Spoilers Ahead - Personalized Web Filtering

Pascal Bissig, Philipp Brandes, Roger Wattenhofer, Roman Willi
ETH Zurich
firstname.lastname@ethz.ch

***Abstract*—Unwanted content on web pages can take many forms, be it ads, malicious code, pointless clutter, or specific topics that the user does not want to read about (yet). Unlike most other work, we focus on the latter. The user can define terms based on which we prevent the disclosure of undesired information (e.g., the latest sports result) and warn the user before it is revealed. The user can decide if and when the filtered elements should be displayed. We define this formally as the node removal problem and show its equivalence to the $\mathcal{NP}$-hard knapsack problem. We developed a proof of concept Firefox extension to filter web pages based on user defined terms and our heuristic. Our evaluation shows that we correctly distinguish between wanted and unwanted content in approximately 9 out of 10 cases.**

***Keywords*-personalization techniques, web personalization, filtering**

I. Introduction

Be it Ads, spam or malicious code, Unwanted content on the web can take many forms. While there exist great filters for ads, spam and malicious code, all these problems share the property that one filter mostly fits all users. For example, if one user considers content to be an ad, most other users will do so as well. The same holds for malware and spam. This means that filters can be constructed once and applied to all users which is how ad- or malware filters usually work. In addition to that, ad filters [1] as well as spam filters heavily rely on blacklisting of hosts that serve either content type.

In this paper, we focus on filtering user specific content. For scalability reasons it is impossible to have a specialized filter for each user specific topic. Even worse, classical host blacklisting techniques do not apply in our case since undesired content may very well be served by the same host as the content the user wants to see. Think of a news site that reports on the latest sports results. Since you did not watch the event yet, you do not want to know the result but may still be interested in the latest political developments, which are shown on the same web page. The upside in our scenario is that we are not competing with companies selling ads or black-hats distributing malware.

We present a personalized filter that removes content from a web page based on user specified terms or topics. The running example in this paper is a spoiler such as a plot development in a TV show or a sports result. Some communities have dealt with this by requiring spoiler tags but this requires manually tagging every post and is topic but not user specific. However, our filter is universally applicable to remove undesired content and we use the notion of a spoiler just as an example.

HTML objects are by definition organized in a tree structure. The Document Object Model Tree (DOM Tree) maps an HTML document to a tree structure whose nodes can be addressed and manipulated easily. These nodes include text, links, images, and all other content displayed on a website. We replace nodes that contain undesired content with placeholder nodes that reveal the original content when clicked. This preserves the overall layout of a web page when removing undesired content.

We explore the tradeoff between removing as many spoilers as possible while not removing too much unrelated content. Only hiding all user defined terms on a web site does not lead to the desired outcome since spoilers will be revealed by text or pictures nearby. For example, sentences like "[redacted] arrested for drunk driving" or "The [redacted] won the Super Bowl 28–24" still reveal a lot of information even though the spoiler terms are hidden. Especially so when we consider the fact that the user herself defined which terms to filter. Blocking the whole web page greatly diminishes the user experience when applied to sites that serve content that is mostly unrelated to the users filter terms, yet contains a small section that reveals spoilers. Thus, neither of the two extremes – solely hiding all user defined terms or blocking the whole web page – are desirable. Unfortunately, the middle ground – removing as many bad words from the web page while removing as little good words as possible – turns out to be an $\mathcal{NP}$-hard problem.

Hence, we use a heuristic to exploit the locality of content in the DOM tree to filter entire paragraphs or sections of a website that may contain spoilers.

II. Related Work

There is a large number of different content filters. Designed to filter specific undesired content types, they span from lightweight browser based filters up to search-engine filters, which may offer safety filters to exclude inappropriate links from the search results. In between those two approaches, there are solutions such as network-based filtering and content-limited filters offered by Internet service providers [2].

Classic Content Filtering Problems. Advertisements and malware are content types that received a lot of attention. Both content types share the property that different people share a common view of what qualifies as malicious content or ads. This means that filtering such content comes down to the task of identifying content that should be filtered once. The obtained list of elements which should be hidden can then be shared with all users such that the filtering logic on the client side is simple and fast.

One example of such a filter is Prophiler [3]. The focus of the system is set on finding websites that serve malicious JavaScript code to its visitors. Malicious behavior is detected by executing JavaScript in a virtual environment which tracks behavior such as drive-by downloads. While such systems identify malicious scripts with high accuracy, it is computationally intensive to sift through large amounts of websites. To reduce the computational requirements, Prophiler uses machine learning to quickly discard benign websites. The results can be used to efficiently blacklist websites that serve malicious code to its visitors.

