Detection of fake web-shops in the .be zone using active learning

Jules Dejaeghere

Research and Development project owner:
Maarten Bosteels – DNS Belgium

Master thesis submitted under the supervision of
Prof. Jean-Noël Colin

in order to be awarded the Degree of
Master in Cybersecurity
System Design

Academic year
2021 – 2022

I hereby confirm that this thesis was written independently by myself without the use of any sources beyond those cited, and all passages and ideas taken from other sources are cited accordingly.

30/05/2022

Title: Detection of fake web-shops in the .be zone using active learning

Author: Jules Dejaeghere
Master in Cybersecurity – System Design
Academic year: 2021 – 2022

Abstract

E-commerce activity in Belgium is growing, with consumers spending more money almost every year. With this popularity of online commerce, some malicious actors saw an opportunity to profit by selling counterfeit goods or not delivering goods at all. DNS Belgium actively takes countermeasures against such malicious actors operating in the .be zone as they threaten the trust that Internet users can have in the .be TLD. With more than 1.7 million domains in the zone, a manual inspection is impossible. With the data it has available, DNS Belgium collected a set of fake web-shops and legitimate domains over the last few years to train a classifier to help detect fake web-shops.

This thesis starts by presenting the existing solution used by DNS Belgium: the features, the classifier and its performance. Based on the literature, we suggest new features to improve the classifier. Next, we attempt to address one of the issues of the existing classifier: it requires a significant amount of labeled data and labeling data is a time-consuming process for the annotators.

We present an active learning architecture and consider multiple query strategy algorithms that the learner may use to identify the instances for which it should request the label. In order to evaluate this architecture, we conduct two main experiments: (i) we apply active learning on the labeled dataset, providing labels only when the learner requests them and (ii) we apply active learning on the .be DNS zone to evaluate how the approach performs and how many fake web-shops we are able to uncover.

Leveraging active learning in this setting limits the number of labeled instances required to start training a classifier. Moreover, it enables the classifier to ask for new labels, limiting the labeling cost to only instances that are actually relevant to the model. We apply active learning on the entire .be zone, leading to the identification of 152 domains that are marked as fake web-shops using significantly less labeled data than the existing classifier used by DNS Belgium.

Keywords: binary classification, active learning, fraudulent web-shop, e-commerce.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations

Over the last few years, e-commerce activity in Belgium has continued to grow. According to BeCommerce [7][1], Belgians spent 12.1 billion euros on e-commerce in 2021, which represents a progression of 18% in one year. BeCommerce also reports that Belgians made, on average in 2021, twenty online purchases for a total value of 1445€, which is 250€ more than the previous year.

In their 2020 version of the *Counterfeit and Piracy Watch List*, the European Commission [15] mentions that consumers turned to e-commerce platforms due to lockdowns. The European Commission [15] states that this is an incentive for e-commerce platforms to increase consumers' choice and their feeling of comfort. However, this also attracts malicious actors seeking to sell counterfeit goods to online shoppers and take advantage of the general trend to use e-commerce platforms.

The European Commission [15] identifies multiple threats posed by the sale of counterfeit goods via Internet: (i) consumers are at risk of buying non-standard and possibly dangerous products, (ii) the brand image of the companies is damaged because of the sale of counterfeit versions of their products, and (iii) the efforts of legitimate e-commerce websites to be regarded as safe are undermined.

As the registry for the .be zone, DNS Belgium is committed to better online security, with the objective to provide a carefree surfing experience on .be websites [13]. They developed a classifier to detect and take down fake web-shops in the .be zone for a few years. Their primary way of removing the fake web-shops from the zone is to look for suspicious web-shops using incorrect WHOIS [12] data when registering their domain. As the ICANN [22] defines, if inaccurate information is provided when registering the domain, it might be suspended or canceled.

For this thesis, we partnered with DNS Belgium to find new ways to improve their classifier. Their central position in the Belgian online ecosystem and their interest in making the .be zone trustworthy motivated this partnership.

1.1.1 Context

DNS Belgium started hunting fake web-shops in 2019. Their research began with a set of fake web-shops provided by the FPS Economy as they receive complaints about online stores that scam buyers in Belgium. Since then, DNS Belgium has explored new ways to improve its classifier and developed a web crawler to get insights about the websites present in the .be zone. Batsleer [4] analyzed, in 2021, how the existing crawler could be

[1] BeCommerce is the Belgian non-profit organization that brings together and represents companies active in the Belgian digital market. `https://www.becommerce.be/`

improved using machine learning techniques allowing to learn only from positive (i.e., fake web-shops) and unlabeled instances.

We think that other approaches to this classification problem are still to be explored. Given the number of websites in the .be zone (currently around 1.3 million) and the nature of the task, we are convinced that an approach taking advantage of human knowledge can be interesting to guide the classification process.

As reviewing each existing and newly registered domain is not tractable, a machine learning solution can be interesting to detect suspicious websites automatically. In order to create a machine learning system, DNS Belgium can extract features from the historical data of the domain in their zone. This includes data about the registrant (i.e., the person buying the domain name) and the registrar (i.e., the company registering the domain on behalf of the registrant). They also use features extracted with their crawler developed since 2019. Currently, the crawler extracts data from the HTML source code of the page hosted on the domain, the DNS records, the SMTP configuration and others.

1.2 Project statement and contributions

In this thesis, we explore how active learning, a machine learning setting that allows the learning algorithm to submit queries to obtain the label of unlabeled instances, can help detect fake web-shops in the .be zone. We review the literature about the previous efforts on fake web-shop detection. We also propose new features, based on the recent literature, to improve the existing classifier. Next, we design an active learning architecture to help with fake web-shop detection.

To this end, we present two experiments to assess how helpful active learning is with the current classification problem. First, we evaluate how active learning performs using the existing labeled data made available by DNS Belgium. In this experiment, we put the active learner in a situation where it does not have access to the labels of most instances until it issues a query to get them. This experiment is the opportunity to compare the performance of multiple query selection algorithms. Three different learners are trained, each with a different query selection algorithm, to allow us to compare their performances. Second, we apply the active learning architecture to the entire .be zone to attempt to uncover previously undetected fake web-shops. For this experiment, we build a dataset using multiple sources for labeled instances, including search engine results.

Both of those experiments are implemented, taking inspiration from the existing solution used by DNS Belgium and using Scikit-Learn [36]. The results of the experiments are presented and interpreted. Next, we put our results into perspective and highlight some considerations to take into account when interpreting them. Finally, we suggest directions for future research related to the detection of fake web-shops using machine learning.

1.3 Organization of this document

The document is structured as follows:

Chapter 2 discusses the previous research in the context of fake web-shop detection and

introduces the key concepts of active learning;

Chapter 3 presents the existing features used by DNS Belgium as well as their existing classifier and its performance;

Chapter 4 describes how we propose to improve the existing classifier. It presents the additional features that we suggest, the architecture for the active learner and a description of the two experiments we conduct;

Chapter 5 presents the results of the two experiments conducted;

Chapter 6 discusses the results obtained and the limitations of validity of the experiments and their results;

Chapter 7 suggests directions for future research in the context of fake web-shop detection;

Chapter 8 summarizes the most important results and concludes this thesis.

Chapter 2

Related work

This chapter starts by presenting the previous research focused on fake web-shop detection. We describe the different methods and features used for the classification. Then, the key ideas behind active learning and the aspects relevant to our specific case are presented.

2.1 Fake web-shop detection

This section gives a general overview of the research related to fake web-shop detection. The second subsection focuses on the different features identified in the research papers.

2.1.1 Overview

Fake web-shop detection has been studied only for a few years. One of the first major contributions in the field is by Wang et al. (2014) [46]. Their work focuses on identifying black-hat search engine optimization campaigns over eight months. These campaigns aim at achieving a high ranking of web pages redirecting to web-shops selling counterfeit luxury goods. The authors highlight how cloaking, a "search engine optimization technique in which the content presented to the search engine spider is different from that presented to the user's browser" [47], can be used by malicious actors to achieve a high ranking in search engine results. In total, they identified 52 search engine optimization campaigns. The different malicious search results were attributed to one of the 52 campaigns using a classifier that relies on HTML-based features. The authors conclude that search engine and domain seizure intervention have the desired effect but neither of those methods is currently employed in a way that is effective against malicious actors. They also highlight the actors' agility to adapt in response to the interventions and fill the gap created by disappearing campaigns.

Concurrently to Wang et al., Wadleigh et al. (2015) [45] inspected the search results for 225 queries across 25 brands. For each brand, the authors used "innocent" (such as "buy online"), "gray" (such as "cheap") and "complicit" (such as "replica") terms in the queries. They independently trained three models: a Logistic Regression model, a Support Vector Machine and an Adaptive Boosting. The first two models performed better than the last one. To train their classifiers, Wadleigh et al. used three main categories of features: URL-level features (mainly the length of the URL), page-level features (such as the number of currencies and the number of brands on the page) and website-level features (mainly WHOIS [12] data and a reputation score). The authors reached some conclusions about the presence of websites selling counterfeit goods in the search results given the query that was made. First, they noticed that "innocent" queries are less likely to yield results pointing to fakes than "gray" or "complicit" queries. Second, brands that actively take countermeasures against counterfeit versions of their products experience a lower rate of fakes in the search results. Finally, they highlight that malicious actors take advantage

of reliable hosting by operating in some specific countries with widespread web hosting infrastructure (namely the United States, the Netherlands and Germany).

Following the ideas introduced by Wadleigh et al., Carpineto and Romano (2017) [10] also used search engine queries to identify fake e-commerce websites. In order to gather the data for their experiment, they used three different search engines and focused on shoes-related searches. As in Wadleigh et al. [45], they used "gray" and "complicit" keywords in addition to the name of the brand. Their results indicate that the number of fake web-shops found in the search results is influenced by the search engine and the search query. Carpineto and Romano used a two-step approach to tackle the classification of the websites: first, they built a classifier to discriminate e-commerce websites from other websites; next, they made another classifier to distinguish between legitimate and fake e-commerce websites.

Cox and Haanen (2018) [11] trained an Adaptive Boosting Classifier relying on 32 content-based features designed to identify fraudulent web-shops. The authors found that it is necessary for operators of fraudulent web-shops to attract customers, have a high ranking in search engines, and scale efficiently. Based on those preconditions, they developed the features used by the Adaptive Boosting Classifier.

Mostard et al. (2019) [33] explored how to include visual features of the web pages into a classifier. Their main hypothesis is that malicious web-shops will mimic legitimate ones by adding known logos (mainly social media and payments methods). However, fake web-shops are not likely to be present on social media, introducing a discrepancy between the visual information (the logos shown) and the contextual information (the website not being represented on the social media advertised using the logos). Mostard et al. used a Convolutional Neural Network to identify the presence of logos on the web pages. They then used the discrepancy between the detected logos and the contextual information, extracted from the HTML, as additional features. Those extra features enabled them to increase the F1-score of their best-performing algorithm, a Random Forest Classifier, from 93% to 98%.

Wabeke et al. (2020) [44] conducted research for SIDN[1], focusing on fraudulent web-shops selling luxury goods. The first classifier developed by the authors was based on observations gathered while detecting phishing in the .nl zone. They came across suspicious web-shops selling luxury goods displaying a long HTML title listing many brands during their phishing detection campaign. Such behavior will likely improve the search engine ranking, as explained in Wang et al. [46]. This provided the authors with a simple but efficient way to detect suspicious websites. While analyzing the results, Wabeke et al. observed the following characteristics.

Domains are cheap. Operators of malicious websites may prefer to register many domains as they are relatively inexpensive. If some are taken down, the remaining ones are enough to remain profitable.

Registrar concentration. More than 87% of the suspicious domains identified by Wabeke et al. [44] are registered by ten different registrars. The authors highlight that the most used registrar is among the cheapest registrars and offers an API for bulk registration, which probably helps fake web-shop operators to automate the process.

[1]SIDN is the registry for the .nl zone. `https://www.sidn.nl/`

Few content-management systems. The home pages of the web-shops are similar yet different. The websites seem to use few content-management systems and share some common design features.

Most domains are drop-catch. Most domains used for fake web-shops were domains that became available and were immediately registered: this practice is called "drop-catching" [21]. This allows the domain's new owner to take advantage of the previously built trust in the domain name.

Using the title as the only feature to detect fraudulent web-shops is not very robust. After a round of take-downs, Wabeke et al. designed a Support Vector Machine Classifier based on nine features. The features selected for the classifier were based on the observations from the previous results. With this new classifier, Wabeke et al. could detect fake web-shops that would not have been detected using their first heuristic. The authors noticed that the characteristics mentioned earlier were already more diverse than previously.

Beltzung et al. (2020) [9] highlighted that fake web-shops are often short-lived, making it harder for retroactive tools (like blacklists) to effectively protect users. They developed a detection system relying only on the source code of the web page (HTML, CSS and JavaScript) able to produce real-time predictions for the users through a browser plug-in. With the developed models, the system was able to warn the users for 48% of the fraudulent web-shops with the highest confidence level.

While most of the research regarding fake web-shop detection is focused on feature engineering and classification, Odekerken and Bex (2020) [35] proposed a high-level architecture of a web-shop checker that systematically involves a human. The so-called "human-in-the-loop" design brings some constraints. The primary constraint identified by Odekerken and Bex [35] is that the system must be able to explain the decisions to an analyst. The proposed architecture relies on explicit rules and similar cases. The advice produced by the checker is then presented to an analyst along with a set of explanations, similar cases, and information that could change the system's decision.

Batsleer (2021) [4] partnered with DNS Belgium to find new fake web-shops in the .be zone. First, the author confirmed an observation previously made by Wabeke et al.: the malicious domains are often re-registered using the *drop-catching* mechanism. Batsleer also found it very unlikely for fake web-shops to display telephone or email address links on their home page. In order to detect new fake web-shops, Batsleer trained classifiers to learn from positive and unlabeled data [8]. The best classifier was able to detect fake web-shops not detected by the current model of DNS Belgium and achieved a similar precision, requiring only positive examples.

Gopal et al. (2022) [19] presented a different take on fraudulent website detection. The authors start from the observation that conventional methods for fraudulent website detection are based on the content of the website. Such systems are generally designed to detect a specific subset of fraudulent websites (e.g., phishing, fake e-commerce, fake news or piracy). Even if classifiers based on the website's content are generally accurate, they cannot generalize to other kinds of fraudulent online activities. Using the requests made to third-parties, Gopal et al. [19] propose an "approach that is able to increase the likelihood of detecting all kinds of fake and fraudulent websites". Using different data sources to

determine if a third-party is safe, is known and what business model it is relying on, the authors show that third-party requests can bring valuable information.

van den Hout et al. (2022) [43] introduced a system able to detect specific logos on web pages. They presented two case studies of their system: the government impersonation, in which the authors try to detect misuses of the Netherlands government's logo, and the trust-mark abuse, in which the authors try to detect misuses of the *Thuiswinkel Waarborg*'s[2] logo. In both case studies, the authors were able to find misuses of the logos: phishing websites mimicking the National Tax Authority or the national online authentication system in the first case, web-shops using the *Thuiswinkel Waarborg*'s logo without obtaining a certificate in the second case. The authors also "discovered" legitimate domains in both cases: legitimate government domains that were missing from the government's website portfolio and domains belonging to members of *Thuiswinkel Waarborg* but that were unknown to *Thuiswinkel Waarborg*. Some potential threats were found during the government impersonation case study: domains typo-squatting legitimate government domains and redirecting to the legitimate government website. van den Hout et al. expect this to be part of a strategy to build up a reputation and achieve high search engine ranking, and, once popular, use the domain to host a phishing website.

2.1.2 Analysis of the features used

Inspired by Batsleer [4], we group the features presented in the previously discussed articles into the following categories: registration features, URL features, product features, merchant features, payment features, page-level features, website-level features and visual features. For each category, we discuss why they are relevant and which articles mentioned features in that category. In addition to the features presented in the previously mentioned articles, we also include features from Bannur et al. [3] and Kazemian and Ahmed [24]. Their research is not focused on fake web-shop detection but an encompassing class: the malicious web pages. We only include features that are relevant to our application.

Similar features are grouped and features in bold were identified to be statistically significant by at least one research paper. Some researchers did not provide a feature importance analysis (Bannur et al. [3], Carpineto and Romano [10], Kazemian and Ahmed [24]). Wabeke et al. [44] and Cox and Haanen [11] only reported the relative importance of the different features used in their classifiers, so we represent the five most important features in bold for those articles.

Registration features

The registration features encompass all the information related to the registrars and registrants. While some information is publicly available via WHOIS [12], DNS registries like DNS Belgium or SIDN have access to more information. Table 2.1 gives an overview of the different features encountered in the research articles. The reported domain score used by Wabeke et al. [44] is defined by the authors as "the ratio of malicious domains reported via the Netcraft abuse list divided by all the domains registered by a given registrar". As mentioned earlier, *drop-catching* is a common practice among spammers [21]. The re-registered

[2] *Thuiswinkel Waarborg* is a quality mark for shops selling via the Internet in the Netherlands. `https://www.thuiswinkel.org/`

flag and the re-registration delay try to capture this behavior. Batsleer [4] highlighted that more than 60% of re-registered domains used for fake web-shops are registered within a day. Features such as the registration hour and the email provider of the registrant may indicate where the domain was registered from. Finally, as Beltzung et al. [9] identified, fake web-shops are often short-lived scams. Therefore, the domain age may give valuable insight into the trustworthiness of the domain.

Feature	Type	Reference
Private or China-registered WHOIS [12]	Boolean	[45]
Domain age	Boolean	[9, 10, 45]
Re-registered domain	Boolean	[4, 44]
Re-registration delay (less than x days)	Boolean	[4]
Domain was transferred to another registrar	Boolean	[4]
Registration hour	Numerical	[4, 44]
Registrar	Categorical	[44]
Email provider of registrant	Categorical	[44]
Reported domain score	Numerical	[44]
Ratio of lowercase characters in registrant's name	Numerical	[44]
Registrant country	Categorical	[10]

Table 2.1: Overview of the registration features and the papers that use those features.

URL features

Table 2.2 lists the different features related to the URL of the web page. Keywords in the URL may help search engine optimization [18]. According to Kazemian and Ahmed [24], the presence of spelling mistakes or suspicious characters in the URL is often an indicator of a malicious web page.

Feature	Type	Reference
Keywords in domain name	Boolean	[10, 45]
Length of domain name	Numerical	[45]
Explicit IP or port in the URL	Boolean	[3]
Presence of spelling mistakes	Boolean	[24]
Suspicious characters in the URL	Boolean	[24]

Table 2.2: Overview of the URL features and the papers that use those features.

Product features

Product features give an indication about the number of products sold on a web-shop, their prices and if discounts are applied. While some of these features are harder to extract automatically, some of the features presented in table 2.3 provide a good approximation. Most reputable web-shops provide a different domain for each geographical zone they operate in, so it is unlikely for those shops to display multiple currencies on the same page. On the other hand, malicious shops may want to attract as many buyers as possible and show multiple currencies on the same website. According to Carpineto and Romano [10],

legitimate web-shops do not display many products on their homepage. In contrast fake web-shops often advertise numerous products on the front page of their website. Most of the time, fraudulent web-shops display large discounts on the products on sale. The number of brands mentioned on fake e-commerce websites is often high to achieve a good ranking in search engines. Finally, the number of duplicate prices on a page may result from a lazy or automated fraudulent web-shop operator copying products on the page and not changing their properties.

Feature	Type	Reference
Number of currencies	Numerical	[4, 10, 11, 45]
Percent savings average	Numerical	[10, 45]
Number of duplicated prices	Numerical	[45]
Unique brand mention count	Numerical	[45]
Number of products	Numerical	[33]
Percentage of discounted products	Numerical	[10]
Products present on the home page	Boolean	[10]
Number of numerical strings (similar to prices)	Numerical	[4]

Table 2.3: Overview of the product features and the papers that use those features.

Merchant features

Legitimate web-shops often give context information about the identity of the merchant. Table 2.4 provides an overview of the features used by different researchers to capture such pieces of information. Most of the research papers identify the email address to be an interesting feature and some argue that an email address from a free web-mail should not be considered trustworthy. Multiple authors consider the presence of links to social media as a reliable feature. Cox and Haanen [11] even refined this further: they make a difference between a "shallow" link (i.e., pointing to the homepage of the social media) and a "deep" link (i.e., pointing to a particular profile on the social media). According to the authors, the second type of link is most likely to appear on legitimate web-shops as fraudulent web-shops are not likely to have accounts on social media. Finally, the researchers often propose information related to a real-world shop or activity: physical address, link to a page with physical store information, jobs offerings, VAT number... This information is often considered to discriminate between legitimate and fraudulent web-shops.

Payment features

According to Carpineto and Romano [10], fraudulent web-shops tend to use payment methods such as Western Union as transfers cannot be canceled or reversed. Mostard et al. [33] counted the number of payments methods mentioned in the HTML page as they expect fraudulent web-shop to accept less payments methods than legitimate ones.

