VERSITA

Journal of Applied Mathematics, Statistics and Informatics (JAMSI), **9** (2013), No. 1

AN AUTOMATED FRAMEWORK WITH APPLICATION TO STUDY URL BASED ONLINE ADVERTISEMENTS DETECTION

PIOTR L. SZCZEPAŃSKI, ADRIAN WIŚNIEWSKI AND TOMASZ GERSZBERG

Abstract

A rapid growth of online advertisements results in unsolicited bulk of data being downloaded during web surfing. To tackle this problem a fast mechanism detecting adverts is required. In this paper we present the usefulness of URL based web-pages classification in the process of online advertisements detection. Our experiments are performed on seven popular classifiers using the real-life dataset obtained by human agents browsing the internet. We introduce a general and fully automated framework that allows us to do a comprehensive analysis by performing simultaneously hundreds of experiments. This study results in solution with 0.987 accuracy and 0.822 F-measure.

Mathematics Subject Classification 2000: 62H30
General Terms: Classification, Pattern recognition

Additional Key Words and Phrases: machine-learning, classification, online advertisement

1. INTRODUCTION

Fast machine-learning online advertisements detection is needed for a few reasons. In order to efficiently access information on internet it is necessary to separate superfluous web content (like ads) from legitimate one. For dealing with thousands evolving adverts we need automated solution that can be easily adapted to the new conditions. Finally, to deal with the voluminous data streams in real-time we need fast solution—thus, classification based only on URLs is a proper approach to this requirement.

There is a growing body of literature that advocates the use of URLs alone in classification of web-pages [Kan 2004; Kan and Thi 2005; Devi et al. 2007; Baykan et al. 2011]. The method is very fast and quite effective as one does not need to process the content of the web page, which is time consuming [Kan and Thi 2005], but only infer necessary information from the URL, the marker of the WWW resources. Furthermore, in some applications one is not able to investigate content of the website, and it has to be checked before downloading.

The most popular usage of URLs to detect online advertisements involved mostly rule-based techniques. These methods basically look for some key words like 'ad', 'advert', or 'banner' and block URLs containing them. Even the best solution from this group can not deal with advertisements not containing these key words, so publishers could easily start a new ad campaign easily tricking these kind of ads detection methods [Singh and Potdar 2009]. Here the need for machine-learning

DOI 10.2478/jamsi-2013-0005

solution arise. This approach has one big advantage over the rule-based techniques. It does not required human effort in the process of identification the key features of each URL, but it uses statistics methods extracting all necessary information from the collection of URLs.

A growing body of machine-learning solutions poses another task. In many well known domains it is already widely known which classifiers give the best performance. However, to tackle a new problem we need to study many machine-learning solutions to find the best one. Expert's knowledge is an essential part of this researcher, but there are many standard methodological procedures that can be automatized. We present an automated framework allowing us to perform many experiments simultaneously. This system is a strong support for any studies that requires testing many machine-learning solutions.

We advocated the usefulness of such automated system by performing comprehensive studies on URL based online advertisements detection. We tested more than 2500 machine-learning solutions and obtain some unpredictable results. For instance, we showed that in order to obtain the best solution the optimal number of features used to describe the instances of training set is quite low (only 0.006% off all extracted features).

Against this background, the key contributions of our paper are as follows:

—We perform the first comprehensive study on URL based machine-learning solution to the problem of online advertisements detection. Our research is based a broad range of experiments that test seven popular classifiers with different parameters.

—We propose the automated framework to test the classifiers performance. Our architecture allows for setting the hundreds of experiments in the fast and human-friendly way using concise representation. It performs tests simultaneously and automatically choose the best solution in the terms of arbitrary chosen evaluation method.

—We suggest the most effective solution detecting online advertisements with 0.987 accuracy and 0.83 F-measure (0.95 weighted F-measure).

This paper is organized as follows: Section 2 gives a brief description of a state-of-the-art in online advertisements detection, Section 3 defines the problem in more formal way and introduce machine-learning algorithms investigated in our studies, Section 4 gives detailed description of our automated framework and techniques used in our experiments, Section 5 presents our main results of classification process, and Section 6 presents conclusions and future work.

2. RELATED WORK

Recent work on automated web-pages classification task can be divided into three categories. Two of them are based on had-coded rules or website content and have been studied in the context of online advertisements detection. The last one that is based only on URLs haven not been yet tested on popular classifiers to detect online ads. Generally, previous research embraces the following fields.

Classification based on hand-coded rules uses hand-coded rules, such as regular expressions, to classify web-pages. The big disadvantage of such methods

is an human effort to build the database of such rules. Furthermore, this approach is vulnerable to the evolution of advertisements, which require to constantly write new rules and rewrite old ones. The leading project using this technique to online advertisements detection is *Adblock Plus*[1], which was shown by Singh and Potdar [2009] that it is the most popular and most effective solution only using information extracted from URLs.

