Discovering of Alternative Marketplaces on the Web for Mobile App Security Monitoring

Massimo Guarascio[1], Ettore Ritacco[1], Francesco S. Pisani[1], Daniele Biondo[2], Rocco Mammoliti[2] and Alessandra Toma[2]

[1] Institute for High Performance Computing and Networking of the Italian National Research Council (ICAR - CNR), Italy
[2] Poste Italiane, Rome, Italy

Abstract. Brand reputation is an open issue for several companies delivering services through dedicated apps. The latter are often targeted by malicious developers who spread unauthorized (fake, malicious, obsolete or deprecated) versions through alternative distribution channels and app stores. The aim of the work is the early detection of these alternative markets advertised through social media such as Twitter of Facebook or hosted in the Dark Web. Specifically, we propose a semi-automatic approach to monitor these media and to recommend web pages that are likely to represent alternative marketplaces. The underlying predictive platform allows to analyze web pages extracted from the Web and exploits an ensemble classification model to distinguish between real app stores and similar pages (i.e. blogs, forums, etc.) which can be erroneously returned by a common search engine. An experimental evaluation on a real dataset confirms the validity of the approach in terms of accuracy.

1 Introduction

Nowadays, smartphones and tablets are widespread devices used by millions of users. Their popularity is mainly due to their reduced dimensions and the availability of a wide range of useful applications provided by marketplaces and app stores. A major threat for such devices is the exposure to counterfeit apps which can compromise the security of the devices and eventually the reputation of the original developer. Indeed, unaware users might install unauthorized (e.g. fake, malicious, obsolete and deprecated) apps, which can potentially harm the device and consequently the brand reputation of the copyright owner. There's an exponential growth of virus and trojans able to attack commonly used devices [3]. It is quite easy to develop variants of well-known malware [2], and many popular security tools are not able to counteract common malware transformation techniques [18].

In many cases, these malicious programs are disguised as popular apps spreading via official or alternative market places (e.g. Amazon app store for Android[3]). As an example [4], in July 2016 approximately two hundred mobile apps were diffused as an official version of the *Pokémon Go* game,[4] most of which being

[3] `https://www.amazon.com/mobile-apps/b?ie=UTF8&node=2350149011`
[4] `http://www.pokemongo.com/`

malicious applications able to permanently lock the device. Other examples are the *PayPal App* clones and counterfeits which aim at stealing the login data of unaware users who accidentally update the original app to the malicious version.

Besides representing a risk for the end user, the spread of these applications represents a serious threat even for the original service providers and developers. Being directly or indirectly associated with harmful apps can cause users to give up on their services, thus causing a reputational damage. That's why the early detection of these apps is becoming strategic from a *Brand Protection* point of view [22]. Notice that the problem is relevant even in situations where there are no security issues, but just the possibility of being associated with poorly designed apps and services. For example, the mobile app *BancoSaldo* is a third-party app providing many utilities for handling financial services of the Italian postal service "Poste Italiane". The whole app (UI and Logo) has been designed to look a legitimate Poste Italiane product. Hence it's easy for an end user to confuse this app for an official one distributed by Poste Italiane, thus attributing all the potential failures to Poste Italiane itself.

The problem is that typically, malicious, fake or obsolete apps spread via unofficial channels (e.g. *alternative marketplaces and app stores*) accessible both via regular and *Dark* Web and therefore they are difficult to be discovered. The capability of identifying and monitoring alternative marketplaces both on Regular and Dark Web is an relevant and challenging task. These marketplaces are extremely dynamic and often do not monitor the published apps, and typically, they are advertised through social media posts. Notably, companies set up teams of experts and specialized personnel to discover these alternative markets and to inspect whether potentially harmful apps are available which can be associated with them. Anyway, the whole process to detect unauthorized app stores is usually performed manually by exploiting suitable queries on well-known research engines (e.g. by employing google hacking techniques) and they strongly rely on the skills of the operator.

Defining semi-automatic monitoring protocols is crucial for effectively counteracting the malicious mobile app diffusion, since it allows human operators to analyze a wider web search space. Our main aim, in this work, is to propose an *intelligent* infrastructure for continuously monitoring and analyzing the Web in order to detect alternative or unofficial marketplaces that may contain unauthorized mobile apps.

The current literature has mainly focused on the problem of monitoring and detecting malicious mobile apps, and little effort has been devoted to the detection of alternative markets.

