shown in Figure 1.

Similarly low accuracies were achieved with SVMs using different kernels and $l2$-logistic regression at different thresholds with the error rate never going below 7% over multiple iterations of the tests and by tuning the the various parameters of the models.

k-Nearest Neighbors proved to have the lowest error rate and false positive rate and highest precision overall. Performing the classification with different search algorithms, the training and testing times can be seen in Table 5.

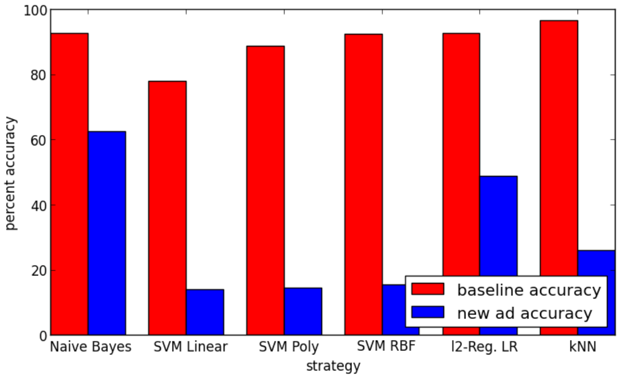

Figure 1: **Baseline accuracy and new-ad accuracy for the six different machine learning approaches. New-ad accuracy is out of a total of 123 examples.**

Figure 2 shows the Receiver Operating Characteristic (ROC) for the kNN classifier [11]. The thresholds were taken to be the number of positive neighbors k to k-(k-1) of a point, with the resulting class labels determined as only a function of having at least the threshold number of positive neighbors. The figure is zoomed in to focus on the parts of the curve where the TP rate and FP rate are optimal and above the diagonal which is shown as the straight line in the lower half of the figure.

Baseline and New-Ad Accuracy: Figure 1 shows the baseline accuracy and new-ad positive accuracy for the 6 different classification methods discussed in the previous sections.

As expected, the kNN classifier had the highest baseline accuracy. However Naïve Bayes and $l2$-regularized Logistic Regression performed the best on the new advertisements and even have decent baseline accuracies. Despite this, Naïve Bayes still wouldn't be considered optimal because of its relatively low overall accuracy, low true positive rate and positive prediction capabilities (86-89%), which are crucial metrics in this domain. From Table 4, we can see that the new-ad accuracy increased up to almost 50% for different feature sets. In each case, it is clear that there is a tradeoff between this improved accuracy, the overall accuracy and baseline accuracy.

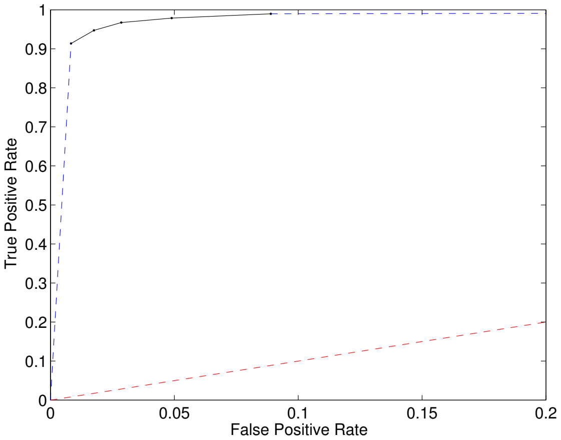

Figure 2: **Receiver Operating Characteristic curve. The region with FP rate 1% or less contains the optimal points of the curve.**

Feature Set (f)	Accuracy	Baseline Acc.	New-ad Acc.
A	90.21%	81.82%	48.78%
B	97.42%	95.20%	48.78%
C	96.82%	95.16%	34.96%
D	95.94%	93.38%	27.64%
E	96.22%	94.21%	21.95%
F	76.88%	57.50%	9.76%

Table 4: **Average accuracy, baseline accuracy, and new-ad accuracy without each feature set (f).**

5.1 Performance of kNN Classifier

Because we found k-Nearest Neighbors to be the most overall effective algorithm for classification, here we discuss different ways we tried to tune it even further by scoring over different feature sets and varying values of k neighbors.