Classification based on Web-page content requires the analysis of web content, or the other web-pages pointing at a given website. Some interesting studies were made by Shih and Karger [2004], who created machine-learning solution to online advertisements detection based only on URL and its place on the website. More sophisticated use of website content was made by Krammer [2008], who develop rule-based algorithm using such features as: surrounding text, css style, etc. The main drawback of all methods based on website content is their time complexity. They are too slow to cope with voluminous data streams.

Classification based on URLs was studied by Devi et al. [2007]. The authors tested three popular algorithms (Naive Bayes, Support vector machine (SVM) and Radial basis function network). However, their experiments are not extensive and their results are poor (0.52 accuracy). Previous work made by Kan and Thi [2005] is more interesting. The authors introduced and adopted a few text mining techniques to extract features from URLs. Then, they classify URLs using maximum entropy model and SVM. The performance results are 0.413 and 0.627 F-measure, respectively. More comprehensive studies were made by Baykan et al. [2011] where authors examine three classifiers (Naive Bayes, SVM, Maximum Entropy) and build meta classifier using boosting technique. They achieved F-measure around $0.8-0.85$ testing many datasets and different feature generation techniques. To the best of our knowledge, no comprehensive study on online advertisements detection using machine-learning algorithms was made in this field. Esfandiari and Nock [2005] described an adaptive system based only on tokens extracted from URls and users feedback to detect adverts images. However, their work covers only weighted majority algorithm and can not deal with other advertisements types than images.

The automated framework to quantify machine-learning algorithms performance was presented by Moran et al. [2009]. The authors made an empirical study on Bayesian Network classifier scrutinizing different configurations. However, the architecture of this framework does not cover a concise way of describing the classifiers configurations and does not support distributed computation. For that reasons it can not be used to make more comprehensive study.

3. PRELIMINARIES

In this section we introduce basic concepts of machine-learning techniques. We also provide the briefly description of seven popular classifiers used in our studies.

Machine-learning is a process of finding some *hypothesis* $h : X \to C$, which is a function from a given set of entities to the set of classes. Our aim is to find such *hypothesis* that is as close as possible to a given function $c : X \to C$, which is called *concept* and represents the actual classes of the instances. In our case there are only two classes $C = \{noAd, ad\}$ representing online advertisements and legitimate

[1] http://adblockplus.org/

content of websites. With each entity $x \in X$ we associate a set of features, denoted by $F(x)$. Machine-learning process is similar to inductive reasoning. We test a few examples from a training set and infer from a few experiments general rules. Then, these rules are applied to new examples to classify them. Similarly to induction machine-learning process allows for the possibility that the conclusion is false.

In our work from each URL the meaningful tokens are extracted. These token are also called features and are described in Section 4.2. Then, each URL is represented as a vector $F = [a_1, a_2, \ldots, a_n]$, where $a_i \in \{0, 1\}$ and n is the number of all extracted features. When 0 occurs in the vector F at position i it means that a given URL does not contain feature f_i. During feature selection process, which is described in Section 4.3, we can reduce n-dimension space by ignoring some unimportant features, i.e. we remove features that do not improve classification process. The desirable output of a machine-learning algorithm is an effective classification of a given URL as *noAd* or *ad*. The detailed description how we evaluate machine-learning solutions is provided in Section 4.5.

In our studies we scrutinized the following seven popular classifiers. For each of them we examine many different configurations and scope of our experiments is briefly described in the next paragraphs.

k-nearest neighbour classifier [Aha et al. 1991] picks the most common class among k closest neighbours of a given instances. We investigated popular distance metrics: Euclidean, Chebyshev, Levenshtein and Manhattan with different k parameter and different distance weights.

Naive Bayes classifier maximizes a conditional probability of a new instance belonging to a particular class by looking at all features individually. Using Bayes' theorem this problem can be expressed with probability of instance's features occurring in specified class, which in turn can be obtained using statistics. To reduce problem even further we naively assume feature independence. This approach was tested on three configurations. We discretize continuous features using an equal frequencies method assuming a normal distribution, or using the kernel estimator introduced by George and Pat [1995]. The third configuration uses the Multi-interval discretization method proposed by Fayyad and Irani [1993].

Bayesian network classifier tries to achieve better accuracy by overcoming the naivety of Naive Bayes. This is done by modelling the relationships between features using a directed acyclic graph. There are many algorithms constructing Bayesian network and learning conditional probabilities. We combined four score measures: Entropy, Akaike information criterion (AIC) metric, minimum description length (MDL) metric and Bayesian metric with different search approaches: hill climbing, repeated hill climbing, augmented tree (TAN) and genetic search. We also tested global score metrics as cross-validation and k-fold cross validation. For more detailed description of these parameters please read work by Moran et al. [2009].

Support vector machine finds a single hyperplane that separates all the instances with maximum distance between plane and instances. Internally this is done by expressing classification task as a constrained optimization problem and solving it with quadric programming. If no such plane exists, SVM can relax a separability constraint by introducing penalty function for misclassified instances. Moreover the dimension of feature space can be greatly increased using kernel function trick.