Page-level features

Table 2.5 lists the features identified by the researchers related to the HTML structure of the web page. According to Wadleigh et al. [45], large iframes are often used to hide

Feature	Type	Reference
Presence of a (free) **email address**	Boolean	[4, 10, 33, 45]
Number of links to social media	Numerical	[4, 10, 11, 33]
Presence of deep links to social media	Boolean	[4, 11]
Business registration number / VAT found	Boolean	[10, 11, 33]
Presence of a phone number	Boolean	[4, 10, 11, 33]
Presence of an address	Boolean	[10, 11, 33]
Presence of bank account number	Boolean	[11]
Link to physical stores	Boolean	[10]
Jobs offerings	Boolean	[10]
Presence of a link to a mobile app	Boolean	[10]
Trust-mark logo misuses	Boolean	[43]

Table 2.4: Overview of the merchant features and the papers that use those features.

malicious content and scripts from the users. The number of internal links gives an appropriate estimate of the website's size, while the number of external links can give insight about related websites. In the case of fraudulent web-shops, it is expected to have fewer external links than legitimate web-shops. Moreover, external links can be checked against lists of known malicious websites, as suggested by Kazemian and Ahmed [24]. Interestingly, Mostard et al. [33] found that the presence of a shopping cart system is statistically significant and tends to indicate that the web-shop is fraudulent. The authors hypothesize that legitimate web-shops would develop their own shopping cart system while fraudulent ones would use well-known existing systems. Cox and Haanen [11] showed that the distance between the HTML title of the page and the domain name is often larger for malicious web-shops. As malicious web-shops often register domains using the *drop-catching* mechanism, operators do not choose the domain to match what they host on the page. The meta description and the meta keywords of the HTML page are often used by search engines [18] to determine the subject of the page to create their index. Therefore, fake web-shops that try to achieve a high ranking in search engine results often use those fields [11].

The content of the HTML page can be processed to be used as features. One option is to use the bag-of-word representation, where the number of occurrences of every word is reported. Another option is to use the *term frequency - inverse document frequency* weighting, which "increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general" [48].

Website-level features

The features related to the website (in opposition to a single web page) are listed in table 2.6. Some features are directly linked to the website's popularity, such as the presence in Alexa top sites [1] list. Other features, like the number of open ports and the presence of a TLS certificate, try to reflect the security of the website, taking the hypothesis that legitimate websites will have a bigger incentive to secure their web-shops. Wabeke et al. [44] found that many fake web-shops were using TLS certificates issued by a small number of certification authorities, making this a simple but effective feature. Multiple researchers (Carpineto and Romano [10], Cox and Haanen [11], Gopal et al. [19], Mostard et al. [33])

Feature	Type	Reference
Large iframes	Boolean	[45]
HTTP headers	Categorical	[3]
Count of the HTML tags	Numerical	[3, 4, 11]
Number of internal / **external** links	Numerical	[3, 4, 11, 33]
Number of unique hostnames in the page links	Numerical	[3]
HTML size / Total word count	Numerical	[3, 33]
Presence of a shopping cart system	Boolean	[33]
Presence of copyright	Boolean	[33]
Links to known malicious web pages	Boolean	[24]
Number of images	Numerical	[4, 11]
Presence of meta Open Graph tags	Boolean	[11]
Distance between HTML title and the URL	Numerical	[4, 11]
Lexical diversity	Numerical	[11]
Number of words in the meta description	Numerical	[4, 11]
Number of meta keywords	Numerical	[4, 11]
Bag-of-words of HTML body	Vector	[3]
TF-IDF of HTML body	Vector	[3, 24]

Table 2.5: Overview of the page-level features and the papers that use those features.

identified the use of analytic trackers or cookies to be a relevant feature. They observed that most fraudulent web-shops do not use analytic trackers or tracking cookies. Carpineto and Romano [10] developed the detection of this feature even further and checked if the website displays a notice and consent banner to be compliant with cookie laws.

Features	Type	Reference
Website in Alexa top sites	Boolean	[10, 45]
Existence of MX record	Boolean	[4, 44]
Presence / Issuer of TLS certificate	Boolean / Categorical	[3, 4, 44]
Autonomous system / Location of the website	Categorical	[4, 10, 44]
Number of pages found	Numerical	[33]
Number of open ports found	Numerical	[33]
Presence of analytic trackers or tracking cookies	Boolean	[10, 11, 19, 33]
Notice and consent banner for cookie law compliance	Boolean	[10]
Website uses cloaking techniques	Boolean	[10]

Table 2.6: Overview of the website-level features and the papers that use those features.

Visual features

As already mentioned, Mostard et al. [33] use the discrepancy between visual and contextual information about social media and payment method as a feature.

Bannur et al. [3] used the *scale-invariant feature transform* (SIFT) [31] on screenshots of web pages. The resulting descriptors can be used to compare different web pages together. Those descriptors can also be compared with a database of known logos to detect their

presence on the web page.

Finally, Kazemian and Ahmed [24] used the *speeded-up robust features* (SURF) [6], a feature descriptor like SIFT [31] but faster. They grouped pages into clusters of similar-looking pages based on the screenshots. The clusters were then fed into the machine learning model to improve the classification.

Summary

In the previous sub-sections, we presented many features authors use in the literature. Even if those features are based on different information sources (e.g., content of the web pages, registration or WHOIS [12] data, structure and configuration of the website in general, reputation of the domain, interaction with third-parties), they all try to capture the trustworthiness of the domain. Strange registration patterns, lazy website management and lack of information about the seller may lead an informed online shopper to stay away from the website. On the other hand, websites that care about their users' privacy by asking for consent to track the activity, presenting information about their business in general, and improving user experience will more easily inspire trust.

The features capture hints that an informed Internet user should look for to assess the trustworthiness of a web-shop (or even a website in general). While the "intuition" that the human user can have when visiting a website is hardly translatable into features, features may capture more diverse information. Some information captured by the features is not immediately seen by humans when browsing a website (such as the registration history or the third-parties used by the website) but has been proved helpful in discriminating against malicious websites.

2.2 Active learning

In order to train a binary classifier, a set of labeled instances is needed to train and test the model. However, in many cases, obtaining labeled data is expensive or time-consuming, but unlabeled data is abundant or can be obtained for cheap. In our current setting, we are facing a similar situation: DNS Belgium has a large amount of data about the websites in the .be zone but labeling instances for classification requires a human annotator and is time-consuming. Active learning relies on the idea that if a machine learning algorithm can choose data it learns from, greater accuracy is achievable with fewer labeled instances needed. Leveraging active learning in our context may be interesting as the machine learning algorithm will be presented with many unlabeled instances to learn from. This section introduces the concepts behind active learning and is based on the survey conducted by Settles [37].

2.2.1 Overview

For any supervised system to perform well, it often needs hundreds (or thousands) of training examples. The key idea behind active learning is that, if the learning algorithm is allowed to choose instances to learn from, it can achieve better performance with less training data. Active learning systems are allowed to submit queries to an oracle. Those

queries are unlabeled instances to be labeled, often by a human annotator. Active learning is well-motivated in domains where unlabeled data is abundant, but the labels are expensive to obtain.

2.2.2 Active learning settings

Settles [37] presents the three main scenarios in which an active learning algorithm may submit queries to an oracle. The three scenarios are (i) membership query synthesis, (ii) stream-based selective sampling and (iii) pool-based active learning.

In a membership query synthesis, the active learner can request labels for any unlabeled instance of the feature space. This enables the learner to generate new instances to submit for the oracle to label. While this approach leaves a lot of freedom in the way queries are generated by the learner, Baum and Lang [5] encountered a problem when using this setting with a human annotator: most of the instances generated could not be classified in one of the classes by the human. Their research was focused on handwritten digits and most of the examples generated by the learner had no natural semantic meaning. King et al. [25] used this setting successfully with a scientific robot able to autonomously perform a series of biological experiments.

In a stream-based selective sampling setting, the assumption is that obtaining unlabeled instances is cheap. Instances are sampled from the natural distribution, one at a time, and the learner can choose to query the oracle to get the label or to discard the instance. If the distribution is uniform, this setting may behave like membership query learning. However, if the distribution is non-uniform, the queries will be more representative of the true underlying distribution than in the query synthesis setting, as they are sampled from the real underlying distribution.

Finally, in a pool-based active learning setting, the assumption is that there is a small set of labeled data and a large pool of unlabeled instances. Queries are drawn from the pool of unlabeled data by the learner and presented to the oracle to get the label. In this setting, the learner is able to scan the entire pool of unlabeled data to make the most interesting decision about the instance to query.

2.2.3 Uncertainty sampling

For all the active learning settings mentioned before, it is needed to evaluate the informativeness of unlabeled instances. Methods to evaluate the informativeness of unlabeled instances are referred to as *query strategies*. Uncertainty sampling is a simple yet commonly used query strategy.

In this query strategy, the learner queries the instance about which it is least certain about the label to predict. For binary probabilistic classifiers, an efficient way to implement uncertainty sampling is to query the instance whose posterior probability of being positive is the closest to 0.5 (Lewis and Gale [27]).

A more general uncertainty sampling strategy uses entropy (Shannon [39]) as an uncertainty measure. When considering only binary classification, using the entropy as an uncertainty measure is identical to choosing the instance with the posterior nearest to 0.5. However, this entropy-based method is easily applicable to probabilistic multi-label

classifiers.

Multiple researchers applied uncertainty sampling to non-probabilistic models. Lewis and Catlett [26] modified the output of a Decision Tree to have a probabilistic output, allowing them to apply uncertainty sampling. Lindenbaum et al. [30] used a similar approach with a K-Nearest Neighbor classifier where each neighbor can vote for the label of a given instance. The proportion of the neighbors' votes represents the posterior label probability. Tong and Koller [42] applied uncertainty sampling to Support Vector Machine Classifiers by querying the closest instance to the decision boundary.

Multiple other query strategies have been proposed in the literature. Such strategies include query-by-committee, expected model change, variance reduction, estimated error reduction and density-weighted methods. Those methods are not covered here and we refer to Settles [37] for more information.

2.2.4 Batch-mode active learning

In most active learning research, the instances submitted to the oracle are selected one at a time. It is expensive to generate only one query for the oracle in some scenarios instead of a batch. Inducing a model on large datasets and selecting the most informative unlabeled instance may take some time and result in ineffective use of the oracle resources (i.e., the human annotator is waiting for the learner to submit a new query). An interesting extension of the active learning setting would be to allow the learner to select a subset of the pool to submit to the oracle. Such a design would allow the human annotator to be presented with multiple instances to label or would enable multiple annotators to work concurrently on the labeling task.

In order to select the most informative instance, the learner is able to inspect the entire pool of unlabeled instances. A naive approach to construct the set of queries is to select the "N-best" instances to be labeled. However, this fails to capture the potential similarity of the selected instances: multiple instances might be highly ranked by the query strategy because they are very similar and will bring similar information to the model once labeled.

Shen and Zhai [40] compared three batch query algorithms: (i) the *Top K* algorithm, which selects the K first instances, (ii) the *Gapped Top K* algorithm, which selects K instances starting from the top but leaving out G instances after selecting one (i.e., with K set to 4 and G to 3, the algorithm selects the 1st, the 5th, the 9th and the 13th instances) and (iii) the *K Cluster Centroid* algorithm, which clusters the top N instances into K clusters and selects the medoids. They found the *K Cluster Centroid* algorithm to perform the best out of the three algorithms. The authors also observed that the *K Cluster Centroid* algorithm "obtains the fewest number of relevant documents from user feedback, yet its performance is the best". The *K Cluster Centroid* algorithm uses the K-Medoid clustering [23] to divide the top N instances into clusters. Shen and Zhai [40] used the J-Divergence [29] as the distance function between two instances.

Settles [38] mentions that, on some datasets, a random batch construction can still outperform active methods, as illustrated by Guo and Schuurmans [20]. Therefore, it is necessary to evaluate if a specific batch construction algorithm is preferable over a random batch construction for a given dataset.

2.3 Scope of this work

Most of the research related to fake web-shop detection presented in the previous sections focuses on feature engineering and behavior analysis. Only a few authors present a functional architecture for a classifier that can be used to detect new malicious websites and improved over time. In this thesis, we try to leverage concepts from the active learning setting to design a classifier that can be used to detect fraudulent web-shops and improved iteratively over time. In our setting, the classifier improvements are unlabeled instances identified as informative by the classifier and presented to a human annotator to label. Such instances are then added to the labeled data used to train the classifier. This should help detect known types of malicious websites and find new types in the instances submitted by the classifier to the annotator.

Chapter 3

Case study

This chapter presents the current approach taken by DNS Belgium to detect fake web-shops in the .be zone. We start by presenting how the organization gathered a labeled dataset of fake web-shops and what features are extracted. Next, we analyze the distributions of the features for three classes: *legitimate websites*, *legitimate web-shops* and *fake web-shops*. Finally, we present the current architecture of the classifier used by DNS Belgium to detect new fake web-shops using their labeled data. This chapter also introduces the challenges encountered by DNS Belgium when classifying websites in a real-world setting.

This existing solution constitutes a stepping stone towards the solution we design to detect fake web-shops. For this reason, we devote a chapter to analyzing the existing solution and give the reader enough insight to understand our motivations for the rest of this thesis.

3.1 Labeled dataset

3.1.1 Dataset construction

DNS Belgium started labeling web-shops in 2019. This dataset has been used to detect fake web-shops in the .be zone. It mainly consists of fake web-shops, legitimate web-shops and some non web-shop websites. The fraudulent web-shops of this first dataset originated from the FPS economy, which receives complaints about fraudulent online stores. The legitimate web-shops present in the dataset were provided by the employees of DNS Belgium and cover some of the most popular web-shops in Belgium. Fraudulent web-shops detected by the classifier were subsequently added to the dataset. For each website in the dataset, DNS Belgium crawled the HTML web page hosted on the domain. Table 3.1 summarizes the number of labeled visits in the dataset per category.

Label	**Count**
Fake web-shop	1843
Legitimate web-shop	1085
Not a web-shop	77

Table 3.1: Number of labeled instances in the DNS Belgium's fake web-shop dataset as of April 2022.

As the registry for the .be zone, DNS Belgium has access to the registration data of each domain in the zone. Some data from the registration information is used in order to classify the domains to detect fake web-shops. The main features used from registration data are flags to specify if the domain has been transferred, re-registered, and multiple booleans to indicate how fast the domain has been re-registered.

For each domain in the labeled dataset, the HTML of the domain's front page was

Month	Fake web-shops crawled	Legitimate web-sites crawled	Legitimate web-shops crawled
January 2019	1565	700	664
February 2019	1	11	11
August 2019	151	103	60
October 2019	47	298	298
December 2019	45	0	0
January 2021	0	2449	0
October 2021	34	2425	52
November 2021	0	2	0
Total	1843	5988	1085

Table 3.2: Crawl date for the labeled websites in the baseline dataset.

crawled and features were extracted. Those features were then merged with features derived from the registration data to create the training dataset.

For their annual report [14], DNS Belgium labels a sample of domains into multiple categories. They also estimate the number of *low-content* websites, displaying a page with very little information (e.g., blank page, default software installation, domain for sale). The proportion of low-content websites in the .be zone is estimated at around 45% by DNS Belgium in their annual reports for 2020 and 2021. To get a more representative sample from the .be zone, we include the websites labeled for the annual reports for 2020 and 2021 (including the low-content websites). Combining the fake web-shop dataset and the dataset of the annual report, we obtain a dataset of 7831 domains. Out of the 7831 domains, 5988 are labeled as legitimate websites (including 1085 legitimate web-shops), and 1843 are labeled as fake web-shops. This dataset, referred to as our baseline dataset for this research, will be used to evaluate the distribution of the different features in the next section. This dataset will also be used to train and compare the performance of the classifiers presented in the following chapters. Table 3.2 gives an overview of the crawl date for the data in the baseline dataset. Most of the fake web-shops were crawled in 2019 and many legitimate websites were crawled in 2021, for the annual reports [14].

Please note that when we refer to *legitimate websites*, we also include the *legitimate web-shops* in the set. In other words, the set of *legitimate web-shops* is a subset of the *legitimate websites*. In the following sections, the distributions of multiple features are analyzed. As we try to set apart *fake web-shops* from *legitimate websites*, we show the distributions for those classes. As both *legitimate* and *fake web-shops* may share some features, we also include the distribution for the *legitimate web-shops*. This allows us to estimate whether the feature helps to discriminate between *fake web-shops* and *legitimate websites* or only between web-shops (either fake or legitimate) and *legitimate websites*.

The websites visited in 2019 and 2021 were not crawled by the same crawler. In late 2019, DNS Belgium started to build its own crawler which is currently used to crawl the entire .be zone regularly. The first full crawl done with their own crawler was launched in January 2021. Websites visited before this date were crawled with another crawler. Both crawlers retrieve the HTML front page of the visited domains and other information about the domain in general. However, the current crawler of DNS Belgium keeps getting updates and is able to extract features that were not extracted by the previous crawler.

As an example, the current crawler takes a screenshot of the website as it is presented to the user. This feature helps when classifying websites as the HTML might not be rendered properly if linked content (e.g., images, style sheets, scripts) is not available anymore. Comparing the two crawlers is not the main focus of this research, but the relevant differences will be highlighted when they may affect the results presented.

3.1.2 Overview of the features

The following paragraphs give an overview of the features collected by the crawler and used for website classification. The features are grouped into categories according to what they try to capture. The following section presents the analysis of the most important features in more detail.

Page-level features

Page-level features give information about the general page layout, content, and size. Those features mainly reflect what elements are present on the page and how the content presented on the page is related to the domain name.

`nb_imgs` **(numerical)** Number of images.

`nb_links_int` **(numerical)** Number of internal links.

`nb_links_ext` **(numerical)** Number of external links.

`nb_input_txt` **(numerical)** Number of text inputs.

`nb_button` **(numerical)** Number of buttons.

`nb_meta_desc` **(numerical)** Word count of the meta description field.

`nb_meta_keyw` **(numerical)** Word count of the meta keywords field.

`nb_tags` **(numerical)** Total number of HTML tags.

`nb_words` **(numerical)** Word count of the body of the web page.

`nb_letters` **(numerical)** Letter count of the body of the web page.

`title` **(string)** HTML title of the page.

`body_text` **(string)** Text present in the body of the page.

`meta_text` **(string)** Text present in the meta description.

`body_text_truncated` **(boolean)** Indicate if the stored `body_text` is shorter than the actual text.

`meta_text_truncated` **(boolean)** Indicate if the stored `meta_text` is shorter than the actual meta text.

`title_truncated` **(boolean)** Indicate if the stored `title` is shorter than the actual title.

`nb_distinct_hosts_in_urls` **(numerical)** Number of unique hostnames found in the links on the page.

`external_hosts` **(list of strings)** Hostnames found in the external links.

`distance_title_initial_dn` **(numerical)** Edit distance between the HTML title and the domain name (before following redirections).

`distance_title_final_dn` **(numerical)** Edit distance between the HTML title and the domain name (after following redirections).

`longest_subsequence_title_initial_dn` **(numerical)** Longest common subsequence between the HTML title and the domain name (before following redirections).

`longest_subsequence_title_final_dn` **(numerical)** Longest common subsequence between the HTML title and the domain name (after following redirections).

`body_text_language` **(categorical)** Language detected by Lingua (Stahl [41]) among a set of 27 common languages.

`body_text_language_2` **(categorical)** Language detected by Lingua (Stahl [41]) among all the spoken languages.

`fraction_words_title_initial_dn` **(numerical)** Number of words in the HTML title that also appear in the domain name (before following redirections) divided by the total number of words in the HTML title.

`fraction_words_title_final_dn` **(numerical)** Number of words in the HTML title that also appear in the domain name (after following redirections) divided by the total number of words in the HTML title.

`html_length` **(numerical)** Number of characters in the HTML source code.

Merchant related features

Merchant related features reflect how the merchant gives information about its identity and its presence on social media.

`nb_links_tel` **(numerical)** Number of telephone links.

`nb_links_email` **(numerical)** Number of email links.

`nb_facebook_deep_links` **(numerical)** Number of Facebook deep links.

`nb_facebook_shallow_links` **(numerical)** Number of Facebook shallow links.

`nb_linkedin_deep_links` **(numerical)** Number of LinkedIn deep links.

`nb_linkedin_shallow_links` **(numerical)** Number of LinkedIn shallow links.

`nb_twitter_deep_links` **(numerical)** Number of Twitter deep links.

`nb_twitter_shallow_links` **(numerical)** Number of Twitter shallow links.

`nb_youtube_deep_links` **(numerical)** Number of YouTube deep links.

`nb_youtube_shallow_links` **(numerical)** Number of YouTube shallow links.

`nb_vimeo_deep_links` **(numerical)** Number of Vimeo deep links.

`nb_vimeo_shallow_links` **(numerical)** Number of Vimeo shallow links.

Product related features

Product related features try to capture the number of products on sale displayed on the page. Those features also capture the number and the diversity of currencies shown.

`nb_numerical_strings` **(numerical)** Number of strings that look like prices.

`nb_currency_names` **(numerical)** Total number of currencies (ISO code or symbol).