ZOZZLE [4] is a browser plug-in that recognizes JavaScript malware, through static code analysis, while a user is visiting websites. Our method follows ZOZZLE's idea of filtering content within the clients browser without using global blacklists. However, since malware is still the same for most users, we do not see the advantage of such a solution when filtering malware instead of user specific content. The growing amount of junk email has led to the development of email spam filters [5]. The arms race between spammers and filter providers has fueled the development of a large number of different filtering mechanisms ranging from host blacklisting [6] to content analysis [7].

User Specific Content Filtering. Systems that allow users to filter content based on personal preferences or circumstances have also been proposed before. While malware- and Ad-filters are vastly popular, user specific filters, to our knowledge, are rarely applied. Since user specific topics should be filtered, building a global blacklist is usually not practical since each topic would require a custom blacklist. This is the main reason why such filters, in general, are implemented to filter content on the client machine. Existing filters do not apply to both: general websites and arbitrary topics.

Goldbeck et al. [8] filter tweets according to user defined TV shows or sporting events. However, their filtering mechanism is only applied to tweets. Using blacklist creation methods that are specific to the given scenario (TV shows or sporting events), the system hides undesired content with high success rates. However, to achieve this performance, many tweets that do not contain any spoilers were hidden. While our approach uses user defined keywords similar to the ones presented in this paper, we do not limit the application scenario to twitter and topics can be chosen freely. Guo et al. [9] use Latent Dirichlet Allocation to hide spoilers in movie reviews found on IMDB. The effectiveness of the system is remarkable. However, its application range is strongly limited due to the focus on IMDB in both design and evaluation of the method. Boyd-Graber et al. [10] show that crowd-sourcing is a viable option to obtain annotated training data for their spoiler filter.

III. Model

We model a website as a rooted tree $G = (V, E)$ with n being the number of nodes $|V|$. Each node $v \in V$ contains $b(v)$ many keywords (bad words) and $g(v)$ many unrelated (good) words. Note that inner nodes of the tree can and do contain good and bad words. We can remove any node v from the graph, we denote this with *cutting*. Upon removing v, by extension, all its children are removed from the graph as well. We denote with $\beta(v)$ and $\gamma(v)$ the sum of the bad and good words of v and all of its children, respectively. Hence, $\beta(v) = b(v)$ for every leaf v of the tree. Let R denote the set of nodes that were removed. This set includes the children of explicitly removed nodes. The user has to specify a threshold t. Our goal is to remove as many bad words as possible subject to $\frac{\sum_{v\in R} g(v)}{\sum_{v\in R} b(v)} \leq t$. We call this the *node removal* problem.

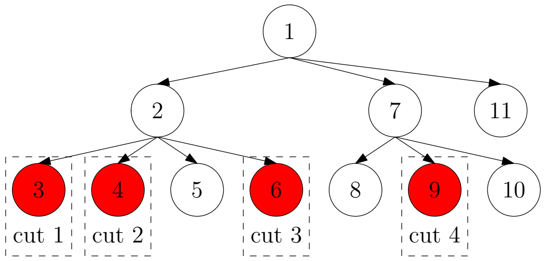


Fig. 1: Placing the cuts only at the leaf nodes to filter the keywords

A. Example

We show a simplified example in Figure 1. The four nodes containing keywords are shown in red and the corresponding text of a node is replaced by the ID of that node.

In order to filter all nodes containing keywords at the leaf node level, we need to perform four cuts in the tree at nodes 3, 4, 6, and 9 (indicated by a box surrounding the corresponding node). If we use the minimal number of cuts, namely one cut at the root node 1, we remove the whole web page.

IV. $\mathcal{NP}$-Hardness

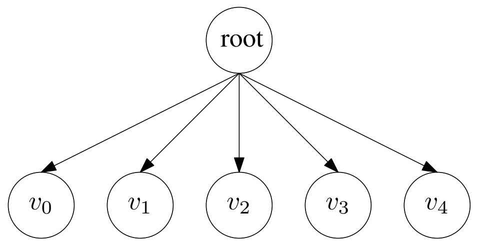
We now show that finding the best set of cuts for a given tree is $\mathcal{NP}$-hard. We show this by a reduction from the knapsack problem, which is well known to be $\mathcal{NP}$-hard [11].

Theorem 1. *The node removal problem is $\mathcal{NP}$-hard.*

Proof: We briefly describe the traditional knapsack problem before we present the main idea of the reduction. There are n items $z_1, \dots, z_n$ and each item z_i has weight w_i and value y_i. The knapsack has a capacity of W. The task is to pack the knapsack and maximize the value of the set S of items that are in the knapsack without putting too many items in it, i.e., maximize $\sum_{z_i \in S} y_i$ subject to $\sum_{z_i \in S} w_i \leq W$.