To the best of our knowledge this is the first attempt to tackle this problem. To summarize, the main contributions of this work are:

- A prototype platform for proactively discovering (alternative) app stores on Regular and Dark Web;
- A learning approach for accurately classifying and ranking web pages according the probability to be a real mobile apps marketplace (*UASD*).

The rest of the paper is organized as follows. After introducing some preliminary concepts in Section 2, we present our solution approach for discovering alternative app stores. An empirical analysis on a real case study is then discussed in Section 3. Finally, the section 4 concludes the paper and presents some interesting future developments.

2 The UASD Framework

UASD stands for *Unauthorized App Store Discovery* , and it is a semi-automatic machine learning approach that supports human operators recommending the most likely alternative app stores in a (Dark) Web research space. *UASD* is composed by three main macro components shown in Fig. 1: Information Retrieval, Knowledge Discovery and Interaction with the operator. The details of the modules are as follows.

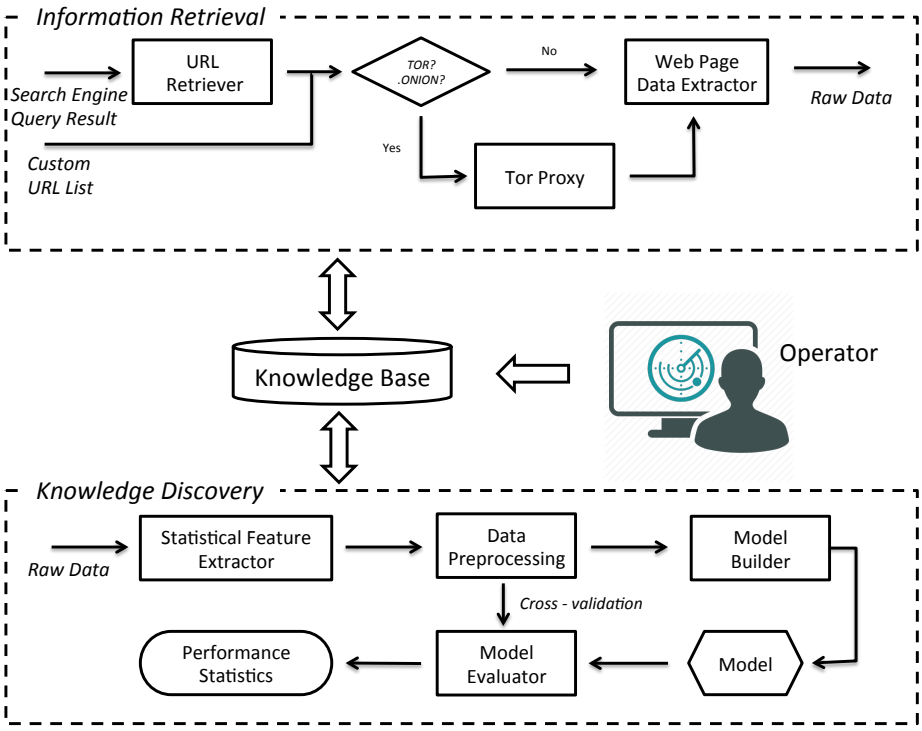

Fig. 1. Logical flow of UASD.

2.1 Information Retrieval

Human experts devise a set of web queries by exploiting Advanced Google Search Operators [1] or specify some `URL`s in order to identify possible alternative mobile markets in regular Web or in the *TOR* network (Dark Web). Typically, these queries contain keywords referring to the names of the most popular mobile operative systems, app categories (such as e.g. "arcade games", "puzzles"), specific mobile apps or even mobile devices. The **Information Retrieval** module is fed with these queries. Its task is to interpret them and extract the related pages. The **URL Retriever** retrieves the `URL`s from the query results. If a `URL` belongs to the *TOR* network, then a **Tor Proxy** allows the system to access the page in the dark web and collect its content. At the end of this process a **Knowledge Base** is populated with all the extracted pages in `HTML` format. The **Web Page Data Extractor** allows to store in the Knowledge Base the html text (raw data) contained in the selected web pages.

2.2 Knowledge Discovery

This component allows to learn a classification/prediction model from the data (knowledge base) gathered by the information retrieval module. We can devise three components in it.