`nb_distinct_currencies` **(numerical)** Number of different currencies (ISO code or symbol).

Website-level features

Website-level features capture general information about the website itself.

`country` **(categorical)** Country code where the website is hosted.

`asn` **(categorical)** Autonomous system number where the website is hosted.

`has_mx` **(boolean)** Indicate if the domain has a mail exchange record set up.

Registration features

As the registry for the .be zone, DNS Belgium can leverage the registration data as a feature. Multiple features can be extracted from the registration data. The following features are common features extracted by DNS Belgium to train their classifier.

`is_reregistered` **(boolean)** Is true if the domain has been re-registered.

`is_transferred` **(boolean)** Is true if the domain has been transferred from one registrar to another.

`agent` **(categorical)** Identifier for the registrar used to buy the domain.

`registration_duration_days` **(numerical)** Number of days elapsed since the beginning of the current registration.

`rant_country` **(categorical)** Country where the registrant lives.

`rereg_1d`, `rereg_10d`, `rereg_30d`, `rereg_90d`, `rereg_365d`, `rereg_older` **(booleans)** Indicate if the domain has been re-registered within the given number of days (1, 10, 30, 90 or 365 days) after it was available. If the domain is re-registered more than a year after it became available, `rereg_older` is set to true. At most one of those features is set to true for a specific instance.

3.2 Feature distributions

This section explores the distribution of the computed features over the labeled dataset. The distributions presented are calculated over the baseline dataset, composed of 7831 instances: 5988 legitimate websites (including 1085 legitimate web-shops) and 1843 fake web-shops.

Analyzing the distributions of the features enables us to estimate how relevant the features are with regard to our task: detecting fake web-shops. If the features show similar distributions for the fake web-shops and the legitimate websites, then it is unlikely to uncover new fake web-shops using those. On the other hand, if the distributions diverge, information can be extracted and classes can be separated. This step will also enable us to highlight aspects of the websites specific to fake web-shops.

Recall that the set of *legitimate web-shops* is a subset of the *legitimate websites*. The distributions for both classes are shown to assess whether features discriminate *fake web-shops* or only web-shops in general. For completeness, the three classes (*legitimate websites*, *fake web-shops* and *legitimate web-shops*) are shown on every graph. In the following graphs, the percentage of domains is always expressed as the percentage of domains for the specific class, not for the entire dataset.

3.2.1 Page-level features

The distributions for the number of images and the number of HTML tags are very compact for the fake web-shops but are more diverse for the legitimate websites, as shown in figures 3.1 and 3.2. Malicious websites tend to mainly use internal links, while legitimate websites use a combination of internal and external links, as shown in figures 3.3a and 3.3b. Very few legitimate websites use meta keywords but fake web-shops extensively use those (figure 3.4a). To a lesser extent, a similar distribution can be observed for the number of words in the web page's meta description (figure 3.4b). Fake web-shops tend to have a title that differs more from their domain name than legitimate websites, as shown in figure 3.5. Finally, fake web-shops operators generally use the same languages as legitimate websites (figure 3.6).

3.2.2 Merchant related features

Merchant related features try to capture information about the identity of the website owner. One of the first ways to retrieve such information implemented in DNS Belgium's crawler was to count the number of links that point to a telephone number or an email address. Those links respectively start with `tel:` and `mailto:` in the HTML source code and allow users to directly interact either by opening the dial pad or the email client to write a message. This approach captures very well the number of links designed for this purpose but fails to capture email addresses and phone numbers that might appear in plain text on the website. Telephone links are generally less popular than email links. For both types of links, fake web-shops hardly ever use them, while legitimate websites use them a little more often, as shown in figure 3.7.

More recently, the crawler has been added the capability to detect social media links

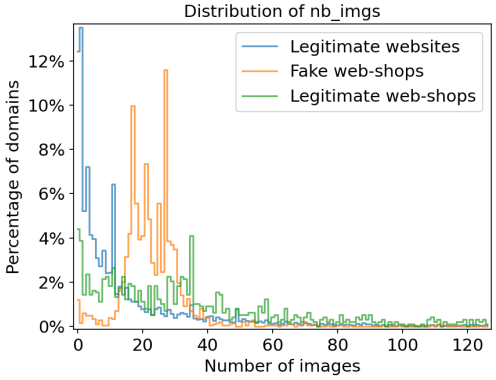


Figure 3.1: The distribution of the number of images on the fake web-shops is more compact than on the legitimate websites. Outliers on the x-axis have been removed for readability.

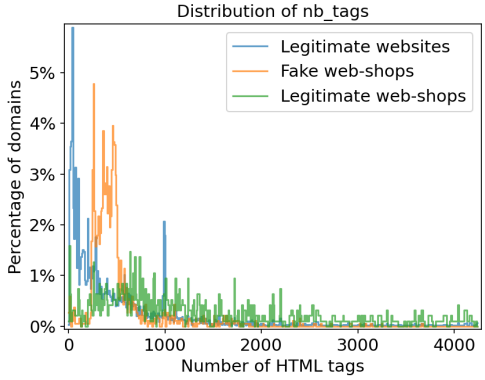
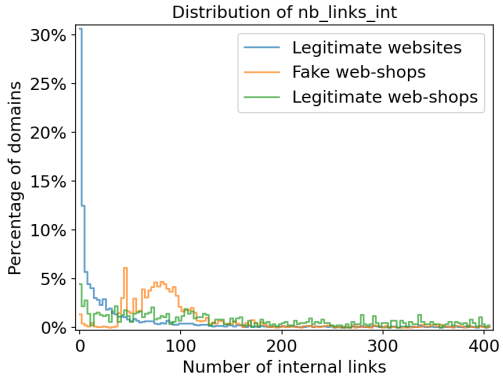
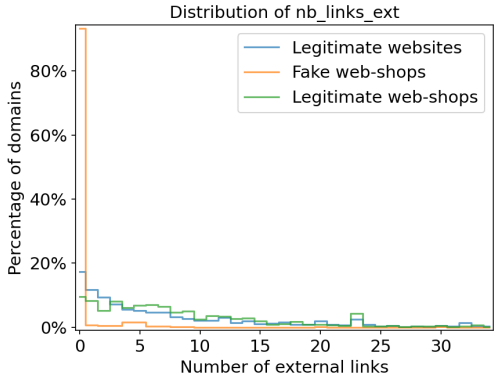


Figure 3.2: The distribution of the number of HTML tags on the fake web-shops is more compact than on the legitimate websites. Outliers on the x-axis have been removed for readability.

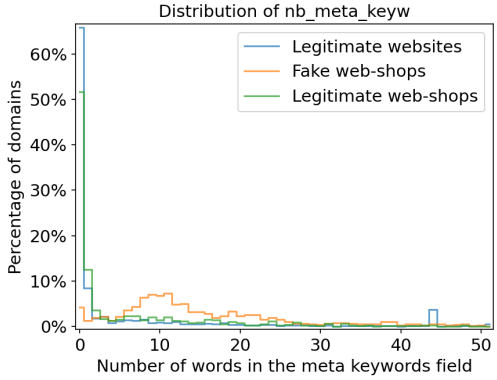


(a) Fake web-shops tend to have more internal links than legitimate websites.

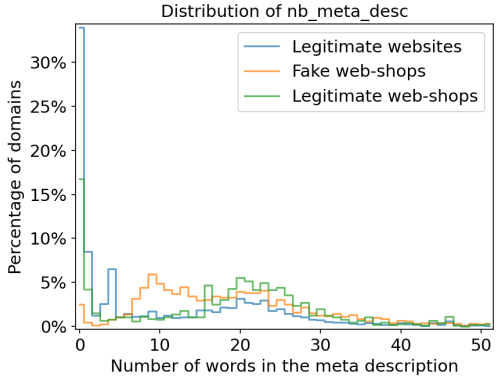


(b) Very few fake web-shops have external links on their front page, which is more common on legitimate websites.

Figure 3.3: Distributions of the number of internal and external links. Outliers on the x-axis have been removed for readability.



(a) Meta keywords are extensively used by fake web-shops but not much on legitimate websites. Search engines often use meta keywords to index the pages.



(b) Around a third of legitimate websites do not use the meta description field, while nearly all fake web-shops do.

Figure 3.4: Distributions of the number of words in the meta keywords and meta description HTML fields. Outliers on the x-axis have been removed for readability.

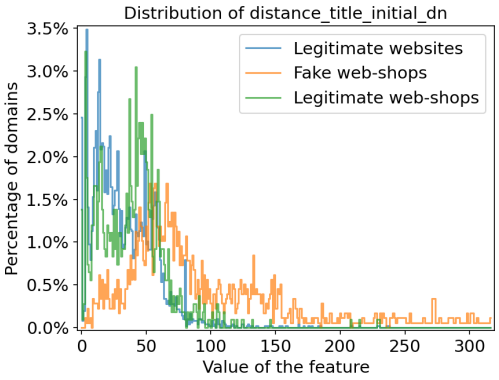


Figure 3.5: The edit distance between the title and the domain name (before following redirections) is higher for fake web-shops than for legitimate websites. This could indicate that the title and the domain name do not carry the same information for fake web-shops. Outliers on the x-axis have been removed for readability.

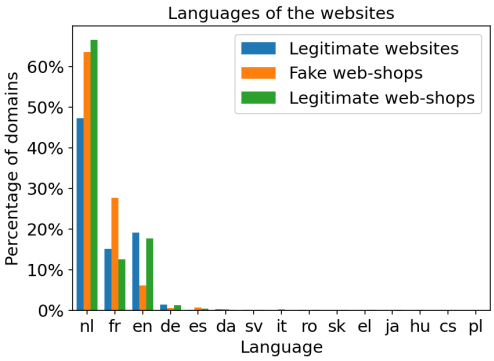


Figure 3.6: Fake web-shops and legitimate websites usually use the same languages. Only the 15 most used languages are depicted for readability.

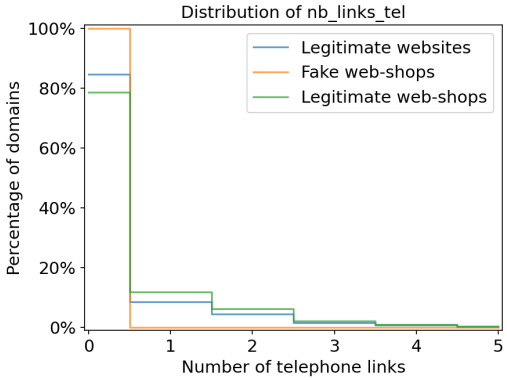
(namely, links to Facebook, Twitter, LinkedIn, YouTube and Vimeo). The crawler looks for two kinds of links: deep and shallow links. Deep links point to a resource on social media created by a user (e.g., a profile, a publication, an event, a video). In contrast, shallow links point to the front page of the website or a generic page (e.g., the terms of service, the privacy policy, the sharing menu). The most popular links on legitimate websites are the deep links as websites owner probably want to advertise their social media accounts. Fake web-shops do not often display social media links on their front page. Figure 3.8 summarizes the number of social media links displayed on the home page of the websites.

3.2.3 Product related features

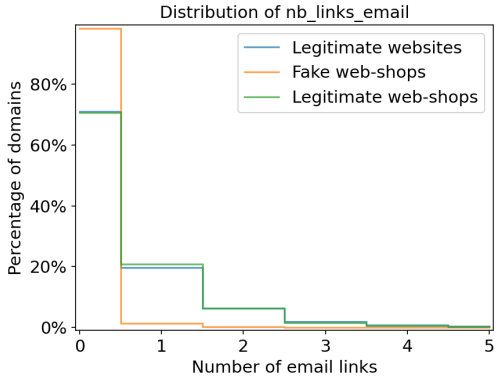
Product related features capture the number of products on sale on a given web page. The first approach followed by DNS Belgium to approximate the number of products was to count the number of strings that look like prices (e.g., numbers with generally two decimal places and optionally thousand separators). More recently, they added a currency detection ability to the crawler to count the total and the unique number of currencies appearing on a web page. Generally, fake web-shops display more numerical strings and currencies (both total and unique) than legitimate websites. Figure 3.9 summarizes the distributions for the number of numerical strings and unique currencies count on the websites in our baseline dataset.

3.2.4 Website-level features

Website-level features offer general information about the domain configuration and the hosting provider. Those features were among the first features used by DNS Belgium to detect malicious features in the .be zone. When analyzing the autonomous systems numbers, the country of the hosting provider and the presence of a MX record, it appears that those features have a very different distribution for fake web-shops and legitimate websites. From

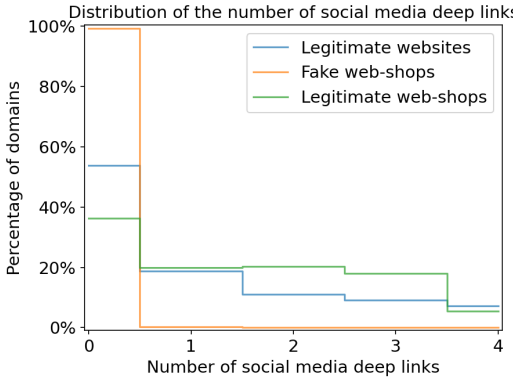


(a) Number of `tel:` links on the home page of the website.

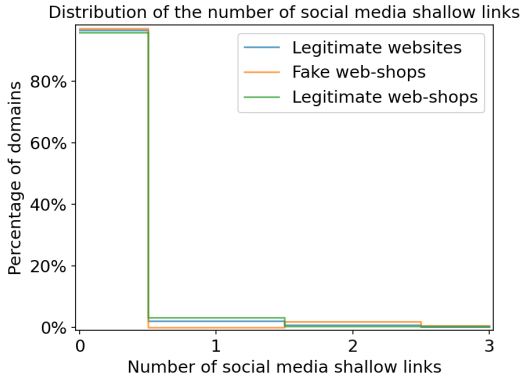


(b) Number of `mailto:` links on the home page of the website.

Figure 3.7: `tel:` and `mailto:` links are not very popular on website home pages. Fake web-shops tend not to use any of them while legitimate websites use them more, especially `mailto:` links. Outliers on the x-axis have been removed for readability.

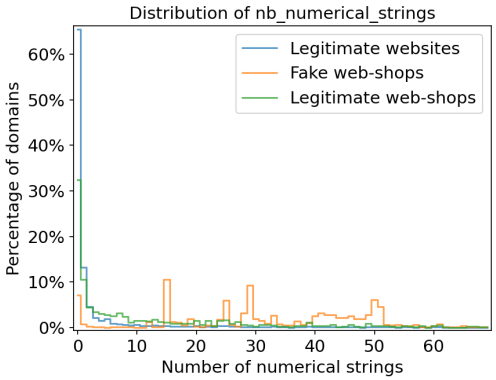


(a) Number of deep social media links on the home page of the websites.

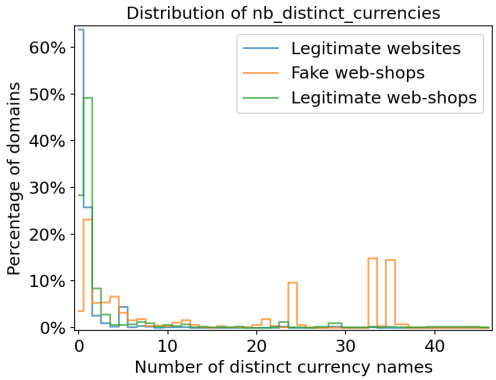


(b) Number of shallow social media links on the home page of the websites.

Figure 3.8: Websites, in general, do not often use shallows links to social media. Fake web-shops hardly ever use deep links to social media, while more than half of the legitimate web-shops do. Outliers on the x-axis have been removed for readability.



(a) Distribution of the number of numerical strings on the home page of the websites.



(b) Distribution of the number of distinct currencies on the home page of the website.

Figure 3.9: Fake web-shops seem to usually display more prices on their home page than legitimate websites. They also display many different currencies on their home page. Outliers on the x-axis have been removed for readability.

the data collected in our baseline dataset, some countries and some autonomous systems host only fake web-shops, as shown in figure 3.10.

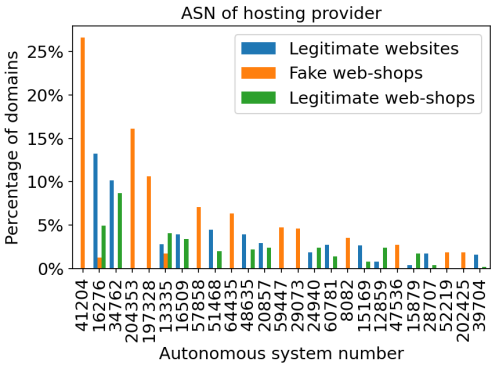
It is important to note that the concentration of fake web-shops in some autonomous systems and some countries is a direct consequence of how DNS Belgium found the first set of fake web-shops. In 2019, DNS Belgium was provided with 112 known fake web-shops operating in the .be zone by the FPS Economy. Those malicious websites were all hosted in a limited number of autonomous systems. Based on this observation, DNS Belgium reviewed all the .be websites in those autonomous systems in order to find new fake web-shops. This led the organization to find a first set of fake web-shops, probably controlled by the same operators.

Because of how DNS Belgium first identified fake web-shops, some of the website-level features are highly biased and do not represent the true distribution of all the fake web-shops present in the .be zone. For the sake of completeness, we include those features here as well.

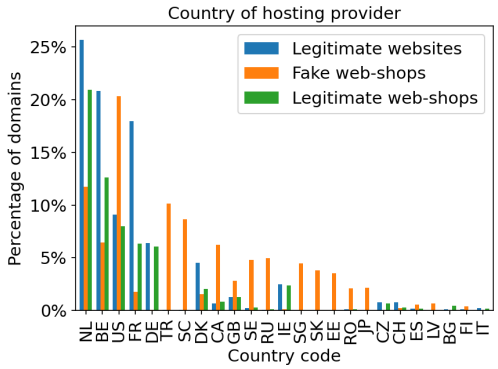
3.2.5 Registration features

Registration features give information about how the domain was registered. The registration data used by the classifier at DNS Belgium mainly covers two things: the person or the organization registering the domain and the history of the domain. The features presented here are more about the history of the domain rather than the registrant's identity, only the country of the registrant is used as a feature. In the literature review, we presented the concept of *drop-catching*, explained by Hao et al. [21]. This behavior can be observed in our baseline dataset: more than 80% of the fake web-shops are re-registered, and more than 75% are within ten days after the domain became available. Figure 3.11 summarizes the dataset for those aspects. The age of the domain gives a good indication of its trustworthiness, as depicted in figure 3.13.

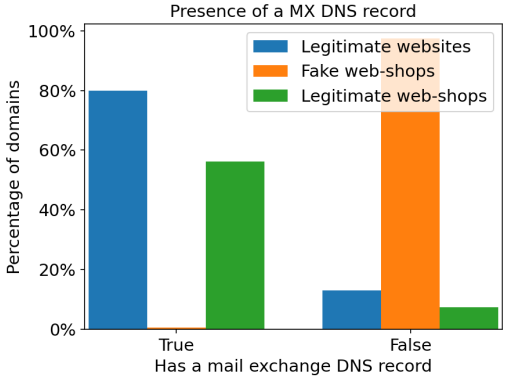
When analyzing the country of residence of the registrant and the registrar (figure 3.12) used to buy the domain name, it appears fake web-shop operators report that they live in



(a) Multiple autonomous systems only host fake web-shops. Fake web-shops are concentrated in a few autonomous systems, with legitimate websites spread across more autonomous systems.

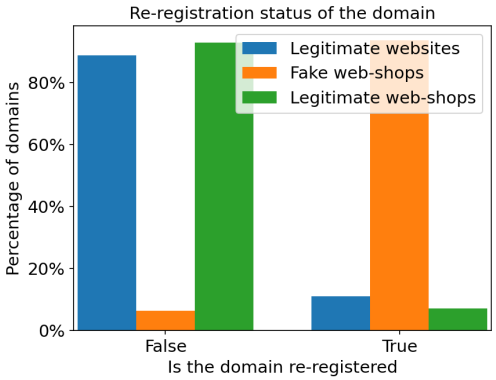


(b) Some countries only host fake web-shops while legitimate domains are concentrated in a few countries.

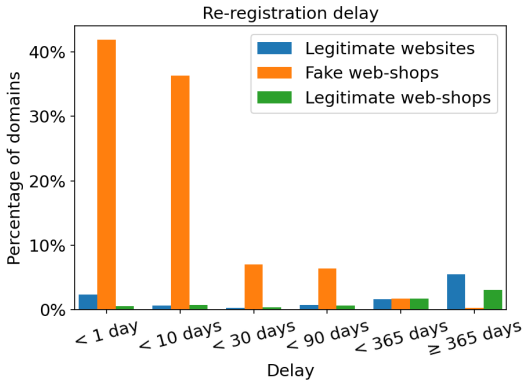


(c) Fake web-shops usually do not set up MX records. The sum for the categories does not always equal 100% because the data is missing for some entries in the dataset.

Figure 3.10: Summary of the website-level features. Sub-figures a and b are limited to the 25 most popular for clarity.



(a) Nearly all the fake web-shops are using re-registered domains.



