Adaptive Filtering of Advertisements on Web Pages

Babak Esfandiari
Dpt of Systems and Computer Engineering
Carleton University
1125 Colonel By Drive
Ottawa, Ontario, Canada K1S5B6
babak@sce.carleton.ca

Richard Nock
GRIMAAG-DSI / Univ. Antilles-Guyane
Campus de Schoelcher
B.P. 7209, 97275 Schoelcher Cedex
Martinique, France
rnock@martinique.univ-ag.fr

ABSTRACT

We present a browser extension to dynamically learn to filter unwanted images (such as advertisements or flashy graphics) based on minimal user feedback. To do so, we apply the weighted majority algorithm using pieces of the Uniform Resource Locators of such images as predictors. Experimental results tend to confirm that the accuracy of the predictions converges quickly to very high levels.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent Agents*; H.4.3 [**Information Systems Applications**]: Communications Applications—*Information Browsers*

General Terms

Algorithms, Theory, Experimentation

Keywords

Advertisement Filtering, Weighted Majority, Interface Agents

1. INTRODUCTION

Many attempts have been made to make Web browsing more pleasant by allowing the user to remove big pictures and unwanted animations that interfere with reading. Some browsers such as Netscape or Mozilla allow the user to collapse such pictures or even create blacklists of internet domains that supply them.

But the most sophisticated approach so far has been proposed by the developers of AdBlock. AdBlock [1] is, according to "Mozdev update" data [6], in the top ten of the most popular extension to the Mozilla Firefox web browser [5], with about 100000 downloads. To use AdBlock, the user has to come up with a collection of regular expressions that describe the URL patterns of images that they want to see filtered. As a result, whenever the browser is pointed to an item whose URL is matched by a regular expression, it is simply ignored, which not only "cleans up" the web page, but also makes page downloading faster.

However, as often discussed in the AdBlock online forum, coming up with regular expressions is a difficult task, especially for the non-computer savvy. Moreover, it would be impossible to come up with a general set of filters that would satisfy every user. Finally, as the advertisement suppliers and browsing habits change, so should the set of regular expressions that are needed.

To address this problem, we propose a machine learning approach that would create filters based on minimal interaction with the user. The user will not need to know how to create regular expressions; all that is required is for the user to click on images that he/she wants to see blocked. Conversely, from time to time the user will need to unblock images that shouldn't have been blocked by the adaptive filter. Based on this simple feedback, our proposed method, an adaptation of the Weighted Majority algorithm [4], will build a set of "experts" (chunks of URLs) that will vote on whether a given image should be blocked or not. Our approach draws on previous works such as [3, 7]. Section 2 presents the algorithm. Section 3 presents the browser extension, and Section 4 presents experimental results. Section 5 concludes.

2. THE ALGORITHM

Algorithm 1: `Receive_New_Example`$((x_t, y_t))$

Input: example (x_t, y_t)

$\mathbf{N} \leftarrow$ `Create_Hypotheses`$((x_t, y_t))$;
`Update_Experts` $(\mathbf{N})$;
foreach $(h, w_t(h)) \in \boldsymbol{E}$ **do**
  $w_{t+1}(h) \leftarrow w_t(h) \times \left(\frac{1+y_t h(x_t)}{2\beta} + \frac{(1-y_t h(x_t))\beta}{2}\right)$;
$t \leftarrow t + 1$;

We denote a couple (observation, class) obtained from the user as an *example*. An observation is a regular expression, and a class belongs to the set {block, unblock}, which we code respectively $\{+1, -1\}$, and refer to as the positive (because we try to predict the "block" status) and negative class. We let $(x_1, y_1), (x_2, y_2), ...$ denote the stream of examples observed from the user, and (x_t, y_t) is thus the t^{th} example of the stream. We build a set of experts $\mathbf{E}$ which is growing with time. Each expert of $\mathbf{E}$ is a couple (hypothesis, weight). An hypothesis is a function h :

WWW 2005, May 10–14, 2005, Chiba, Japan.
ACM 1-59593-051-5/05/0005.

$X \rightarrow \{-1, 0, +1\}$ which is allowed to *abstain* (this is the output "0"). More precisely, each hypothesis' output is either $\{-1, 0\}$ or $\{0, +1\}$, which means that the corresponding expert is authorized to say "I don't know", thus delegating the decision on the class of an observation to the other experts. The weight associated to hypothesis h is denoted $w_t(h) \in \mathbb{R}^+$. It is a function of t since it is updated each time an example is received. At the very beginning of the algorithm, prior to seeing the first example, we initialize the following set of parameters: $\beta \in (0, 1)$ is a learning constant chosen by the user; $\mathbf{E} \leftarrow \emptyset$ is the initial set of experts; $t \leftarrow 1$ is the "time stamp" labeling the examples received. Algorithm 1 above displays more formally what happens when example (x_t, y_t) is received. Function `Update_Experts` takes the new hypotheses from $\mathbf{N}$ and create as many experts, whose weights are initialized to 1. Then, it puts them into $\mathbf{E}$. The class assigned to an arbitrary URL is obtained by making a weighted majority vote of the experts of $\mathbf{E}$. The sign of the output gives the class assigned. Its absolute value may be thought as a confidence in the choice; whenever the weighted vote returns zero, a random class choice is made.