Data Transformation. Collected data need to be transformed and filtered in order to be provided as input to the machine learning process. A crucial step for this task is performed by a **Statistical Feature Extractor**, that allows to devise a set of (discriminative) structural features for each web page, denoted as **embedded attributes**. These features are based on the assumption that the Web can be modeled as a graph whose nodes are the web pages and edges are the hyperlink references. Each node has a neighborhood composed by those nodes that are directly linked to it. Hence, given a raw web page, the current implementation of the Statistical Feature Extractor builds a set of discriminative features (embedded attributes) based on target page's neighborhood. Table 1 summarizes the main features extracted from the pages.

Besides the embedded attributes, content features can be extracted directly from the HTML code of the page. Specifically, the page is described by the *META* tags and the keywords belonging to its *BODY* (where HTML tags are removed). The **Data Preprocessing Module** filters such a content and produces further attributes: specifically, META tags are directly converted in (logical) relational attributes, text extracted from the BODY is converted by exploiting *Text Mining* techniques.

The merging of embedded attributes, meta tags and textual features represents the general schema upon which to characterize an extracted web page.

Prediction Model. The output of the Data Preprocessing Module is a *Cleaned Data Set* that can be used for generating a classifier capable to discriminate

Table 1. Embedded attributes.

Attribute	Description
`isTorLink`	a boolean flag highlighting whether the page was extracted from the Dark Web
`NumberIntraDomainLinks`	the number of neighbors within the same web domain of the target page
`IntraDomainLinkPercentage`	the ratio between the `NumberIntraDomainLinks` and the neighborhood size
`IntraDomainDistinctLinkPercentage`	similar to `IntraDomainLinkPercentage` but distinct links are considered
`NumberDownloads`	number of direct download links for apps (e.g. *.apk*) or keywords (e.g. *install*) that allow a user to get a mobile application
`NumberKeywords`	Number of keywords typically included in app stores (suggested by domain experts): e.g. *android*, *apk*, *ios*, *access*, `app`, *categories*, *market*, *price*, *best*, `popular`, *top*, *rated*, *ios*, *windows phone*, etc.
`NumberDownloadFirstLevel`	the cumulative *NumberDownloads* for the neighbors within the same domain of the target page
`NumberKeywordsFirstLevel`	the cumulative *NumberKeywords* for the neighbors within the same domain of the target page

between records corresponding to actual app store pages and records referring to regular Web. UASD relies on an Ensemble classifier [9].

An Ensemble is a combination of two or more classifiers (typically at least three) according to different strategies in order to achieve a better prediction: the idea is that classification errors are less likely with several classifiers rather than a single classifier. In literature different combination strategies have been proposed, the most known of them are three: Bootstrap aggregation (bagging) [5], Boosting [20] and Stacking [23]. In our solution, we use a stacking approach since the amount of labeled data is limited. Small data sets are more prone to the overfitting problem and it is well known from the literature that stacking is more robust in this setting. The Stacking approach, also called stacked generalization, is a two-step strategy. In the first step, several different classifiers, *Base Learners*, are trained on the whole data set: each one of them will enrich the data with its prediction. A new data set, often called *Stacked-View*, is then built by the

combination of all the predictions and the original data. The stacked-view is provided as input for one last classifier, called meta-classifier. The prediction process of a new record will follow two steps: firstly, the record will be equipped with the base learners' predictions, then the meta-classifier returns the final prediction about the enriched record.

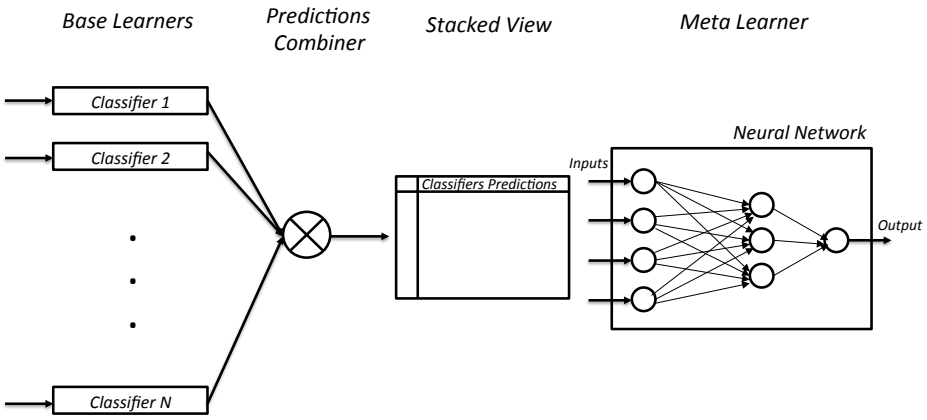

Fig. 2. UASD Ensemble Model.