Higher dimension separating plane projected back to the original feature space can model the variety of shapes. Our tests covered four kernel function (linear, polynomial, RBF and sigmoid) with different degrees and coefficients [Chang and Lin 2001].

Decision tree classifier [Quinlan 1992] is a tree that divides whole feature space into hyper-rectangular areas corresponding to particular classes. Inner nodes of the tree represent separating planes and each leaf indicates a class assigned to instances occupying that region of the feature space. Decision trees are built recursively. Starting in the root node, algorithm must decide whether to stop branching and create a leaf node or continue division. Tests for inner nodes are selected greedily based on the information gain estimation, which is very often expressed as an entropy reduction. For better generalization after initial generation tree nodes can be pruned and substituted with leaves. We studied two methods of pruning the tree by setting different confidence thresholds and different folds for confidence-based pruning and reduced error pruning, respectively. We also examined trees with Laplace smoothing and binary splits only. We also set different values of minimal instances per node for stop condition.

Random forest algorithm developed by Breiman [2001] is a meta-classifier based on the number independent decision trees created from the random subset of features. The output of this classifier is chosen during voting between individual trees. Random forest does not suffer from overfitting as much as single decision trees built in greedy way. We examined different number of trees in the forest with different number of features randomly selected to build it. Furthermore, we tested different limitations to the depth of each tree.

AdaBoost introduced by Freund and Schapire [1995] is another meta-classifier Initially it weights all the training instances evenly. Then, it performs the series of iterations in which a new sub-classifier is created and trained taking weights into account. The weights of properly classified instances are decreased and weight of misclassified instances are increased. This allows next sub-classifiers to focus on wrongly classified instances. Final decision is made based on weighted vote of all sub-classifiers. Their weight is strictly related to their error rate. As sub-classifier we used a simplified decision tree having at most one inner node called decision stump. We investigated different number of iteration and resampling technique described by Freund and Schapire [1996].

In order to examine above classifiers using all configurations we performed more then 2500 experiments.

4. AUTOMATED FRAMEWORK

In this section we present our automated framework. We also described in details the configuration of our experiments on URL based online advertisements detection.

The automated framework for testing different machine-learning solutions is depicted in Figure 1. It consists of modules that should be appropriately configured to meet requirements of a given application. This architecture is crucial to examine simultaneously hundreds of machine-learning solutions. Each experiment consists of the following steps:

(1) The preparation of training set, which is a collection of URLs described by

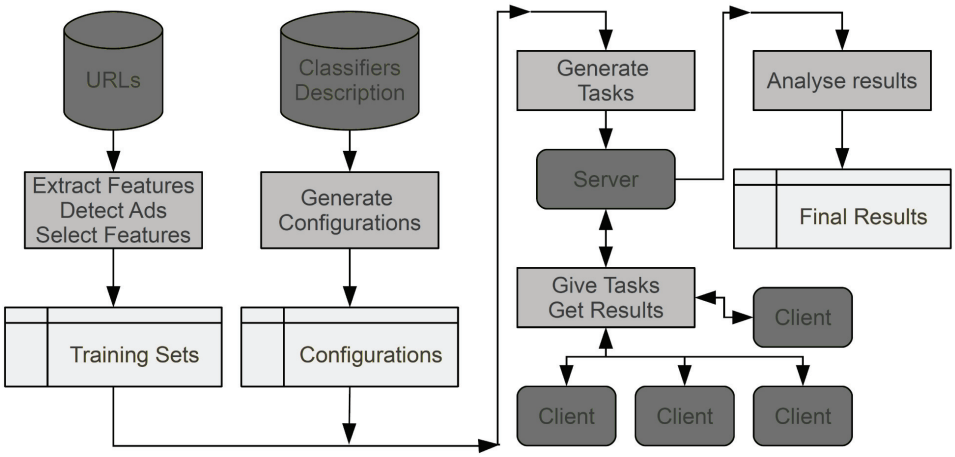

Fig. 1. The automated framework for investigation machine-learning solutions.

features. Each URL is identified as an advertisement or legitimate content and the number of features describing it is reduced in feature selection process. The detailed description how we obtained the dataset of URLs and marked ads within it is in Section 4.1, then in Section 4.2 we present text mining techniques to extract features from URLs and in Section 4.3 we describe our feature selection methods to improve classification performance.

(2) The selection of classifier and generating its correct configuration. The technique how to effectively describe the scope of our experiments is presented in Section 4.4 and all classifiers used in our studies have been already described in previous Section 3.

(3) The generation of task, which is a tuple consisting of a training set and configuration of classifier generated in previous steps.

(4) Building on each task a classifier and evaluating its performance. In Section 4.5 we propose the evaluation metrics, which allows our framework to choose the best classifier.