(b) Fake web-shops tend to re-register domains quickly after they become available. This is known as *drop-catching*. Re-registered legitimate web-shops do not exhibit such behavior.

Figure 3.11: The data in our baseline dataset contains cases of *drop-catching*, almost exclusively for fake web-shops.

two countries and use mainly two registrars. As already mentioned in sub-section 3.2.4, this is probably a consequence of how DNS Belgium gathered their set of fake web-shops in the first place. It is interesting to notice that the countries of the registrants of fake web-shops (figure 3.12a) do not match the countries hosting the fake web-shops (figure 3.10b).

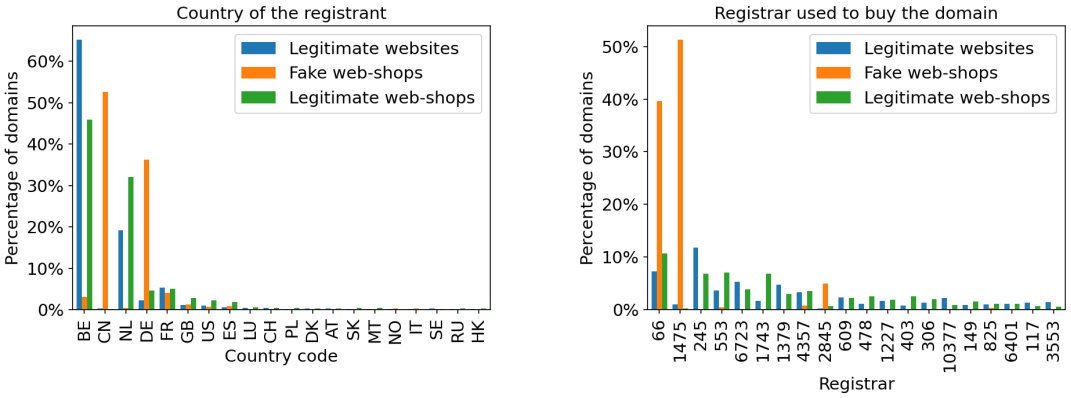
3.2.6 Summary

According to the distributions presented in the previous sub-sections, one could draw a rough approximation of what an average fake web-shop looks like as follows. Fake web-shops seem to have a fixed number of tags in their HTML source code (around 500) and display relatively more images than legitimate websites (generally between 15 and 30 on their homepage). Those malicious web-shops tend to keep Internet users within their websites by creating many internal links and very few links to other websites. HTML tags generally meaningful for search engine spiders (such as the meta description and the meta keywords) are extensively used by fake web-shops, while this is not so popular among legitimate websites.

When considering the contextual information that the websites provide, fake web-shops often lack contact information (i.e., telephone, email, social media links). This is also observed when analyzing the DNS records: very few fake web-shops have MX records, used for email exchange.

Fake web-shops, in general, are more likely to advertise many products on their home page: we found more numerical strings (looking like prices) and distinct currency names on fake web-shop homepages than on legitimate web-shops (or websites in general).

Finally, malicious web-shops exhibit a registration pattern that is not found for other websites: they re-register domains very quickly after they became available. This is known as *drop-catching* and is a common practice to take advantage of the existing reputation of a domain name that is no longer used.



(a) Registrant report that they live mainly in China and Germany. This information is provided by the registrants when buying a domain name.

(b) Two registrars account for around 90% of the fake web-shops.

Figure 3.12: Our baseline dataset shows that registrants of fake web-shops mainly originate from China and Germany and use almost exclusively two registrars. Those features are probably biased due to the way the dataset was constructed. Only the 20 most popular countries and registrars are shown for clarity.

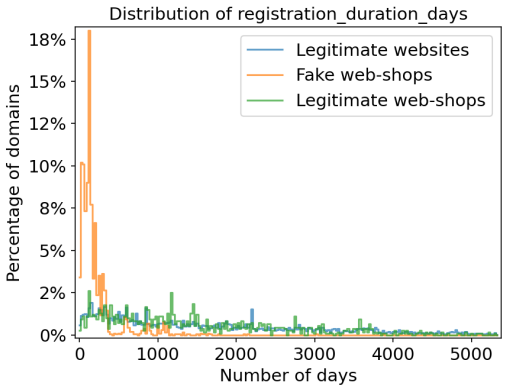


Figure 3.13: Fake web-shops are generally not very old when detected. The age of legitimate websites is more diverse. Outliers on the x-axis have been removed for readability.

3.3 Existing classifier

Currently, DNS Belgium uses a supervised classifier to detect new fake web-shops. They mainly used a Random Forest Classifier but also trained a Gradient Boosting Classifier, as mentioned by Batsleer [4] in order to compare the performances. Both classifiers achieved similar precision and recall.

In this section, we describe the workflow of DNS Belgium to train their Random Forest Classifier. We train a Random Forest Classifier using this workflow and evaluate its performance using our baseline dataset.

3.3.1 General architecture

The current solution used by DNS Belgium to build a classifier uses 80% of the labeled data as a training set and 20% as a test set. The features are then pre-processed using the following steps.

- The missing boolean and numeric features are imputed with the median value.
- The numeric features are standardized by removing the mean and scaling to unit variance.
- The textual fields (`meta_text`, `body_text` and `title`) are independently turned into TF-IDF features. Only words of at least three characters and appearing in at least 0.1% and at most 90% of the documents are used.
- The list of hosts in `external_hosts` is converted to a string (all the elements are grouped in a single string, separated by a space) and turned into TF-IDF features.
- Other features, such as categorical features, are dropped by the current implementation of the classifier.

Since registration data is stored in a different location than the crawler data, registration features are not always used when performing fake web-shop predictions with the classifier. The website-level features are also often left out because they increase the query time due to the structure of the stored data.

Once the features are pre-processed, a Random Forest Classifier is trained. A grid search with a set of hyper-parameters is used to find the best combination of hyper-parameters with a 5-fold cross-validation.

3.3.2 Performance evaluation

We trained a Random Forest Classifier as described above, using 51 features (see subsection 3.1.2). Table 3.3 summarizes the metrics of the trained classifier on the baseline dataset. The most significant features (excluding TF-IDF features), based on the mean decrease in impurity, are depicted in figure 3.14. Those features are consistent with the distributions observed in section 3.2.

The trained Random Forest Classifier achieves a high recall and a perfect precision so, at first glance, this should lead DNS Belgium to find most of the fake web-shops in the .be zone. However, in practice, the classifier mainly reports false positive instances. To assess the practical usability of the classifier, we reproduced the most common workflow at DNS Belgium to detect fake web-shops. For this detection, we consider only the domains hosting a website and crawled during the last full crawl (which ran on March 1st, 2022) and take the following steps.

1. We extract the features (HTML, registration and DNS features) for all the websites crawled (around 1.3 million).
2. We instantiate a pipeline to pre-process the data as described in sub-section 3.3.1.
3. We use our baseline dataset to train a Random Forest Classifier.
4. For each crawled website, we get the prediction and store the result in a relational database.
5. We review the websites with the highest probability of being fake web-shop.

Of the crawled websites submitted to the classifier, only 15 received a probability higher than 0.5 of being a fake web-shop. The 13 first domains (the most suspicious according to the classifier) are all similar: a page showing some content and then a lot of links to shoes. From those 13 domains, 11 were already taken down by DNS Belgium between the crawl time and the prediction. This kind of websites is malicious because they try to sell very cheap shoes from popular brands, but they are not usual web-shops. When clicking on one of the links shown at the bottom of the page, we are prompted to send a WhatsApp message to a given phone number. Such examples are in the training set and labeled as fake web-shops, which explains why those appear at the top.

The remaining two websites that are flagged with a probability greater than 0.5 as fake web-shops really look like web-shops and are suspicious. When sorting the predictions by ascending probability of being a fake web-shop, websites further down in the list are somewhat more interesting. Some of them are web-shops and look very suspicious (selling many different unrelated items, large discounts and no information about the owner), some redirect to websites selling pills and some look like honest web-shops. However, those websites have been predicted with a slightly higher probability of being legitimate websites than fake web-shops.

The poor performance of the classifier when applied to actual data has already been analyzed by Batsleer [4]. He identified three possible causes for the problem, which we think are still relevant.

1. Fake web-shops tend to use other TLDs than .be as the operators know that DNS Belgium actively takes countermeasures against them.
2. Some parts of the .be zone are not well represented in our baseline dataset, causing the classifier to produce unexpected results when predicting the class for instances lying in those parts of the zone.

Metric	Value
F1	0.972
Accuracy	0.987
Recall	0.945
Precision	1.0

Table 3.3: Metrics for the existing Random Forest Classifier trained on the baseline dataset.

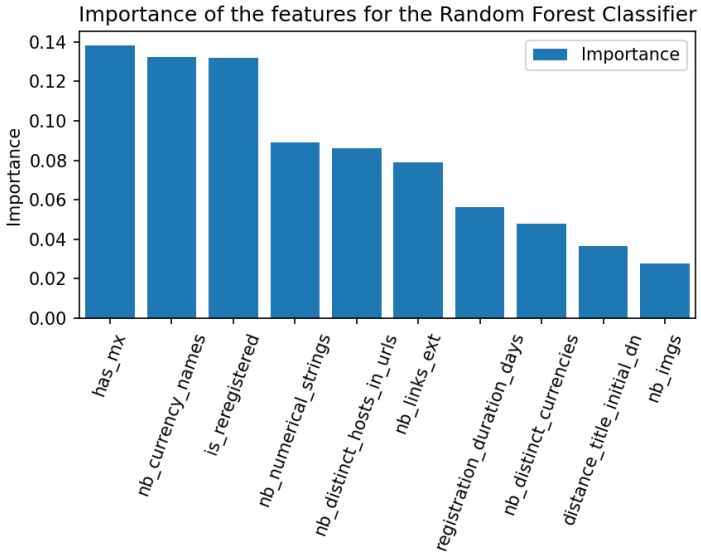


Figure 3.14: Most significant features (not TF-IDF) based on the mean decrease in impurity for the Random Forest Classifier.

3. The strategy of fake web-shop operators evolves over time to evade detection. As most of the labeled fake web-shops were crawled in 2019, they may not represent the newest types of fake web-shops operating in the .be zone.

The two last hypotheses seem more likely than the first. The next chapter presents how active learning can be implemented to allow the classifier to ask the label for instances that may bring valuable information.

3.3.3 Challenges of building a classifier

Gathering labeled data in order to train a classifier may require some effort. In this section, we briefly cover some of the main challenges encountered by DNS Belgium when gathering and leveraging labeled data.

Biased data. The initial dataset created by DNS Belgium was constructed using 112 fake web-shops provided by the FPS Economy. Those websites were reported by users that filled a complaint to the FPS Economy. The benign web-shops were provided by employees of DNS Belgium, listing the web-shops that they know are trustworthy. This approach created a dataset that is not representative of the set of web-shops in the .be zone and may lead a classifier to mainly predict websites that are very similar to the labeled ones.

Adversarial setting. The situation of DNS Belgium regarding fake web-shop detection could be qualified as a doubly adversarial setting. On the one hand, the operators adapt their techniques to evade detection and continue to operate as long as possible. On the other hand, the interesting examples (i.e., new fake web-shops) detected by the classifier are taken down as soon as possible. This means that the website is no longer accessible and should have been copied and archived with all the relevant data in order to learn from it in the future.

Dealing with the legacy. As we already briefly mentioned, DNS Belgium crawled most of its labeled fake web-shops with a previous crawler. This crawler had different features than the one currently used. For this reason, the data collected by both crawlers do not present the same features and some adaptation needs to be done to translate the data gathered by the old one into the data schema of the new one. Even if some features can be computed perfectly (e.g., features solely based on the HTML as the HTML was captured by the old crawler), some may only be approximated (e.g., detection of the technologies by Wappalyzer [2]) and some cannot be computed (e.g., taking a screenshot of the page, looking across pages to find a VAT number, SMTP configuration detection). For this reason, some features collected by DNS Belgium are not currently used to detect fake web-shops as they are missing from a significant part of the labeled data.

Constant implementation of new techniques. The current crawler used by DNS Belgium is still under development and new features are added on a regular basis. Obviously, data crawled before the implementation of new features are missing those. Recently, DNS Belgium slightly modified the crawler to be able to re-extract features based on the HTML to easily compute the value of the new features on older data if needed. However, as for the previous point, some new features may not be computed for older data. This requires to find new data to label so the labeled dataset has all the newest features and it can be leveraged for classification.

Limited resources. Manually reviewing hundreds of websites takes time and DNS Belgium may not reasonably spend days or weeks reviewing websites in the hope of finding some suspicious ones to take down and label. Telling apart fake web-shops from legitimate websites requires some insight and is more time-consuming than telling apart pictures of cats and dogs.

Chapter 4

Methodology

In this chapter, we present the different steps we take to build an active learning classifier able to tell apart legitimate websites from fake web-shops in the entire .be zone. We start by implementing additional features that could help with our classification problem based on the literature. We analyze the distributions of those features on the different classes of interest to assess whether they bring valuable information for the classification.

Next, the design of the active learner that will be used for classification is explained, and the design choices are motivated. In order to evaluate how interesting the active learning approach is, we present two experiments to assess the capacity of the learner to predict fake web-shops in two settings: (i) using the baseline dataset and providing labels to the learner as it queries for instances to be labeled and (ii) using all the websites available in the .be zone. We also lay out our expectations for those two experiments.

4.1 Implementation of new features

In addition to the features already extracted by the DNS Belgium's crawler, we implemented some new features discussed during the literature review. We implemented features presented in previous works that are either not too computationally expensive or calculable from data already present in DNS Belgium's database. This criterion has been chosen to select the features to implement as the features need to be extracted for each website in the entire .be zone, currently containing more than 1.7 million domains.

4.1.1 Open Graph tags

The Open Graph protocol "enables any web page to become a rich object in a social graph" [17]. Websites use this protocol to enable social media to display their content as rich objects instead of plain links once shared on a platform. As mentioned by Cox and Haanen [11], legitimate websites may have an incentive to adopt such protocols to increase their visibility on social media and attract more viewers. On the other hand, fake web-shops operators may also be interested in being represented on social media, but this requires more effort and will probably not be implemented by many.

The Open Graph tags are included in the HTML source code as meta tags in the head of the web page. Each tag is represented as a meta HTML tag with a property name and the value for that property. Listing 1 shows an example of how Open Graph tags can be used to represent a movie on social media. The property attribute always starts with `og:` (to indicate that this tag is an Open Graph tag) followed by the property name. The property name can be composed of multiple levels: a website can include the tag `og:image` to specify the image to use when sharing the content to a social media and include the tag `og:image:alt` to specify an alternative description of the image for accessibility purposes.

```
<html prefix="og: https://ogp.me/ns#">
<head>
<title>The Rock (1996)</title>
<meta property="og:title" content="The Rock" />
<meta property="og:type" content="video.movie" />
<meta property="og:url" content="https://www.imdb.com/title/tt0117500/" />
<meta property="og:image"
↪  content="https://ia.media-imdb.com/images/rock.jpg" />
...
</head>
...
</html>
```

Listing 1: Example of Open Graph tags for the movie The Rock [17]

When extracting the Open Graph tags, we only considered the top-level tags (i.e., `og:image` and `og:image:alt` are both considered as an `image` tag).

Given that DNS Belgium stores the HTML hosted at the root of the crawled domains, we can extract the Open Graph tags present on each website in our dataset. Multiple features represent the presence of Open Graph tags on the page. For 15 of the top-level tags defined by the Open Graph protocol, we count the number of times the tag appears on the web page. We also include a global counter to indicate the total number of Open Graph tags on the page. In order to account for non-standard tags that websites may use, we add a feature to count the number of Open Graph tags that are not defined by the Open Graph protocol [17].

After extracting those features from the baseline dataset, around 40% of the legitimate websites use Open Graph tags and commonly use five to seven tags. Almost no fake web-shop uses Open Graph tags, which was expected. Figure 4.1 shows the distribution of the number of Open Graph tags.

As depicted in figure 4.2, six tags are used by at least 20% of the legitimate websites, and only seven out of the fifteen used as features are commonly used. Again, we can clearly see that fake web-shops do not often use Open Graph tags.

4.1.2 Technologies and third-party services

Cox and Haanen [11] identified the use of third-party analytics tools as a feature to tell apart fake web-shops and legitimate websites. They hypothesize that fake web-shops operators are less interested in audience measurement than owners of legitimate websites. The presence of third-party analytics tools on the website is among the most important features of the classifier they built.

Gopal et al. [19] focused their research on an extensive analysis of requests to third-parties. Their research shows how the analysis of the different levels of third-party requests can bring valuable information to classify websites. They study the different levels of requests: the first level being the third-parties directly called by the visited website, the second level being the third-parties called by the first level third-parties and so on. In their dataset, they observed up to 27 levels of requests. While such an analysis would not

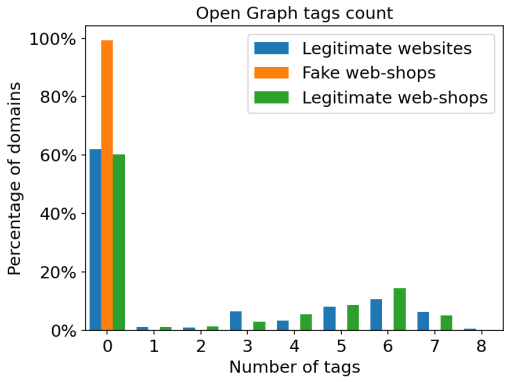


Figure 4.1: Almost none of the fake web-shop in our baseline dataset use Open Graph tags on the front page, while around 40% of legitimate websites do.

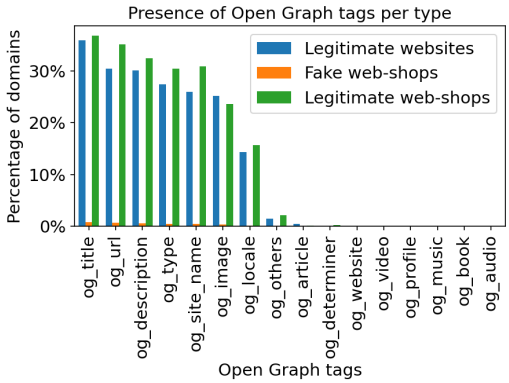


Figure 4.2: Six Open Graph tags are used by more than 20% of the legitimate websites. Only a small set of Open Graph tags are used in practice.

be tractable for the entire .be zone, we draw inspiration from their results to build new features.

The current version of DNS Belgium's crawler leverages the Wappalyzer library (Alias [2]) to get information about the different technologies used on the websites. Technologies detected by Wappalyzer [2] fall under different categories, including analytics which we are interested in. The current version of Wappalyzer [2] categorizes the technologies in 105 different categories. For each category, we created a feature to indicate the number of the detected technologies that fall under that category. In addition to those features, we added a feature to indicate the number of distinct categories found on the website.

Wappalyzer [2] uses the content of the HTML page and JavaScript variables, response headers, and more to identify the technologies used on a given website. While the current version of Wappalyzer [2] scans the live version of the websites, a significant part of our dataset (containing nearly all of the fake web-shops) was collected using the previous crawler used by DNS Belgium. This previous crawler did not collect such information about technologies used on the website. In order to extract information about technologies used on those websites, we served the crawled HTML pages to Wappalyzer [2] for analysis. This enabled us to extract some information based on the HTML content. Still, some technologies may not have been detected as Wappalyzer [2] also uses data found in the response header, which we could not reproduce. All the web pages were served using Nginx [34], and we removed the technology corresponding to Nginx [34] from the result to avoid any future bias toward this particular technology. However, this resulted in very few fake web-shop using web servers according to the features, which may also bias the model. For this reason, we remove the feature corresponding to web servers technologies when training the model with our baseline dataset.

Figure 4.3 shows how many different technology types are detected on websites. Fake web-shops tend to use one, two of for types of technologies, while the numbers for the legitimate websites are more diverse. Malicious web-shops mainly use JavaScript libraries, fonts, UI frameworks and e-commerce technologies, as shown in figure 4.4 while the types of technologies found on legitimate websites are more diverse. Again, the results obtained for the fake web-shops may be biased because of the way the technologies were detected

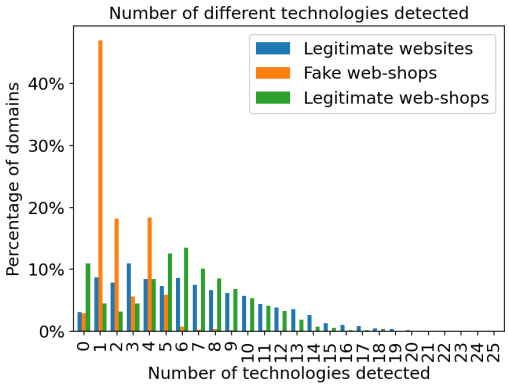


Figure 4.3: Fake web-shops usually use one to five different technologies on their website. The number of technologies on legitimate websites varies more and is generally comprised between 0 and 14.

