Antyscam – Practical Web Spam Classifier
Marcin Luckner, Michał Gad, and Paweł Sobkowiak

Abstract—To avoid of manipulating search engines results by web spam, anti spam system use machine learning techniques to detect spam. However, if the learning set for the system is out of date the quality of classification falls rapidly. We present the web spam recognition system that periodically refreshes the learning set to create an adequate classifier. A new classifier is trained exclusively on data collected during the last period. We have proved that such strategy is better than an incrementation of the learning set. The system solves the starting-up issues of lacks in learning set by minimisation of learning examples and utilization of external data sets. The system was tested on real data from the spam traps and common known web services: Quora, Reddit, and Stack Overflow. The test performed among ten months shows stability of the system and improvement of the results up to 60 percent at the end of the examined period.

Keywords—Web Spam Detection, Spam Detection, Imbalanced Sets Classification, Automatic Classification, Machine Learning.

I. INTRODUCTION

SPAM is a serious problem for the Internet community [1]. However, the unsolicited message are not limited to emails. Spam is present in SMS [2], MMS [3], image files [4], [5], video files [6], [7], and on web pages [8].

The last form of spam – web spam – is still one of the most challenging issues. The most common type in the list is web spam that exploits vulnerabilities and gaps in the web 2.0 to inject links to spam content into dynamic and sharable content such as blogs, comments, reviews, or wikipages.

Figure 1 presents a few examples of web spam. The examples are comments on WordPress blogs. The comments are diversified. The first comment (Figure 1(a)) looks like a valid comment, but hides a link in the user name. The second comment (Figure 1(b)) contains links in nearly normal text. The last comment (Figure 1(c)) is a typical example of web spam comment with several injected links without normal text. This spam was created to promote link farms and provide credibility to the spammer website.

Because of variety of web spam, web spam detectors use machine learning techniques to create a model of spam from training sets that include spam and non-spam examples. On of the most popular training sets is WEBSPAM-UK collection. The collection includes two datasets from 2006 and 2007 [9].

Many projects have used these data sets to test web spam detectors.

However, works [10], [11], [12], [13], [14] proved that using data from 2006 to create classifier that recognises web spam

The Antyscam was developed in EU POIG.01.04.00-14-031/11 project.

M. Luckner is with the Faculty of Mathematics and Information Science, Warsaw University of Technology, Koszykowa 75, 00–662 Warszawa, Poland, (e-mail: mluckner@mini.pw.edu.pl).

M. Gad is with EO Networks, ul. Głuszycka 5, Warszawa, Poland, (e-mail: michal.gad@eo.pl).

P. Sobkowiak is with Sensi Soft, ul. Głuszycka 5, Warszawa, Poland, (e-mail: pawel.sobkowiak@gmail.com).
in data from 2007 decreases the accuracy of the classification in comparison to classifiers trained on the data from the same year.

The problem of reduction of the classification accuracy over time was stressed in [14], [15], [16], [10], [17]. The works shows that it is impossible to create the classifier that will keep the similar level of quality over years.

In this work, we proposed a new system for web spam classification. The system collects data on web spam from web spam traps and information on non-spam from trusted web services to create a new classifier for each approaching period. The classifiers are based on fast learning bagging methods and the RUSBoost algorithm. The creation of new classifiers allows the web page administrator to keep the anti-spam protection on the high level. The system can work in a fully automated mode when the learning process is based on external learning data or master the spam rejection when the learning process is based on supervised internal data from the protected service.

We present test of the classifiers on three common knows web services: Quora, Reddit, and Stack Overflow. The test proved high quality of the system. Using the data sets, we discussed issues of the classification in the case of lacks in positive or negative spam examples during starting–up of the system. We also proved an unobvious fact that using web spam rejectors trained on limited data from the last period gives better results than using rejectors trained on the whole history.

The remainder of this work is organised as follows. First, the results obtained in other works on web spam detection are described in Section II. This is followed by a description of the proposed dynamic web spam recognition system in Section III. Section IV presents the testing methodology. Section V contains the results and discussion. This is followed by the conclusions in Section VI.

II. Previous Works

The web spam recognition issue was analysed by several works summarized in the following articles. In work [18], the authors compared results obtained by various machine learning techniques used for web spam recognition. The work was focused on the commonly known techniques that were tested over the WEBSNAP-UK2006 and WEBSNAP-UK2007 datasets. That review includes such works as [14], [13], [19], [12]. More results obtained on the same data sets were published in [10], [17], [20], [21], [22], [23], [24], [25]. The discussion of these results was presented in [14]. The present research focuses on systematically analysing and categorizing models that detect review spam were summed in [26].