3. DESIGN OF THE BROWSER EXTENSION

Our filtering algorithm and the test drivers were both implemented as extensions to the Mozilla Firefox Web browser [5] in Javascript.

Our extension provides two extra menu items in the browser's context menu: "Block Me" appears only when the user right-clicks on an URL (*e.g.* an image) that he/she wishes to block; "Unblock" is always available should the user want to unblock an URL that appears to be blocked by mistake.

To generate the new set of experts $\mathbf{N}$ in Algorithm 1, we tokenize the example URLs using the character "/" as delimiter. The tokens obtained represent items such as domain names and folders. They are then compared with the existing set of experts. If no match is found, the new token is added to the corresponding list of experts using function `Update_Experts` above.

4. EXPERIMENTAL RESULTS

In our experiments, we have fixed $\beta = 1/\sqrt{e} \approx 0.61$. At each step consisting of k examples (*e.g.* visited image URLs), we freeze a copy of the learner's knowledge base up to that point. While the "unfrozen" version keeps evolving and accepting feedback from the oracle, the "frozen" copy is used to evaluate the learning accuracy of the accumulated knowledge so far by populating a confusion matrix based on its predictions on the incoming examples. After n such steps, and for a total of $n \times k$ examples, the user is notified that the testing is finished. The resulting logs store $n - 1$ confusion matrices.

Our first set of tests were designed to see whether our algorithm was able to correctly predict which URLs to block on a single "busy" (*i.e.* littered with annoying images) web page, and if so, after how many visits. We used a common set of AdBlock regular expressions, as collaboratively devised on AdBlock discussion forums, as oracle. We set $k = 10$ and $n = 10$. The total 100 examples were usually quickly reached after only hitting the "reload" button a few times.

Figure 1 (left) traces the evolution of learning accuracy over the 10 steps. As can be seen on that figure, except

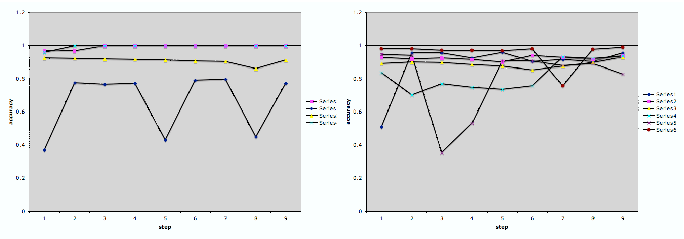

Figure 1: Evolution of accuracy on a single site (left), and during a typical browsing session (right). Each curve represents a browsing session.

for one test, the algorithm converges very quickly (in fact usually within *one* step) to nearly perfect prediction. This corresponds to less than two reloads of the same page, which is good given that commercial web sites use dynamic loading of advertisements using cookies, and as a result hitting reload usually brings up a different set of images and URLs.

The next set of tests measures robustness to overfitting. We asked the users to simply follow their usual browsing habits, and we set $k = 100$ (to absorb some of the variability) and $n = 10$. It is worth noting that the total of 1000 examples was reached very quickly. The results are charted on figure 1 (right). This time a few dips can be observed on the learning curve. This happens when the learner is faced with a set of fresh URLs that it usually misclassifies as false negatives at first.

5. CONCLUSION

We presented a method to dynamically create custom filters to avoid downloading unwanted URLs with minimal interaction with the user, using a weighted-majority type algorithm. Both the standalone extension (AdBlockLearner) and the test driver (AdBlockLearnerTest) are available in [2], including source code and documentation. They are compatible with most versions of Mozilla Firefox.

6. ACKNOWLEDGMENTS

We would like to thank the AdBlock team for their timely help and enthusiasm for our ideas. R. Nock would like to thank Ottawa University and Stan Matwin for an invitation grant, during which part of this work was achieved.

7. REFERENCES

[1] AdBlock, 2005. http://adblock.mozdev.org.

[2] B. Esfandiari and R. Nock. Adblocklearner project page, 2005. http://adblocklearner.mozdev.org/.

[3] Y. Lashkari, M. Metral, and P. Maes. Collaborative interface agents. In *Proc. of AAAI-94*, pages 444–449, 1994.

[4] T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[5] Mozilla Firefox, 2005. http://mozilla.org/products/firefox.

[6] Mozilla Firefox extensions, 2005. http://update.mozilla.org/extensions/.

[7] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A bayesian approach to filtering junk email. In *AAAI Workshop on Learning for Text Categorization*. AAAI Press, 1998.