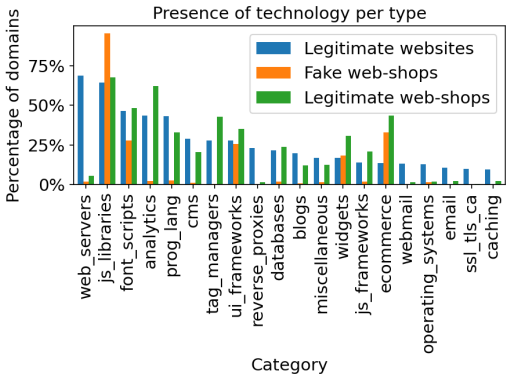


Figure 4.4: Five technologies are mainly used on fake web-shops, while legitimate websites use a more diverse set of technologies. Please note that the way this information was computed for websites crawled in 2019 (i.e., most of the fake web-shops in the dataset and some legitimate websites) may bias the result, see sub-section 4.1.2.

for those instances.

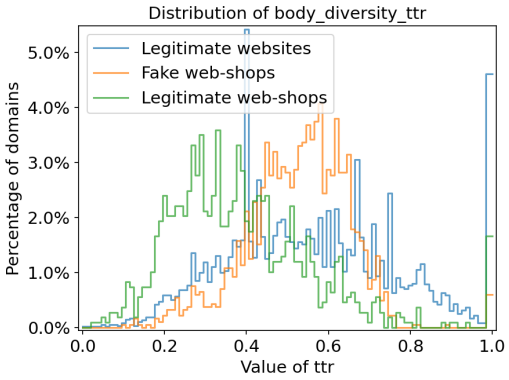
4.1.3 Lexical diversity of the body text

Cox and Haanen [11] suggest using lexical diversity as a feature. Their hypothesis is that fake web-shops will have lower lexical diversity than legitimate websites because they offer little other content than the products on sale. To evaluate the lexical diversity of a document, multiple metrics exist. We focus on metrics using the *Type-Token Ratio (TTR)*, which is defined as the ratio between the number of *types* (i.e., the number of unique words in the text) and the number of *tokens* (i.e., the total number of words in the text).

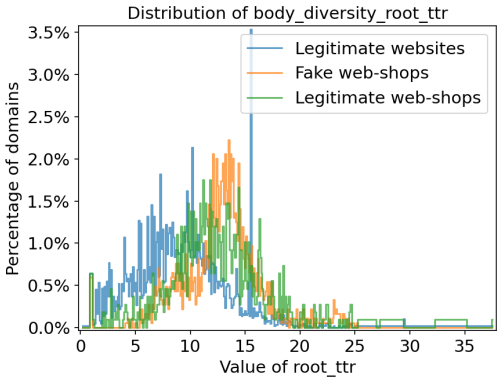
The following measures for lexical diversity were compared:

- `ttr` defined as $\frac{nb\_types}{nb\_tokens}$.
- `root_ttr` defined as $\frac{nb\_types}{\sqrt{nb\_tokens}}$.
- `msttr` defined as the average of the `ttr` computed for all the non-overlapping segments of 50 words in the text.
- `mattr` defined as the average of the `ttr` computed for all the (possibly overlapping) windows of 50 words in the text.

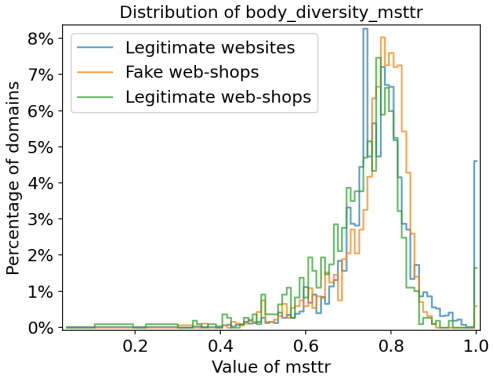
Before computing the different metrics for the lexical diversity, we apply lemmatization using spaCy (Explosion [16]), a natural language processing library in Python, for the text written in one of the following languages: Dutch, French, English or German. When the text is written in another language, only basic tokenization is applied. To avoid storing an unbounded amount of data when crawling the websites, DNS Belgium's crawler truncates the text to a defined maximum length. The default value currently in use is 20,000



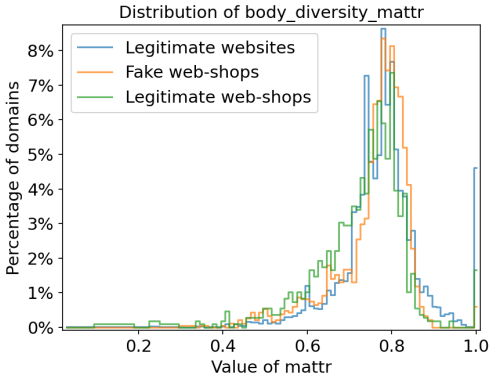
(a) Distribution of the `ttr` computed on the body text of the HTML pages.



(b) Distribution of the `root_ttr` computed on the body text of the HTML pages.



(c) Distribution of the `msttr` computed on the body text of the HTML pages.



(d) Distribution of the `mattr` computed on the body text of the HTML pages.

Figure 4.5: Distributions of the different metrics used to reflect the lexical diversity of the body text.

characters maximum. Web pages with a body text longer will only have the 20,000 first characters stored, and the feature `body_text_truncated` will be set to `true`.

The four metrics presented were applied to the baseline dataset and their distributions are shown in figure 4.5. Unlike what Cox and Haanen [11] expected, the values for the lexical diversity are often somewhat higher for the fake web-shops than for the legitimate websites when computed on our baseline dataset. Based on the definitions of the different metrics and our intuition, the first metric implemented to represent the lexical diversity was `mattr` as the moving average seemed more suitable for texts of varying length. However, the distributions for the fake web-shops and the legitimate websites (figure 4.5d) are not very distinct. Building on this observation, we computed other metrics and observed their distributions.

Simpler metrics like `ttr` and `root_ttr` gave more distinct distributions. In order to select the best metric to represent the lexical diversity, we computed the information gain for each of the four metrics using the entropy as split criterion. As shown in table 4.1, the `root_ttr` yields a higher value for the information gain. For our dataset, this metric seems to be the most effective in representing the lexical diversity of the text present on the HTML pages and telling apart fake web-shops and legitimate websites.

Metric	Information gain	
	Legitimate websites versus fake web-shops	**Legitimate web-shops versus fake web-shops**
`ttr`	0.749	0.919
`root_ttr`	0.782	0.942
`msttr`	0.461	0.687
`mattr`	0.776	0.942

Table 4.1: Information gain for the different lexical diversity metrics. The first column gives the information gain when separating legitimate websites from fake web-shops. The second column gives the information gain when separating legitimate web-shops from fake ones.

4.1.4 Semantic similarity between the domain name and the title

DNS Belgium's crawler currently computes the edit distance and the longest sub-sequence between the title and the domain name (before and after the redirections). The crawler also computes the fraction of words of the title that appear in the domain name (again once before and once after following the redirections). Those metrics are suitable to reflect the similarity of two strings but fail to capture if the strings are semantically close.

In order to capture the semantic similarity between the title and the domain name, we leverage spaCy [16] to compute the semantic similarity of two strings. spaCy [16] uses word2vec (Mikolov et al. [32]) as in [11] to represent words in a vector space. To compute the similarity of two strings, spaCy [16] computes the cosine similarity using an average of word vectors. This results in a score between -1 and 1, representing how similar the strings are: the higher the value, the more similar the strings are.

Granted, the definition of semantic similarity cannot be defined precisely and strongly depends on the application. For example, a sentence about a dog and another about a cat might be considered similar in a general way but quite different if the goal is to sort sentences based on what animal is mentioned. Moreover, spaCy [16] uses an average of the word vectors to compute the representation of the entire string. This means that the order of the words in the string is not taken into account and that the wording of the phrases might influence their similarity score.

In order to create a meaningful representation of the string, spaCy [16] needs words. The title of the HTML page is usually naturally split into words. However, domain names do not always contain separators between words. To overcome this, we draw inspiration from Cox and Haanen [11] and divide the domain name into words as follows:

1. We generate all the possible sub-strings from the domain name.
2. We keep only the words appearing in the dictionary for the relevant language.
3. We remove all the words that are sub-strings of other words.
4. We sort the words by descending length and keep the n first words such that the total length does not exceed the length of the domain name.

Obviously, this heuristic is far from perfect and often results in a sub-optimal split of the domain name for multiple reasons. Here are multiple examples of how the described

method performs to extract words from the domain name.

From `postgraduatespaceandservicedesign.be`, the method extracts the following string: `postgraduates serviced design espace`. The word *serviced* has been preferred over *service* as the second is a sub-string of the first word. The same holds for *espace* being preferred over *space* because *espace* appears in the English words that are used (even if the word is not an English word).

From `aannemingsbedrijfvda.be`, the method extracts the following list of words: `aannemingsbedrijf v`. As *VDA* is an abbreviation that is not present in the Dutch dictionary that we use, the word is left out.

From `abcdudeveloppement.be`, the method extracts the following list of words: `a du developpement`. The word *abc* is not part of the French dictionary used here, but it carries some information in this context.

Multiple factors may influence the quality of the results. The content of the dictionary plays an important role here: if too few words are present in the dictionary, the meaning of some parts may be missed. On the other hand, if the dictionary contains random words, those may appear in the final result where a more suitable split was possible. The number of natural words present in the domain name also influences the result: if the domain name is composed of one or two words, the result is generally the expected one.

Other aspects of the solution should be considered when evaluating the performance. Choosing the words from all the possible sub-strings of the domain names brings a major drawback: some words in the final result may overlap (as in the first example, *serviced* and *design* overlap in the original domain name). Removing words that are sub-string of other words might be indicated in some cases, but this is not always the best option. The same is true when sorting the words by their length to keep only the n first.

A final thing to consider is that the language in which the domain name is written cannot be known precisely. In our implementation, we use the language detected from the body text and assume that the domain name is also in that language. When this assumption does not hold, the described method performs poorly.

Even with those limitations, the semantic similarity between the title and the domain name still provides valuable insight. Figure 4.6 shows the distribution of the semantic similarity computed between the HTML title and the domain name of the home page for each website in our baseline dataset. Domains with a similarity higher than 0.5 are almost all legitimate websites. The peak at 0 is caused by two main cases either (i) the language of the body text is not one of the four most popular in the .be zone (i.e., Dutch, French, English and German), and the score is set to 0 or, (ii) either the title or the domain name could not be represented in the vector space, making it impossible to compute their similarity.

4.1.5 Relative value of existing features

Currently, the features that represent the similarity between the title of the HTML page and the domain name (the edit distance, the longest sub-sequence and the fraction of words of the title that also appear in the domain name) are expressed as absolute values. Expressing the value of those features with regard to the length of the title, as suggested by Maarten

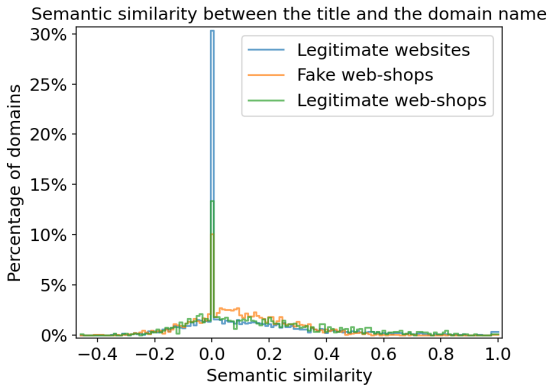


Figure 4.6: Legitimate websites generally have a somewhat higher semantic similarity between the title and the domain name. When the similarity cannot be computed, a score of 0 is attributed for the instance, which explains the peak.

Feature	Information gain	
	Absolute feature	Relative feature
`fraction_words_title_initial_dn`	0.225	0.311
`distance_title_initial_dn`	0.335	0.400
`longest_subsequence_title_initial_dn`	0.079	0.424

Table 4.2: Those features bring more valuable information when divided by the HTML title length. The values are computed when separating the legitimate websites from the fake web-shops.

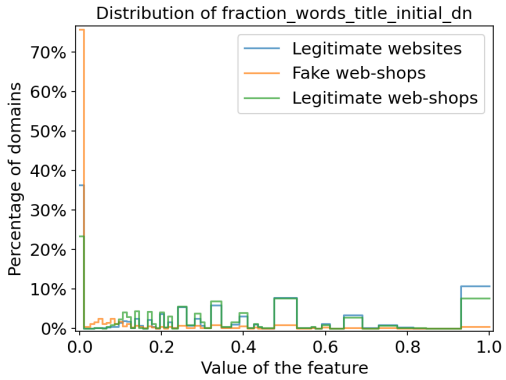
Bosteels from DNS Belgium [private communication], could help to differentiate between the distribution of the fake web-shops and the distribution of the legitimate websites.

The different distributions are depicted in figure 4.7. The most noticeable change is with the feature `longest_subsequence_title_initial_dn`: once divided by the title length, the distribution for the fake web-shops stays very compact while the distribution for the legitimate websites spreads out.

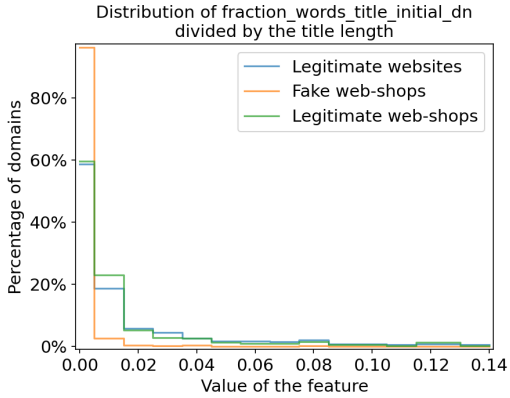
Table 4.2 shows the information gain for both the initial features and the features once divided by the title length. As expected, the most interesting increase of information gain is observed for the feature `longest_subsequence_title_initial_dn`. The two other features also benefit from the division by the title length.

4.2 Design of an active learning setup

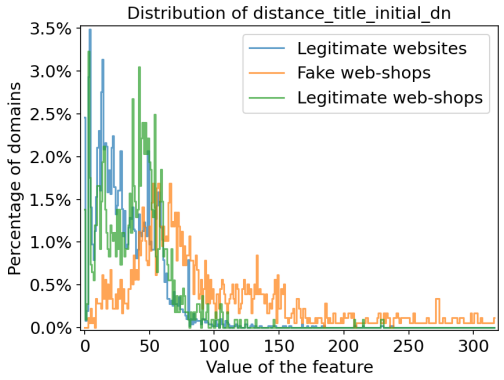
Given the setting of DNS Belgium and the literature about active learning, we believe that active learning may be suitable in our situation. This section presents how we plan to implement active learning and how we will assess its performance to confirm or refute our intuition. First, we detail how we implement active learning in our case and motivate our design choices. Next, we introduce two experiments designed to assess if active learning is more interesting than the current solution at DNS Belgium. Finally, we describe the expected results of the two experiments.



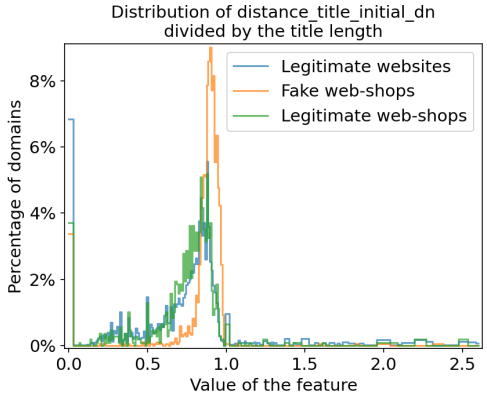
(a) Distribution of the `fraction_words_title_initial_dn` computed on the baseline dataset.



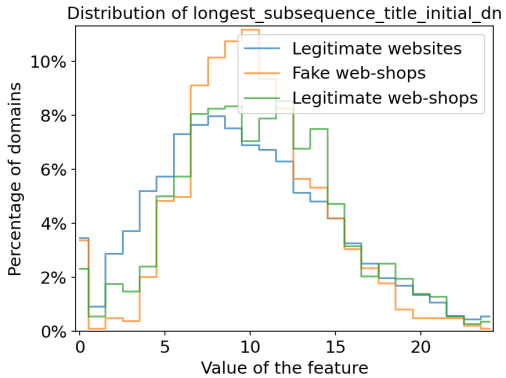
(b) Distribution of the `fraction_words_title_initial_dn` divided by the length of the title of the HTML computed on the baseline dataset.



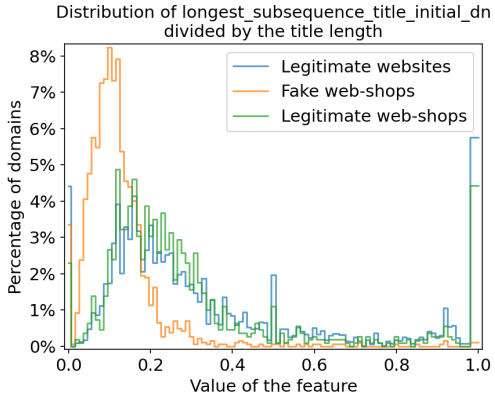
(c) Distribution of the `distance_title_initial_dn` computed on the baseline dataset.



(d) Distribution of the `distance_title_initial_dn` divided by the length of the title of the HTML computed on the baseline dataset.



(e) Distribution of the `longest_subsequence_title_initial_dn` computed on the baseline dataset.



(f) Distribution of the `longest_subsequence_title_initial_dn` divided by the length of the title of the HTML computed on the baseline dataset.

Figure 4.7: Comparison between the distribution of features with their absolute values and the same features divided by the HTML title length. Some distributions, especially for `longest_subsequence_title_initial_dn`, are more distinct when the feature is divided by the title length. Outliers on the x-axis have been removed for readability.

4.2.1 Implementation of an active learner

In order to implement an active learner, we need to make multiple design decisions. Those decisions concern the following aspects of the solution (i) the source of labeled and unlabeled data, (ii) the underlying algorithm used for classification and (iii) the query selection algorithm. This section details our design choices and highlights the motivations that led to the final choice.

Source of the data

An active learner is provided with two sets of instances. The first set, often referred to as the labeled dataset, is a collection of instances for which the label is known. In our case, we will use our baseline dataset (as described in sub-section 3.1.1). Creating a dataset from scratch takes time and DNS Belgium already has a decent number of labeled websites that can serve that purpose. By no means is this dataset perfect, but we consider it is representative of the kind of dataset on which a real-world classifier could be trained.

The second set of instances provided to an active learner, often referred to as the pool, is a collection of unlabeled instances. The active learner does not directly learn from those instances, but it will be allowed to select instances to be labeled and added to the labeled dataset. In our case, we use the websites crawled by DNS Belgium's crawler during their full crawl launched on March 1st, 2022. This dataset contains information about all the domains hosting a website in the .be zone at the crawl time. This amounts to around 1.3 million instances. For those instances, the features presented in sub-section 3.1.2 are extracted at crawl time or available in the registration database for the registration features.

Using the full crawl of March 2022 as the pool of unlabeled data is interesting for two main reasons. First, the data is recent, meaning that fake web-shops present in the pool are more likely to represent the latest techniques and strategies used by fake web-shops operators. Second, the data is a complete snapshot of .be zone. As the registry of the .be zone, DNS Belgium has a global view of the currently registered domains and can, therefore, crawl every domain. Please note that DNS Belgium does only crawl the second level domain and the `www` sub-domain for any given registered domain. This means that for `example.be`, only `example.be` and `www.example.be` will be crawled; a website hosted on `blog.example.be` will not be crawled.

Throughout the active learning algorithm iterations, the labeled dataset and the pool will evolve as the learner submits queries to the oracle. At each iteration, the learner is allowed to submit a query to the oracle with instances to label. Labeled instances will be removed from the pool and added to the labeled dataset, enabling the model to be trained using those new instances during the next iterations.

Classification algorithm

An active learning setting can be designed using any classifier that can provide the probability that a given instance belongs to a class. Some classifiers naturally provide a probabilistic output, but others, like Decision Trees and K-Nearest Neighbors, do not. Authors such as Lewis and Catlett [26] modified non-probabilistic classifiers, like Decision Trees, to have a probabilistic output. A probabilistic output allows the classifier to determine the

instances where it is the most uncertain, enabling techniques like uncertainty sampling.

Multiple papers in the literature review make use of Support Vector Machine Classifiers and use the distance between the instance and the decision boundary as a measure of uncertainty: the closest to the boundary, the most unsure the classifier is. For this reason, we first tried to implement a Support Vector Machine Classifier. Training a Support Vector Machine Classifier on our baseline dataset takes multiple times the time needed to train a Random Forest Classifier. DNS Belgium currently uses a Random Forest Classifier. Because of the significantly longer training time for the Support Vector Machine Classifier, we decided to use a Random Forest Classifier for this active learning setting.

The probabilistic output of the Random Forest Classifier is computed as the mean of the probability of the trees in the forest. The probability for the Decision Trees is defined as the fraction of instances of the same class in the leaf. While the certainty of the model for a Support Vector Machine Classifier can be computed based on the distance between the instance and the decision boundary, the certainty of the model for a Random Forest Classifier is reflected by "the purity" of the leaves in the trees on average for the given class.