Several new approach were proposed in last years. In work [27], the authors proposed a general mathematical framework, which proves beneficial for grouping classifiers using a convex ensemble diversity measure. The ant colony optimization designed to let an ant start from a non-spam seed, and compilation of the created path to non-spam classification rules was discussed in [28].

Our solution depends on features that discriminate web spam. Two approaches are common in web spam detection. In the first one, features are selected using data mining methods [21], [29]. In the second approach expert knowledge is used.

Several works proposed their own set of features. In work [30], changes in the distribution of the set of the selected detection features according to the page language were examined. The authors of work [31] proved that historical web page information was an important factor in spam classification. Work [32] described how to use the HTML style similarity clusters to pinpoint dubious pages and enhance the quality of spam classifiers. Work [33] proved that certain linguistic features could be useful for a spam-detection task when combined with features studied elsewhere.

Part of the features used in our system was inspired by the above–mentioned papers.

We propose a complete system for the web spam rejection from blogs. The following two systems aimed at email spam were created before.

Paper [16] evaluated spam filters derived from different optimisation problems on chronologically ordered future emails. The Nash–equilibrial prediction models used outperformed reference methods. However, the execution time is 10 thousand time higher for the Nash–equilibrial prediction models than for the SVM.

The recognition of spam through years was discussed in several works. Work [15] presented two mechanisms. The predictive defence algorithm combines game theory with machine learning to model and predict future adversary actions for synthesising robust defences. The extrapolative defence algorithm involves extrapolating the evolution of defence configurations forward in time, in the terms of defence parametrisations, as a way of generating defences. The algorithms were tested over 18 quarters. The results showed a 5 percent reduction in accuracy over 5 quarters for both methods.

Finally, a new web spam filtering framework WSF2 was presented in [34], [35]. The authors proposed the approach being able to dynamically adjust different parameters to ensure continuous improvement in filtering precision with the passage of time. The framework was tested over the WEBSPAM-UK2007 using combinations of different filtering techniques including regular expressions and well-known classifiers.

In our work we discuss several issues that were not stressed in description of the WSF2 system such as updating the learning set and first period issue, when the classifiers do not have a full knowledge on characteristic of the recognized data.

III. Methodology

A. System

Our research was done as part of Antyscam project, which is a commercially developed SaaS (Software as a Service) application designed to identify and minimise volume of unwanted content on web pages. Its goal is to provide a simple API that allow users to quickly classify content of any kind as spam or non-spam.

Most available spam detection solutions are limited to a single type of content - SpamAssassin handles unwanted email, Akismet fights spam WordPress comments and search engines filter out malicious web pages. Current common practice is
to build a domain-specific tool from scratch, rely on human moderators or to ignore the issue completely.

Although, in this work we are focused on web spam, Antiscam provides a cross-domain solution by focusing on lexical-based features that can be extracted from any textual content, be it blog post, web page, email or a comment.

In order to classify documents Antiscam user must first log in to the panel and configure a document feed schema by defining a list of typed fields of the documents that will be analysed.

Figure 2 shows how the admin can define a datafeed for WordPress comments consists of four fields: a 'content' field of a type HTML for comment content, 'author' field for comment's author nickname, 'url' field for author's website link and 'email' field for author's email. This allows the application to properly extract features - HTML is treated differently than plain text or URL.

After document feed is defined, user can start sending documents to Antiscam via API. Labelled documents (i.e. ones that the user is sure whether they are spam or non-spam) are stored and used for training the classifier and unlabelled documents are evaluated and classified as spam or non-spam.

Each user has his own classifier, which is initially trained on a corpora of heterogeneous documents collected by Antiscam sub-modules. This allows to quick start the spam classification process, user can get a working solution without the necessity of providing a premoderated dataset. Later, as user adds labelled examples to his document feeds, the classifier is retrained to include them.

Antiscam corpora consist of different kinds of datasets, collected with different methods. We use some publicly available complete datasets, for example a WEbspam-uk dataset of web pages crawled by Yahoo. We also use public APIs to collect a snapshot collection of documents, for example comments to Reddit threads or answers to Quora questions described in detail later in the article. And finally we collect spam documents by creating spam traps (honeypots), for example by creating multiple fake WordPress blogs to attract bots posting spam comments.