The application independent part of our system is *Server*, which is collecting *tasks* (the configuration of a classifier with a training set) and distributing them among *Clients*, which are performing tests simultaneously and returning results. Having performed all experiments, we automatically choose the best solution.

4.1 Dataset

We collected $55k$ instances of URLs from internet by browsing the most popular websites. We used proxy server to filter all URLs that have occurred during web surfing. Then, we construct training set by applying *Adblock Plus* to all collected instances. This solution imposes limitation to the real-life accuracy of our experiments, while even this best rule-based ad detector is fallible. However, in order to test the usefulness of URL based online ad detection this approach is sufficient. This experiment shows that near 6% of all instances are online advertisements.

4.2 Feature extraction

After obtaining dataset, each URL must be converted into a numeric feature vector. In general each URL consists of the following components:

$$scheme : //userinfo@hostname : port/path?query\_string\#fragment\_id$$

The list of features, we based our tests on, is shown below. Examples use the following URL: $http : //support.google.com/ads/?hl = en$.

Textual length of URL (total and per URL component) - Short and very long URLs are unlikely to be ads. Moreover, lengths of URL's components can be used as a valuable hint.
ComponentLength$Total = 29, ComponentLength$Host = 18, ...

URL component presence - If query component is absent or userinfo component present, we can be sure that test result for URL will be negative.
ComponentMissing$Host = false, ComponentMissing$UserInfo = true, ...

Token occurrences - Using text mining methods, URLs are splitted into words (precisely division is made at special characters). This allows us to count occurrences of each word in URL as a feature. Terms related to advertising are much more frequent among ad URLs.
Token: com = 1, Token: google = 1, Token: ads = 1, ...

Token occurrences by URL component - To distinguish advertisement link from blog post about advertising we can count token occurrences for each component. Some important tokens, as names of big advertisement companies, appear only in host component of ads URLs.
Token: Host$com = 1, Token: Path$ads = 1, ...

Sequential n-grams - Instead of counting single words we can count occurrences of token sequences. This provides to detect sequences as *Ads Fighter*, which are unlikely to occur in an advert URL.
Ngram: com>google = 1, Ngram: google>support = 1, ...

Full token n-grams - This feature extractor counts occurrences of sets of tokens with regard to succession relation. If token ädïs preceded by token "block", even not directly, given URL probably won't be an advertisement.
Ngram: com>support = 1, Ngram: com>hl = 1, ...

Token count (total and per URL component) - Ads are likely to have many parameters in query component.
TokenCount$Total = 7, TokenCount$Host = 3, TokensCount$Query = 2, ...

Numeric tokens count (total and per URL component) - Numeric values appear in URLs quite often. They generate many tokens, which contain almost no useful information. However, ad links can contain many of those including an advert size, site id and others.
NumericCount$Total = 0, NumericCount$Host = 0, ...

After applying the above extractors to our dataset we obtained $330k$ features, which was reduced by frequent value elimination to $64k$.

4.3 Feature selection

The number of features grows very quickly with the size of dataset, but most of them do not contribute significantly to classification process. They can be seen as

noise, especially when classifiers are vulnerable to the large numbers of features such as k-nearest neighbours or random forest classifiers are used. These features can be discarded, resulting in decreased training time and better accuracy.

Feature selection heuristics try to estimate the usefulness of features, so that unimportant ones can be filtered out. Depending on method, estimation refers to a single feature or whole subset of features. The former are generally faster, but they don't take a correlation between features into account. The latter can discard redundant information and produce better results, but searching in space of all available subsets is much more time-consuming.

For our tests we used two feature selection methods. Initially dataset was filtered using information gain heuristics, which estimates usefulness of feature by calculating the class value entropy reduction over whole dataset. Forman [2003] showed that this method is fast and quite effective and since the number of feature exceeds $330k$ we are able to use only method that evaluate each feature individually. After filtering the most superfluous features, second iteration is performed using greedy stepwise search with Correlation Feature Selection (CFS) estimator. This method evaluates the subsets of features, rewarding for a high correlation with a class value and penalizing for a correlations between features considered in a subset. This is very time consuming method, but after initial selection using information gain we need to traverse a space of 1000 features, what makes this method applicable.

We take different number of top features to our experiments using: 100, 200, 500, 1000, 2000, and 5000. For the number of features not exceeding 1000 we used both aforementioned methods. In order to select 1000 features and more we used only information gain algorithm.

4.4 Representation

In this section we introduce the compact representation scheme for describing different parameters of classifiers. The number of classifiers and its configurations spans a huge space of possible optimal solutions. In order to search it we introduce the syntax of formulas ($\mathcal{F}$), which gives us opportunity to describe number of tests in concise way.