Varying the number of neighbors k, the performance significantly decreased with more than five neighbors likely due to overfitting, so five was the standard k-value we used throughout our experiments.

Table 4 shows the accuracy of the classifier using all the feature sets except for the one listed where the letter corresponds to the matching feature set from Section 3.2.

Feature set A and F appear to have the most impact on the classifier with the absence of F dropping the average accuracy to a startlingly low 76%. Absences of all the other feature sets decrease the accuracies by 1-3% each, so we do not exclude any of these either. Hence, we conclude that all 12 features are effective for achieving high overall accuracy on the dataset.

The runtime of the prediction algorithm is dependent on the type of search algorithm used. There are many search algorithms, of which brute force, KD-tree and ball-tree are very common. Brute force is the most simplistic method but has a very large runtime in n where n is the number of training examples. In this case, n is extremely high, on the order of tens of thousands. hence it doesn't seem practical in a production environment. We experimented with the KD-tree and Ball-tree search algorithms under different granularities of cross validation, the training and times of which are listed in Table 5 along with the number of folds the testing was done in. It is clear that the KD-tree outperforms the ball-tree search algorithm with such an abundance of data points.

# Folds	Train Time	KD-Tree	Ball-Tree
3	2.81	4.23	12
4	3.41	3.59	10.68
6	4.51	2.36	7.71

Table 5: **Average training and testing times in seconds of kd-tree and ball-tree search algorithms for different values of k in k-fold CV.**

6. ANALYSIS

In this section, we will analyze the results from the previous section as well as discuss possible causes for errors in classification and their impact in real world ad blocking.

6.1 Minimizing False Positives

A false positive in ad-blocking can be seen in the form of real content that is mistakenly blocked on a web page. Minimizing false positives is of utmost importance as it is extremely off-putting for a user to miss out on content that is misclassified as advertisement-related when it could actually be crucial to the functioning of the webpage. From Table 4, we can see that the false positive rate according to our ground truth is around 1% on average using kNN. While a very minute percentage, it is a non-trivial amount of falsely categorized resources.

To understand the cause for this better, we first checked all the false positives against a very recent EasyList version from June 7th, 2014. There were an average of 27 matches out of the 419 false positives against this list, over different tests. On manual inspection of the remaining URLs and their feature vectors, it is obvious that an extremely high percentage of these are actually advertisement URLs that were simply not caught by the "new" EasyList. That is, approximately every 7 out of 10 false positives were determined to be ad-related based on human observation. Figure 3 shows a few examples that were erroneously classified as false positives that are clearly advertisement or tracking URLs. Using the EasyList blacklist of third-party advertisers, some of these would likey be caught, but our classification, with the goal of avoiding blacklists, classified these as ad-related or tracking without it.

In the manual world of ad-blocking, these could be accommodated in the form of updated filters over time, based on user feedback or addition of new ad servers to the list. New kinds of advertisements are not necessarily accommodated, as it can be difficult to construct specific regular expressions to match them in limited ad-related contexts. In this case, "ground truth" simply refers to the current level of human recognition of ad-related regexes and requires constant updates. While crafting regular expressions to both filter new ads and not induce new false positives is a taxing affair, our results show that a feature based learning approach can automatically classify many of these new ads without a high false positive rate.

6.2 Baseline and New-Ad Accuracy

While our kNN classifier was able to maintain an extremely high baseline accuracy and extremely low baseline false positive rate, the low new-ad accuracy is low enough to be

```
http://ums.adtechus.com/mapuser?providerid=1024;userid=ochyuqazeolxp&cfp

http://t1.visualrevenue.com/?idsite=516&url=http%3A%2F%2Fwww.cbsnews.com%2F&seen_url=http%3A%2F%2Fwww.cbsnews.com
%2F&t=CBS%20News%20-%20Breaking%20News%2C%20U.S.%2C%20World%2C%20Business%2C%20Entertainment%20%26%20Video&
c=1395434846773WiyhLqq65BlwV3Jtv9UE3BO1tOC6oOuP&r=1395434846775&ypos=-1&debug=loadTime.3&content_type=UTF-8&v=11

http://l.betrad.com/pub/p.gif?pid=254&ocid=1155&ii=1&r=0.01754436711780727

http://ad.dedicatedmedia.com/seg?add=581028&t=2
```

