Towards Website Domain Name Classification Using Graph Based Semi-supervised Learning

Azadeh Faroughi[c], Andrea Morichetta[a,b], Luca Vassio[a,*], Flavio Figueiredo[d], Marco Mellia[a], Reza Javidan[c]

[a]Politecnico di Torino, Italy
[b]TU Wien, Austria
[c]Shiraz University of Technology, Iran
[d]Universidade Federal de Minas Gerais, Brazil

Abstract

In this work, we tackle the problem of classifying websites domain names to a category, e.g., mapping `bbc.com` to the "News and Media" class. Domain name classification is challenging due to the high number of class labels and the highly skewed class distributions. Differently from prior efforts that need to crawl and use the web pages' actual content, we rely only on traffic logs passively collected, observing traffic regularly flowing in the network, without the burden to crawl and parse web pages. We exploit the information carried by network logs, using just the name of the websites and the sequence of visited websites by users. For this, we propose and evaluate different classification methods based on machine learning. Using a large dataset with hundreds of thousands of domain names and 25 different categories, we show that semi-supervised learning methods are more suitable for this task than traditional supervised approaches. Using graphs, we incorporate in the classifier aspects not strictly related to the labeled data, and we can classify most of the unlabeled domains. However, in this framework, classification scores are lower than those usually found when exploiting the page-specific content. Our work is the first to perform an extensive evaluation of domain name classification using only passive flow-level logs to the best of our knowledge.

Keywords: semi-supervised learning; domain names; network measurements; classification; passive measurements.

*Corresponding author
Email address: `luca.vassio@polito.it` (Luca Vassio)

1. Introduction

The latest estimations show that there are over 1.6 billion websites on the Internet, distributed over more than 268 million domains.[1] With this ever-growing nature of the Web, researchers and practitioners resort more and more to automated approaches bases on machine learning to process and understand such vast variety. A common machine learning classification task is to assign a category to a domain (i.e., mapping `bbc.com` to a category such as "News and Media")[1]. This task is the focus of our paper and has several applications such as information integration [2], building efficient focused crawler or vertical search engines [3], helping to choose the appropriate model for extracting information from a web page [4], improving quality of search results [5], constructing and expanding web directories [6], web filtering [7] and advertising [8].

In this paper focus, websites are URLs such as `https://www.bbc.com/news/today.html`, whereas domains refer to the URL domain only, i.e., `www.bbc.com`. Our goal is to perform domain classification using large flow level logs only. We specifically consider logs collected by passively monitoring the network traffic, where a passive sniffer identifies TCP or UDP flows, and recovers the name of the servers serving such flows. Each flow contains the client identifier (e.g., the client IP address - properly anonymized in our data) and the domain name of the server (recovered directly from the HTTP request, or DNS and TLS negotiation when HTTPS is in place). Given the flow sequence, each with the timestamp of when the request was observed, we want to automatically assign each domain a category, assuming to know only a small subset of domain categories (e.g., via manual labeling). This problem falls in the supervised classification class, and we aim at training a classifier based on different machine learning approaches.

The problem of classifying domain names is not new; various researchers studied it in the past [9, 10, 11]. However, differently from prior efforts, we focus on passive flow level traces, and we are limited to little information. No information on web page content (e.g., their HTML content, metadata, objects, images, etc.) is available to our classifier by assumption - since we rely on passively collected data. Since we do not require to collect and analyze the content of the pages, our approach is scalable, and it naturally factors the popularity of websites (the more the users are visiting a website, the more data we collect about it), the users' habits, and the diversity of the devices and applications they use to access the Internet. Note also that the rise of encryption in the web via HTTPS/TLS, flow traces are the only information that is still available to Internet Service Providers (ISP) and network administrators [12].

Given the importance of such a form of data, and the easiness of collecting them, some prior efforts studied the website classification problem using passive traces only (see Section 2 for a discussion). However, previous works usually limit themselves to a handful of domains and very small datasets. Here, we leverage

[1] `https://news.netcraft.com/archives/2019/01/24/january-2019-web-server-survey.html`, accessed 2020-07-25

the information contained in data collected from real users from our University campus in Turin. We collect 40 days of traces, where we extract several hundred thousand unique domains. From these, we extract the domains related to explicit visits to websites (called Core domains) building on our previous work [13]. We obtain more than 14 000 unique core domains.

For labeling, we take as ground truth the categories given by SimilarWeb, an open service providing 25 non-overlapping different categories. When compared to other publicly available repositories of domain classes such as Curlie (formerly known as DMOZ), SimilarWeb provides better coverage in our use case (see [14] for how critical this labeling could be). SimilarWeb's richness in the number of classes (25) implies that some of them may overlap. Hence, we do not expect that the results can compare to the ones with fewer classes. Despite offering a large dataset, SimilarWeb allows us to label only a small fraction of domains in our trace (about 2 000). This limitation strengthens our work motivation to use a machine learning approach that can classify the remaining 86% of data.

For the classification task, we thoroughly compare several different methods, including the ones previously proposed in the literature in other contexts, that we adapt to our use case. Next, we investigate graph-based semi-supervised methods [15] where the nodes are the domains, while the edges factor the different similarities between domains.

Results show that our semi-supervised method can achieve the best results with average accuracy in the order of 0.52. Albeit low at first glance, these scores represent a gain of 300% when compared to the naive random classifier (0.52 vs. 0.13), and a gain of about 27% compared to the best performing supervised method (0.52 vs. 0.41). Granting that we tackle a classification problem with 25 classes, our results represent an improvement to prior proposals that considered both a much smaller dataset and a reduced number of classes.

Our main contributions in this paper are i) the specific focus on data that can be collected by simply passively observing network traffic, ii) the exploitation of the semi-supervised method, and iii) the in-depth comparison of different approaches. Our analysis may serve as a guide for future works aiming at exploring passive flow level data, possibly integrating it with active crawlers. We aim at fostering future research and the reproducibility of results. For this, we release the labeled dataset used for this analysis and the Python scripts containing the code for the machine learning approaches and the used parameters [16].

The rest of this paper is organized as follows. In Section 2, we present related work on the subject. In Section 3, we discuss on the possible approaches. In Section 4, we present the supervised methodology to solve the classification task, while in Section 5, we describe the semi-supervised methods. In Section 6, we define our dataset and its preprocessing, reporting in Section 7 all the results of the methodologies. Finally, we conclude the paper in Section 8 with an outline of the results.

2. Related Work

In this section, we review some of the previous efforts focused on domain and website classification tasks. Previous works may be categorized into three broad cases. Content-based methods, which explore HTML, Images or Video (Section 2.1). URL and link methods, which explore the textual tokens in the URLs as well as hyperlinks across pages (Section 2.2). Finally, some approaches focus on traffic requests (Section 2.3). Given that we are limited to TCP traces, we focus on adapting methods taken from the second and third classes.

2.1. Content based approaches

Content-based approaches rely on document contents or their brief descriptions as extracted by visiting the web page. The web page content can include texts, audios, images, videos, and structure records. In [17], the authors express document content as n-grams feature-vectors; the n-grams frequencies vector represents each web page. Afterward, they apply supervised methodologies for web page classification. In [18], the authors study the influence of different significance indicators for automatic web pages classification. The indicators are the title, the headings, the internal metadata, and the main web page text. They showed that it is possible to obtain the best classification with a well-tuned linear combination of these four elements. Shen et al. [19] proposed an approach to classify web pages topics through web page summarization algorithms. These algorithms aim at extracting the most relevant features from web pages. By preprocessing web pages with summarization techniques, they get an improvement in the classification accuracy, compared to plain content-based classifiers. In the field of web content classification, the work in [20] employes Ant Colony optimization [21] for classification rules discovery. The work proves that the Ant Colony is a powerful classification tool and produces high accuracy in the results.

Know and Li in 2003 [22] proposed a web page classifier based on k-nearest-neighbor (k-NN). In this method, they use HTML tags and structure features, where different parts of HTML tags have different weights. Considering two documents, the higher the co-occurrence of terms between the two, the stronger their relationship is.

De Boer et al. in [23], use visual-based features, such as simple color and edge histograms to provide an aesthetic classification of web pages. In [24], the authors propose a visual-based approach, where they classify web pages into three main categories, namely information pages, research pages, and personal home pages, using both structural and visual features. Kovacevic et al. [25] proposed a method based on visual layout analysis. They represent a web page as a hierarchical structure called visual adjacency multi-graph in which the nodes represent HTML objects, and the directed edges reflect spatial relations between objects on the browser screen. By visual information of the multi-graph, it is possible to define heuristic rules for recognizing common logical areas of web pages.

At last, authors in [26] build a supervised classifier that targets 5 sensitive categories (ethnicity, health, political belief, religion, sexual orientation). They leverage web page contents, comparing the text against a list of keywords that may identify each category. A naive Bayes classifier suffices for this simple task.

All the works related to this category rely on features extracted by directly visiting and rendering the page. It is thus necessary to first have the complete URL of the page and then to access it to analyze its content, structure, or visual features, adding computational and time complexity in the process. Note that active crawling is also becoming more and more complicated. For instance, the simple landing page of a website may not reveal the actual content until the user has accepted the so-called cookie-policy, or performed a login, or entered the inner page of the website. Our methods are simpler, as they require only to receive as input the name of the domains users visits when regularly accessing the web.

2.2. Link and URL based approaches

In link-based approaches, features can be pulled out from other pages related to the pages under analysis with hyperlinks. This approach aims at supplying additional information for the classification step. This category of methodologies requires extensive crawling sessions. Typically, these approaches incorporate the creation of links-graphs. Utard and Furnkranz [27] proposed a method that uses the information present in hyperlinks toward the page of interest. They use the region in the neighborhood part of the predecessor document. These parts can be the anchor text, the anchor text neighborhood, or the text in a paragraph around the hyperlink. Moreover, they also use the text on the target web page.