B. Algorithm

In the described system, the classification process performs in separated periods. During a period $T_i$ all incoming comments are classified by the same classifier $c_i$. The classifier is trained on data from the previous period $T_{i-1}$. Parallel to the classification process the system collects information on web spam from the spam traps and on non-spam comments from trusted web services. After the period $T_i$ the system possesses the following data sets

- $exS_i^+$ - external spam examples collected by the spam traps;
- $exS_i^-$ - external non-spam examples downloaded from trusted web services;
- $inS_i^+$ - internal spam examples labelled by the existing classifier;
- $inS_i^-$ - internal non-spam examples labelled by the existing classifier;

The collected sets can be used to create a new classifier $c_{i+1}$. Spam collected in the same period from various sources should be similar. Therefore, it can be assumed that $inS_i^+ \subset exS_i^+$ and the set $exS_i^+$ should be used to create the classifier. The selection of data set with non-spam comments is disputable. If the web page is well supervised then the set $inS_i^-$ is the most representable. In other case, the set $exS_i^-$ provides data for the full automated classification.

A separate issue is classification of data in the first period $T_1$ when the system lacks in collected examples. The first period starts when

$$inS_1^+ \cup inS_1^- = \emptyset$$

(1)

However, we assumed that the set $eS_0^+$ always exists. Therefore, it is necessary to create a set with non-spam comments. One option is to find the set $eS_0^-$. An alternative is to create a set $in_nS_0 \subset eS_0^+$ that contains $n$ manually labelled non-spam comments from the working web page. In this second option it is necessary to minimise the number $n$. In such case the first period ends when

$$inS_0^+ \neq \emptyset \land |in_nS_0| \geq n$$

(2)

The issue of the first period is discussed in Section V-C. When the number of non-spam comments $n$ is minimised the disproportion between sets $in_nS_0^+$ and $exS_0^+$ grows. As the result, the classifier may label all comments as a spam. To avoid this situation a special classification algorithm for unbalanced learning set must be used.

This period ends when

$$|in_nS_0^+| - |exS_0^+| < \epsilon$$

(3)

where $\epsilon$ is an acceptable difference between cardinal numbers of the sets. After that the classification problem can be solve by any binary classifier.
C. Classifiers

The discussed issue of a continuous working web spam rejector contains two classification problems. The first problem is data classification at the beginning of function of the system when the full knowledge of data from the previous period. The following classification functions can be used to recognise web spam under given circumstances.

The RUSBoost algorithm [36] is dedicated to discriminate imbalance classes. The algorithm creates an ensemble of week classifiers similarly as bagging techniques. However, in the created learning set a number of examples from the majority class is reduced to reach a given percentage of examples of the minority class in the learning set.

The One–Class Support Vector Machine (SVM) algorithm [37] creates an SVM classifier [38] for one class. Assuming that \( x_i \in \mathbb{R}^d \) are vectors of features of training elements, for \( i = 1, 2, \ldots, N \) and for \( N \) being the cardinality of the training set. Assume also \( K(x, x') \) is a kernel function. Then a one-class SVM decision function is implemented as:

\[
f(x) = \text{sgn} \left( \sum_{i=1}^{N} \alpha_i * K(x, x_i) - \rho \right)
\]

where \( \alpha_i \) and \( \rho \) are obtained by maximization of the following convex quadratic programming (QP) problem:

\[
\frac{1}{2} \left( \sum_{i=1}^{N} \alpha_i * K(x, x_i) \right)^2 - \rho - \sum_{i=1}^{N} \alpha_i \left( \sum_{i=1}^{N} \alpha_i * K(x, x_i) \right) - \rho
\]

with constrains:

\[
\bigwedge_{i \in \{1, 2, \ldots, N\}} 0 \leq \alpha_i \leq \frac{1}{\nu N} \land \sum_{j=1}^{N} \alpha_i = 1
\]

where \( \nu \) is an equivalent of the regularization coefficient \( C \) in the binary SVM classification.

Comparison of various SVM techniques in recognition with rejection is given in [39].

The second problem is classification of data from the current period using a full knowledge on data from the previous periods. The classifier should provide a high accuracy and short learning and testing time. A very good candidate is Random Forest that uses a bagging for classification [40].

Bagging bags a weak classifier such as a decision tree on a dataset, generates many bootstrap replicas of this dataset and grows decision trees on these replicas. To find the predicted response of a trained ensemble, the algorithm takes an average of predictions from individual trees.

D. Features extraction

In the preprocessing, before calculating actual features, each analysed comment was transformed into three separate forms.

The first form was a Visible Text. The HTML document was stripped of all mark–up. For that BeautifulSoup4 library with lxml backend was used. In the result, we obtained the pure text between tags.

The second form was a Non–blank Visible Text. To obtain this form, we removed all space characters from the Visible Text. With this form it is easier to detect content given by spaced out letters such as v i a g r a.