We will mimic the knapsack problem in our graph. Each node v that we remove will add value (bad words), but use up space (removes too many good words compared to the number of bad words, i.e., $\frac{g(v)}{b(v)} > t$). The size of the knapsack is mimicked by a single node v_0 that has $b(v_0) = W$ and $g(v_0) = 0$. Hence, maximizing the number of bad words that are removed while staying below the threshold is equivalent to maximizing the value of items in the knapsack while staying below the capacity of the knapsack.

Given an instance I of the knapsack problem with $I = (W, (z_1, \dots, z_n))$ with each item $z_i = (w_i, y_i)$ and size of the knapsack W, we create a rooted tree $G = (V, E)$ with $n + 2$ nodes and set the threshold t to any positive number. The root node has $n + 1$ children, $v_0, \dots, v_n$. Note that this tree has depth 1. Node v_0 has $b_0 = W$ and $g_0 = 0$. Every



nodes	v_0	v_1	v_2	v_3	v_4
$g(v_i)$	0	5	3	7	4
$b(v_i)$	5	3	1	4	2
node removal problem					

items		z_1	z_2	z_3	z_4
w_i		2	2	3	2
y_i		3	1	4	2
knapsack problem					

Fig. 2: Instance of knapsack problem with $W = 5$ (and therefore $b(v_0) = 5$) and the transformed instance of the node removal problem. We set $t = 1$. Thus, the number of bad words $b(v_i)$ is the value of the item z_i, and the number of good words $g(v_i)$ is the sum of the value and the weight of the item z_i. The optimal solution S of the knapsack problem is z_2 and z_3 (total value is 5). The optimal solution R of the node removal problem is v_0, v_2, v_3. The number of bad words that are removed is $5 + 2 + 3$. The ratio is $\frac{\sum_{v_i \in R} g(v_i)}{\sum_{v_i \in R} b(v_i)} = \frac{0+3+7}{5+2+3} \leq 1 = t$.

node v_i with $1 \leq i \leq n$ has $b(v_i) = y_i$ and $g(v_i) = w_i t + y_i t$. Hence, there is a bijection between the items $z_1, \dots, z_n$ from the knapsack instance and the nodes $v_1, \dots, v_n$ of our graph. A simple example of this construction is depicted in Figure 2.

We now claim that if and only if there exists a set of items S such that $\sum_{z_i \in S} y_i \geq Y$ and $\sum_{z_i \in S} w_i \leq W$, then there exists a set of nodes R in the node removal problem where we remove at least $W + Y$ many bad words from the tree while preserving $\frac{\sum_{v \in R} g(v)}{\sum_{v \in R} b(v)} \leq t$.

Let S be such a set. We claim that if we remove $R := S \cup \{v_0\}$ from the tree, then we have $\frac{\sum_{v \in R} g(v)}{\sum_{v \in R} b(v)} \leq t$ and we remove at least $W + Y$ many bad words. Since $\sum_{v_i \in R} b(v_i) = W + Y$ by construction, we now look at the ratio. We obtain

$$\frac{\sum_{v_i \in R} g(v_i)}{\sum_{v_i \in R} b(v_i)} = \frac{\sum_{v_i \in R} (w_i t + y_i t)}{W + \sum_{v_i \in R \setminus \{v_0\}} y_i} \leq \frac{Wt + Vt}{W + Y} = t,$$

which establishes the claim.

Let R be the set, which, if removed, removes $W + Y$ many bad words from G. It is easy to see that v_0 must be part of any optimal solution. We know that $\sum_{v_i \in R} b(v_i) \geq W + Y$ and therefore

$$\begin{aligned} \frac{\sum_{v_i \in R} g(v_i)}{\sum_{v_i \in R} b(v_i)} &= \frac{\sum_{v_i \in R} (w_i t + y_i t)}{W + \sum_{v_i \in R \setminus \{v_0\}} y_i} \\ &= \frac{\sum_{v_i \in R} w_i t + Yt}{W + Y} \leq t \end{aligned}$$

i.e., $\sum_{v_i \in R \setminus \{v_0\}} y_i = Y$. Since $\frac{\sum_{v_i \in R} w_i t + Yt}{W+Y} \leq t$ holds, we know that $t \sum_{v_i \in R} w_i \leq tW$ and thus those items fit in the knapsack.