The structure of the stacking method adopted in UASD is shown in Fig. 2. We chose 9 different base learners, in order to promote the diversity of the single predictions. These base learners are algorithms which work particularly well in unbalanced scenarios (i.e. where a dominant class exists). This is exactly the situation we cope with: the frequency of apps stores in the training data is extremely lower than the regular web pages' one. The complete list of the chosen algorithms is the following: *AODE* [21], *Hidden Naive Bayes* [24], *Maximum Entropy* [14], *Mine Rule AODE* [7], *Mine Rule Naive Bayes* [7], *Bayesian Approach (Discretization)* [10], *Bayesian Approach (Kernel Transformation)* [10], *Nearest Neighbors* [8] and *Logistic Regression* [13]. The predictions of these base learners are combined in a stacked view that feeds a Neural Network [16] as a meta-learner.

Evaluation We assess the quality of the devised predictive engine by computing some accuracy metrics. As aforesaid, the discovering of novel app stores is an unbalanced classification problem: the number of positive examples (actual app store) is overwhelmed by the negatives (rest of the web pages). Different evaluation metrics have been proposed in literature for testing the effectiveness of classification models in presence of a rare class. Indeed, the usage of metrics that do not adequately accounts for the rarity of such a minority class may lead to overestimating the real capability of a classifier to correctly recognize the instances of that class. Typically, some core metrics are based on the following

statistics: *(i) True Positives* (TP), i.e. the number of positive cases correctly classified as such; *(ii) False Positives* (FP), i.e. the number of negative cases incorrectly classified as positive; *(iii) False Negatives*, i.e. the number of positive cases incorrectly classified as negative; and *(iv) True Negatives*, i.e. the number of negative cases correctly classified as such.

We exploit such statistics to compute the standard *Precision* (P) and *Recall* (R) measures[6] on the minority class, in order to support fine grain analyses on the misclassification errors made over those instances:

$$Prec = \frac{TP}{TP+FP}$$
$$Rec = \frac{TP}{TP+FN}$$

The F-measure, defined as

$$F_1 = \frac{2 * Prec * Rec}{Prec + Rec}$$

represents the harmonic mean of the above measures and it is used to combine them in a single score [19].

2.3 Human Interaction and Incremental Learning

The set up phase is the initial process of the system, during which the first model is generated. This phase requires a small seed set of manually labeled web pages. The overall process is monitored by human experts, as described in Fig. 3 which represents the steady and operational states of the system. An operator defines a set of queries and selects a set of target links to submit to the URL Retriever. Relevant data is extracted and preprocessed from the links, and the resulting pages are submitted to the prediction engine which performs the prediction and sorts the pages according to their likelihood to represent a marketplace. The list is returned to the operator, and her feedbacks are again exploited to enrich the original training data and consequently update the prediction model.

3 Experimental Evaluation

In this section, we discuss the experiments carried out on a real application scenario. The scenario and the data at hand are discussed first. We next discuss the setting adopted to evaluate the quality of discovered models and evaluate the results of the performed test.

Case Study. The evaluation is performed on a crawl of 440 pages examined by domain experts at Poste Italiane. Within these, 40 web pages represent actual app stores and the remaining pages represent normal sites. The pages underwent the preprocessing module and a final dataset characterized by the feature described in Section 2 was created.