Some previous works focus on URLs and extract features by dividing the URLs into meaningful portions. Using only URLs, the execution is faster since there is no need to retrieve and analyze web page content. Kan and Thi [9] proposed a supervised method based on URLs features. The proposed method divides the URLs into meaningful tokens. These tokens constitute the feature set, together with correlated information, such as the token position in the URL, the token lexical kind, or, again, information about the token successor and predecessor. The feature set is the input for a supervised maximum entropy model, a classical method in text classification. Baykan et al. [28],[29] presented a supervised classification based again on URLs. They split each URL in tokens, using any punctuation mark as separators, extracting strings, numbers, or other non-letter characters. The feature set consists of four different categories: tokens, n-grams derived from tokens, n-grams directly derived from the URL, and positional information explicitly encoded in tokens or n-grams. Then they use those features to build a Support Vector Machine (SVM) [30] and a naive-Bayes classifier. The work presented in [31] proposed an unsupervised web page classification solution based on URLs. Each resulting cluster includes a set of web pages that are assigned to the same class. Unlike classification methods that need a training set of labeled pages, the proposed solution builds several URL patterns representing the different classes of pages on a website. It is then possible to classify additional pages by matching their URLs to the patterns.

In our work, we consider having access to the domain name and not the entire URL, as in TLS traffic.

2.3. Traffic based approaches

Jiang et al. [32] is the only work that proposed a method based on patterns in mobile application access logs. They extracted the logs from the traffic flow data captured in an ISP core network and target the classification of domain names into four coarse classes. Then they extracted the latent vector representation from users' visiting sequences, taking inspiration from the Word2Vec model [33]. In this context, mobile server domains stand for words, while a user visiting sequence corresponds to a sentence. The resulting vector is the input for a Support Vector Machine classifier. In our work, we implement the methodology of Jiang et al. [32] (see Section 4.2) and compare it with the other proposed approaches. Finally, it is essential to point out that the authors of [32] perform their evaluation on a limited dataset filtering only a handful of classes (5) and focusing on the most popular, by access, domains. Here we explore a much broader set of data and a finer-grained classification. In this sense, the results in numbers (e.g., accuracy or precision) reported in [32] are not comparable to our work.

3. Discussion over the adopted methodologies

In this paper, we want to address the problem of websites domains classification. We extract three features from the TCP traces: the source IP, the timestamp, and the domain name. We know that some domains belong to a category, and we want to build a model to enhance this knowledge. Using the available features, we investigate the problem over two dimensions. In the first, we consider domain names as strings. In the second, we consider the temporal evolution of how users move from one website to another by examining the sequence of domains they visit inside a time window.

We try different solutions based on machine learning, including some previously proposed ones for similar problems. Worth to mention is the analysis of Neural Network performances. More specifically, we refer to deep learning methodologies that work with sequences of data, like strings or time sequence. While these off-the-shelf solutions are easy to execute, they perform reasonably well given the complexity of the problem and the limitation of getting large amounts of labeled data. For this, we define more ad-hoc features and methodologies and compare them in detail.

In the following sections, we will discuss the proposed approaches, supervised and semi-supervised. In Section 4, we discuss how to extract relevant attributes from the data collection in use and how to define appropriate classification methodologies. In Section 5, we examine the semi-supervised classification approaches. This category of methodologies is particularly suited for problems in which the classified set of elements is small with the overall dataset size, and it is insufficient for building a model. Given the complexity of the task, as discussed before, we have to use specific features. Furthermore, we show the necessity of computing specific similarity values for the nodes in the graph. Finally, we describe a graph pruning technique that allows having fewer edges, limiting the computational and memory complexity. The package containing the code for all the used methodology and all the tuned parameters is available online [16].

4. Supervised classification approaches

In this section, we outline different methodologies for classifying domain names. We describe only methods based on features obtained by passive measurement traces. This kind of data allows the extraction of sequences of domains visited by users. The rising encryption of traffic (i.e., HTTPS) is nowadays limiting access to more detailed information. Most notably, a passive sniffer can capture only the timing and flow information, along with the domain name of the contacted server.

Among the other data, the logs elements include a client IP address (anonymous in our analysis), $s \in S$; a requested domain name, $d \in D$; and a timestamp when the request was sent $t \in [0, T]$. We are able to define a category, $c \in C$ for a small subset of domains (e.g., via manual labeling). Overall, our data has the form of quadruples $T = \{(t, s, d, c)\}$ where each entry is a request. $T_l = \{(t_l, s_l, d_l, c_l)\}$ is a labeled subset of data, with sub-scripts representing if this is the case (labeled data). Our goal is to create a function $F_\theta : D \to C$, from the set of domains D to the set of classes C, whose objective is to accurately uncover c_u for an unlabeled subset of domains: $T_u = \{(t_u, s_u, d_u)\}$. We will define the function F_θ starting from our labeled data T_l. This function is parametrized by θ (e.g., in a logistic regression, θ are the regression parameters) and is applied as $F_\theta(d) = \hat{c}$, where $\hat{c}$ is the predicted class.

4.1. Supervised classification based on domain names

A domain name is a string composed of strings separated by 'dots,' i.e., the different domain levels. We consider, for the forthcoming experiments, n-grams as sequences of characters present in all the domain levels. For instance, in `google.com`, the 1-grams are $\{g, o, o, g, l, e, c, o, m\}$, whereas the 2-grams (bi-grams) are $\{go, oo, og, gl, le, co, om\}$. For each domain, we extract all its n-grams with $3 \leq n \leq k$, where k is a maximum threshold and obtained by parameter tuning. For each category c, we count the frequency among domains of each n-gram g i.e., $f(g, c)$. This data represents the training features. Then, for an unlabeled domain name d_u, we compute its n-grams. We will assign the category $\hat{c}$ with maximum similarity $sim(d_u, c)$ between this domain d and all classes c.

$$\hat{c} = \arg\max_{c \in C} sim(d_u, c) \tag{1}$$

We use two different metrics to find the similarity between each unlabeled domain name and each category.

4.1.1. Similarity based on TFIDF on n-grams

The first metric we consider is TFIDF [34]. TFIDF is the product of two statistics, term frequency (TF) and inverse document frequency (IDF). It is widely used in information retrieval and gives a weighting scheme for each term [35]. TFIDF reflects how important an n-gram g is for a category $c \in C$ to the set of all categories in the training model. Term Frequency (TF) measures the importance of an n-gram in category c. We use the so-called augmented frequency version of TF [36] in order to prevent a bias towards longer documents, i.e., raw n-gram frequency divided by the raw frequency of the most occurring n-gram in the class c. The equation for TF is:

$$TF(g,c) = 0.5 + 0.5 \cdot \frac{f(g,c)}{\max_{g\,'}\{f(g\,',c)\}} \tag{2}$$

IDF measures how important an n-gram is in the whole collection of categories, i.e., whether it is common or rare. If an n-gram is very common, it has little importance to distinguish among the categories. Thus, we assign less weight to frequent n-grams, while we scale up the rare ones. IDF is defined as:

$$IDF(g,C) = \log \frac{|C|}{|\{c \in C : f(g,c) > 0\}|} \tag{3}$$

where $|C|$ represents the total number of categories and $|\{c \in C : f(g,c) > 0\}|$ is the number of categories where the term g appears. Then the TFIDF is calculated as:

$$TFIDF(g,c,C) = TF(g,c) \cdot IDF(g,C) \tag{4}$$

Given a domain d we calculate its similarity to a class c as the sum of TFIDF values $sim(d,c)$ of the n-grams of the domain d in a class c:

$$sim(d,c) = \sum_{g=1}^{k} TFIDF(g,c,C) \cdot f(g,d) \tag{5}$$

where $f(g,d)$ is the frequency of the n-gram g in the domain name d and k is the number of unique n-gram in d. At last, we assign d to the category with the highest similarity.

4.1.2. Similarity based on NFA on n-grams

The second evaluated metric is the Number of False Alarms (NFA) metric. NFA expresses a similarity measure between a domain and a class [37]. NFA algorithm employs the Helmholtz principle [38]: meaningful features and notable events appear as significant deviations from randomness or noise. For these reasons, humans can perceive the significance of the characteristics mentioned above. A low value of NFA connotes a perceptually meaningful event.

Following this approach, we can calculate the meaning of an n-gram g for a category c[37]. Given the sum of frequencies of n-grams in a class, i.e., $B(c) = \sum_g f(g,c)$, we define $N(c)$ as:

$$N(c) = \frac{\sum_{c \in C} B(c)}{B(c)} \tag{6}$$

Then we compute $NFA(g, c)$:

$$NFA(g, c) = \binom{\sum_{c' \in C} f(g, c')}{f(g, c)} \frac{1}{N(c)^{f(g,c)-1}} \tag{7}$$

If the NFA value is less than one, then the frequency of g can be reflected as a meaningful event since our calculations do not expect it. Thus, n-gram g can be considered as a meaningful or significant term in the category c. Since the values of NFA can be exponentially large or small, in order to define a function that computes the meaning of an n-gram within a class, we use the logarithmic value of NFA [38]. Finally, we obtain the distance $Meaning(g, c)$ as:

$$Meaning(g, c) = -\frac{\log NFA(g, c)}{f(g, c)} \tag{8}$$

The bigger the meaning score $Meaning(g, c)$ of an n-gram g in a class c is, the more significant the n-gram is for that class. Finally, we categorize the unlabeled domain d by computing the similarity as the total meaning value of d for the category c, i.e., $sim(d, c)$:

$$sim(d, c) = \sum_{g=1}^{k} Meaning(g, c) \cdot f(g, d) \tag{9}$$

where $f(g, d)$ is the frequency of the n-gram g in the domain name d and k is the number of unique n-gram in d.