Firstly, we propose three atomic formulas (AF) that are commonly used in many machine-learning algorithms. Namely, *DoubleParameter* (DP), *IntegerParameter* (IP) and *StringParameter* (SP), which are not empty sets of ordered pairs (l, v_i), where l is some label and v_i is double, integer and string, respectively. Secondly, we introduce *ObjectParameter* (OP), which is an ordered pair $(l, \mathcal{F})$, where l is label and $\mathcal{F}$ atomic formulas, or other *ObjectParameters* connected with some logical connectives. In real-life application each of these parameters is labelled with some symbol, which is the name of the parameter.

The single configuration of classifier (model) is a n-tuple $\langle x_0, \ldots, x_n \rangle$, where $x_i \in AF \cup (l, c)$, where (l, c) is an ordered pair with label l and configuration c corresponding to some *ObjectParameter*. Now, we say that a given set of configurations M meets a given atomic formula AF if the set of all n-tuple generated from a given formula is identical to M. Formally:

$$M \models AF \iff M = AF^{\times},$$

where operation $AF^{\times}$ transforms the set of elements into the set of singletons (1-

tuples).

Example 1. Let us define *StringParameter* $SP = \{(-W, first), (-W, last)\}$. We have that $SP^{\times} = \{\langle(-W, first)\rangle, \langle(-W, last)\rangle\}$ and:

$$\{\langle(-W, first)\rangle, \langle(-W, last)\rangle\} \vDash SP$$
$$\{\langle(-W, first)\rangle\} \nvDash SP$$

Now, we introduce the syntax of our formulas ($\mathcal{F}$) , which is:

$$\begin{aligned} AF &:= DP \mid IP \mid SP \\ \mathcal{F} &:= AF \mid OP \\ \mathcal{F} &:= (\mathcal{F}) \mid \mathcal{F} \wedge \mathcal{F} \mid \mathcal{F} \vee \mathcal{F} \end{aligned}$$

From a given syntax and from the fact that $OP = (l, \mathcal{F})$ we see that introducing *ObjectParameters* is necessary to express the configurations of the classifiers having parameters with their own inner (nested) configuration.

We say that a given set of configurations M meets a given formula $\mathcal{F}$ if the set of all n-tuples generated from a given formula is identical to M. Formally:

$$M \models \mathcal{F} \iff M = \mathcal{F}^{\times},$$

where the generation process for atomic formulas has been already defined, and for complex formulas it is defined in the following way:

$$\begin{aligned} M \vDash (\mathcal{F}) &\iff M = \mathcal{F}^{\times} \\ M \vDash OP &\iff M = \{\langle(l, t_i)\rangle\} \text{ , where } t_i \in \mathcal{F}^{\times} \text{ and } OP = (l, \mathcal{F}) \\ M \vDash \mathcal{F}_1 \wedge \mathcal{F}_2 &\iff M = \mathcal{F}_1 \times \mathcal{F}_2 \\ M \vDash \mathcal{F}_1 \vee \mathcal{F}_2 &\iff M = \mathcal{F}_1^{\times} \text{ or } M = \mathcal{F}_2^{\times} \end{aligned}$$

The notation $\mathcal{F}_1 \times \mathcal{F}_2$ denotes the standard Cartesian product, which is defined:

$$\mathcal{F}_1 \times \mathcal{F}_2 = \{\langle(l_1, x), (l_2, y)\rangle \mid \langle(l_1, x)\rangle \in \mathcal{F}_1^{\times} \wedge \langle(l_2, y)\rangle \in \mathcal{F}_2^{\times}\}$$

Let us introduced the real-life example, where the labels of parameters were taken from WEKA project.

Example 2. Firstly, we would like to test Random Forest algorithm with the different number of trees and selected features (parameter *-I* and *-K*, respectively). We set up the following formula:

$$\begin{aligned} \mathcal{F}_{forest} &:= (RandomForest, \mathcal{F}) \\ \mathcal{F} &:= \mathcal{F}_1 \wedge \mathcal{F}_2 \\ \mathcal{F}_1 &:= \{(-I, 10), (-I, 30), (-I, 50), (-I, 100)\} \\ \mathcal{F}_2 &:= \{(-K, 5), (-K, 10), (-K, 15), (-K, 20)\} \end{aligned}$$

The above formula will create 16 different configurations.

Now, we would like to improve our results by running AdaBoost, which is meta classifier and requires other classifier as a parameter. We test different number of iterations of AdaBoost (*-I*).

$$\begin{aligned} \mathcal{F}_{ada} &:= (AdaBoostM1, \mathcal{F}_3) \\ \mathcal{F}_3 &:= \mathcal{F}_4 \wedge \mathcal{F}_5 \\ \mathcal{F}_4 &:= \{(-I, 50), (-I, 100), (-I, 200), (-I, 400)\} \\ \mathcal{F}_5 &:= \{(-W, \mathcal{F}_{forest})\} \end{aligned}$$

The above formula will create 64 different configurations.

The compact representation introduced in this section is crucial in order to examine a wide range of classifiers with different parameters. We also believe that the future work on searching for optimal solution in the space of all possible configurations requires formalization.