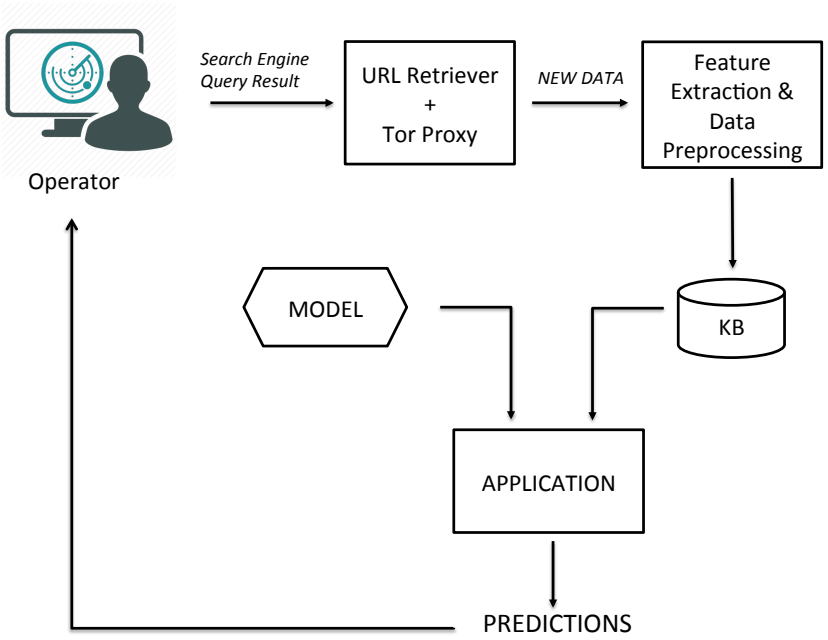

Fig. 3. Interaction with the operator.

The dataset clearly devises an unbalanced classification problem where the number of examples, belonging to the class of interest, is significantly lower than the majority class. The data preparation phase generates almost 12,000 attributes (mostly textual features and META tags), which are further preprocessed by the feature evaluator and consequently reduced to approximately 3,000 relevant attributes.

Performance Measures and Evaluation Setting. The model evaluation is accomplished relying on Stratified 10-fold Cross Validation [17], and measuring Precision (i.e., confidence of a prediction), Recall (i.e., coverage of a prediction) and F-Measure (the harmonic mean between precision and recall, also called *F1-Score*) [15].

Experimental Results. The purpose of the analysis is twofold.

- First, we aim at evaluating the role of the embedding attributes. Under this perspective, we evaluate the performance of the base classifiers by considering the full set of features and by excluding the embedding attributes.
- Second, we evaluate the role of the stacking architecture. In principle, the addition of a further level where the set of available features is enriched should allow to better balance the prediction.

In the context we are analyzing, the objective is to obtain a good balance between precision and recall with regard to the minority class label. In fact,

the aim is to increase the number of covered marketplaces by simultaneously minimizing the number of false positives, i.e. the incorrect suggestions to the operator. In this respect, $F_1 - Score$ represents a good choice to evaluate the performance of the classification model.

Table 2. Evaluation of $F_1 - Score$, Precision and Recall on base models compared to UASD.

Use embedded attributes	Model	F_1-Score	Precision	Recall
N	AODE	0,712	0,788	0,650
	Hidden Naive Bayes	0,687	0,852	0,575
	Max-Ent	0,693	0,743	0,650
	Mine Rule AODE	0,699	0,674	0,725
	Mine Rule Naive Bayes	0,658	0,694	0,625
	Bayesian Approach (Discretization)	0,692	0,711	0,675
	Bayesian Approach (Kernel Transformation)	0,687	0,917	0,550
	Nearest Neighbors	0,667	0,846	0,55
	Logistic Regression	0,694	0,781	0,625
Y	AODE	0,740	0,818	0,675
	Hidden Naive Bayes	0,722	0,813	0,650
	Max-Ent	0,707	0,690	0,725
	Mine Rule AODE	0,690	0,659	0,725
	Mine Rule Naive Bayes	0,700	0,700	0,700
	Bayesian Approach (Discretization)	0,725	0,725	0,725
	Bayesian Approach (Kernel Transformation)	0,725	0,862	0,625
	Nearest Neighbors	0,694	0,781	0,625
	Logistic Regression	0,743	0,867	0,650
UASD Approach		**0,757**	0,824	0,700

In Table 2 we report the results of this analysis. The role of the embedding attributes can be clearly appreciated and it increases the general performance of the base classifiers by 5%. The stacker is also a winner in the evaluation, as its adoption further boost the prediction quality.

3.1 UASD in action

UASD is currently employed by a Poste Italiane expert team to monitor the introduction of potentially harmful apps. The platform, integrated in MASM[5] an advanced security system for the analysis of mobile apps, provides the operator with a set of graphical tools for querying the recommendation engine which returns a ranked list of URLs ordered according the estimated probability to be a market.

In figure 4, we show an example screenshot, where the result of a specific query returns a set of potentially matching (unknown) marketplaces. Within the figure, we can notice the first four matches, representing three true positive examples (with a probability above 97%) and a false positive example (the wikipedia page for "app store" deemed as an app store with probability 60%).