4.2. Supervised classification based on sequences of visited domains

So far, the proposed methods consider only domains in isolation. We now turn our attention to sequential accesses to domains within a session. We aim at using the context of an unlabeled domain d_u to help its classification. In Section 6.2, we present a methodology to extract only explicitly visited websites (called Core). In this way, we can remove all the domains contacted for advertisements, trackers, and other parts of the page, as well as for other applications, system updates, etc. The concept is that two (Core) websites that users actively visit in a sequence have a more significant probability of belonging to the same class. Indeed, as in [32] and from our observations, users often visit same-category websites at a short distance of time. In this work, we consider as a session the sequence of visited domains by a user in a time window. We considered sessions of one hour, but we report the results with different time window length in Section 7.3. We implement the same methodology presented by Jiang et al. [32]. First, we use a Word2Vec model [33] to represent the session as a vector in a vector space. Word embedding is the most widely used text representation model. It represents each word with a very low-dimensional vector with semantic meaning. Afterward, we use a supervised classification method on the embedding space to assign categories to domains.

Let $v = [d_i : i = 1, 2, \ldots, S]$ be the ordered sequence of domains that a user visits in a session, where d_i is the $i - th$ visited domain, and S is the number of visits. Based on the model proposed by Mikolov et al. [39], we build the vector space using a multi-layered neural network arranged with the skip-gram model. In the current use case, the neural network has the job of predicting a target domain given a set of domains called *context domains*. The context domains of a target domain are the set of domains present at the same time in the visiting sequences in the corpus. More formally, the goal of word vectors is to maximize the average log probability:

$$\frac{1}{S}\sum_{t=1}^{S}\sum_{s=1,s\neq t}^{S}\log p(context|d_t) \tag{10}$$

where S is the number of domains in each sequence, and d_t is the target domain. The *context* may be either a sequence of co-occurring domains based on a sliding window or just the domain preceding d_t in time. In the latter case, we represent the probability $p(d_s|d_t)$ using a softmax function $p(d_s|d_t) = \frac{exp(\nu_t \cdot \nu_s)}{\sum_{t'} exp(\nu_{t'} \cdot \nu_s))}$. Softmax is a function that returns a vector that describes the probability distribution of potential assignment. Here, ν_s and $\nu_t \in \mathbb{R}^K$ represent the K dimensional vector space and $\cdot$ is the inner product. When the domains d_t and d_s frequently co-occur in a sequential manner, the parameters $\nu_{s,t}$ should have similar values, increasing the softmax probability. In order to compute the parameters, some techniques like hierarchical softmax [40], negative sampling [41] and sub sampling of frequent words [39] are used. We refer the reader to [39] for further details.

As a consequence, each domain has a vector representation $\nu_d \in \mathbb{R}^K$. Under those circumstances, we can use the resulting vectors as classification features. For supervised classification based on the vector of domains, we use a support vector machine (SVM) algorithm [30], as already used in [32]. In our implementation, we generate the representation of the domains using FastText [42], an implementation of Word2Vec, using the default parameters. The default sliding window is of size 5. Hence, with a session of size 5 (domains) $[d_1, d_2, d_3, d_4, d_5]$, when using skip-gram, for each of the 5 domains in the session, random words (within the window) are chosen to update the model. That is, when training, we try the predict d_1 using one other random domain. This is done iteratively until the model converges. In our dataset, we point out that only 28% of our sessions have a size greater than 5. Thus increasing this window size should have a limited effect on our results.

5. Semi-supervised classification approach

The previously described approaches study a traditional supervised classification problem. When the number of labeled data is small compared to the

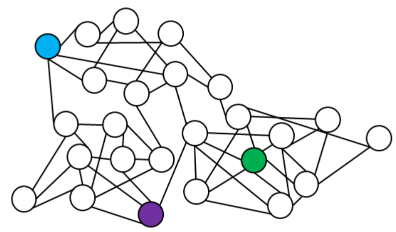


Figure 1: An example graph, where only coloured nodes are labeled, and the others are unlabeled. Edges can be built following different criteria.

unlabeled data, sometimes this approach does not obtain accurate predictions. A solution to overcome this limitation is to use semi-supervised classification. The intuition is that, in semi-supervised techniques, the unlabeled data can somehow be useful to improve the classifier. Here, by using a semi-supervised methodology, we exploit a few labeled domains and their relationship with the remaining large amount of unlabeled domains to extend our knowledge.

In our work, we rely on a graph-based approach, proposed in [43], where domains are vertices, and their similarities define edges and weights. This algorithm belongs to the class of transductive methods. This category's main characteristic is leveraging unlabeled data for training the method, using a graph data representation. [44, 45] The graph structure enables the propagation of the few available labels through its network until all the domains in a connected component are labeled. The graph structure is defined as $G = (V, E)$, where V is the set of vertices that include both the labeled and unlabeled domains, and E is a set of edges. The graph structure uses distance metrics or kernel functions like the Gaussian kernel to define the edge weight between pairs of nodes and represent them by an adjacency matrix W. As an example, Figure 1 illustrates an undirected connected graph in which colored vertices represent the labeled domains.

Here, we have a vector representation of domains, and we use the cosine similarity [46] to measure how similar domains are. We have two representations, one obtained using domain2vec, and the other using NFA. The first type considers co-occurring domains. The domain2vec process extracts 100-dimension vectors. With this methodology, domains that often appeared together in the same sessions have similar vector values. The second class of vectors looks at the domain names. The vectors have 25 elements, each representing the distance to the SimilarWeb categories, computed using NFA. Similar domains have similar vector values. As we describe later in the text, in the SSDS approach, we use the former domain2vec representation; in SSDB, we use the latter. In SSB, we use both, combining the similarities in the final weight.

Toward increasing efficiency and robustness against noise, we extract a sparse weighted subgraph from the fully connected graph. There are different possible solutions to recover a sparse subgraph. The most common algorithm is the k-Nearest Neighbor algorithm (k-NN); it keeps k-nearest neighbor edges, extracted with the use of similarity functions, for each node. Another viable approach is the ϵ-neighborhood graph. This subgraph extraction technique removes all the data whose pair-wise similarity is smaller than ϵ [47].

A fundamental assumption of semi-supervised learning problems, called *smoothness*, is that nodes close to each other in the network are likely to have the same labels. Let $D_l = \{d_1, d_2, \dots, d_l\}$ be the set of labeled domains, with $|D_l|$ the number of them. Let $D_u = \{d_{l+1}, d_{l+2}, \dots, D_{l+u}\}$ be the unlabeled ones. There are $|C|$ classes, and each class $c \in C$ comprises a subset of domains in D_l. We define a matrix $Y_l \in \{0,1\}^{|C| \times |D_l|}$ with $Y_{ij} = 1$ if $d_j \in C_i$. Y_l maps domain D_l into classes. The training data $D = D_l \cup D_u$ produce a weighted graph $G = (D, W)$, where D has $N = |D_l| + |D_u|$ domains, and $W \in \mathbb{R}^{N \times N}$ is the adjacency matrix.

The prediction is based on assumption of consistency: (1) nearby points are likely to have the same label (2) points on the same structure (cluster or a manifold) are likely to have the same label prediction of labeled domains. To formalize the assumption, we use a classifying function [48], which is sufficiently smooth for the structure of labeled and unlabeled domains. The objective function is:

$$\arg\min_F \frac{1}{2} \left(\sum_{i,j=1} W_{ij} \left| \frac{1}{\sqrt{\mathbb{D}_{ii}}} F_i - \frac{1}{\sqrt{\mathbb{D}_{jj}}} F_j \right|^2 + \mu \sum_{i=1} n \left| F_i - Y_i \right|^2 \right) \tag{11}$$

where $F \in \{0,1\}^{|C| \times N}$ is the final mapping of domains (labeled and unlabeled) into classes and $\mathbb{D}$ is the diagonal degree matrix given by $\mathbb{D}_{ii} = \sum_j W_{ij}$.

The objective function has two terms. The first one represents the smoothness constraint, which expresses the dissimilarity between the results of the classifying function of nearby nodes. In a nutshell, the classification outcome should not differ too much when considering two adjacent elements. The second term refers to the difference between the output of the classifying function and the initial labeling. In a few words, the final classification should be compatible with the ground truth labels. The idea of smoothness constraint can be expressed using graph Laplacian. The Laplacian matrix is obtained by $L = \mathbb{D} - W$ and regularization Laplacian is often used to constrain the labels to be consistent with the graph structure [44]. A positive weight parameter μ captures the trade-off between these two terms.

A fundamental step in the semi-supervised approach is the extraction of the adjacency matrix W, which represents the edge weights, using a meaningful similarity measure. In our work, we define the weight of the edges in graphs via different similarity functions. We choose metrics that refer to the functions defined for the supervised classifications in Section 4. Hereafter, we assign the weights in three ways: (i) using a similarity function associated with the domain names, (ii) considering the sequence of visited domains, and, lastly, (iii) combining the distances obtained using these two features. These three solutions include pruning mechanisms to reduce the number of edges in graphs, assigning a weight of 0 under a certain threshold. The pruning is necessary to avoid weak connections and prevent the creation of complete graphs, computationally intractable when the number of nodes is large. In the following, we define and describe the three weight functions for extracting values for W.