The third created form was a Distinct Domains. The Distinct Domains is a set of unique domain names including the domains defined by Internationalised Domain Names in Application (IDNA) standards [41], which were in a language–specific script or alphabet, such as Arabic, Chinese, Russian, or the Latin alphabet-based characters with diacritical marks, such as Polish.

The domains were extracted not only from the visible part of a comment, but also from the whole of origin the HTML document to include the domains written down inside tags. It is easy to distinguish domains names from the surrounding text due to well defined constraints, especially the set of characters a domain name can consist of and the limited set of valid top–level domains.

The features used in web spam detection were detailed described in our previous work [14] including the Python codes that calculate the features.

IV. Tests

A. Data sets

Data were collected from June 2013 to February 2014 and divided into ten monthly periods labelled \( T_i \) where \( i = 0 \ldots 9 \). The first period \( T_0 \) is unique because there is not a preceding period and it is not possible to use any preceding data to create a classifier to recognise classes from this period.

The system collected two data sets. The first set \( S^+ \) contains web spam comments collected by a spam trap and is labelled as spam. The second set \( S^- \) consist of non–spam comments and is labelled as non–spam. This set is heterogeneous and contains non–spam comments from the three web communities: Quora, Reddit, and Stack Overflow. Examples of comments are presented on Figure 3. In the test, the comments from one web service are treated as an internal dataset in\( S^- \) when the other two are treated as separate external datasets ex\( S^- \). All spam data are treated as an internal spam data in\( S^+ \).

Quora dataset consists of the best answers to most popular questions posted on Quora.com. Quora does not provide an API, so web scraping method was chosen. Bot started crawling from pages with most followed topics in 2014 and 2015 and collected 105 links to top topic pages.

On each topic page the bot visited an overview, top answers and FAQ page to extract a total of 2804 links to individual question pages. On each question page 5 top answers were extracted and saved, for a total of 9520 answers.

Reddit JSON API was used to collect non-spam comments for Reddit dataset. Bot starts with top topics page and enters each topic in order. On each topic page all comments are examined. Comments are considered non-spam when the following conditions are met:

- Comment is parsed correctly according to Reddit API docs
- Comment is ranked positively - it has 5 more thumbs-up than thumbs-down
- Comment is not too short - it has at least 100 characters
Yes it is definitely possible. Amazon focuses more on customers than their sellers unfortunately. I have sellers getting banned for no particular reasons and Amazon will not reveal specifically the reason sellers getting banned so it could be anything and it could be a mistake on Amazon’s behalf.

Amazon tracks an account through IP and MAC addresses, browser fingerprinting, cookies, flash objects, and all personal details. So if a person tried to create another account and used the same IP address for example, Amazon will ban him/her again immediately.

There is a way to get back onto Amazon and start selling again. I have crafted a training course which is tested and proven. Feel free to check it out at 8 Steps to Creating Your Amazon ‘Ghost’ Account.

Thanks.

(a) Quora

Read the YouTube comments, this is his reply to someone telling people that it’s stopped: Or maybe ita me holding the Golden Retriever that kept getting in my shot. He almost pulled me down as he wanted to run with Stella. Also a huge point for the untrained, you can hear Chewy (the golden) panting late in the video... long after Stella was out of view.... Your "trained" eye should have saw by the number of other high quality videos I have posted..... it in no way have the capability of spotting....he I apparently have scoliosis case I don't know how to use panoramic view!!! LOL

Keep studying other people's work, analyze their faults, learn from them and maybe, maybe you can make it in Hollywood someday.

(b) Reddit

It prevents JSON hijacking.

Contribute example: say Google has a URL like mail.google.com/jsonaction/index which returns the first 50 messages of your inbox in JSON format. Evil websites on other domains can’t make AJAX requests to get this data due to the same-origin policy, but they can include the URL via a <script> tag. The URL is visited with your cookies, and by overriding the global array constructor or accessor methods they can have a method called whenever an object (array or hash) attribute is set, allowing them to read the JSON content.

The widj[33]; or mail.google.com prevents this: an AJAX request at mail.google.com will have full access to the test content, and can strip it away. But a <script> tag insertion blindly executes the Javascript without any processing, resulting in either an infinite loop or a syntax error.

This does not address the issue of cross-site request forgery.

(c) Stack Overflow

Fig. 3. The examples of comments from various datasets

- Comment is not reported as offensive by any user

When feed was collected a total of 6521 topic pages were visited. 529158 comments were parsed. Among them 130604 were rejected because they had the ups/downs balance lower than 5. Among all the comments, 176688 comments were considered too short (less than 100 characters). Finally, 221866 were collected and included in the feed data.

StackExchange API was used to download all answers to top rated questions on StackOverflow. When feed was collected a list of 68410 top rated questions was downloaded via API and a total of 500000 answers were extracted and saved. Answers with upvote score greater or equal to 30 were selected for a total of 101161 highly rated answers.