[5] `http://www.posteitaliane.it/en/innovation/technology_centre/certcyb.shtml`

It is interesting to highlight that the latter contains many terms that typically characterize an actual marketplace and exhibits a similar structure in terms of intra-domain links, and at the same time the lack of other embedded features (such as direct download links to apk) lowers the positiveness of the likelihood.

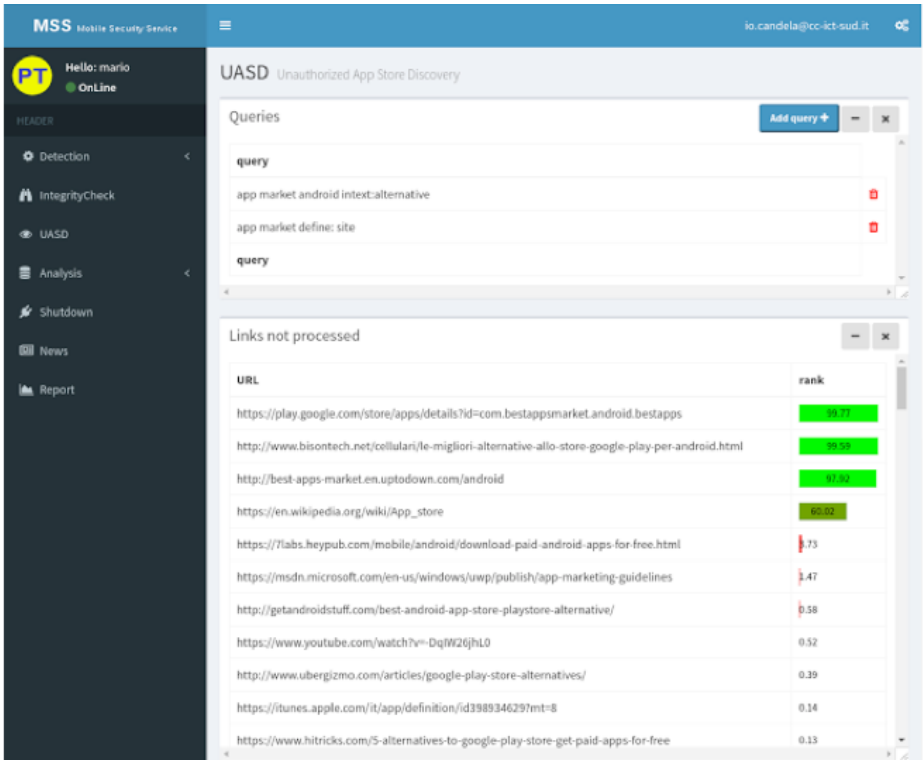

Fig. 4. Dashboard of MASM integrating UASD.

4 Conclusion

Detecting malicious behaviors of mobile apps is a challenging task and it comprises a continuous monitoring of the apps available in marketplaces. Most of the current approaches assume prior knowledge about the marketplaces to be monitored. Due to the continuous growth and dynamism of mobile app providers, identifying these app stores is becoming a difficult and time-consuming task. In this work, we propose a semi-automatic machine learning approach based on a suitable set of (derived) discriminative features and an ensemble learning method for discovering alternative mobile app marketplaces. Our approach has been im-

plemented in a prototype platform for the proactive searching of marketplaces both on Regular and Dark Web, called UASD.

UASD provides a list of URLs (extracted from a set of queries defined by the Human operator) sorted according the probability to be an actual app store, then, the operator can evaluate the provided URL list. The platform has been integrated as a service of MASM, an advanced security system for the analysis of mobile apps that is developed and currently employed by Poste Italiane. Experimental findings on a real use case confirm that our approach is effective in identifying these marketplaces.

As future work, we plan to investigate the possibility to use Deep Learning approaches [11] to automatically discover higher-level features from raw data. This could allow to extract further discriminative features that are difficult to be manually defined. Moreover, we want to investigate the possibility to equip our approach with some collective mining techniques, e.g. [12], in order to exploit other informations related to the structure of the web page links.