5.1. Edge weighting with similarity based on the domain names

The first similarity function for edge weighting uses the NFA-based vectors extracted in Section 4.1. Using the similarity function $sim(i, c)$ between a domain i and a class c as a building block, we use cosine similarity for computing the pairwise domain similarity between the domains i and j in the following way:

$$sim_{name}(i, j) = \frac{\sum_{k=1}^{|C|} sim(i, k) \cdot sim(j, k)}{\sqrt{\sum_{k=1}^{|C|} sim(i, k)^2}\sqrt{\sum_{k=1}^{|C|} sim(j, k)^2}} \tag{12}$$

We generate an edge between i and j if the resulting similarity is higher than a threshold ϵ, with weight equal to the output of $sim_{name}(i, j)$:

$$W_{ij} = \begin{cases} sim(d_i, d_j) & \text{if } sim(i, j) > \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

In order to choose the best threshold, tune parameter ϵ is done by performing 5-fold cross-validation with different ϵ values in the range $[0.9, 0.99]$ (using steps of 0.005). The selected value is the one with the best performance in our cross-validation procedure. Moreover, for the best final algorithm, we also employed a dedicated test set. We refer to Section 7 for the definition of the parameter values.

5.2. Edge weighting with similarity-based on domain sequences

The second way to define the weights of the edges uses the vectors extracted from the sequences of visited domains described in 4.2. Recall that each domain in the word vector model is represented as a $\nu_d \in \mathbb{R}^K$ vector. With such vectors, we can now compute a pairwise similarity matrix for every pair of domains. Here, we again make use of the cosine similarity based on multi-dimensional vectors:

$$sim_{sequence}(i, j) = \frac{\sum_{t=1}^{K} \nu_{it}\nu_{jt}}{\sqrt{\sum_{t=1}^{K} \nu_{it}^2}\sqrt{\sum_{t=1}^{K} \nu_{jt}^2}} \tag{14}$$

Afterwards, we create an edge between two domains according to equation (13). We use 5-fold cross-validation to tune the parameter ϵ with different values in the range $[0.4, 0.8]$ (using steps of 0.01). We report in Table A.3 in Appendix A the optimal value of ϵ that has been found.

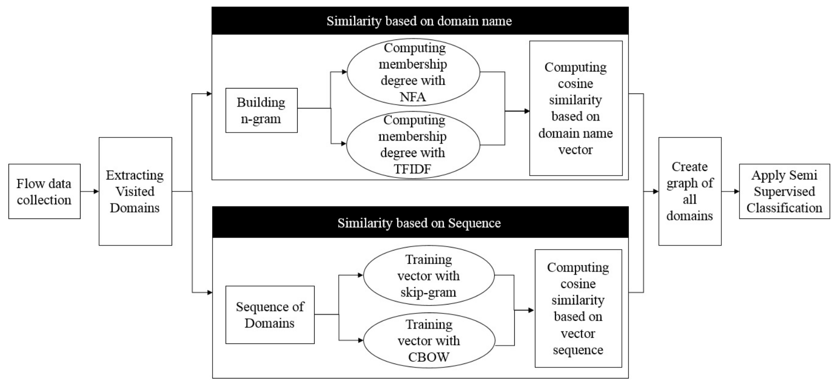


Figure 2: Employed schema for semi-supervised method based on both the domain name and the sequence of visited domains.

5.3. Edge weighting combining metrics using domain names and domain sequences

The third and last edge weighting function considers the conjunct impact of both features, i.e., domain names and sequence of visited domains. In this way, we can exploit both the concepts of similarity, enriching edges information. To reach this goal, we compute the average similarities of the equations (12) and (14):

$$sim_{name\&sequence} = \frac{sim_{name} + sim_{sequence}}{2} \tag{15}$$

Then, based on the resulting similarity, we again use the equation (13) to assign the weights (see Table A.3 in Appendix A for the adopted ϵ).

In Figure 2, we show the whole flow used for applying this semi-supervised method.

6. Dataset collection, preparation and characterization

6.1. Network traffic collection

Our analysis relies on a dataset collected in our university campus in Turin (Italy). In the dataset, users' terminals (usually PCs) are directly connected to the Internet via campus network (wired Ethernet) and uniquely identified by a statically assigned IP address associated with one and only one terminal. In other setups, like, for example, in the presence of a NAT, the users should be identified using different strategies as proposed in the literature [49]. However, this is outside the scope of this work. Moreover, often clients are contacting from their terminal also domains outside of the browser session (e.g., software updates or other background applications). The Core domain approach presented in Section 6.2 helps in removing such domains.

We rely on Tstat [50] to collect data. Tstat monitors each TCP flow, exposing detailed information. Here, we are interested in retrieving the domain name of the server being contacted. Tstat implements three techniques to get it. For HTTP flows, the `Host:` header is parsed directly from HTTP requests. In the case of HTTPS/TLS, Tstat DPI module extracts the Server Name Indication (SNI) field in the Client Hello message. SNI is a TLS extension by which the client indicates the server domain that it is trying to contact. At last, Tstat reports the domain name clients resolved via DNS queries prior to flows [51]. We combine these three mechanisms to label each TCP flow with the server name, giving higher priority to Host and SNI fields where more than one is present.

In this work, for each TCP flow, we consider: (i) the anonymized client IP address as terminal identifier s, (ii) the starting time of the flow t and (iii) the server domain name d - as retrieved via HTTP, TLS, or DNS protocols.

Our dataset contains the traffic of approximately 2 500 terminals, collected at our university Campus in Torino in 40 days in 2017. The dataset includes 4691 million flows and 404 thousand unique domains. For our train/validation/test set definition, we extract the domains visited in one day by the users (see Section 6.4).

Information about user behavior is sensitive, and the collection of these data might be privacy-invasive [52]. To reduce as much as possible to possible privacy violations, we followed the best practice of limiting the data collection to only the necessary information for the experiment. Both the data collection process and the collected data have been discussed, reviewed, and approved by the ethical board of our University. In collaboration with our campus network administrators, we took all possible actions to protect the leakages of private information from users. In particular, Tstat was installed and configured i) to process packets in real-time, ii) to anonymize the IP addresses of clients using an irreversible hash function, whose key was selected by the network administrators, and iii) to save only flow level logs with the needed information.

6.2. Identification of Core domains

Here we present a methodology to extract only explicitly visited websites. This approach is instrumental in removing all the domains contacted for advertisements, trackers, and other content of the page and traffic of other applications, system updates, and other elements running in the background. Indeed, when visiting a web page, the browser application first downloads the main HTML document and then fetches all the page objects (images, scripts, advertisements, and other content). These objects often lie on external servers that have different domains [53]. We call *Core domain* a domain initially contacted to download the main HTML document of a page. Core domains are essential since users intentionally visit them, like `www.facebook.com` and `en.wikipedia.org`. We call *Support domains* those domains automatically contacted by visiting a Core domain, or by background applications, like `static.10.fbcdn.net` and `dl-client.dropbox.com`. Support domains do not contain useful information about user intention. Hence, we build on our previous methodology [13, 54] to identify and consider only Core domains. Here we briefly report the Core domain extraction methodology.

We build a labeled dataset that we use for training and testing. We consider 500 Core and 500 Support domains, a balanced labeled dataset that we make publicly available [16]. We visit each domain using a headless browser and extract an extensive list of features guided by domain knowledge. Features include the length and the content type of the main HTML document (if present); the number of objects of the page and domains contacted by the browser to fetch all objects; HTTP response code (e.g., 2xx, 3xx and 4xx); and whether the browser has been redirected to an external domain. We then let the classifier choose the ones that better allow it to separate Core and Support domains. We solve the classification problem using a decision tree classifier. The final model results in a simple, efficient, and descriptive tree which reads as it follows: a domain is Core if a) the main HTML document size is bigger than 3357B and b) the browser is not redirected to an external domain, i.e., the HTTP response code of the website homepage is not 3xx or, if it is, the homepage is still redirected to another page on the same domain. Intuitively, support domains typically lack real home pages. When directly contacted, the server reply with short error messages. In some cases, Support domains redirect visitors to the service home page (e.g., `fbcdn.net` redirects on `www.facebook.com`). Despite its simplicity, overall accuracy is higher than 96% when tested against 1 000 labeled domains. For more details, refer to[13].

Considering the dataset obtained in Section 6.1, we identify 161 333 unique Core domains (14 712 for the single day labeled and used for training/validation/testing). This dataset of Core domains is released to the public [16]. IP addresses are obfuscated, and the class is provided, where available.

6.3. SimilarWeb dataset with domain category

To obtain Core domain classes, we conducted several tests using different categorization systems. Note indeed that there is not a unique taxonomy, and each service provides a different definition of classes and offers a different coverage [14]. Here, we rely on SimilarWeb[2], a website that provides web analytics services. It results in the most reliable and offers good coverage of domains, even for Italian websites. As a result of several manual inspections, SimilarWeb performed consistently better than other publicly available datasets to categorize our study country domains.