4.5 Evaluation methods

The specification of our problem requires an appropriate evaluation method. We need to take into account two important issues. Firstly, only 6% of all instances are ads, so basic evaluation metric *accuracy* is very hight even if our classifier can not detect any online advertisements. Secondly, online advertisement filtering is a cost-sensitive classification task. Misclassifying legitimate web content is generally more costly then misclassifying unwanted ads. To deal with the second problem we can use *cost sensitive accuracy*, which is given by a formula:

$$Accuracy = \frac{\alpha TP + TN}{\alpha TP + \alpha FP + TN + FN},$$

where TP, FP, TN, FN are true positive (detected ads), false positive (undetected ads), true negative (detected legitimate content) and false negative (undetected legitimate content) rates, respectively. We assume $\alpha = 10$, which means that misclassification legitimate content is 10 time more costly than not detecting an advertisement. This assumption was proposed by Berry and Kogan [2010] as an appropriate distinction between spam and legitimate mail in spam filtering problems.

In order to deal with dataset, where adverts are rare, we will use *F-measure*, which is an harmonic mean of *precision* and *recall*. The latter tells us how many advertisements we can truly detect and is given by the formula: $\frac{TP}{TP+FN}$. On the other hand precision gives us information about the misclassification of legitimate content and is given by formula: $\frac{TP}{TP+FP}$. Now, meeting the above two requirements we introduce *cost sensitive F-measure* known in literature as F_β*-measure*:

$$F_\beta\text{-}measure = \frac{(1+\beta^2) \cdot precisio \cdot recall}{\beta^2 \cdot precision + recall},$$

where we assume that $\beta = 0.25$, which means that we are looking for solutions with high *precision*.

In our framework we evaluate results using *5-fold cross validation*.

5. EXPERIMENTS

In this section we present our main results. The implementations of all classifiers are taken from WEKA project[2]. All performance were made on AMD 64-cores (4x16 cores Opteron 6276/2.3GHz) 320 GB RAM.

The best performance of each classifier is shown in Figure 2. We choose the $F_{0.25}$*-measure* as a criterion and contrasted it with other metrics. We want to maximize *precision*, so almost all results (except AdaBoost) have a low number of false

[2]http://www.cs.waikato.ac.nz/ml/weka/

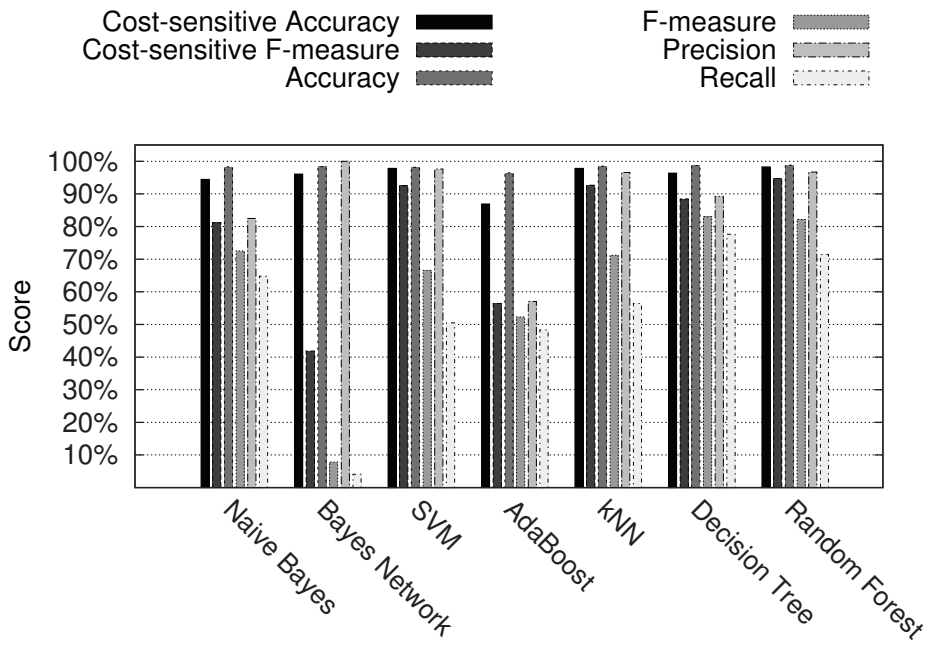

Fig. 2. Comparison of all classifiers performance measured with different metrics.

positive instances. We see that in our cost-sensitive problem the best compromise between *precision* and *recall* gives Random Forest algorithm. It achieved 0.967 *precision* and 0.714 *recall* what results in 0.822 *F-measure* and 0.947 *cost sensitive F-measure*. The best configuration for Random Forest consists of: 200 trees, 50 randomly chosen features and 50 – the maximum depth of each tree. The second best classifier is kNN algorithm with 0.927 *cost sensitive F-measure*. However, it belongs to the group of so called *lazy* classifiers. It can learn fast, but the process of classifying new instances is too time consuming. This disadvantage is clearly depicted in Figure 3.