Figure 3: **Examples of false positives that are advertisement-related.**

undesirable. However, this corresponds to a low false negative rate on new-ads which can be considered a significantly more tolerable tradeoff for low false positives with respect to the user experience. Furthermore, false negatives for both baseline and new ads in this case could be tuned by certain feedback based improvements to our learning algorithms as part of future work.

7. DISCUSSION

In this paper, we show as a proof of concept, how a classifier trained on a few features can efficiently determine current ads from old filters, new ads and ads that haven't been discovered yet. We do this through batch training and testing of relevant current-day ads determined by the current-day ad blocking filters. The low dimensionality of the data is beneficial to the overall prediction run-time and the features outlined can be extended for use in other kinds of research regarding ad-ware and other unwanted internet resources. The reason that ad blockers are getting updated frequently is due to the fact that ad networks are coming up with new schemas of URLs that get injected into a page either to combat blockers or to spread different kinds of content. It is this ability to keep up with new trends in advertisements that our classifier should acquire.

Future work suggests a way to transform the concept into a real-time learning system that trains on old ads, but constantly updates its labels incrementally every time a new type of ad example is discovered. An ideal classifier for this purpose would grow as the domain of ad structures grows and changes over time. To maintain the supervised nature of the classifier, the best kind of approach would be semi-supervised incremental learning. As k-Nearest Neighbors was the most promising algorithm to determine accurate labels on old examples, we propose future steps to utilize the same approach in an online learning environment. Because k-Nearest Neighbors is a non-parametric estimator, the probabilistic output would be determined as usual by the majority vote of the neighbors' respective classes. However, as the number of examples increases, kNN can become slow and inefficient and might be subject to overfitting because of high density in certain regions over time. Possible implementation solutions could be in clustering of neighbor groups to condense the number of data points in a given region similar to in self-organizing maps [17], keeping an overall time and accuracy efficiency on the data.

8. FUTURE WORK

While we were able to achieve high accuracy, precision and low false positive rates on old and new advertisements, ad URLs are constantly updating and changing their structure to combat ad blockers and hence there needs to be a way for a classifier to periodically train on new data. To improve the new-ad accuracy, the classifier would need to be able to keep up with the new types of ads and update its model periodically to accommodate these new rules. A possible implementation would be to have a feedback mechanism browser extension, which allows a user to identify ad elements, similar to feedback-based spam filters in email clients. All the URLs requested within this element could be added to the trainer with their features computed. To maintain a balance of examples between the two classes, for every ad URL label incoming, one non-ad URL can be added to the training data. This sort of feedback mechanism learning could be applied to our proposed semi-supervised online learning approach with initial training done on the most recent EasyList data.

9. CONCLUSION

While the traditional ad blocking model has been successful and efficient since its creation in 2002, the manual efforts behind the scenes might be better placed elsewhere. Undesirable resources that could be irritating, privacy-violating or maliciously loaded as ads, need a more automatic way to be detected in the future. The landscape of ads is continuously changing either to embody more security threats or simply to combat ad blockers. We have developed a machine learning based classifier for ads which was able to automatically learn currently known ads with a high accuracy and upto 50% of new ads that would otherwise necessitate manual filter creation. Our classifier was also able to detect a large number of ad and tracking URLs that current filter lists aren't able to identify, either due to their lack of defining URL tokens from which filters can be constructed or because the filters matching these haven't been manually identified yet. We believe that this line of research has the potential to further enable user choice regarding what ads, tracking beacons, and other undesirable web assets are loaded on their machines, improving several aspects of the experience and web security for web users.

10. ACKNOWLEDGEMENTS

The authors would like to acknowledge the help of MonztA at EasyList and would also like to thank the anonymous reviewers for their helpful feedback. This work was made possible in part by NSF grant CNS-1351058.