The collected comments were limited to comments that overlap the monthly periods when the spam comments were collected. The number of data in division on data sources and periods are given in Table I.

B. Methods of tests and evaluations

We considered the ten monthly periods. During the periods both spam comments and non–spam comments were collected. We compared the three following strategies of creation of the learning set for the classification: Dynamic, Incremental, and Static.

The Dynamic strategy uses data in $S^+_i$ and in $S^-_i$ from the previous period to classify data in $S^+_i+1$ and in $S^-_{i+1}$ from the current period.

The Incremental strategy uses all collected data $\bigcup_{j=0}^{i} S^+_j$ and $\bigcup_{j=0}^{i} S^-_j$ – including the previous period – to classify data in $S^+_i+1$ and in $S^-_{i+1}$ from the current period.

The Static strategy uses the static learning set in $S^+_0$ and in $S^-_0$ – the set collected during the first period – to classify data in $S^+_i+1$ and in $S^-_{i+1}$ from the current period.

For all strategies the test were done separately on nine monthly periods from $T_1$ to $T_9$. The evaluation of the results obtained on the testing sets was done by the following measures:

TP (true positive) the number of correctly recognised spam entries. TN (true negative) the number of correctly recognised non–spam entries. FP (false positive) the number of incorrectly recognised spam entries. FN (false negative) the number of incorrectly recognised non–spam entries. Accuracy the fraction of correctly recognised entries (both spam and non–spam)

$$ACC = \frac{TP + TN}{TP + FP + TN + FN}.$$  

(7)

Sensitivity or True Positive Rate the fraction of correctly recognised spam entries among all spam entries

$$TPR = \frac{TP}{TP + FN}.$$  

(8)

Specificity the fraction of detected non–spam entries among all non–spam entries

$$SPC = \frac{TN}{TN + FP}.$$  

(9)

F–measure the weighted average of the spam predictive value and sensitivity,

$$F1 = \frac{2TP}{2TP + FP + FN}.$$  

(10)

V. RESULTS AND DISCUSSION

The detailed results – obtained for each training set $T_1 \ldots T_9$ by three examined strategies – are given for each non–spam dataset separately. Table II presents statistics for the Quora dataset, Table III shows results for the Stack Overflow dataset, and Table IV gives information on the Reddit dataset. In all cases, the data set was treated as the internal non-spam dataset in $S^-_0$ when data from the spam traps were used as the internal spam data set in $S^+_0$.

In each table, the stressed results are the best obtained results for a given measure among three learning strategies. The Dynamic strategy and the Incremental strategy give good classification results. The detailed comparison of the strategies is given in Section V-B. The Static strategy gives the worst results. The reasons of this situation are discussed in Section V-A.

Data collected in the tables does not include results for period $T_0$. The first period issue is raised separately in Section V-C.
A. Reasons of fail of Static strategy

The main reason why the Static strategy fails is a concept drift. The statistical definition of spam and non-spam changes over time in unforeseen way. This causes problems because the predictions become less accurate as time passes. To prove this reasoning we have tested changes of features importance over time in unforeseen way. This causes problems because the classification results may be still important, but we can limit the current discussion to the most important features.

An estimation of predictor importance for decision trees was calculated. Feature importance is calculated for a split defined by the given feature. Importance is computed as the difference between Mean Squared Error (MSE) for the parent node and the sum of changes in the MSE is calculated due to splits on every feature used in the recognition process. Next, the sum is divided by the number of branch nodes.

Importance is normalised to the range $[0, 1]$ with 0 representing the smallest possible importance.

Figure 4(a), Figure 4(b), and Figure 4(c) show how the normalised importance was changed among time. For clarity, we have limited the number of presented features to the set of features with the maximum normalised importance that exceeds 0.8. The influence of the other features on the classification results may be still important, but we can limit the current discussion to the most important features.

The most important features are mostly connected with the Distinct Domain document that was used to calculate the fraction of non–alpha characters in the domain length, and standard deviation of the length of domains. Additionally, the fraction of non–alpha characters in the domain names was calculated.