Query selection algorithm

Multiple query algorithms were presented in the literature review. We leave out more complex query selection algorithms from the outset, such as expected model change or estimated error reduction. Those query selection algorithms often require the learner to train a new model for each instance in the pool. As the pool contains around 1.3 million instances, we focus on methods that do not require training a new model for each possible instance. In addition, we want the learner to present the human oracle with multiple instances at each iteration. In fact, every iteration can take a fair amount of time and having the human wait for the learner to submit a single new query seems counterproductive. By allowing the learner to submit multiple instances to the oracle, the human operator can label those instances and then start the next iteration while working on something else. This approach does not reduce the time needed for the iteration but increases the workload of the human operator between two iterations.

Those constraints led us to three query algorithms: *random sampling*, *Top K* and *K Cluster Medoid*. As there exists no general rule for selecting a query algorithm, we consider the three algorithms and will compare their performances during one of our experiments. The following paragraphs explain how the active learner chooses instances to submit to the oracle using those algorithms and why we consider them relevant.

Random sampling. This algorithm is the simplest and the most naive of the three selected. When the learner has to submit k instances to the oracle to label, it selects k random instances from the pool.

While such behavior may not be efficient, we keep this algorithm as a baseline for the query selection. This will enable us to compare the two other algorithms to a random learner and evaluate how interesting they are.

Top K. In this setup, the learner evaluates the probability of being a fake web-shop for every instance in the pool before selecting the instances to query, using the instances currently in the labeled dataset to build a model. When the learner has to submit

k instances for the oracle to label, it selects the k instances where the probability is the closest to 0.5. This enables the learner to obtain labels for instances about which it is the most uncertain.

Granted, the k instances submitted to the oracle may look similar and not bring diverse information to the learner. However, we keep this algorithm as a middle ground between random sampling and the next algorithm. We also argue that the time spent by the human oracle to label two similar instances is smaller than the time needed to label two very different instances. In fact, the instances are websites for which a screenshot is available, making it easy for a human operator to spot similar instances.

K Cluster Medoid. With this setting, the learner first evaluates the probability of being a fake web-shop for all instances in the pool before selecting instances to query, exactly like for the *Top K* algorithm. Once the probability for each instance is computed, the learner chooses the n instances with the probability closest to 0.5 and clusters those in k clusters. The medoids of the k clusters are then submitted to the oracle. Here, n is a parameter of the query selection algorithm.

Clustering instances allows the learner to submit queries that should bring diverse information to the model. Selecting medoids instead of centroids enables the human operator to make sense of the queries. If we allowed the learner to submit instances not present in the pool (i.e., the centroids of the clusters), the human would be unable to label the instances. In fact, instances are represented by a vector of features that is easy to construct from a crawled website but constructing the crawled website based on the vector of features is not doable in our setting.

4.2.2 First experiment: active learning over the baseline dataset

For the first experiment we carry out, we compare how the three different query selection algorithms perform using the baseline dataset. For this experiment, we divide the labeled dataset into two parts: 20% of the baseline dataset will be used as a holdout set to evaluate the performance of the model after each iteration, and the 80% remaining will be used as the pool of unlabeled data. Please note that the holdout set is never presented to the learner: the learner cannot select instances from the holdout set to be labeled. Moreover, for consistency, the holdout set is the same across all the iterations of this experiment.

As our baseline dataset consists of 7831 instances, the holdout set is made of 1567 instances and the pool is made of 6264 instances. Both subsets of the baseline dataset contain the same percentage of fake web-shops as the baseline dataset.

Three active learners are trained independently of each other. Each active learner uses a different query selection. All the learners start with the same set of labeled instances, but they select instances according to their own query selection algorithm, leading to potentially different labeled datasets during the experiment.

As we only use our baseline dataset for this experiment, we, as humans, already have the labels for all the instances. However, we present the pool as unlabeled data to the learner and only provide the learners with the labels it issues queries for. This setting allows us to run the experiment with an automated oracle: the program can give the labels when the learner requests them without requiring a human operator to inspect every instance.

Initially, all three learners are provided with the same labeled dataset, composed of 15 instances with the same percentage of fake web-shops as the baseline dataset (i.e., 4 fake web-shops and 11 legitimate websites). Those 15 instances are drawn randomly from the pool (expect that the percentage of fake web-shops must be the same as in the baseline dataset). Given the pool size (6264 minus the 15 initially labeled instances), we set to 5 the number of instances that the learners are allowed to query at each iteration. For the *K Cluster Medoid* query selection algorithm, we cluster the 50 instances with the probability the closest to 0.5. For every iteration until the pool of unlabeled data is exhausted (i.e., contains less than 5 instances), the three learners independently repeat the following steps.

1. Fitting the pipeline to transform the input data, as described in sub-section 3.3.1, except that categorical features are included using one-hot encoding and features presented in section 4.1 are included.
2. Finding the best Random Forest Classifier among a set of predefined hyper-parameters using 5-fold cross-validation.
3. Executing the query selection algorithm to choose 5 instances to be labeled for the next iteration.
4. Evaluating the performance of the classifier against the holdout set and saving the metrics for analysis.

At the end of every iteration, the queried instances are added to the respective labeled dataset and removed from the respective pool of the learners and the next iteration starts.

This experiment should highlight differences between the query selection algorithms and how effective active learning is for this dataset. In fact, once the pool of unlabeled is exhausted, the learners have the dataset they would have received for training in the current DNS Belgium setting. Thus, the performance obtained for the last learners (trained on the entire pool) should match what DNS Belgium could expect from their existing classifier if they used one-hot encoding for the categorical features and implemented the new features we suggest in section 4.1.

4.2.3 Second experiment: active learning over the entire .be zone

As the primary goal of DNS Belgium regarding fake web-shop classification is to find new fake web-shops operating in the .be zone, our work would not be complete without an overview of how well our approach performs for this task.

For this second experiment, we leverage active learning to predict fake web-shops for the entire .be zone. The following paragraphs explain how we build our first dataset and how we apply active learning on the .be zone.

The current dataset of DNS Belgium for fake web-shop detection is somewhat biased on some features and lacks some newly implemented features. For those reasons, we believe that building a new dataset may yield better predictions. Moreover, active learning is an iterative process where the labeled dataset evolves as the learner submits queries. Therefore, using an existing dataset may not be suitable, especially if the dataset contains some known weaknesses.

DNS Belgium labels websites for multiple purposes: detecting fake web-shops, creating their annual reports [14] and detecting low-content websites in the zone. Using recent labeled data gathered by DNS Belgium seems relevant, especially for the legitimate websites class. When labeling data for their annual report, DNS Belgium defines multiple categories of websites. Taking a sample of websites in each category could help to have a small but representative subset of the .be zone for the legitimate websites.

Gathering examples of legitimate websites is probably the easiest part of creating a labeled dataset. Collecting fake web-shops is more challenging as DNS Belgium took down most known ones, and the unknown are unknown by definition. We consider two primary sources for fake web-shops. First, the existing classifier of DNS Belgium yielded a few suspicious web-shops when they ran on the full crawl of March 2022. Fake web-shops identified by the classifier can be used as the few first training instances for our active learner. We also consider other examples that the organization is still interested in discovering in the zone.

Second, we draw inspiration from previous research (Carpineto and Romano [10], Wadleigh et al. [45]) and use search engine results to find new fake web-shops in the .be zone. To this end, we created a list of 12 popular shoe brands and used each brand for two queries: one with a "gray" keyword (in our case, "cheap") and one with a "complicit" keyword (in our case "replica"). We added the keyword "shoes" for every query and limited the results to websites in the .be zone. This is what our queries looked like: `nike cheap shoes site:.be`. Those queries were submitted to Google and Bing using a fresh install of LibreWolf [28] (a fork of Firefox sending standard HTTP headers in the requests). We connected to Google and Bing using residential Internet access to get similar results to what an average Belgian citizen would get. The first ten result pages were taken into account for each query: every link on those pages was saved, including paid links. After each query, the browser was closed and reopened, clearing the browser's cache and cookies to avoid tracking by search engines across visits.

From all the results, we only keep the domains in the .be zone. When results point to sub-domains, we add the corresponding domain in our list. This list of domains is then manually reviewed to label each instance based on the content crawled by DNS Belgium.

Once the first dataset is constructed with legitimate websites and fake web-shops, we iteratively train an active learner, allowing it to query 20 instances per iteration. A human annotator then labels those 20 instances before running the next iteration.

As we do not know in advance which query selection algorithm will perform best, we decide to use both *Top K* and *K Cluster Medoid*. One iteration out of two, *Top K* will be used to select the instances to label. For the other half of the iterations, *K Cluster Medoid* will be used. For the *K Cluster Medoid*, we select the 500 most uncertain instances and cluster them in 20 clusters. With 500 instances clustered each time the *K Cluster Medoid* is used, the query selection algorithm should bring websites that are different enough to add relevant information to the labeled dataset while keeping the execution time reasonable.

The stopping criterion of this experiment will mainly depend on the time available for the training of the active learner and for the human annotator to label the instances. It will also depend on the relevance of the most probable fake web-shops detected by the active learner at each iteration.

Given the size of the pool, some complex features we proposed are not used for this experiment. The lexical diversity of the body text and the semantic similarity between the title and the domain name are computationally expensive to extract and, therefore, are not used for this experiment. Other features such as the Open Graph tags [17] and the technologies detected by Wappalyzer [2] are extracted for the pool and used in this experiment.

4.2.4 Expected results for the experiments

We briefly describe the results we expect from both experiments. The actual results are presented in the following chapters.

For the first experiment, we expect the three learners to start with exactly the same performance as they are provided with the same set of labeled data. The *random sampling* is expected to have poorer performances than *Top K* and *K Cluster Medoid*. We believe that the *K Cluster Medoid* will yield better results as the queries selected by this algorithm are more likely to bring more diverse information than the *Top K*. The active learners should all reach the same performance once they have all the labels for the data in the pool (i.e., they will be in the current DNS Belgium's setting, with the baseline dataset as the training set). Finally, we hope that active learners will reach the performance currently achieved by the existing classifier using less labeled data.

For the second experiment, we expect to find a few fake web-shops using the search engines. However, we are aware that our search queries are very specific and may not yield many relevant results. We also expect to be able to construct a reasonably sized dataset in order to train an active learner with the data gathered using the search engines. Using those labeled instances, the active learner is expected to improve its predictions over the first few iterations, reaching an acceptable level of accuracy with significantly fewer labeled instances than the few thousand currently present in our baseline dataset. As the active learner will be provided with only a few examples of fake web-shops, the first queries submitted to the oracle are likely to be near random. The queries provided to the human annotator are expected to be more informative as the iterations go on. We hope to find new types of fake web-shops using less data than the existing crawler.

Chapter 5

Results of the experiments

This chapter presents the results obtained during the two experiments described in the previous chapter. We highlight the important observations done when running the experiments and provide illustrations to understand better the general trends and some special cases worth mentioning.

5.1 First experiment: active learning over the baseline dataset

We start by analyzing the results of the first experiment. Recall that we used the baseline dataset for this experiment and trained three active learners, each using a different query strategy. During the experiment, we recorded several metrics about the model, which are presented in the following sub-sections.

For this experiment, our entire dataset contains 7831 instances, separated as follows: 1567 instances are kept in a holdout set that is never presented to the learners for training, the 6264 remaining instances are presented as unlabeled at first, and the labels are provided to the learners as they issue queries. Each learner is allowed to query 5 instances per iteration according to its own query strategy algorithm. With this methodology, all the learners have the same number of labeled instances in their training set. Still, the constituting instances of the sets are different because this depends on the query strategies. For the learner using the *K Cluster Medoid* query selection algorithm, the 50 most uncertain instances are clustered in 5 clusters to select the instances to query. The experiment starts with the same labeled dataset for all the learners: 11 legitimate websites and 4 fake web-shops. We ran the experiment until the pool was exhausted (i.e., less than 5 instances in the pool) and compared the learners on different aspects.

5.1.1 F1 scores of the learners

The metric used to report how well the learners perform is the F1 score. Multiple metrics are available to evaluate the performance of a model. In our setting, we leverage a grid search method to find the best hyper-parameters for our model. As the grid search can only optimize one metric, we choose the F1 as it combines the precision and the recall. Without going into the details, one could interpret the precision as an indication of the "purity" of the model: all the instances identified as positive by a very precise model will be positive; however, it might not find all the positive instances. On the other hand, a model with a very high recall will find all the positive cases but might also include some negative cases. Ideally, we want the model to have good precision and recall. The F1 score combines both metrics using a harmonic mean.

We measure the F1 score using the predictions made by each learner on the holdout set: a set of 1567 instances that is never used for training. After each iteration, we instruct

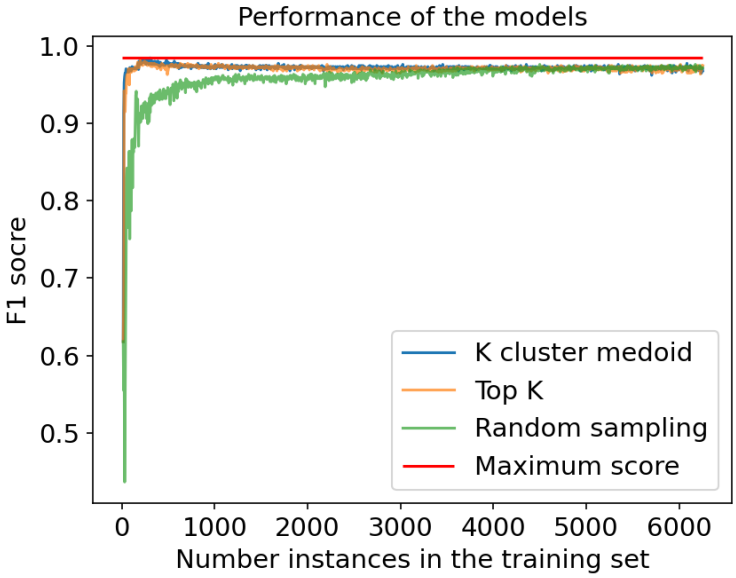


Figure 5.1: F1 score of the different learners as more instances are added to their training set. Both *Top K* and *K Cluster Medoid* perform better than *random sampling*.

each learner to make predictions for the holdout set and we save the F1 score.

Figure 5.1 reports the F1 scores of each learner as the iterations go on. The x-axis represents the number of instances currently labeled. All learners start with 15 instances in this set and 5 new instances are added at each iteration. The y-axis represents the F1 score of the model as a function of the number of instances in the labeled dataset. The blue, orange and green lines each represent a different learner (with a different query selection algorithm). The algorithm used by the learner is specified in the legend. In addition to the three learners, we depicted, in red, the best F1 score ever achieved by one of the learners.

As expected, all three learners start with the same F1 score and have a similar F1 score once the pool is exhausted. It is interesting to note that the performance starts to decrease for both *Top K* and *K Cluster Medoid* once the best F1 has been reached.

The three learners benefit, in the beginning, from the new instances added to their labeled dataset. The performance of the three models rapidly starts to increase when new instances are added. After a certain amount of instances is added to the labeled dataset, the performance decrease for every learner. All the learners do not reach their best performance at the same time.

Table 5.1 summarizes the best F1 score achieved by each learner, along with the number of labeled instances needed to reach the score. As observed in figure 5.1, the *random sampling* takes much longer to achieve its best score, and this score is the lowest of all three. Almost the entire pool is needed for this score to be achieved. On the other hand, *Top K* and *K Cluster Medoid* achieve their best score much faster. With only 200 labeled instances, the *Top K* learner reaches its best score while the *K Cluster Medoid* learner needs 305 instances to reach its best score, which is somewhat higher than the *Top K* learner.

The score of the *random sampling* learner is aligned with the score of the existing

Learner	Best F1	Number of instances
Random sampling	0.976	6175
Top K	0.982	200
K Cluster Medoid	0.985	305

Table 5.1: The *Top K* and *K Cluster Medoid* learners achieve a better F1 score than the *random sampling*. They also achieve their best score with significantly fewer labeled instances.

classifier used by DNS Belgium (presented in table 3.3). On the baseline dataset, the current classifier scored a F1 of 0.972, and the *random sampling* learner scored a F1 of 0.976 with slightly fewer instances needed. While this difference between the two scores is relatively low, this indicates that the active learning approach with the most naive query selection algorithm is at least as good as the existing solution.

Another interesting point to highlight on the graph in figure 5.1 is the point where the *random sampling* is at least as good as the other learners. For the *random sampling* to be at least as good as one of the other learners, it takes 2675 instances in the labeled dataset. For the *random sampling* to be at least as good as both of the other learners, it takes 3757 instances in the labeled dataset. Both intersections are located where the *random sampling* learner still improves its score after each iteration, and the other learners have a decreasing score as the iterations go.

The evolution of the F1 score for the three learners is particularly interesting when the first few hundreds of labeled instances are added to the training set. Figure 5.2 depicts the F1 scores of the three learners for the first 150 iterations (time needed to achieve a labeled dataset of around 750 instances). In the very first iterations, the *random sampling* learner has sudden drops in performance while the other two steadily increase their performance. The *K Cluster Medoid* is slightly faster in getting better performance at the beginning than the *Top K*. However, after around 150 instances in the labeled dataset their performances are similar. One can also notice the performance gap between the *random sampling* learner and the two others, even with 750 instances in the labeled dataset.

5.1.2 Percentage of fake web-shops in the labeled dataset

Another metric collected during this first experiment is the percentage of fake web-shops in the labeled dataset of each learner. This enables us to understand how balanced the datasets created by the three learners are during the experiment. Generally speaking, in machine learning, imbalanced datasets can lead to issues if not handled properly. Multiple techniques have already been explored to prevent dataset imbalance. For this reason, we are quite interested to understand how an active learner can deal with an imbalanced dataset: in our case, only 1843 instances are fake web-shops out of 7831 instances in total.

Figure 5.3 shows the evolution of the percentage of fake web-shops in the labeled dataset during the experiment. On the global trend depicted in sub-figure 5.3a, both *Top K* and *K Cluster Medoid* reach a balanced dataset in the early iterations before getting an imbalanced dataset. As the iterations go, the dataset tends to be imbalanced mainly because a limited number of instances is available in the pool and the choice becomes limited. Unlike the other learners, the *random sampling* learner never gets a balanced dataset. Instead, it keeps a percentage of fake web-shops in the labeled dataset that is

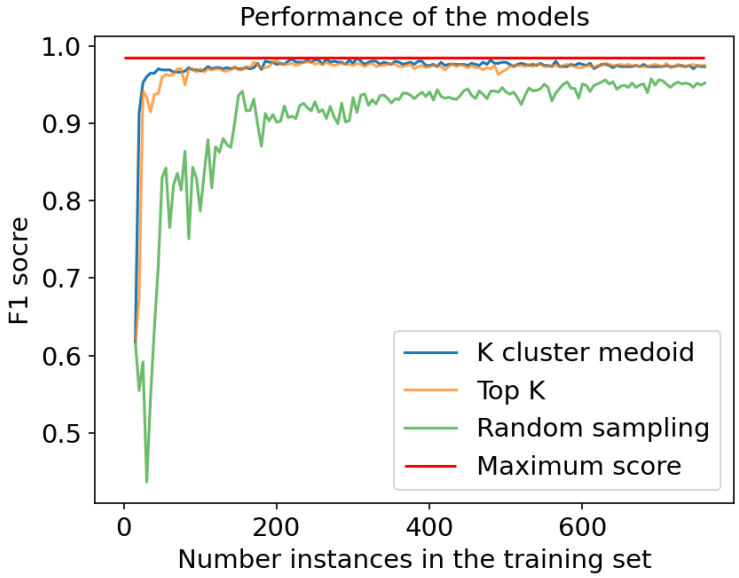


Figure 5.2: F1 score of the different learners as more instances are added to their training set. The graph is focused on the first 150 iterations to highlight how the learners perform with only a few instances in their labeled dataset.

close to the percentage of fake web-shops in the labeled dataset. This is not surprising considering how the instances are added to the labeled dataset for this learner.

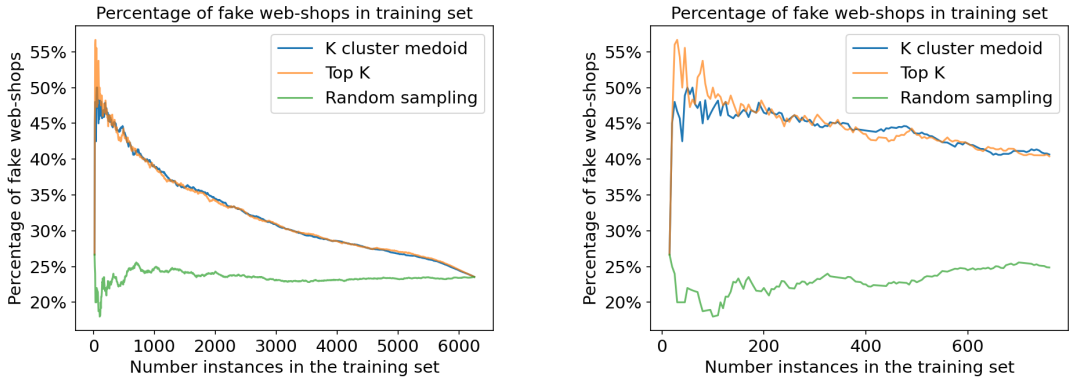
As the most interesting iterations are the few first hundreds, sub-figure 5.3b depicts the evolution of the percentage of fake web-shops in the labeled dataset for the 150 first iterations (which leads to approximately 750 instances in the labeled dataset). As shown in table 5.1, *Top K* and *K Cluster Medoid* reach their best F1 score at, respectively, 200 and 305 labeled instances. When this score is reached, both learners have between 45% and 50% of fake web-shops in their labeled dataset, which is almost a perfectly balanced dataset.