Among the other information, they offer an extensive database of categorization of second-level + top-level domains. We use this as our ground truth. The total number of categories is 25. This number is significant, and many classes may have some overlap. For example, many domains could be assigned to both "Internet and Telecom" and "Computer and Electronics". We could have merged multiple categories, but we decided to keep the original categorization

[2] `https://www.similarweb.com/`

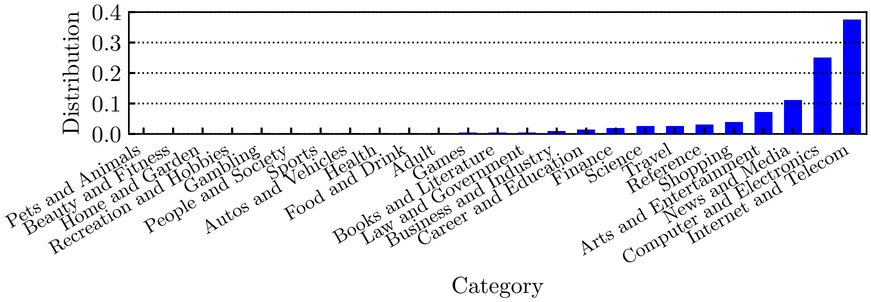


Figure 3: Distribution of popularity in terms of visits of labeled domains in each of the 25 classes.

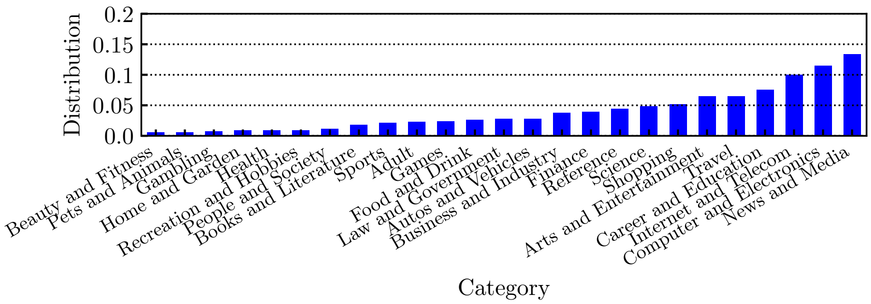


Figure 4: Distribution of unique labeled domains in each of the 25 classes.

of SimilarWeb as ground truth to make our results easily comparable by other scientists.

We intersect our dataset of Core domains obtained in Section 6.2 with the dataset of labeled domains of SimilarWeb referring to 2017. We obtain 2 178 labeled domains out of 14 712 unique Core domains used for training/validation/testing (around 14%). Hence, SimilarWeb contains only a small fraction of the domains for our trace in Italy. Once more, the limited coverage of available classification services motivates the need for automatic means of solving the classification problem.

For comparison, we also checked the DMOZ labeled dataset [3]. Besides having fewer classes than SimilarWeb (15), it covers only 8% of domains in our data.[4]

Figure 3 and Figure 4 show a characterization of the categories in the dataset.

[3]`https://www.kaggle.com/shawon10/url-classification-dataset-dmoz`. DMOZ was abandoned in 2017 by Mozilla, and now accessible under Curlie.org.

[4]We also tried to merge the two services, but desisted due to the difficulty in matching the categories and the different criteria they use to assign a website to a class.

Figure 3 depicts the categories' popularity on the overall set, measured as the total number of visits for each domain. The figure helps to understand which are the most popular categories. "Internet and Telecom" and "Computer and Electronics" cover more than 60% of the overall traffic. The strong prevalence of tech-related categories is not surprising since the dataset collects users' activity on our campus, where the research on these topics is predominant.

Figure 4 outlines the distribution of unique domains over the different categories. The results show a different distribution than Figure 3. Here "Internet and Telecom" and "Computer and Electronics" now include less than 20% of the unique domains, and "News and Media" results to be the category with more distinct elements, suggesting a broader heterogeneity in the fruition of this kind of content.

6.4. Preparation of training, validation, and testing sets

We consistently use for all the methods the same approach. We split the labeled data into train, validation, and test data. We use training and validation sets for parameter tuning for each method, using 5-fold cross-validation. The 5-fold cross-validation is performed for both the supervised and semi-supervised methods, with the same set of labeled elements. For the semi-supervised method, we build the graph with all the Core domains, of which only a fraction is labeled (see Section 6.3). The ones that are not labeled will eventually obtain an estimated label after performing the method, but they cannot be considered for evaluating the performance. Only the labeled ones are taken into consideration, following the same 5-fold cross-validation procedure as for the supervised ones.

The test set is a separate and independent sample of data that we use to provide an unbiased evaluation of the related final model. It is used only to obtain an independent evaluation of the final algorithm, and the result on the test set cannot lead to changes in the choice of the algorithm or the parameters since we will then have no way to measure the true performance. Hence, we can use it only on the best algorithm [55].

To obtain the test set, we consider 20% of the original (randomized) data. The remaining 80% is the dataset used for our 5-fold cross-validation. For each fold, we train each model on 80% on this set and validate on the remaining 20%. We select the algorithm with the best performance on the cross-validation step, and finally, verify its performance on the test set to indicate how it will perform in practice.

7. Experimental results

In this section, we report the experimental evaluation of the considered methodologies. Results can be reproduced by using the code and dataset provided in [16].

7.1. Evaluation metrics and parameters selection

Overall, we have six different approaches to compare. In addition to our six classifiers, we also consider two naive classifiers as a baseline. The first one assigns all domains to the most frequent category, i.e., "News and media" as in Figure 4. We call it "Naive-MostFrequent". The second one assigns one category uniformly at random. We call it "Naive-Uniform".

As said, we perform, for each solution, 5-fold cross-validation on the training set. The cross-validation generates new train and validation datasets with different combinations of elements. For each execution, we consider 80% of the trained data for training and 20% for validation. This process allows us to obtain better performance estimations and better tune the algorithms by combining different parameter values. Regarding the latter, we report the selected parameters in Table A.3 in the Appendix A. We evaluate the performance of each solution using standard classification metrics. For each validation fold, we obtain the *confusion matrix*, a numerical representation of how the classifier predicted the instances of each label. From it, we compute the Accuracy, i.e., the fraction of correct predictions. Moreover, we compute separately for each class the Precision, Recall, and F-Measure [55], offering a detailed analysis of the results. Furthermore, we compute the average of Precision and Recall over the different classes (weighting all classes equally), called macro-averages. Finally, the macro-average for F-Measure is computed as the harmonic mean of the macro-average of Precision and Recall.

Given a labeled instance x and a list τ_x ranking its confidence of x to belong to the different categories, the Position Error (PE) [56], is a measure of the deviation of x correct label position (λ_x) from the top-rank in the τ_x list. For example, if the actual label is in the first position in τ_x, then the error is 0. The maximum error is $m-1$, where m is the number of classes. The Normalized Position Error (NPE) over the number of classes is defined as:

$$NPE(\tau_x, \lambda_x) = \frac{\tau_x(\lambda_x) - 1}{m-1} \in \{0, 1/(m-1), ..., 1\} \tag{16}$$

NPE allows us to evaluate how off is the classification from the correct class. This is a softer metric compared to the ones defined over the confusion matrix, which only consider if a decision is correct or wrong. For example, if the second (last) most probable class is the correct one, we have a PE equal to 1 (24, respectively), even if the decision is wrong.

7.2. Overall and per class results

Table 1 depicts the overall results, obtained with a 5-fold cross-validation process. Observe in general how the naive classifiers perform poorly. This outcome is predictable; having 25 classes, and assigning a domain to a random class or the most popular, results with high probability in a wrong choice. The Naive-MostFrequent has higher accuracy (0.133, equal to the most common frequency as in Figure 3) than Naive-Uniform (0.033). However, the former is deterministically wrong in 24 out of 25 classes resulting in poor average Precision, Recall, and F-Measure.

Moving to Machine learning approaches, we recognize how using domain name structures improves performance. Measuring the similarity with NFA performs better than TFIDF, topping to 0.410 accuracy. When considering just the domain sequence ("SVM-Supervised-DomainsSequence"), we obtain similar performance. Worth to mention, we also tried an approach based on LSTM. We focused on domain names, using character-level models. A character-level model reads each word as an ordered series of characters. The final prediction tells us to which category the domain name belongs. For this aim, we used LSTM as implemented in Keras [57]. The obtained accuracy for LSTM is equal to 0.416. Even if LSTM performance is similar to that of NFA, with the latter, we can implement the similarity metric used in the (better) semi-supervised methods.

Focusing on semi-supervised approaches instead, we can notice a further improvement in the classification. The outcome results correctly in 47% and 44% of the cases when using edge weights based only on domain names or domain sequences. When coupling the information bought by both the domain name and the sequence, we observe a significant improvement, reaching overall accuracy of more than 52%. Overall, all semi-supervised methods improve the performance of supervised classification.

The same behavior is registered analyzing macro-average scores, that help in summarizing the per-class classification results. In this case, as well, the ranking of the methodologies is unchanged. Overall, this outcome shows the better capability of the Semi-supervised techniques in predicting the categories.

Despite the increasing complexity of the classifier, the overall results are still far from a perfect categorization. This outcome is due to the heterogeneity of the dataset, a considerable number of classes, and limited information. Recall, indeed, that we rely just on the information offered by the domain name and sequence of visits.

At last, the definition of a category for a website is, per se, a complex problem. By manually checking some labeled domains of SimilarWeb, we found some domains with misleading labels. This occurrence further complicates the engineering of an automatic model. By looking at the NPE, we observe that the correct class usually lies in the top-most positions in the returned similarity hierarchy. For instance, the NPE of the best classifier ("SemiSupervised-both") is 0.093, i.e., on average, the correct class is found in the top-2 categories (obtained as NPE times the number of categories). The NPE outcome is instrumental for supporting the classification of a domain, restricting the choice among a few options.

Finally, in Table 2 we report the result of the best configuration (i.e., "Semi-Supervised-both") on the test set. As explained in Section 6, the test set is used only to obtain an independent estimate of the performance of the chosen algorithm, and it cannot be used to compare different methods [55]. The test set results align with those in the 5-fold cross-validation set, being even slightly better on the final test set. This shows the fact that the semi-supervised method will work well, even with unseen data.