### Table I

<table>
<thead>
<tr>
<th>Datasets</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$T_5$</th>
<th>$T_6$</th>
<th>$T_7$</th>
<th>$T_8$</th>
<th>$T_9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spam trap</td>
<td>2588</td>
<td>7673</td>
<td>7371</td>
<td>4176</td>
<td>1783</td>
<td>7746</td>
<td>17419</td>
<td>15323</td>
<td>14112</td>
</tr>
<tr>
<td>Quora</td>
<td>45</td>
<td>40</td>
<td>41</td>
<td>53</td>
<td>59</td>
<td>47</td>
<td>54</td>
<td>53</td>
<td>73</td>
</tr>
<tr>
<td>Reddit</td>
<td>105</td>
<td>362</td>
<td>51</td>
<td>555</td>
<td>319</td>
<td>40</td>
<td>24</td>
<td>867</td>
<td>626</td>
</tr>
<tr>
<td>Stack Overflow</td>
<td>1104</td>
<td>951</td>
<td>972</td>
<td>1028</td>
<td>1057</td>
<td>1007</td>
<td>849</td>
<td>717</td>
<td>884</td>
</tr>
</tbody>
</table>

### Table II

<table>
<thead>
<tr>
<th>Stat</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$T_5$</th>
<th>$T_6$</th>
<th>$T_7$</th>
<th>$T_8$</th>
<th>$T_9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC</td>
<td>1.0000</td>
<td>0.9981</td>
<td>0.9991</td>
<td>0.9973</td>
<td>0.9995</td>
<td>0.9994</td>
<td>0.9997</td>
<td>0.9997</td>
<td>0.9997</td>
</tr>
<tr>
<td>TPR</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.9993</td>
<td>0.9983</td>
<td>0.9999</td>
<td>0.9996</td>
<td>0.9997</td>
<td>0.9997</td>
<td>0.9997</td>
</tr>
<tr>
<td>SPC</td>
<td>1.0000</td>
<td>0.7455</td>
<td>0.9804</td>
<td>0.9655</td>
<td>0.9388</td>
<td>0.9216</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>FI</td>
<td>1.0000</td>
<td>0.9990</td>
<td>0.9995</td>
<td>0.9986</td>
<td>0.9997</td>
<td>0.9997</td>
<td>0.9998</td>
<td>0.9998</td>
<td>0.9999</td>
</tr>
</tbody>
</table>

### Table III

<table>
<thead>
<tr>
<th>Stat</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$T_5$</th>
<th>$T_6$</th>
<th>$T_7$</th>
<th>$T_8$</th>
<th>$T_9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC</td>
<td>0.9686</td>
<td>0.9942</td>
<td>0.9773</td>
<td>0.9299</td>
<td>0.9721</td>
<td>0.9931</td>
<td>0.9948</td>
<td>0.9945</td>
<td>0.9851</td>
</tr>
<tr>
<td>TPR</td>
<td>1.0000</td>
<td>0.9997</td>
<td>0.9995</td>
<td>0.9981</td>
<td>0.9992</td>
<td>0.9970</td>
<td>0.9953</td>
<td>0.9953</td>
<td>0.9808</td>
</tr>
<tr>
<td>SPC</td>
<td>0.7782</td>
<td>0.9547</td>
<td>0.8984</td>
<td>0.8432</td>
<td>0.8079</td>
<td>0.9150</td>
<td>0.9832</td>
<td>0.9808</td>
<td>1.0000</td>
</tr>
<tr>
<td>FI</td>
<td>0.9820</td>
<td>0.9967</td>
<td>0.9857</td>
<td>0.9410</td>
<td>0.9840</td>
<td>0.9964</td>
<td>0.9973</td>
<td>0.9971</td>
<td>0.9902</td>
</tr>
</tbody>
</table>

### Table IV

<table>
<thead>
<tr>
<th>Stat</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$T_5$</th>
<th>$T_6$</th>
<th>$T_7$</th>
<th>$T_8$</th>
<th>$T_9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC</td>
<td>0.9676</td>
<td>0.9942</td>
<td>0.9710</td>
<td>0.8908</td>
<td>0.9481</td>
<td>0.9681</td>
<td>0.9981</td>
<td>0.9977</td>
<td>0.9938</td>
</tr>
<tr>
<td>TPR</td>
<td>1.0000</td>
<td>0.9999</td>
<td>1.0000</td>
<td>0.9993</td>
<td>0.9999</td>
<td>0.9994</td>
<td>0.9999</td>
<td>0.9998</td>
<td>0.9918</td>
</tr>
<tr>
<td>SPC</td>
<td>0.7682</td>
<td>0.9538</td>
<td>0.8719</td>
<td>0.7736</td>
<td>0.6895</td>
<td>0.5942</td>
<td>0.9804</td>
<td>0.9786</td>
<td>1.0000</td>
</tr>
<tr>
<td>FI</td>
<td>0.9809</td>
<td>0.9967</td>
<td>0.9816</td>
<td>0.9048</td>
<td>0.9698</td>
<td>0.9830</td>
<td>0.9990</td>
<td>0.9988</td>
<td>0.9959</td>
</tr>
</tbody>
</table>