The more imbalanced the labeled dataset, the worst the F1 score is. This is also probably because the model tends to overfit on some clusters that are present in the pool because it has not much choice when the pool is nearly exhausted.

5.1.3 Number of features

While the number of features provided as input to the pre-processing pipeline is constant during the entire experiment, the number of features generated by the pipeline can vary. Parts of the pipeline transform texts into TF-IDF vectors, and those vectors can have different sizes at each iteration. For this reason, we recorded the number of features after pre-processing at each iteration.

Without any surprise, the number of features increased sharply in the first iterations, as depicted in figure 5.4. The number of features always drops when the number of instances reaches a multiple of one thousand. This is due to the configuration of the TF-IDF transformers used in the pipeline. As mentioned in sub-section 3.3.1, the TF-IDF transformers for the body text, the meta text and the title only keep words that appear in



(a) Percentage of fake web-shops in the labeled dataset of the learners across all the iterations.

(b) Percentage of fake web-shops in the labeled dataset of the learners for the first 150 iterations.

Figure 5.3: Both *Top K* and *K Cluster Medoid* manage to create a balanced labeled dataset. In contrast, the percentage of fake web-shops in the labeled dataset for the *random sampling* learner is close to the percentage of fake web-shops in the baseline dataset.

at least 0.1% and at most 90% of the documents. When the labeled dataset reaches one thousand instances, words appearing in exactly one document, which were kept up to now, are dropped by the transformer as their frequency drops below the 0.1% threshold. The same applies to the other multiples of one thousand.

Another TF-IDF transformer is used to represent the hostnames found in the links on the web pages. However, this transformer does not have lower and upper thresholds for the frequency of the terms.

The *Top K* and *K Cluster Medoid* learners always have more features after pre-processing than the *random sampling*. As already mentioned, this difference in the number of features is only due to the TF-IDF transformers used in the pre-processing pipeline. This difference comes from the diversity in the body text, meta text and title, and the number of external hostnames found on the web page. In fact, all those attributes are transformed using TF-IDF.

5.1.4 Time complexity

In addition to the three metrics already presented, we also recorded the time needed for each learner at each iteration to train the model and to select the instances to label. Those measurements are depicted in figure 5.5. While the absolute time needed to train the models and choose the new instances to label is not very relevant as it will mostly depend on the hardware used, the relative values are interesting.

The *random sampling* learner is generally faster than the other two (sub-figure 5.5a). Both the *Top K* and the *K Cluster Medoid* learners have similar training times during the entire experiment. The three learners end up with roughly the same training time at the end of the experiment because they will ultimately have the same set of labeled instances.

When we analyze the time needed to choose the instances to be labeled by the oracle (sub-figure 5.5b), the *random sampling* has a constant time across the iterations. Indeed, choosing five random numbers is a constant time operation with regard to the number of labeled instances in the dataset. The *Top K* and the *K Cluster Medoid* learners show a

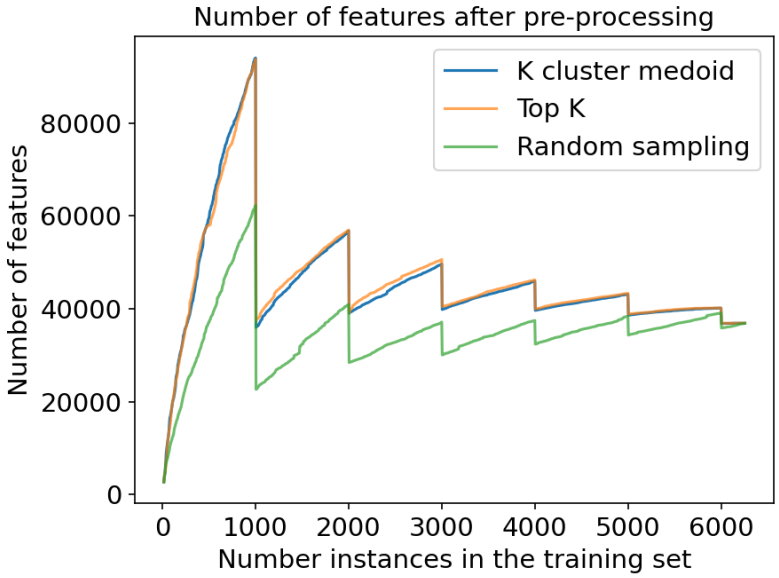


Figure 5.4: The number of features after pre-processing varies across iterations. Those variations are due to the TF-IDF transformers included in the pre-processing pipeline.

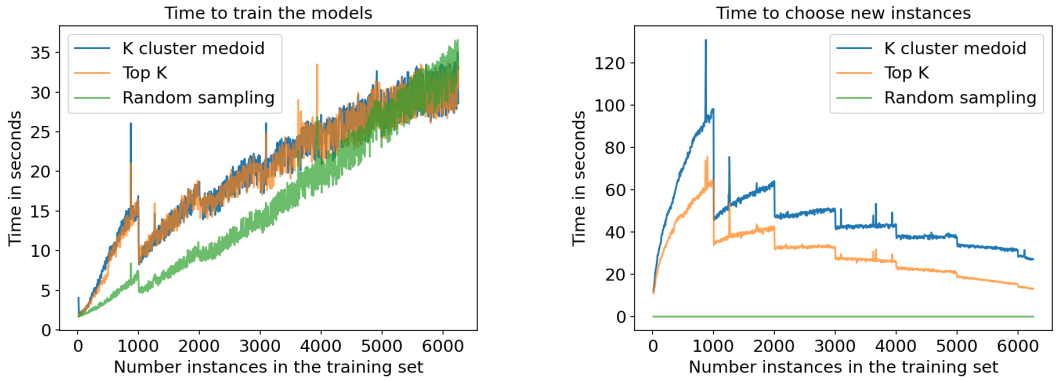
similar pattern: first, the time needed for the operation increases, just like the number of features, then it tends to decrease over time. Recall that for those two learners to choose the next instances to label, they evaluate the predictions for the entire pool. The more complex the instances are (i.e., the more features after pre-processing they have), the longer it takes to run the predictions. The time needed to run the predictions is also dependent on the number of instances to predict: as the iterations go, the number of remaining instances in the pool decreases. Those two effects are visible on the graph (sub-figure 5.5b). The *K Cluster Medoid* query selection algorithm is always slower than the *Top K* for the entire experiment. Not only is it needed to evaluate the predictions for the whole pool, but the learner also has to cluster the 50 most informative instances. Because of the clustering step, this query selection algorithm is slower.

On both sub-figures (5.5a and 5.5b), the impact of the number of features after pre-processing is quite visible, especially around 1000 instances in the labeled dataset. Please note that sporadic peaks are likely due to other activity on the machine and should be considered outliers.

5.2 Second experiment: active learning over the entire .be zone

We now focus on the results obtained during the second experiment we described. As a reminder, this experiment can be summarized by the three following steps.

1. Use Bing and Google to find new fake web-shops in the search results.
2. Construct the first dataset using websites found in the previous step, in the data labeled by DNS Belgium and in the predictions of their existing classifier.



(a) Time needed for each learner to train the model at each iteration.

(b) Time needed for each learner to select five instances to submit to the oracle at each iteration.

Figure 5.5: Time needed for the training and the query selection algorithm of each learner at each iteration.

3. Apply active learning for multiple iterations.

The following sub-sections present the results for each step of the experiment.

5.2.1 Discovering new fake web-shops via search engine results

To find new fake web-shops, we drew inspiration from previous authors (Carpineto and Romano [10], Wadleigh et al. [45]) and used search engines to discover more of those malicious websites. Using 12 brands of shoes, we created 24 queries with either a "gray" or a "complicit" keyword to obtain interesting results. Those 24 queries were executed on both Bing and Google. The 10 first pages of results were saved, but for some queries the search engines did not yield 10 pages.

One of the first observations is that Bing generally shows more results than Google for the same query. However, results tend to be irrelevant after the first few pages. As an example, Bing sometimes showed a link to the home page of Google Search for some of the queries. Because of this, many websites collected are not even related to web-shops.

From all the results gathered, we extracted the domain name. If the domain name is a sub-domain (e.g., `shop.example.be`), we also add the main domain in our results. This is done because DNS Belgium only crawls the main domain and the `www` sub-domain. We remove the domains that are not in the .be zone, as some appeared in the sponsored links. We also remove the domains corresponding to the search engine used to launch the queries.

This resulted in a list of 338 unique domains gathered from the search results. From those domains, some were not crawled by DNS Belgium's crawler as they are sub-domains. In total, 276 domains were crawled. The crawled domains were divided into legitimate web-shops, not web-shops, and suspicious web-shops. Table 5.2 summarizes the number of results found for each category. Most of the results are not web-shops due to Bing results being irrelevant after a few pages.

Most of the websites in the *suspicious web-shop* category are more suspicious than fake: they display items on sale, but there is little to no information about the website owner. When classifying the websites, we mainly looked at the general layout, the legal details

Category	Number of results
Legitimate web-shop	114
Not web-shop	149
Suspicious web-shop	13

Table 5.2: Number of results crawled by DNS Belgium for each category.

about the owner of the website and the presence of (popular) accounts on social media. For most of the suspicious web-shops, the layout is not suspicious but we could not find information about the owner of the online store. Some show an error when trying to read the terms and conditions, others do not even display a phone number or email address. Those websites can be described as websites where an informed online shopper would not buy anything with peace of mind.

5.2.2 Creating a labeled dataset

In order to create the labeled dataset that will be used for the first iteration of the active learner, we needed instances from both classes: legitimate websites representing the .be zone and fake web-shops. To get examples for the first category, we used the labeled websites that DNS Belgium used for their annual report for 2021. DNS Belgium defines 28 categories, and we selected two random websites from each category. In addition to those websites, we also added some results from the search engines. As those instances were manually reviewed, we could spot interesting edge cases that we wanted to include in our dataset as legitimate websites. Those edge cases are mainly web-shops that are poorly designed but not malicious. We added 16 of those in the labeled dataset. Up to now our dataset contains 72 examples of legitimate websites from the .be zone.

To add malicious web-shops, we used three sources of labeled websites: the existing dataset of DNS Belgium, the predictions of the existing classifier and the results of the search engines. In total, 15 examples of fake web-shops were manually selected among those three sources. We made sure that the examples were diverse enough not to have obvious clusters in the fake web-shops, leading the model to mainly discover those.

Our dataset is now composed of 87 websites: 72 legitimate and 15 fake web-shops. Most of the time needed to construct this dataset (once the data from the search engine was gathered) has been spent on selecting the fake web-shops to include. As for the first experiment, the dataset is quite unbalanced: only 17% of the instances belong to the class we are interested in.

5.2.3 Iterative training of the active learner

Starting with the dataset we described, we trained an active learner iteratively. After each iteration, the learner submits 20 queries to be labeled by a human annotator. As already mentioned, we do not know beforehand what the best query strategy is in our setting. For this reason, when the iteration number is odd, the learner uses the *Top K* strategy, and when the iteration number is even, it uses the *K Cluster Medoid* strategy.

The difference between the queries submitted by the two query strategies was noticeable

when labeling the instances. When the *Top K* query strategy was used, multiple similar websites were often submitted to the oracle. Those websites were mainly fake web-shops as the classifier predicted fake web-shops with a pretty low probability. In some iterations, the most probable fake web-shops only had a probability of around 0.6. However, the most probable fake web-shops were still relevant. The classifier progressively predicted fake web-shops similar to those already labeled with a higher probability in the few last iterations. As the *Top K* query strategy mainly asks for labels for (similar) fake web-shops, those are often very well predicted in the next iterations.

When the *K Cluster Medoid* query strategy was used, the instances were more diverse and there were often fewer fake web-shops queried than with the *Top K* strategy. Multiple times, the queries issued using the *K Cluster Medoid* strategy were edge cases created by the *Top K* strategy: some legitimate websites looking like fake web-shops labeled in previous iterations were getting a probability closer to 0.5. They were queried by the learner using the *K Cluster Medoid* strategy.

Overall, the combination of the two query strategies allowed the learner to be confident about fake web-shops queried by *Top K* and diverse enough, thanks to the queries issued by *K Cluster Medoid*.

One of the interesting aspects of the query strategies used is that they issue a batch of queries. This enables the human annotator to skip some instances when they are harder to label. Not labeling some instances about which the oracle is unsure avoids mislabeling them. In the next iterations, the instances that were skipped are still in the pool and might be submitted to the oracle in a future iteration.

In total, we ran 15 iterations of the active learner. At each iteration, 20 instances were presented to the annotator. The newly labeled instances were added to the labeled dataset and removed from the pool before starting the next iteration.

Table 5.3 gives an overview of how many instances were labeled after each iteration. Recall that odd iterations used the *Top K* query selection algorithm while even iterations used *K Cluster Medoid*. This can be observed in the number of fake web-shops labeled after each iteration. When the number of websites added to the labeled dataset is lower than 20, this is because some of the queries of the active learner were not answered. During the 11th iteration, we discovered a cluster of fake web-shops that the classifier did not pick up. We labeled a few instances of the cluster and added them to the labeled dataset. This explains why more than 20 instances were labeled after the 11th iteration.

Figure 5.6 shows the evolution of the percentage of fake web-shops in the labeled dataset during the experiment. While we never reach a balanced dataset, we can see that the percentage of fake web-shops tends to increase. The learner quickly found the interesting areas to query: most of the websites submitted to the annotator were web-shops or looked similar to some fake web-shops.

Recall that both query strategies used in this experiment need to evaluate the predictions for the entire pool before deciding which instances to submit to the oracle. As those predictions were made at each iteration, we reviewed them to find the fake web-shops detected by each version of the active learner. After running the 15 iterations, we manually inspected the highest-scoring instances and labeled them. In total, we found 50 more fake web-shops in the predictions.

	Number of websites added	
Iteration number	**Fake web-shop**	**Legitimate website**
1	1	19
2	0	19
3	10	9
4	0	19
5	6	8
6	2	15
7	2	18
8	1	19
9	14	5
10	1	18
11	26	6
12	1	15
13	18	2
14	1	18
15	4	14

Table 5.3: Number of websites labeled after each iteration of the active learner.

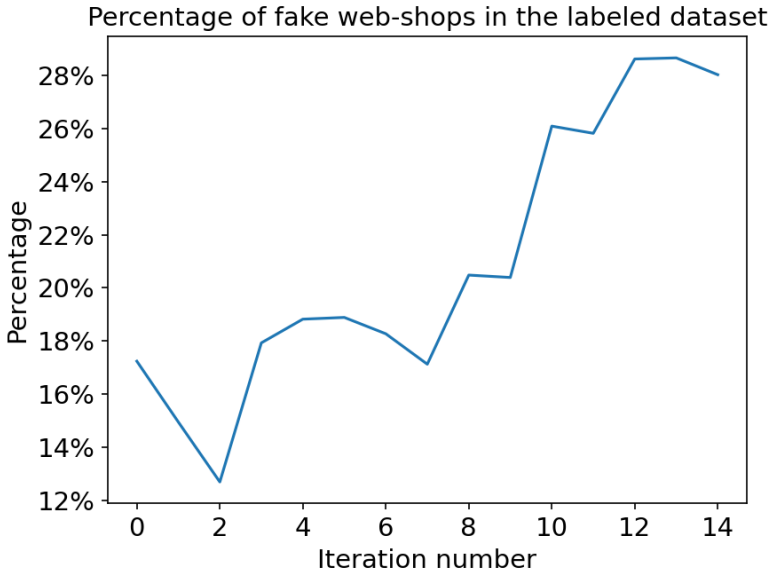


Figure 5.6: Percentage of fake web-shops in the labeled dataset across iterations.

To evaluate if a website is a fake web-shop, we look at multiple hints. First, we take into account the general design of the website and the look and feel. Next, we look for terms and conditions, cookies policies and other legal information to find the identity of the owner of the shop. We mainly look for a physical address, a phone number, an email address and a VAT number. When all those elements can be found, the website is generally honest. If some are missing, we try to find links to their social media accounts and see how popular they are and how people engage. If needed, we also inspect the registration information provided by the registrant when buying the domain. When it is harder to tell between a fake web-shops and a poorly designed (or abandoned) web-shop, we do not label the instance.

During the labeling process, we found multiple clusters of fake web-shops. Some fake web-shops look like usual online stores where it is possible to buy goods. Some others do not sell anything and do not even mention products. Those are generally websites hosting content where hidden links were added to the HTML source code. Listing 2 shows an example of such links. Those links are often found at the end of the HTML source code, sometimes after many blank lines and on a single line to make them harder to spot. We think that those links are part of a search engine optimization campaign to improve the ranking of the websites they point to. In fact, most search engines use the number of back-links (links pointing to the page they are currently indexing) to evaluate the popularity of the page. While the links are not visible to the user due to the position of the `div` element, the spiders of search engines take them into account as they are in the source code. During the experiment, we saw other techniques designed to hide the links: some websites use JavaScript code to hide the links from the user, some include them in very small surrounding HTML tags, and some use CSS style to make the `div` element invisible.

Granted, simple pages with links hidden in the source code are not really what a web-shop usually looks like. However, those pages are probably used by fake web-shops operators to increase their ranking in search engines. For this reason, we classify those websites as fake web-shops as well as they serve their purposes in a certain way.

In total, 117 new fake web-shops were labeled during the iterations because the active learner issued queries for those, including the 15 fake web-shops that were part of the initial labeled dataset. 50 more have been discovered in the predictions of the different versions of the active learners. Among those 167 malicious web-shops, 132 were not found by the existing classifier of DNS Belgium. Those websites were provided to DNS Belgium for them to review and take action if needed.

In addition to those web-shops that we evaluated as malicious, we also provided DNS Belgium with 39 websites that looked suspicious during the labeling process but not enough to classify as fake. Those instances are mainly part of the queries issued by the active learners during the iterations that we did not label. We also provided the dataset that we built using the active learner and the dataset created from the search engine results, enabling the organization to use this data as ground truth for future classifiers.

```
<div style="position: absolute; top: -822px;left: -822px;">
<a href="http://www.nikedunksales.com/nikesbdunklow-c-10.html">nike dunk
↪  low</a>
<a href="http://www.shoesretails.com/tory-burch-boots-c-494.html">tory
↪  burch boots</a>
<a href="http://www.nikedunkshow.com/">Nike SB Dunk</a>
<a href="http://www.airforce1fashion.com/air-force-1-classic-mid-c-238.htm
↪  l">Air Force 1 Classic Mid</a>
<a href="http://www.toplacoste.com/">Lacoste Shoes Men</a>
<a href="http://www.toplacoste.com/">Lacoste Shoes Online</a>
<a href="http://www.nikedunkshow.com/">Nike Dunk Shoes</a>
<a href="http://www.christianlouboutinkick.com/">christian louboutin
↪  shoes</a>
<a href="http://www.frchristianlouboutin.com/christian-louboutin-ankle-boo
↪  ts-c-1.html">christian louboutin boots</a>
<a href="http://www.nikeairmaxsite.com/">Nike Air Max</a>
<a href="http://www.christianlouboutinkick.com/christian-louboutin-pumps-c
↪  -17.html">christian louboutin pumps</a>
</div>
```

Listing 2: Example of links hidden in the HTML source code of legitimate-looking sites. The position of the `div` tag is very negative such that the links are rendered outside of the screen.

Chapter 6

Discussion

We now take a critical look at the results presented in the previous chapter. We present the limitations of the experiments as well as some lessons learned. For each experiment, we explain different aspects that should be considered when interpreting the results.

6.1 First experiment

Given the encouraging results encountered during the literature review and the knowledge about DNS Belgium's dataset, we were confident enough about this first experiment. In fact, we already knew that the dataset contained some clusters of similar websites, enabling an active learner to learn from one member in the cluster and accurately predict the class of the other members of the cluster. Moreover, the results obtained using the existing classifier were decent on the baseline dataset, suggesting that an active learner would achieve at least the same performance.