Table 1: Performance of the different classifiers obtained on the 5-fold cross validation set.

Method	Accuracy	$Precision^{macro}$	$Recall^{macro}$	$F - Measure^{macro}$	NPE
TFIDF-Supervised-DomainsName	0.359	0.342	0.358	0.313	0.181
NFA-Supervised-DomainsName	0.410	0.414	0.331	0.348	0.121
SVM-Supervised-DomainsSequence [32]	0.404	0.335	0.367	0.334	0.135
SSDN-SemiSupervised-DomainsName	0.471	0.486	0.390	0.404	0.112
SSDS-SemiSupervised-DomainsSequence	0.441	0.390	0.344	0.344	0.109
SSB-SemiSupervised-both	0.522	0.528	0.456	0.465	0.089
Naive-Most-Frequent	0.133	0.005	0.040	0.008	–
Naive-Uniform	0.033	0.064	0.063	0.061	–

Table 2: Performance obtained for the best tuned algorithm on the test dataset.

Method	Accuracy	$Precision^{macro}$	$Recall^{macro}$	$F - Measure^{macro}$	NPE
SemiSupervised-both	0.562	0.503	0.474	0.465	0.085

We now move to the detailed description of the results per class. The following figures report the different evaluation metric results. The categories sequence follows the distribution of unique labeled domains reported in Figure 4 in descending order.

Figure 5 shows the obtained Precision for each domain category, considering the six methodologies. The semi-supervised approaches (yellow, cyan, and magenta bars), produce the best results. Analyzing Precision among the classes, we observe promising values for "heterogeneous" categories, in terms of domain distributions, and for the "homogeneous" ones, with all the considered solutions. For the first group, worth to mention are "Career and Education," and "Computer and Electronics," while for the second "Travel," and "Reference," (i.e., subscription-based portals for scientific research). This outcome may suggest that these categories are peculiar both in the domain structure and in terms of user navigation targets, distinguishing them from the others. On the other hand, classes like "Recreation and Hobbies," "Books and Literature," and "People and Society," which more likely cover a large variety of topics, are more challenging to model and create a more significant number of False Positives.

Figure 6 shows the Recall measure results. These outcomes mostly confirm our previous considerations. It is worth to remark the groups with worse values in Recall measurements. In particular, "Recreation and Hobbies," "People and Society," and "Books and Literature" confirm to have a reduced capability of attracting their actual elements. Again, a low distinctiveness of these categories may play an essential role in the model generation, and so in final results.

Finally, Figure 7 and Figure 8 wrap up the aforementioned findings, by showing the F-Measure values. The semi-supervised combined methodology has, in almost all the categories, the best performances, confirming the results depicted in Table 1. Figure 8 details the results of F-Measure for the semi-supervised combined methodology (the same plots for all the analyzed methods are reported in Appendix B). It correlates the F-measure obtained for each category using a specific classifier (x-axis), with the size of the category in unique domains

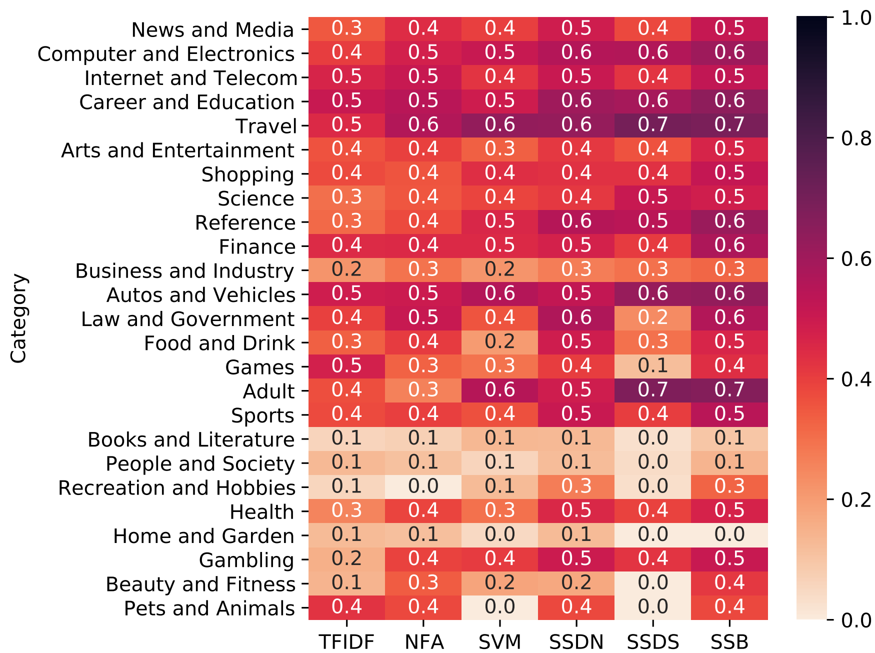


Figure 5: Precision results per class for the six methodologies.

(y-axis). The varying color and radius of the points are directly proportional to the size of each class. The dashed purple vertical line represents the macro F-Measure obtained with the Naive-Uniform algorithm. The dashed dark blue vertical line instead depicts the SSB macro F-Measure. The Figure shows that we obtain good prediction results for the most prominent classes and categories with a small number of elements, i.e., not prevalent in our observation dataset. This outcome suggests a promising behavior of the classifier in the ability to classify both prominent and underrepresented classes accurately. Comparing this Figure with Table 1, we can again appreciate how this classifier works better than simple naive approaches that predict well the most represented classes. An exception is the category “Home and Garden” for which the F-Measure score is zero. Inspecting the root cause for this outcome, we can deduce that the very low number of domains for that class and the difficulty of finding related domains in the same session, since other similar web pages are categorized differently, negatively influence the performance.

In general, the proposed approach shows encouraging results. The categories that are less capable of producing reliable predictions are also more difficult to classify for all the other methods, suggesting an intrinsic complexity of the data.

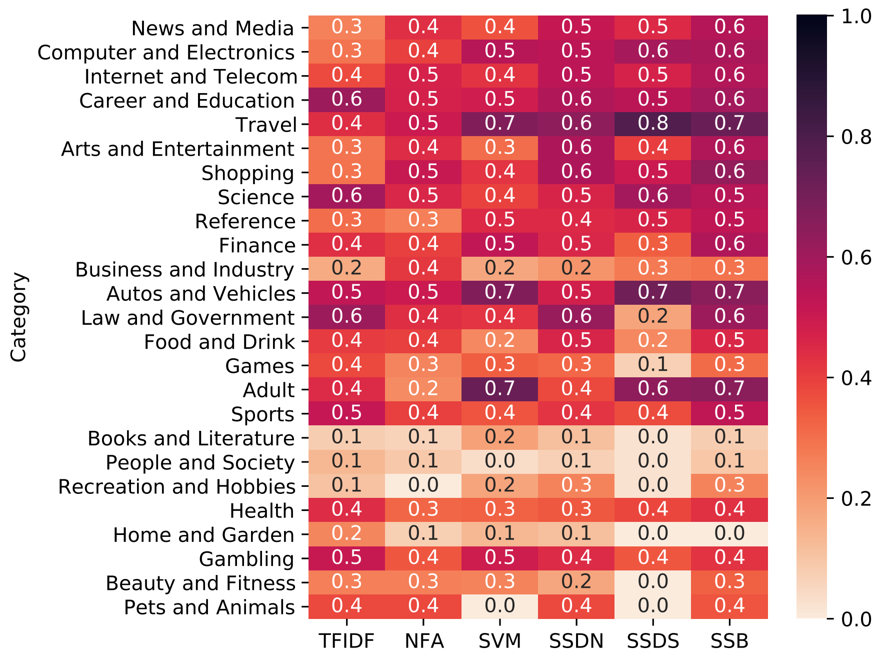


Figure 6: Recall results per class for the six methodologies.

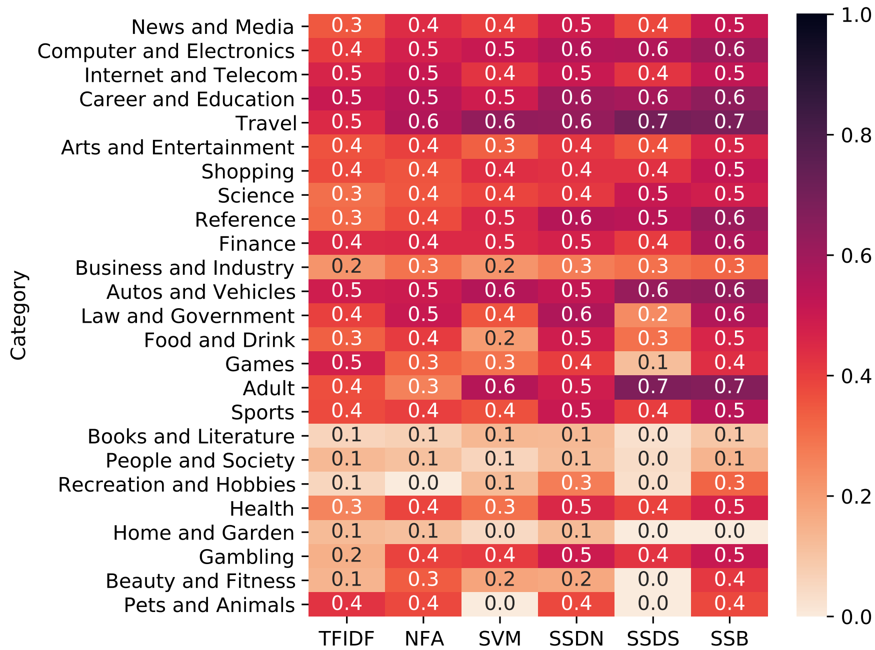


Figure 7: F-Measures results per class for the six methodologies.