This establishes the claim. ■

Hence, the problem is $\mathcal{NP}$-hard. Even though there exists a factor 2 approximation and even FPTAS for the knapsack problem [11], we chose not to use either of these algorithms. We do so for two reasons. None of these algorithms takes the tree structure of the web page into account. Furthermore, even the simple ones require the nodes to be sorted according to their ratio of good to bad words. This takes $\mathcal{O}(n \log n)$ time. Hence, we opt for a simpler algorithm that runs in linear time (in the number of nodes) to ensure a smooth surfing experience.

V. Concept

Since the initial problem is $\mathcal{NP}$-hard, we decided to use a heuristic to select which nodes to remove. This heuristic is based on the following assumptions:

- A node v' that is close to a node v that contains bad words has a higher chance of revealing unwanted content than nodes further away (close in the number of hops in the tree).
- Every node v that contains bad words must be removed.

The reason for the first assumption is based on the fact that these nodes are also close on the web page as it is shown to the user. Hence, the content of these nodes tends to be closely related and thus node v should be considered for removal. The second assumption ensures that we remove every user defined keyword. Our algorithm traverses the graph three times; each time starting from the root. Since the values $\beta(v)$ and $\gamma(v)$ at every node v are not known to us in the beginning, we store these during the first traversal of the tree. During the second traversal, we consider each node v with $\beta(v) > 0$ and its children. The bad words must originate from either the node v itself or one of its descendants. Since every child v' of v has a value $\beta(v')$, we know which children contribute to $\beta(v)$. Furthermore, the number of children that contribute to $\beta(v)$ are a lower bound on the number of cuts that we need to perform if we do not remove v. We denote this value with $c(v)$. An example of this is shown in Figure 3.

Armed with this information, we traverse the tree again – starting from the root. At every node v we consider the ratio $r(v) = \frac{c(v)}{\gamma(v)}$, i.e., the minimal number of cuts we have to perform if we do not remove v divided by the number of good words in v and all its descendants. As soon as the ratio of a

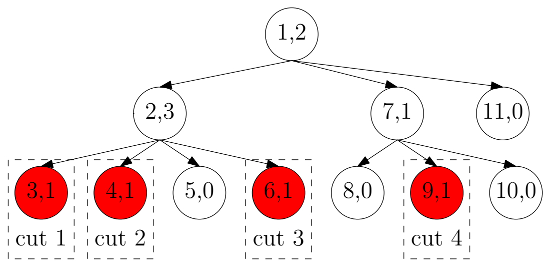


Fig. 3: An example showing the number of children that contain bad words $c(v)$ for each node in addition to its Id (notation: Id,$c(v)$).

TABLE I: News sites including sections, keywords, and page size in number of nodes used for evaluation.

Website	Section	Keyword	# Nodes (bad / all)
20min.ch	International	IS-	188 / 1753
	Finance	Bank	388 / 2333
	Switzerland	Ecopop	121 / 2135
	Sport	Sieg	200 / 2529
blick.ch	International	IS-	62 / 1203
	Economy	Bank	50 / 1159
	Politics	Ecopop	90 / 801
	Sport	Sieg	58 / 1909
nzz.ch	International	IS-	186 / 4565
	Finance	Bank	530 / 4125
	Switzerland	Ecopop	67 / 2904
	Sport	Sieg	215 / 1913

node v exceeds a predefined threshold T, we remove it (and thereby all its children) from the tree. Let us first assume that $c(v) = 1$, then this ratio is high once there are not many good words left. For larger values of $c(v)$, we implicitly assume that the good words are spread out over its children. Thus, in order to avoid too many cuts, we increase the chance to remove node v – keeping in mind that it is better to err on this side.

VI. Evaluation

As a proof of concept, we implemented a Firefox extension that uses our filtering mechanism. The extension was developed and tested with Firefox versions 34.0 to 36.01 [12]. In the following, we discuss the performance evaluation based on the Firefox extension. Our test set contains three different news sites as shown in Table I for which we downloaded the overview page of the sections International, Sports, Economy/-Finance and Switzerland/Politics. For each section we defined a keyword that matches part of the content to filter with.

This set up provides us with nine different pages to process and allows us to explore the tradeoff by filtering too much or too little by varying T. All sites are tested against all spoiler terms which means that we also test sites that do not contain any content that matches the spoiler term. We downloaded the sites on 5th of November 2014 and manually annotated the parts of the web pages that contain spoilers before the experiments were performed. Note that our approach is also designed to remove pictures with using the alt attribute.