Figure 3 shows training and testing time for all classifiers. These times are chosen for the best performance for each classifier. Our training set consists of $55k$ instances and using *5-fold cross validation* we divide it into two pieces with $44k$ and $11k$ instances, respectively. Training time stands for an average time to learn a given classifier using $44k$ instances and testing time stands for an average time necessary to classify $11k$ new instances. Although near one hour is required to learn Random Forest algorithm, its testing time is only $1.5s$ (almost the same time like Naive Bayes), what makes this solution applicable to real life problems.

Figure 4 presents how the number of features influences the algorithm performance. Our experiments shows that the percentage of all features needed to properly classify URLs is quite low. For Random Forest, our optimal solution, it is only 0.006%. This result stems from the fact that there exists a small specific group of words and patterns usually occurring in advertisements URLs. This figure also

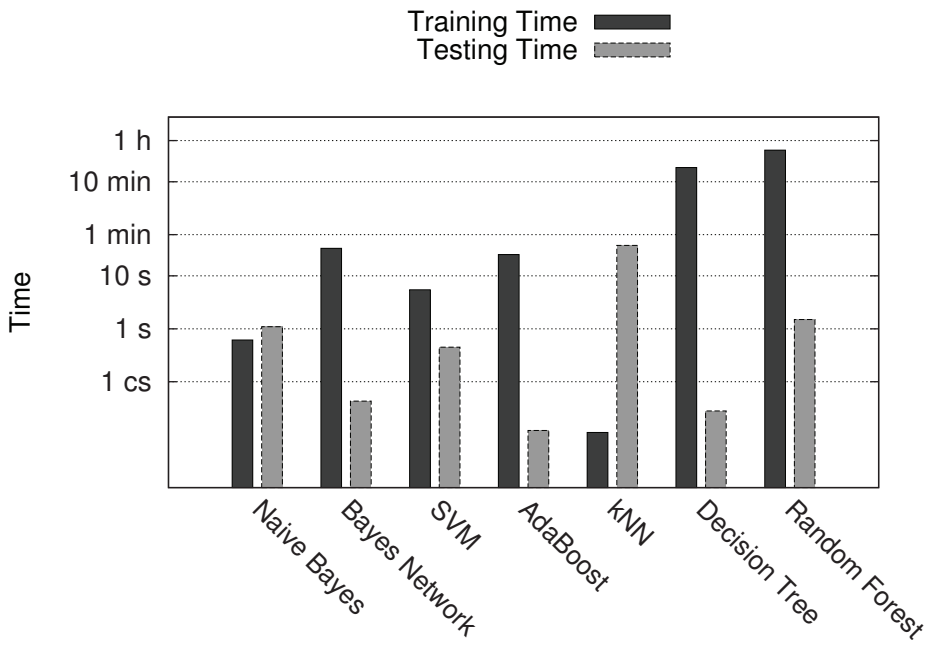

Fig. 3. Training and testing time for the best performance of all seven classifiers.

presents how unimportant features could disturb machine-learning classifier performance.

6. CONCLUSIONS AND FUTURE WORK

The experts knowledge is expensive while the computational power is becoming much more cheaper. We see a great potential in using automated framework to investigate the number of machine-learning solutions. We examined such framework to find the best solution to online advertisements detection.

Our results shows that URL based online advertisements detection is very effective. Our investigation of a few machine-learning classifiers with hundreds of different parameters shows that Random Forest classifier gives the best performance. The evaluation metrics give 0.987 accuracy and 0.83 F-measure.

The number of different classifiers with wide range of parameters spans the huge space of possible solutions. In some cases, the expert knowledge is enough to deal with the problem. However, in many new domains the best machine-learning solutions are unknown. In both cases the formal description of the problem is required. To express the expert knowledge formal logic with rules of inference would be a useful tool. To develop efficient search algorithms to find optimal solutions the good and effective space representation is necessary. We strongly believe that these two fields (formal logic, search algorithms) are very interesting direction of future research.

Acknowledgments.. This work was supported by PTC T-Mobile Poland. We would like to express our gratitude toward Prof. Mieczysław Muraszkiewicz and