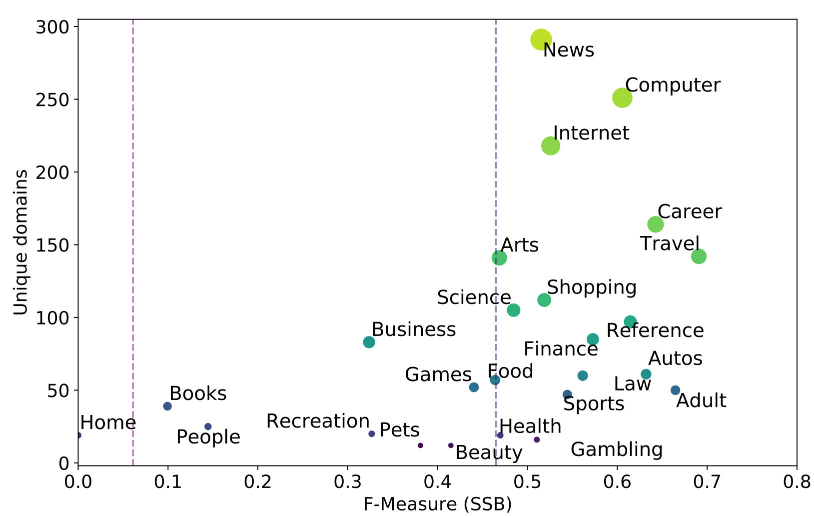


Figure 8: Scatter plot of F-Measure values obtained with the "Semi-supervised both" approach and the size of the considered categories in terms of unique domains.

7.3. Impact of categories, number of samples and time window duration

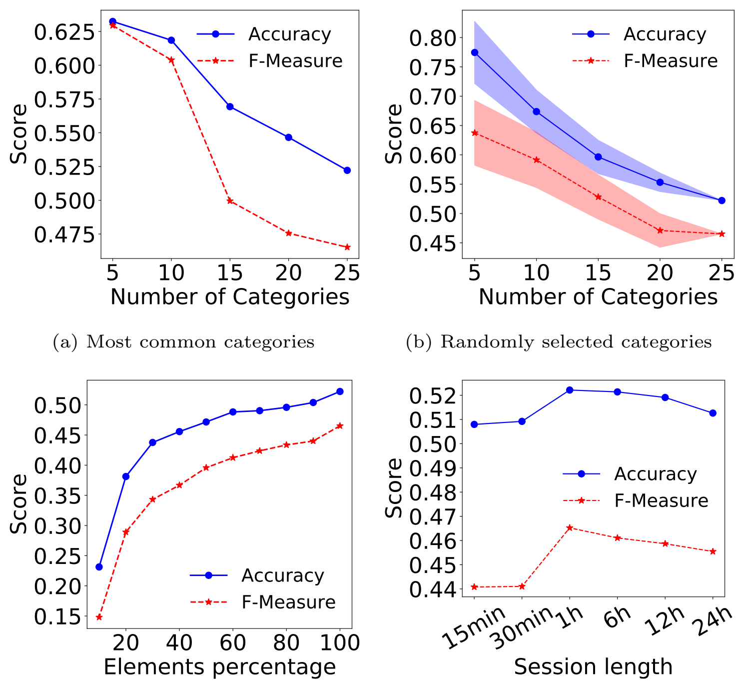
Here we discuss the sensitivity of the tuned SemiSupervised-both (SBB) method with respect to different parameters. In particular, we analyze accuracy and macro F-Measure with respect to: (i) the different number of categories, (ii) different percentage of samples, and (iii) different session length. Again, we use a 5-fold cross-validation approach.

Figure 9a shows the impact of the number of categories when considering the K most common categories according to our dataset 4. Figure 9b instead consider a random choice of K categories with 10 different runs. Curves represent the average over the 5-fold performance of accuracy and macro F-Measure. For the random category selection cases, the area represents the standard deviation on 10 independent runs around the average. The last point reports the single result on all 25 categories. As expected, the performance (both accuracy and macro average F-Measure) tends to decrease with the increase of K. The more categories we consider, the harder the classification problem becomes. Restricting to the most common K categories impacts more performance than a random choice of K categories. This is likely due to the fact that the two most popular classes, "Internet and Teleco" and"Computer and Electronic" may have some overlap and are harder to distinguish (as discussed in Section 6.3).

Figure 9c reports the learning curve when all categories are considered, but only a percentage of flows is used for training. Reducing the training set size reduces performance. Interestingly, with about 60-70% of training, the learning curve already shows signs of saturation. As expected, the results with 100% samples are a bit higher because we tuned the parameters on this exact case (Section 7.1).

Finally, Figure 9d reports the sensitivity with respect to the time window duration to consider co-occurring domains. We hypothesize that users visit similar websites in the same time-window. Here, we consider time windows different from 1 hour, reducing it to 15 minutes and 30 minutes, and increasing it to 6, 12, and 24 hours. Here we observe a smaller impact on the results. Widening the time window to more than one hour slightly reduces the performance. From the literature, we know that users browse continuously in sessions that are usually shorter than 1 hour (about 85% of them, according to [53]). Hence there are few sessions longer than one hour that can provide added value for the analysis. In addition, a too-large session duration can forcibly cause the joining of several independent sessions. Therefore we are likely aggregating sessions of uncorrelated content (e.g., considering 12 hours, we might aggregate a session in the morning with one in the evening, with likely independent topics).

Similarly, reducing the session duration reduces performance. Co-occurring domains about the same topic usually appear very close in time, and hence the performance is still good with time windows of 15 minutes. However, the results show that a 15-minutes time window is not enough to capture the effect of co-occurring domains.



(a) Most common categories

(b) Randomly selected categories

(c) All categories - percentage samples

(d) Varying session length

Figure 9: Performance results changing number of categories, percentage of used elements and time window used for the session.

8. Conclusions

In this paper, we proposed a comprehensive evaluation of classification methodologies for website domain name classification from a network observer's perspective. We considered the main category of websites as classes, and we relied on the category labels provided by the SimilarWeb dataset. We analyzed algorithms that make use of information about the lexical structure of the domains and the co-occurrence of domains in users' sessions, not inspecting web pages content. We created different representations of the data to explore different solutions and models.

We considered methodologies based on the similarity in terms of n-grams extracted from the domain names, using TFIDF and NFA. We tested a linear SVM classifier over data vectors generated by FastText. Furthermore, we proposed semi-supervised solutions to incorporate in the classifier aspects not strictly related to the labeled data. Those semi-supervised methodologies leverage graphs. The graph nodes are the domains; the weighted edges represent their similarity. We expressed the similarity between n-grams, looking at domains co-occurrence in sessions, and as a combination of both. The latter implementation is the one that offers the best performance.

There are still some limitations in our work that we can address in the future. First of all, the nature of the traces demarcates the analysis to the collected domains, excluding in-depth analysis regarding other countries web traffic. The use of SimilarWeb, as discussed in the paper, adds a specific viewpoint to the categorization. Future work could include collecting new traces and comparing the results with other domains classification sources. This work does not contemplate the use of active crawling for the analysis. This choice is justified by the difficulty of selecting a specific page, content, and how the website reacts to active crawling. However, in the future, it could be interesting to focus on crawling-based techniques and understand how they differ from our approach, weighting and merging advantages and disadvantages.

The results show the complexity of the website topic classification task. The lack of an exhaustive classification of domains calls for ingenuity in building semi-supervised solutions. However, the limited but readily available information provided by passive network traffic traces shows that a good classification is possible. To foster studies, we make available the code and data [16] we used in this paper, as a guide for future work exploring passive flow level data for the classification problem.

References

[1] P. V. Nainwani, P. Prajapati, Comparative study of web page classification approaches, International Journal of Computer Applications 179 (45) (2018) 6–9. doi:10.5120/ijca2018916994.
URL `http://www.ijcaonline.org/archives/volume179/number45/29433-2018916994`

[2] A. Y. Halevy, N. Ashish, D. Bitton, M. Carey, D. Draper, J. Pollock, A. Rosenthal, V. Sikka, Enterprise information integration: successes, challenges and controversies, in: Proceedings of the 2005 ACM SIGMOD international conference on Management of data, ACM, 2005, pp. 778–787.

[3] L. Blanco, N. Dalvi, A. Machanavajjhala, Highly efficient algorithms for structural clustering of large websites, in: Proceedings of the 20th international conference on World wide web, ACM, 2011, pp. 437–446.

[4] H. A. Sleiman, R. Corchuelo, Tex: An efficient and effective unsupervised web information extractor, Knowledge-Based Systems 39 (2013) 109–123.

[5] C. Kohlschütter, P.-A. Chirita, W. Nejdl, Utility analysis for topically biased pagerank, in: Proceedings of the 16th international conference on World Wide Web, ACM, 2007, pp. 1211–1212.

[6] C.-C. Huang, S.-L. Chuang, L.-F. Chien, Liveclassifier: creating hierarchical text classifiers through web corpora, in: Proceedings of the 13th international conference on World Wide Web, ACM, 2004, pp. 184–192.

[7] Z. Chen, O. Wu, M. Zhu, W. Hu, A novel web page filtering system by combining texts and images, in: 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06), IEEE, 2006, pp. 732–735.

[8] A. Broder, M. Fontoura, V. Josifovski, L. Riedel, A semantic approach to contextual advertising, in: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 2007, pp. 559–566.