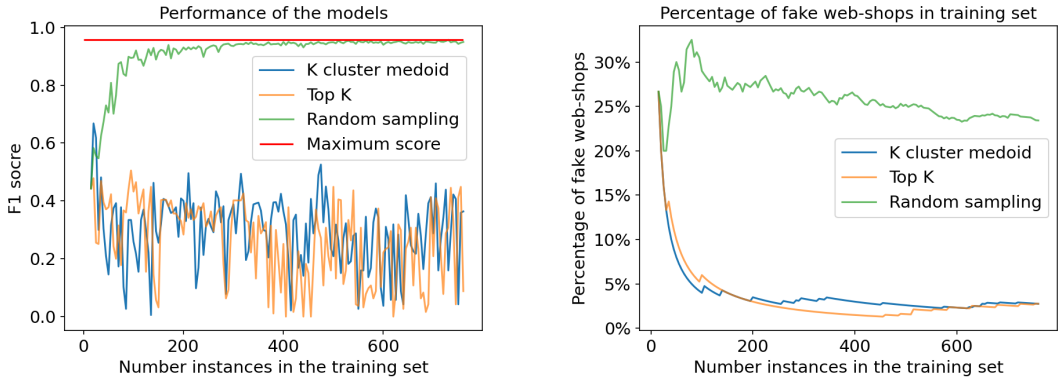
Even though the baseline dataset contains some biases, it is interesting that the performance of the active learners is better with very few instances (around 200-300) than with the entire dataset. These results may indicate that, in this case, it is best to prefer quality over quantity when it comes to the instances we provide the learners with. A few labeled instances that bring diverse information to the model are preferable over a more extensive set of instances that do not bring much more information.

6.1.1 Limitations of validity

While this experiment supports the fact that active learning may help in our setting, it does not prove its usability in practice. The dataset used here is not representative of the entire .be zone or the population of web-shops in the .be zone. Moreover, the proportion of fake web-shops (1843 instances out of 7831) is not in line with the actual proportion of fake web-shops in the entire .be zone. This experiment can be understood as a confirmation that such a large dataset is not needed if the instances are carefully chosen.

For this experiment to be more representative of an actual use case, one should first draw a random subset of the entire .be zone and label those instances. Next, the experiment can be repeated as we described. However, this setting is much more time-consuming as a significantly more extensive dataset will be needed to have a sufficient amount of fake web-shops labeled as such.

This experiment can be seen as a necessary condition for the second experiment: if active learning cannot help in a biased setting, then how can it help with the entire dataset? Again, the fact that the experiment yielded interesting results does not indicate that active learning will have good performance in an actual application. However, an apparent failure would have suggested that it was not worth launching the second experiment. The main



(a) F1 score of the different learners as more instances are added in their training set.

(b) Percentage of fake web-shops in the labeled dataset of the learners.

Figure 6.1: Overview of the performance of the three active learners when instances that the model is the most certain about are queried by *Top K* and *K Cluster Medoid*. Only the first 150 iterations are depicted.

advantage of this experiment is the relatively low cost of human resources. Apart from designing the global architecture and selecting an adequate labeled dataset, the experiment is completely automated.

6.1.2 Lessons learned

During the first global architecture design, a programming mistake was made, resulting in the instances being sorted with the most certain at the top. With this mistake, the *Top K* and *K Cluster Medoid* issued queries for instances they were the most certain about (i.e., with a probability for one class close to 1). The performance for those learners quickly dropped after a few iterations, while the *random sampling* learner kept improving its performance. Almost only legitimate websites were added to the labeled dataset, resulting in a very imbalanced dataset for the *Top K* and *K Cluster Medoid* learners. Figure 6.1 gives an overview of the performance.

Once this mistake was detected, we made a connection with what DNS Belgium is currently doing with fake web-shops detected by the model. When fake web-shops are detected (i.e., their probability is close to 1 for the fake web-shop class), they are added to the labeled dataset. While our flawed design led *Top K* and *K Cluster Medoid* to add both fake web-shops and legitimate websites to the labeled dataset (as long as they had a high probability for one of the classes), DNS Belgium mainly adds fake web-shops that are correctly predicted. In the DNS Belgium's setting, this may help reduce the dataset imbalance but does not necessarily help improve the quality of the predictions. The instances that were already confidently predicted by the model will still be, and instances that are less similar but still interesting to discover will probably not be detected.

In DNS Belgium's setting, it is probably advisable to keep a set of known fake web-shops and only a subset that is actually used for training the models. This could prevent clusters from forming in the labeled dataset and likely prevent the model from over-fitting on those clusters.

6.2 Second experiment

Despite the positive results from the first experiment, we were unsure about the performance to expect from the active learner on the entire .be zone. The first experiment is based on a dataset that we think is easier to predict.

However, the first few iterations of this second experiment were pretty interesting: the active learner quickly started to issue queries for instances belonging to both classes. The probability given to the most probable fake web-shops during the first iterations was relatively low: most of the time, only five or six websites (out of 1.3 million) were labeled with a probability slightly above 0.5 as fake web-shops. The learner thus queried those most probable fake web-shops as they were close to the decision boundary. In this setting, we think that the combination of *Top K* and *K Cluster Medoid* was indicated as the two algorithms provided distinct queries. The first mainly issued queries for clusters very close to the decision boundary, most of the time fake web-shops. The second mainly issued queries for diverse instances, adding more diversity to the labeled dataset and allowing the learner to uncover new types of malicious websites.

Starting from 15 hand-picked examples of fake web-shops, discovered 152 more in 15 iterations. This indicates that this architecture might help create classifiers in a reasonable amount of time and enable users to find similar instances quickly. Moreover, the iterative process allows the human to remain in control of what data is added to the classifier.

6.2.1 Limitations of validity

One of our motivations for using active learning in our setting was that it was supposed to require less time from the human operator. While we did not measure the total time spent by the operator for this experiment, we believe that this might not always be true and should be nuanced. During the active learning part of the experiment (i.e., not taking into account the time needed to label the data from the search engine results), we manually labeled 532 websites and discovered 152 fake web-shops. To put things into perspective, during one of the recent executions of their existing classifier, DNS Belgium labeled 154 websites to find 21 fake web-shops. If we only consider the labeling time (and assume that each instance takes about the same time to be labeled), our solution led to 28.6% of the labeled instances being fake web-shops against 13.6% for DNS Belgium's last predictions.

However, this is only half of the story. Our approach requires at least 15 times more time for the classifier to run. As our process is iterative and we executed 15 iterations (similar to a single run of DNS Belgium's existing classifier), our architecture is much slower from this point of view. Moreover, the human has to work sporadically: after every iteration, 20 new instances to label are presented, while the existing classifier gives a list of probable fake web-shops after the first run.

6.2.2 Lessons learned

For completeness, we reviewed the predictions made by every version of the active learner to find fake web-shops once all the iterations were done. This happened to be a waste of resources. We started by reviewing the predictions of the last version of the learner (i.e.,

the one trained on the larger labeled dataset) and checked the predictions starting from the latest version to the first. In order to avoid labeling twice the same websites, already labeled websites were hidden from the list. The last four versions of the classifier identified all of the fake web-shops we found by reviewing all the predictions, making it useless to review the predictions of the other 11 versions.

While this was time-consuming, we learned that the most recent versions of the classifier are also the most interesting as they could identify the fake web-shops at least as good as the first versions.

Chapter 7

Future work

7.1 Learning from a subset of the .be zone

In this thesis, we trained and used a classifier on websites from the entire .be zone. While this avoids excluding websites (and maybe fake web-shops) from our dataset, this approach can be more complex and less efficient than learning only from web-shops. In order to learn only from web-shops, one could design a classifier to separate websites between web-shops and not web-shops and only use web-shops to detect fake web-shops.

This approach might be more suitable for finding malicious online stores that look like usual web-shops. However, during our research, we discovered websites that are likely to be part of the fake web-shops strategies to increase their ranking in search engine results but do not look like web-shops. Using a two-fold technique may not uncover such websites.

7.2 Using label propagation

In our setting, leveraging semi-supervised learning algorithms such as automatic label propagation could help create a more extensive set of labeled data and positively impact the classifier's performance. Unfortunately, due to the limited time available for this research, we did not implement such a solution. We identify two main prerequisites for label propagation to be helpful in our context.

Limited propagation. The label propagation algorithm should be able to only propagate labels to instances that are very likely to have the predicted label. Simple algorithms that propagate labels to all the unlabeled instances in the pool may yield poor results: labeling 1.3 million unlabeled instances from a few hundreds labeled instances may not give the expected results, especially in areas not well represented by the labeled instances. Therefore, the label propagation algorithm should be able to leave unlabeled instances, or the architecture should only select a subset of the pool to be labeled using label propagation.

Scalability. Currently, around 1.3 million websites are hosted using a .be domain. To be tractable, the label propagation algorithm should either have a low time and memory complexity or only be applied to a subset of the unlabeled pool.

Chapter 8

Conclusion

This thesis started with an introduction to the existing features used by DNS Belgium to detect fake web-shops in the .be zone. First, we found that fake web-shops are unlikely to show contact information, such as an email or telephone link, on their homepage. Similarly, they are not likely to have MX DNS records, used for mail exchange, configured. Second, as suggested by the literature, fake web-shops operators use drop-catching to register their domain names. Finally, meta information fields in HTML source code, like the meta keywords and the meta description, are extensively used by fake web-shops. This is probably part of their search engine optimization techniques.

We also reviewed the existing classifier used by DNS Belgium, its performance and the main challenges encountered while building this classifier. While the performance on the labeled dataset is promising, the predictions generated by the classifier when used on the entire .be zone contain many false positives. We hypothesized that the training set used with the existing classifier may contain biases and clusters and may not represent the entire .be nor the new techniques used by current fake web-shops operators.

Before designing a solution to overcome those issues, we proposed multiple new features inspired by the literature and ideas from DNS Belgium. We included features representing the number of Open Graph tags, mainly used by social media to integrate web pages as rich objects. As third-party integrations used by websites can give information about their business and trustworthiness, we added features to represent technologies present on the websites in our dataset. We capture the semantic distance between the HTML title and the words found in the domain name by using natural language processing. We incorporate features that represent the lexical diversity of the text found in the body of the page. Finally, we expressed some existing features capturing the syntactic similarity between the HTML title and the domain name as a fraction, dividing the feature by the length of the HTML title.

We then designed an active learning approach to improve the existing classifier and overcome some of the issues mentioned previously. Active learning allows the learning algorithm to ask for the label of unlabeled instances to improve the predictions. This allows to start with a smaller labeled dataset and iteratively train a classifier that can issue queries for specific instances, thus expanding the labeled dataset. As the active learner queries instances about which it is the most uncertain, the parts of the dataset that are less covered with labeled data are more likely to be queried, leading to a labeled dataset that is more representative of the .be zone.

Next, we presented two experiments to evaluate how active learning can improve classification in our setting. First, we evaluate how active learning performs on our baseline dataset, presenting the instances as unlabeled to the learner and providing labels only for the instances it queries. We ran this experiment with three different learners, each using a different strategy to choose the instances they wanted to query the label for. The best-performing learners achieved better performance than the existing classifier using sig-

nificantly less labeled data.

In the second experiment, we leverage active learning to detect fake web-shops in the entire .be zone. After creating the initial dataset with instances from the existing labeled data from DNS Belgium, results from search engines and predictions from the existing classifier, we iteratively trained 15 versions of the active learner. The active learner alternatively used the two best performing query strategies from the previous experiment. Starting with a dataset containing as few as 15 examples of fake web-shops, we were able to identify 152 more fake web-shops. Most of the fake web-shops we discovered during this experiment were instances queried by the active learner. The remaining were predicted with a high probability by the last versions of the active learner.

After presenting the results of both experiments, we put them into perspective and highlight some key points consider when interpreting the results. While active learning allowed us to discover new fake web-shops, it requires more computational resources as 15 iterations were needed in our experiment to achieve such results. Moreover, the human operator is not simply presented with a list of predictions at the end of the execution. Instead it has to label instances after each iteration of the algorithm.

Finally, we suggest new directions for future research in the domain, such as using a two-fold method, first separating web-shops from not web-shops and only detecting fake web-shops in the first set or leveraging semi-supervised learning with techniques like automatic label propagation.

Bibliography

[1] Alexa Internet, Inc. Alexa – Top sites. https://www.alexa.com/topsites, 2022. [Online; accessed 2022-04-05].

[2] Elbert Alias. Wappalyzer – Identify technology on websites. https://github.com/wappalyzer/wappalyzer, June 2021. Version 6.7.4.

[3] Sushma Nagesh Bannur, Lawrence K Saul, and Stefan Savage. Judging a site by its content: learning the textual, structural, and visual features of malicious web pages. In *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, pages 1–10, 2011. doi:10.1145/2046684.2046686.

[4] Senne Batsleer. The detection of fake webshops in the .be zone. Master's thesis, Katholieke Universiteit Leuven, 2021. URL https://assets.dnsbelgium.be/attachment/final-ThesisMAI_SenneBatsleer.pdf.

[5] Eric B Baum and Kenneth Lang. Query learning can work poorly when a human oracle is used. In *International Joint Conference on Neural Networks*, volume 8, page 8. Beijing China, 1992.

[6] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded Up Robust Features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg, 2006. Springer. ISBN 9783540338338. doi:10.1007/11744023_32.

[7] BeCommerce. BeCommerce Market Monitor: Belg spendeerde in 2021 12,1 miljard euro in e-commerce. https://www.becommerce.be/en_US/becommerce-market-monitor-belg-spendeerde-in-2021-121-miljard-euro-in-e-commerce, March 2022. [Online; accessed 2022-05-18].

[8] Jessa Bekker and Jesse Davis. Learning from positive and unlabeled data: A survey. *Machine Learning*, 109(4):719–760, 2020. doi:10.1007/s10994-020-05877-5.

[9] Louise Beltzung, Andrew Lindley, Olivia Dinica, Nadin Hermann, and Raphaela Lindner. Real-time detection of fake-shops through machine learning. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 2254–2263. IEEE, 2020. doi:10.1109/BigData50022.2020.9378204.

[10] Claudio Carpineto and Giovanni Romano. Learning to detect and measure fake ecommerce websites in search-engine results. In *Proceedings of the international conference on web intelligence*, pages 403–410, 2017. doi:10.1145/3106426.3106441.

[11] Mick Cox and Sjors Haanen. Content-based classification of fraudulent webshops. Master's thesis, University of Amsterdam, 2018. URL https://www.sidnlabs.nl/downloads/o1B7l2nASJeOl-DvZ5flqg/46c60d5373e83c4e242adfa78110be87/Whitepaper_Content-based_Class._of_Fraudulent_Webshops.pdf.

[12] Leslie Daigle. WHOIS Protocol Specification. Request for Comments RFC 3912, Internet Engineering Task Force, September 2004. URL https://datatracker.ietf.org/doc/rfc3912/.

[13] DNS Belgium. Prevention. https://www.dnsbelgium.be/en/internet-security/prevention. [Online; accessed 2022-05-18].

[14] DNS Belgium. Annual reports. https://www.dnsbelgium.be/en/about-dns-belgium/annual-reports, 2022. [Online; accessed 2022-05-07].

[15] European Commission. Counterfeit and Piracy Watch List. Commission Staff Working Document SWD(2020) 360 final, European Commission, Brussels, December 2020. URL https://trade.ec.europa.eu/doclib/docs/2020/december/tradoc_159183.pdf.

[16] Explosion. spaCy – Industrial-strength Natural Language Processing in Python. https://github.com/explosion/spaCy, March 2022. Version 3.2.4.

[17] Facebook. The Open Graph protocol. https://ogp.me/, 2022. [Online; accessed 2022-04-22].

[18] Google. SEO Starter Guide: The Basics | Google Search Central | Documentation | Google Developers. https://developers.google.com/search/docs/beginner/seo-starter-guide, 2022. [Online; accessed 2022-04-01].

[19] Ram D Gopal, Afrouz Hojati, and Raymond A Patterson. Analysis of third-party request structures to detect fraudulent websites. *Decision Support Systems*, 154:113698, 2022. doi:10.1016/j.dss.2021.113698.

[20] Yuhong Guo and Dale Schuurmans. Discriminative batch mode active learning. *Advances in neural information processing systems*, 20, 2007. URL https://proceedings.neurips.cc/paper/2007/file/ccc0aa1b81bf81e16c676ddb977c5881-Paper.pdf.

[21] Shuang Hao, Matthew Thomas, Vern Paxson, Nick Feamster, Christian Kreibich, Chris Grier, and Scott Hollenbeck. Understanding the domain registration behavior of spammers. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 63–76. Association for Computing Machinery, 2013. doi:10.1145/2504730.2504753.

[22] ICANN. WHOIS Data and Accuracy. https://www.icann.org/resources/pages/whois-data-accuracy-2017-06-20-en. [Online; accessed 2022-05-18].

[23] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009. doi:10.1002/9780470316801.

[24] Hassan B Kazemian and Shafi Ahmed. Comparisons of machine learning techniques for detecting malicious webpages. *Expert Systems with Applications*, 42(3):1166–1177, 2015. doi:10.1016/j.eswa.2014.08.046.

[25] Ross D King, Kenneth E Whelan, Ffion M Jones, Philip GK Reiser, Christopher H Bryant, Stephen H Muggleton, Douglas B Kell, and Stephen G Oliver. Functional

genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427 (6971):247–252, 2004. doi:10.1038/nature02236.

[26] David D Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Machine learning proceedings 1994*, pages 148–156. Elsevier, 1994. doi:10.1016/B978-1-55860-335-6.50026-X.

[27] David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *SIGIR '94*, pages 3–12. Springer, 1994. doi:10.1007/978-1-4471-2099-5_1.

[28] LibreWolf. Librewolf. `https://librewolf.net`, 2022. Version 99.0.1-4.

[29] Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991. doi:10.1109/18.61115.

[30] Michael Lindenbaum, Shaul Markovitch, and Dmitry Rusakov. Selective sampling for nearest neighbor classifiers. *Machine learning*, 54(2):125–152, 2004. doi:10.1023/B:MACH.0000011805.60520.fe.

[31] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004. ISSN 0920-5691. doi:10.1023/B:VISI.0000029664.99615.94. URL `http://link.springer.com/10.1023/B:VISI.0000029664.99615.94`.

[32] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. doi:10.48550/arXiv.1301.3781.

[33] Wouter Mostard, Bastiaan Zijlema, and Marco Wiering. Combining visual and contextual information for fraudulent online store classification. In *IEEE/WIC/ACM International Conference on Web Intelligence*, pages 84–90, 2019. doi:10.1145/3350546.3352504.

[34] Nginx. Nginx. `https://nginx.org/en`, 2022.

[35] Daphne Odekerken and Floris Bex. Towards transparent human-in-the-loop classification of fraudulent web shops. In *Legal Knowledge and Information Systems*, pages 239–242. IOS Press, 2020. doi:10.3233/FAIA200873.

[36] Scikit-Learn. scikit-learn: machine learning in Python — scikit-learn 1.1.0 documentation. `https://scikit-learn.org/stable/`. [Online; accessed 2022-05-18].

[37] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison Department of Computer Sciences, January 2009. `https://digital.library.wisc.edu/1793/60660`.

[38] Burr Settles. From theories to queries: Active learning in practice. In *Active learning and experimental design workshop in conjunction with AISTATS 2010*, pages 1–18. JMLR Workshop and Conference Proceedings, 2011. `https://proceedings.mlr.press/v16/settles11a`.

[39] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948. doi:10.1002/j.1538-7305.1948.tb01338.x.

[40] Xuehua Shen and ChengXiang Zhai. Active feedback in ad hoc information retrieval. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 59–66, 2005. doi:10.1145/1076034.1076047.

[41] Peter M. Stahl. Lingua – The most accurate natural language detection library for Java and the JVM, suitable for long and short text alike. `https://github.com/pemistahl/lingua`, May 2021. Version 1.1.0.

[42] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66, 2001. doi:10.1162/153244302760185243.

[43] Thijs van den Hout, Thymen Wabeke, Giovane C. M. Moura, and Cristian Hesselman. LogoMotive: Detecting logos on websites to identify online scams - a TLD case study. In *International Conference on Passive and Active Network Measurement*, pages 3–29. Springer, 2022. doi:10.1007/978-3-030-98785-5_1.

[44] Thymen Wabeke, Giovane Moura, Nanneke Franken, and Cristian Hesselman. Counterfighting counterfeit: detecting and taking down fraudulent webshops at a ccTLD. In *International Conference on Passive and Active Network Measurement*, pages 158–174. Springer, 2020. doi:10.1007/978-3-030-44081-7_10.

[45] John Wadleigh, Jake Drew, and Tyler Moore. The e-commerce market for "lemons" identification and analysis of websites selling counterfeit goods. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1188–1197, 2015. doi:10.1145/2736277.2741658.

[46] David Y Wang, Matthew Der, Mohammad Karami, Lawrence Saul, Damon McCoy, Stefan Savage, and Geoffrey M Voelker. Search + seizure: The effectiveness of interventions on SEO campaigns. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, pages 359–372, 2014. doi:10.1145/2663716.2663738.

[47] Wikipedia contributors. Cloaking — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Cloaking&oldid=1079171070`, 2022. [Online; accessed 2022-03-22].

[48] Wikipedia contributors. Tf–idf — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Tf%E2%80%93idf&oldid=1071253989`, 2022. [Online; accessed 2022-04-01].