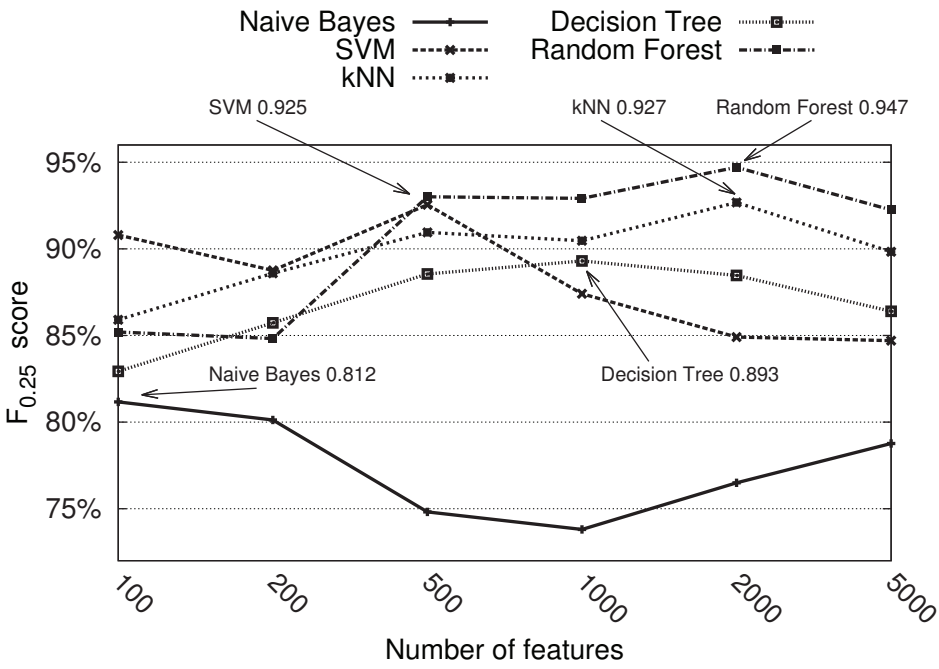

Fig. 4. The best five classifiers performance for a given number of features.

Dariusz Mastalerz for their kind co-operation which helped us in successfully completion of this research.

REFERENCES

Aha, D. W., Kibler, D., and Albert, M. K. 1991. Instance-based learning algorithms. *Machine Learning 6,* 1 (Jan.), 37–66.

Baykan, E., Henzinger, M., Marian, L., and Weber, I. 2011. A comprehensive study of features and algorithms for url-based topic classification. *ACM Transactions on Web 5,* 3 (July), 15:1–15:29.

Berry, M. W. and Kogan, J., Eds. 2010. *Text Mining: Applications and Theory.* Wiley, Chichester, UK.

Breiman, L. 2001. Random forests. *Machine Learning 45,* 1 (Oct.), 5–32.

Chang, C.-C. and Lin, C.-J. 2001. Libsvm - a library for support vector machines. The Weka classifier works with version 2.82 of LIBSVM.

Devi, M. I., Rajaram, R., and Selvakuberan, K. 2007. Machine learning techniques for automated web page classification using url features. In *ICCIMA '07: Proceedings of the International Conference on Computational Intelligence and Multimedia Applications.* Vol. 2. IEEE Computer Society, Washington, DC, USA, 116–120.

Esfandiari, B. and Nock, R. 2005. Adaptive filtering of advertisements on web pages. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web.* ACM, New York, NY, USA, 916–917.

Fayyad, U. and Irani, K. 1993. Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. *IJCAI '93: Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 1022–1027.

Forman, G. 2003. An extensive empirical study of feature selection metrics for text classification. *The Journal of Machine Learning Research 3*, 1289–1305.

Freund, Y. and Schapire, R. E. 1995. A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT '95: Proceedings of the 2nd European Conference on Computational Learning Theory*. Springer-Verlag, London, UK, 23–37.

Freund, Y. and Schapire, R. E. 1996. Experiments with a New Boosting Algorithm. In *ICML '96: Proceedings of the 13th International Conference on Machine Learning*. 148–156.

George, J. and Pat, L. 1995. Estimating continuous distributions in bayesian classifiers. In *In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 338–345.

Kan, M. Y. 2004. Web page classification without the web page. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*. ACM, New York, NY, USA, 262–263.

Kan, M.-Y. and Thi, H. O. N. 2005. Fast webpage classification using url features. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*. ACM, New York, NY, USA, 325–326.

Krammer, V. 2008. An effective defense against intrusive web advertising. In *PST '08: Proceedings of the 2008 Sixth Annual Conference on Privacy, Security and Trust*. IEEE Computer Society, Washington, DC, USA, 3–14.

Moran, S., He, Y., and Liu, K. 2009. Choosing the best bayesian classifier: an empirical study. *IAENG International Journal of Computer Science 36,* 4, 322–331.

Quinlan, J. R. 1992. *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)*, 1 ed. Morgan Kaufmann.

Shih, L. K. and Karger, D. R. 2004. Using urls and table layout for web classification tasks. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*. ACM, New York, NY, USA, 193–202.

Singh, A. K. and Potdar, V. 2009. Blocking online advertising - a state of the art. In *ICIT '09: Proceedings of the 2009 IEEE International Conference on Industrial Technology*. IEEE Computer Society, Washington, DC, USA, 1–10.

PIOTR L. SZCZEPAŃSKI AND ADRIAN WIŚNIEWSKI,
Institute of Informatics, Warsaw University of Technology,
Pl. Politechniki 1, 00-661 Warsaw,
Poland

TOMASZ GERSZBERG,
T-Mobile Poland,
Al. Jerozolimskie 181, 02-222 Warsaw,
Poland.

Received March 2013