Depending on the user preference, T can be set in the preferences of our plug-in. Decreasing the value of T will lead to lower false negative rates and hence minimize the probability of spoilers being shown. However, smaller values of T also lead to higher false positive rates which means that the likelihood of unrelated content being filtered grows. Figure 4 shows the performance trade-off for varying values for T. An ideal system would reach the top left corner and maximize the true positive rate while maintaining low false positive rates. The figure shows, for example, that our system can reach a true positive rate of more than 90% with reasonable false positive rates of less than 15%. The ideal value for T can vary for different web pages because of the different structures in the DOM Tree. However, our test set indicates that a global threshold still delivers good results.

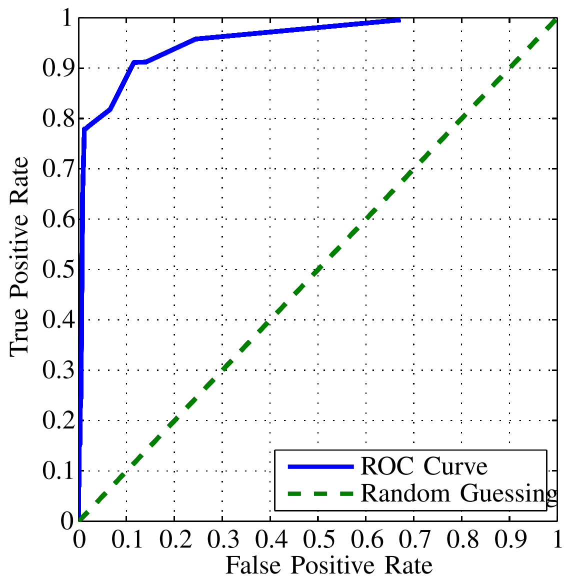


Fig. 4: ROC curve plot for varying values of T

References

[1] H. Pomeranz, "A simple dns-based approach for blocking web advertising," Aug. 2013.

[2] D. Danchev, "South korea to block port 25 as anti-spam countermeasure," in *http://www.zdnet.com/article/ south-korea-to-block-port-25-as-anti-spam-countermeasure/*, Nov. 2011.

[3] D. Canali, M. Cova, G. Vigna, and C. Kruegel, "Prophiler: A fast filter for the large-scale detection of malicious web pages," in *Proceedings of the 20th International Conference on World Wide Web*, ser. WWW '11. New York, NY, USA: ACM, 2011, pp. 197–206. [Online]. Available: http://doi.acm.org/10.1145/1963405.1963436

[4] C. Curtsinger, B. Livshits, B. Zorn, and C. Seifert, "Zozzle: Fast and precise in-browser javascript malware detection," in *Proceedings of the 20th USENIX Conference on Security*, ser. SEC'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 3–3. [Online]. Available: http://dl.acm.org/citation.cfm?id=2028067.2028070

[5] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, "A bayesian approach to filtering junk e-mail," 1998.

[6] C. Dietrich and C. Rossow, "Empirical research of ip blacklists," in *ISSE 2008 Securing Electronic Business Processes*, N. Pohlmann, H. Reimer, and W. Schneider, Eds. Vieweg+Teubner, 2009, pp. 163–171. [Online]. Available: http://dx.doi.org/10.1007/978-3-8348-9283-6_17

[7] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly, "Detecting spam web pages through content analysis," in *Proceedings of the 15th International Conference on World Wide Web*, ser. WWW '06. New York, NY, USA: ACM, 2006, pp. 83–92. [Online]. Available: http://doi.acm.org/10.1145/1135777.1135794

[8] J. Golbeck, "The twitter mute button: A web filtering challenge," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '12. New York, NY, USA: ACM, 2012, pp. 2755–2758. [Online]. Available: http://doi.acm.org/10.1145/2207676.2208673

[9] S. Guo and N. Ramakrishnan, "Finding the storyteller: Automatic spoiler tagging using linguistic cues." in *COLING*, C.-R. Huang and D. Jurafsky, Eds. Tsinghua University Press, 2010, pp. 412–420. [Online]. Available: http://dblp.uni-trier.de/db/conf/coling/coling2010.html#GuoR10

[10] J. Boyd-Graber, K. Glasgow, and J. S. Zajac, "Spoiler alert: Machine learning approaches to detect social media posts with revelatory information," in *Proceedings of the 76th ASIS&T Annual Meeting: Beyond the Cloud: Rethinking Information Boundaries*, ser. ASIST '13. Silver Springs, MD, USA: American Society for Information Science, 2013, pp. 45:1–45:9. [Online]. Available: http://dl.acm.org/citation.cfm?id=2655780.2655825

[11] V. V. Vazirani, *Approximation Algorithms*. Springer, 2004.

[12] MDN, "Firefox-version," in *https://www.mozilla.org/en-US/firefox/ releases/*, Mar. 2015.