### Table V

<table>
<thead>
<tr>
<th>Stat</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$T_5$</th>
<th>$T_6$</th>
<th>$T_7$</th>
<th>$T_8$</th>
<th>$T_9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC</td>
<td>0.9681</td>
<td>0.8632</td>
<td>0.8100</td>
<td>0.6067</td>
<td>0.3728</td>
<td>0.2069</td>
<td>0.1436</td>
<td>0.2279</td>
<td>0.3635</td>
</tr>
<tr>
<td>TPR</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>SPC</td>
<td>0.7757</td>
<td>0.4660</td>
<td>0.5097</td>
<td>0.4862</td>
<td>0.1550</td>
<td>0.0554</td>
<td>0.0496</td>
<td>0.0709</td>
<td>0.2821</td>
</tr>
<tr>
<td>FI</td>
<td>0.9818</td>
<td>0.9161</td>
<td>0.8657</td>
<td>0.5839</td>
<td>0.4511</td>
<td>0.2880</td>
<td>0.1877</td>
<td>0.3045</td>
<td>0.2625</td>
</tr>
</tbody>
</table>
The observed changes of importance are very chaotic but that supports the concept drift hypothesis. For example, we observe for Quora dataset (Figure 4(a)) that the count of word that was extremely important in the first month loose it position in the following months. On the other hand, the ratio of non–alpha characters in the domain names grows rapidly in the 3rd month despite that it was not an important discriminator in the first month.

Therefore, it is not possible to create a classifier based on Static strategy that obtains a stable accuracy of the webspam detection.

B. Dynamic strategy versus Incremental strategy

The main important question is which strategies brought the best results. Table II, Table III, and Table IV show that the ACC obtained by the Static strategy is definitely the worst but the victory of any strategy from the two remaining strategies can be questioned.

In the tables the highest values of statistics ACC and F1 were marked. In some months the Dynamic strategy was the best options in other the Incremental strategy was better. To prove that there is a significant difference between results obtained by the strategies, we performed Wilcoxon’s Signed–Rank test for paired scores [42].

If the ACC obtained by the Dynamic strategy is significantly better the test should show that the ACC calculated for the Dynamic strategy is greater than for the Incremental strategy in most of the tests and smaller in a few tests by only a small amount. We compared both strategies in all 27 combinations of datasets and months. For 16 pairs ACC calculated for the Dynamic strategy was greater. An opposite situation has place in 8 cases. In the rest of cases the results was the same for both strategies.

Wilcoxon’s Signed–Rank test rejected the null hypothesis ($p = 0.098741$), which stated that the results obtained by the two strategies were not significantly different, at the 0.1 level. Moreover, the modified test accepted ($p = 0.051170$), at the 0.1 level, the alternate hypothesis that the difference in F1 between the Dynamic strategy and the Incremental strategy come from a distribution with median greater than 0.

Therefore, the results obtained by the Dynamic strategy significantly better than results obtained by the Incremental strategy when the strategies are evaluated using ACC and F1.

The second aspect that can be compared is time necessary to learn and classify. Times that should be compared are the learning time as a time necessary to train a web spam detector and the testing time as a time necessary to classify all entries from the given period.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Strategy</th>
<th>Dataset</th>
<th>Learning time [s]</th>
<th>Testing time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF</td>
<td>Incremental</td>
<td>Quora</td>
<td>19.33</td>
<td>0.00</td>
</tr>
<tr>
<td>RF</td>
<td>Incremental</td>
<td>Reddit</td>
<td>22.30</td>
<td>0.00</td>
</tr>
<tr>
<td>RF</td>
<td>Incremental</td>
<td>Stack Overflow</td>
<td>36.60</td>
<td>0.00</td>
</tr>
<tr>
<td>RF</td>
<td>Dynamic</td>
<td>Quora</td>
<td>3.54</td>
<td>0.00</td>
</tr>
<tr>
<td>RF</td>
<td>Dynamic</td>
<td>Reddit</td>
<td>3.89</td>
<td>0.00</td>
</tr>
<tr>
<td>RF</td>
<td>Dynamic</td>
<td>Stack Overflow</td>
<td>6.89</td>
<td>0.00</td>
</tr>
<tr>
<td>SVM</td>
<td>Incremental</td>
<td>Quora</td>
<td>19.06</td>
<td>0.08</td>
</tr>
<tr>
<td>SVM</td>
<td>Incremental</td>
<td>Redidt</td>
<td>118.17</td>
<td>0.66</td>
</tr>
<tr>
<td>SVM</td>
<td>Incremental</td>
<td>Stack Overflow</td>
<td>959.77</td>
<td>6.41</td>
</tr>
<tr>
<td>SVM</td>
<td>Dynamic</td>
<td>Quora</td>
<td>2.79</td>
<td>0.04</td>
</tr>
<tr>
<td>SVM</td>
<td>Dynamic</td>
<td>Reddit</td>
<td>9.86</td>
<td>0.21</td>
</tr>
<tr>
<td>SVM</td>
<td>Dynamic</td>
<td>Stack Overflow</td>
<td>35.42</td>
<td>1.06</td>
</tr>
</tbody>
</table>