11. REFERENCES

[1] Easylist statistics: August 2011. https://easylist.adblockplus.org/blog/2011/09/01/easylist-statistics:-august-2011,
2011.

[2] 10 best ad blockers and privacy extensions. http://www.tomsguide.com/us/pictures-story/565-7-

best-adblockers-privacy-extensions.html,
2013.

[3] Disconnect: Online privacy and security. https://disconnect.me/, 2014.

[4] Easylist devlopment. https://easylist.adblockplus.org/en/development, 2014.

[5] Easylist mercurial changelogs. https://hg.adblockplus.org/easylist/, May 2014.

[6] Adblock Plus community. EasyList. https://easylist.adblockplus.org, 2014.

[7] Alexa. Alexa top sites. http://www.alexa.com/topsites, 2013.

[8] Aydemir, O., Ozturk, M., and Kayikcioglu, T. Performance evaluation of five classification algorithms in low-dimensional feature vectors extracted from eeg signals. In *34th International Conference on Telecommunications and Signal Processing (TSP)* (2011).

[9] Beil, A. Q1'10 web-based malware data and trends. http://blog.dasient.com/2010/05/q110-web-based-malware-data-and-trends.html,
2012.

[10] Electronic Frontier Foundation. Privacy badger. https://www.eff.org/privacybadger, 2014.

[11] Fawcett, T. An introduction to roc analysis. In *Pattern Recognition Letters* (2006).

[12] Garera, S., Provos, N., Chew, M., and Rubin, A. D. A framework for detection and measurement of phishing attacks. In *WORM '07 Proceedings of the 2007 ACM workshop on Recurring malcode* (2007).

[13] Gonzalez, N. Half of destructoid's readers block our ads. now what? http://www.destructoid.com/half-of-destructoid-s-readers-block-our-ads-now-what--247904.phtml, 2013.

[14] Gundlach, M. Adblock. https://getadblock.com/, 2014.

[15] Hurps, M. Ad blocker: Ad muncher. http://www.admuncher.com/, 2012.

[16] Joachims, T. Svm-light support vector machine. http://svmlight.joachims.org, 2008.

[17] Kohonen, T. The self-organzing map. In *Proceedings of the IEEE* (1990).

[18] Krammer, V. An effective defense against intrusive web advertising. In *Privacy, Security and Trust, 2008. PST '08. Sixth Annual Conference on* (2008).

[19] Krishnamurthy, B., and Wills, C. E. Cat and mouse: Content delivery tradeoffs in web access. In *WWW '06 Proceedings of the 15th international conference on World Wide Web* (2006).

[20] Kushmerick, N. Learning to remove internet advertisements. In *Proceedings of the third annual conference on Autonomous Agents* (1999).

[21] Ma, J., Saul, L. K., Savage, S., and Voelker, G. M. Beyond blacklists: Learning to detect malicious web sites from suspicious urls. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (2009).

[22] McDonald, M. The adblock project. http://adblock.mozdev.org/, 2002.

[23] Nethercote, N. Adblock plus's effect on firefox's memory usage. https://blog.mozilla.org/nnethercote/2014/05/14/adblock-pluss-effect-on-firefoxs-memory-usage/, 2014.

[24] Nock, R., and Esfandiari, B. On-line adaptive filtering of web pages. In *9th European Conference on Principles and Practice of Knowledge Discovery in Databases, Porto, Portugal, October 3-7, 2005. Proceedings* (2005).

[25] Orr, C. R., Chauhan, A., Gupta, M., Frisz, C. J., and Dunn, C. W. An approach for identifying javascript-loaded advertisements through static program analysis. In *Proceedings of the 2012 ACM workshop on Privacy in the electronic society* (2012).

[26] Palant, W. Adblock Plus. https://adblockplus.org, 2014.

[27] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research 12* (2011), 2825–2830.

[28] Signanini, J. M., and McDermott, B. Ghostery. https://www.ghostery.com/en/, 2014.