References

1. Google hacking. https://en.wikipedia.org/wiki/Google_hacking
2. McAfee threats report: Fourth quarter 2011. http://www.intel.se/content/dam/www/public/us/en/documents/reports/mcafee-threats-quarterly-report.pdf (2011)
3. McAfee threats report: First quarter 2012. https://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q1-2012.pdf (2012)
4. Security researchers find 215 fake pokemon go apps and issue android ransomware warning. http://www.silicon.co.uk/security/fake-pokemon-go-mobile-apps-195141 (2016)
5. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (Aug 1996), http://dx.doi.org/10.1023/A:1018054314350
6. Buckland, M., Gey, F.: The relationship between recall and precision. J. Am. Soc. Inf. Sci. 45(1), 12–19 (Jan 1994), http://dx.doi.org/10.1002/(SICI)1097-4571(199401)45:1<12::AID-ASI2>3.0.CO;2-L
7. Costa, G., Guarascio, M., Manco, G., Ortale, R., Ritacco, E.: Rule learning with probabilistic smoothing. In: Proceedings of the 11th International Conference on Data Warehousing and Knowledge Discovery. pp. 428–440. DaWaK '09, Springer-Verlag, Berlin, Heidelberg (2009), http://dx.doi.org/10.1007/978-3-642-03730-6_34
8. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theor. 13(1), 21–27 (Sep 2006), http://dx.doi.org/10.1109/TIT.1967.1053964
9. Jurek, A., Bi, Y., Wu, S., Nugent, C.: A survey of commonly used ensemble-based classification techniques. The Knowledge Engineering Review 29(5), 551?581 (2014)
10. Langley, P., Iba, and, W., Thompson, K.: An analysis of bayesian classifiers. In: Proceedings of the Tenth National Conference on Artificial Intelligence. pp. 223–228. AAAI'92, AAAI Press (1992), http://dl.acm.org/citation.cfm?id=1867135.1867170
11. Lecun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature 521(7553), 436–444 (5 2015)

12. Loglisci, C., Appice, A., Malerba, D.: Collective regression for handling autocorrelation of network data in a transductive setting. J. Intell. Inf. Syst. 46(3), 447–472 (Jun 2016), http://dx.doi.org/10.1007/s10844-015-0361-8
13. McCullagh, P., Nelder, J.A.: Generalized linear models (Second edition). London: Chapman & Hall (1989)
14. Phillips, S.J., Dudík, M., Schapire, R.E.: A maximum entropy approach to species distribution modeling. In: Proceedings of the Twenty-first International Conference on Machine Learning. pp. 83–. ICML '04, ACM, New York, NY, USA (2004), http://doi.acm.org/10.1145/1015330.1015412
15. Powers, D.M.W.: Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation. Journal of Machine Learning Technologies 2(1), 37–63 (2011)
16. Priddy, K.L., Keller, P.E.: Artificial Neural Networks: An Introduction (SPIE Tutorial Texts in Optical Engineering, Vol. TT68). SPIE- International Society for Optical Engineering (2005)
17. Purushotham, S., Tripathy, B.K.: Evaluation of Classifier Models Using Stratified Tenfold Cross Validation Techniques, pp. 680–690. Springer Berlin Heidelberg, Berlin, Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-29216-3_74
18. Rastogi, V., Chen, Y., Jiang, X.: Catch me if you can: Evaluating android anti-malware against transformation attacks. Trans. Info. For. Sec. 9(1), 99–108 (Jan 2014), http://dx.doi.org/10.1109/TIFS.2013.2290431
19. Rijsbergen, C.J.V.: Information Retrieval. Butterworth-Heinemann, Newton, MA, USA, 2nd edn. (1979)
20. Schapire, R.E.: The strength of weak learnability. Mach. Learn. 5(2), 197–227 (Jul 1990), http://dx.doi.org/10.1023/A:1022648800760
21. Webb, G.I., Boughton, J.R., Wang, Z.: Not so naive bayes: Aggregating one-dependence estimators. Mach. Learn. 58(1), 5–24 (Jan 2005), http://dx.doi.org/10.1007/s10994-005-4258-6
22. Wilson, J.M.: Brand protection 2020. In: Tech. Rep. Michigan State University (2015)
23. Wolpert, D.H.: Stacked generalization. Neural Networks 5, 241–259 (1992)
24. Zhang, H., Jiang, L., Su, J.: Hidden naive bayes. In: Proceedings of the 20th National Conference on Artificial Intelligence - Volume 2. pp. 919–924. AAAI'05, AAAI Press (2005), http://dl.acm.org/citation.cfm?id=1619410.1619480