Table V presents comparison of time consumption for both strategies. The tests have been done on personal computer with the processor 2.9 GHz Intel Core i5 supported by 16 GB 1867 MHz DDR3 memory. The difference in learning time between the Dynamic strategy and the Incremental strategy is very big. The Incremental strategy is over five times slower.
The learning time among the months is presented on Figure 5. The learning time for the Incremental strategy grows rapidly when the learning time for the Dynamic strategy stays nearly on the same level. Therefore, even if the learning times obtained on the datasets are not very long the learning time for the Incremental strategy will increase month by month. The testing times for both strategies stay similar for all datasets. The testing time is shorter than one second.

The discussed problem arises for more time consuming classification methods. The additional test have been done on SVM classifiers. The tests stress differences between strategies. Depending on the dataset the Incremental strategy can be from six to twenty-seven times slower. Moreover, the classification times is from two to six time slower than for the Dynamic strategy.

To sum up, the Dynamic strategy obtained the significantly better results – evaluated by ACC and F1 – than the Incremental strategy. Moreover, the learning time for the Dynamic strategy was static in time when for the Incremental strategy the learning time grew. Therefore, the Dynamic strategy is better option for implementation.

C. First period issue

During the first period the classifier does not possess the full characteristic of the web spam in $S^+$ and non–spam comments in $S^-$ from the previous period. To create any classifier we have to had a representation of at least one of the classes. Because the historical web spam data sets exist – spam collected before the system start-up or spam from public repositories such as WEBSHAM-UK [9] – we assume that a set $\mathcal{S}^+$ can be created. However, To optimize the classification process, the knowledge on web spam characteristic should be supplemented by a partial knowledge on the non–spam comments. This knowledge is represented by a set $\mathcal{S}^-$ that contains $n$ examples of the comments.

Especially, for the small value of $n$ the both sets are imbalanced. Therefore, the dedicated classifier – such as one-class SVM or RUSBoost – should be used as a discriminator.

Figure 6(a) presents the accuracy obtained by the RUSBoost algorithm on the sets in $\mathcal{S}^+$ and in $\mathcal{S}^-$ using as the learning sets in $\mathcal{S}_0^+$ and in $\mathcal{S}_0^-$ (the whole information on spam from the previous period and $n$ non–spam comments).

From $n = 7$, the results obtained on the testing set are more or less stable for all data sets. Moreover, for the Stack Overflow set and the Quora set the obtained accuracy is similar.
70 percent accuracy. The classifier accepted all non–spam comments as spam obtaining the classification process and the accuracy was constant. The learning comments was too small to change the structure of results.

A dataset that needs over 50 spam comments to stabilise the accuracy for the first period in all cases except the Quora of 10 spam comments is enough to exceed 90 percent of the spam comments is limited. Figure 6(b) shows that the number than the first one – that assumes that the number of known spam rejection. A new classifier is trained exclusively on data collected during the last period. We have proved that such strategy is better than an incrementation of the learning set. The system contains the start-up mechanism that allows the web page administrator to protect the service despite of lacks in learning sets. Assuming the full information on current form of web spam received from the spam traps, the system can works with minimal information on non–spam comments.

All elements of the classification process were tested on real data from the spam traps and common known web services: Quora, Reddit, and Stack Overflow. In all cases, the quality of the system was satisfactory.

VI. CONCLUSIONS

We have presented the web spam recognition system. The system periodically replaces the classifier used for the web spam rejection. A new classifier is trained exclusively on data collected during the last period. We have proved that such strategy is better than an incrementation of the learning set. The system contains the start-up mechanism that allows the web page administrator to protect the service despite of lacks in learning sets. Assuming the full information on current form of web spam received from the spam traps, the system can works with minimal information on non–spam comments.

All elements of the classification process were tested on real data from the spam traps and common known web services: Quora, Reddit, and Stack Overflow. In all cases, the quality of the system was satisfactory.

REFERENCES

rithms to image spam evolution,” in Emerging Paradigms in Machine Learning, ser. Smart Innovation, Systems and Technologies, S. Rama-