[9] M.-Y. Kan, H. O. N. Thi, Fast webpage classification using url features, in: Proceedings of the 14th ACM international conference on Information and knowledge management, ACM, 2005, pp. 325–326.

[10] O.-W. Kwon, J.-H. Lee, Web page classification based on k-nearest neighbor approach, in: Proceedings of the fifth international workshop on on Information retrieval with Asian languages, ACM, 2000, pp. 9–15.

[11] A. S. Patil, B. Pawar, Automated classification of web sites using naive bayesian algorithm, in: Proceedings of the international multiconference of engineers and computer scientists, Vol. 1, 2012, pp. 14–16.

[12] D. F. Witmer, The association (of). internet. researchers: Formed to support scholarship in and of the internet, Information, Communication & Society 2 (3) (1999) 368–370.

[13] L. Vassio, D. Giordano, M. Trevisan, M. Mellia, A. P. C. da Silva, Users' fingerprinting techniques from tcp traffic, in: Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks, Big-DAMA '17, ACM, New York, NY, USA, 2017, pp. 49–54. doi:10.1145/3098593.3098602.
URL `http://doi.acm.org/10.1145/3098593.3098602`

[14] P. Vallina, V. Le Pochat, A. Feal, M. Paraschiv, J. Gamba, T. Burke, O. Hohlfeld, J. Tapiador, N. Vallina-Rodriguez, Mis-shapes, mistakes, misfits: An analysis of domain classification services, in: Proceedings of the ACM Internet Measurement Conference, IMC '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 598–618. doi:10.1145/3419394.3423660.
URL `https://doi.org/10.1145/3419394.3423660`

[15] X. Zhu, Z. Ghahramani, J. Lafferty, Semi-supervised learning using gaussian fields and harmonic functions, in: Proceedings of the Twentieth International Conference on International Conference on Machine Learning, ICML'03, AAAI Press, 2003, pp. 912–919.
URL `http://dl.acm.org/citation.cfm?id=3041838.3041953`

[16] A. Faroughi, A. Morichetta, L. Vassio, F. Figueiredo, M. Mellia, R. Javidan, A partially labeled dataset for domain name classification with semi-supervised learning, https://smartdata.polito.it/domains-web-users/, accessed: 2020-02-28 (2020).

[17] D. Mladenic, Turning yahoo into an automatic web-page classifier (1998).

[18] K. Golub, A. Ardö, Importance of html structural elements and metadata in automated subject classification, in: A. Rauber, S. Christodoulakis, A. M. Tjoa (Eds.), Research and Advanced Technology for Digital Libraries, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 368–378.

[19] D. Shen, Z. Chen, Q. Yang, H.-J. Zeng, B. Zhang, Y. Lu, W.-Y. Ma, Web-page classification through summarization, in: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 2004, pp. 242–249.

[20] Y. Wei, W. Wang, B. Wang, B. Yang, Y. Liu, A method for topic classification of web pages using lda-svm model, in: Z. Deng (Ed.), Proceedings of 2017 Chinese Intelligent Automation Conference, Springer Singapore, Singapore, 2018, pp. 589–596.

[21] A. Colorni, M. Dorigo, V. Maniezzo, et al., Distributed optimization by ant colonies, in: Proceedings of the first European conference on artificial life, Vol. 142, Cambridge, MA, 1992, pp. 134–142.

[22] O.-W. Kwon, J.-H. Lee, Text categorization based on k-nearest neighbor approach for web site classification, Information Processing & Management 39 (1) (2003) 25–44.

[23] V. de Boer, M. van Someren, T. Lupascu, et al., Classifying web pages with visual features., in: WEBIST (1), 2010, pp. 245–252.

[24] A. P. Asirvatham, K. K. Ravi, A. Prakash, et al., Web page classification based on document structure, in: IEEE National Convention, 2001.

[25] M. Kovacevic, M. Diligenti, M. Gori, V. Milutinovic, Visual adjacency multigraphs-a novel approach for a web page classification, in: Proceedings of SAWM04 workshop, ECML2004, 2004.

[26] S. Matic, C. Iordanou, G. Smaragdakis, N. Laoutaris, Identifying sensitive urls at web-scale, in: Proceedings of the ACM Internet Measurement Conference, IMC '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 619–633. doi:10.1145/3419394.3423653.
URL https://doi.org/10.1145/3419394.3423653

[27] H. Utard, J. Fürnkranz, Link-local features for hypertext classification, in: Semantics, web and mining, Springer, 2005, pp. 51–64.

[28] E. Baykan, M. Henzinger, L. Marian, I. Weber, Purely url-based topic classification, in: Proceedings of the 18th international conference on World wide web, ACM, 2009, pp. 1109–1110.

[29] E. Baykan, M. Henzinger, L. Marian, I. Weber, A comprehensive study of features and algorithms for url-based topic classification, ACM Transactions on the Web (TWEB) 5 (3) (2011) 15.

[30] C. Cortes, V. Vapnik, Support-vector networks, Machine learning 20 (3) (1995) 273–297.

[31] I. Hernández, C. R. Rivero, D. Ruiz, R. Corchuelo, Cala: An unsupervised url-based web page classification system, Knowledge-Based Systems 57 (2014) 168–180.

[32] W. Jiang, K. Yu, X. Wu, F. Wei, B. Wang, Domain2vec: Vector representation of mobile server's domain based on mobile user visiting sequences, in: 2018 IEEE 3rd International Conference on Big Data Analysis (ICBDA), 2018, pp. 232–236. doi:10.1109/ICBDA.2018.8367683.

[33] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, arXiv preprint arXiv:1301.3781 (2013).

[34] K. Sparck Jones, A Statistical Interpretation of Term Specificity and Its Application in Retrieval, Taylor Graham Publishing, GBR, 1988, p. 132–142.

[35] S. E. Robertson, K. S. Jones, Relevance weighting of search terms, Journal of the American Society for Information science 27 (3) (1976) 129–146.

[36] C. D. Manning, P. Raghavan, H. Schütze, Scoring, term weighting, and the vector space model, Cambridge University Press, 2008, p. 100–123. doi:10.1017/CBO9780511809071.007.

[37] H. Balinsky, A. Balinsky, S. Simske, Document sentences as a small world, in: 2011 IEEE International Conference on Systems, Man, and Cybernetics, IEEE, 2011, pp. 2583–2588.

[38] B. Dadachev, A. Balinsky, H. Balinsky, S. Simske, On the helmholtz principle for data mining, in: 2012 Third International Conference on Emerging Security Technologies, 2012, pp. 99–102.

[39] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in neural information processing systems, 2013, pp. 3111–3119.

[40] F. Morin, Y. Bengio, Hierarchical probabilistic neural network language model., in: Aistats, Vol. 5, Citeseer, 2005, pp. 246–252.

[41] M. U. Gutmann, A. Hyvärinen, Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics, Journal of Machine Learning Research 13 (Feb) (2012) 307–361.

[42] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, T. Mikolov, Fasttext. zip: Compressing text classification models, arXiv preprint arXiv:1612.03651 (2016).

[43] M. Belkin, P. Niyogi, Towards a theoretical foundation for laplacian-based manifold methods, Journal of Computer and System Sciences 74 (8) (2008) 1289–1308.

[44] J. E. van Engelen, H. H. Hoos, A survey on semi-supervised learning, Machine Learning 109 (2) (2020) 373–440. doi:10.1007/s10994-019-05855-6. URL `https://doi.org/10.1007/s10994-019-05855-6`

[45] Y. Chong, Y. Ding, Q. Yan, S. Pan, Graph-based semi-supervised learning: A review, Neurocomputing 408 (2020) 216 – 230. doi:https://doi.org/10.1016/j.neucom.2019.12.130.

[46] T. Jebara, J. Wang, S.-F. Chang, Graph construction and b-matching for semi-supervised learning, in: Proceedings of the 26th annual international conference on machine learning, ACM, 2009, pp. 441–448.

[47] M. S. Baghshah, F. Afsari, S. B. Shouraki, E. Eslami, Scalable semi-supervised clustering by spectral kernel learning, Pattern Recognition Letters 45 (2014) 161–171.

[48] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, B. Schölkopf, Learning with local and global consistency, in: Advances in neural information processing systems, 2004, pp. 321–328.

[49] M. Husák, M. Cermák, T. Jirsík, P. Celeda, Network-based https client identification using ssl/tls fingerprinting, in: 2015 10th International Conference on Availability, Reliability and Security, 2015, pp. 389–396. doi:10.1109/ARES.2015.35.

[50] M. Trevisan, A. Finamore, M. Mellia, M. Munafo, D. Rossi, Traffic analysis with off-the-shelf hardware: Challenges and lessons learned, IEEE Communications Magazine 55 (3) (2017) 163–169.

[51] I. N. Bermudez, M. Mellia, M. M. Munafo, R. Keralapura, A. Nucci, Dns to the rescue: Discerning content and services in a tangled web, in: Proceedings of the 2012 Internet Measurement Conference, 2012, pp. 413–426.

[52] L. Vassio, H. Metwalley, D. Giordano, The exploitation of web navigation data: Ethical issues and alternative scenarios, in: F. D'Ascenzo, M. Magni, A. Lazazzara, S. Za (Eds.), Blurring the Boundaries Through Digital Innovation, Springer International Publishing, Cham, 2016, pp. 119–129.

[53] L. Vassio, I. Drago, M. Mellia, Z. B. Houidi, M. L. Lamali, You, the web, and your device: Longitudinal characterization of browsing habits, ACM Trans. Web 12 (4) (Sep. 2018). doi:10.1145/3231466. URL `https://doi.org/10.1145/3231466`

[54] L. Vassio, M. Mellia, F. Figueiredo, A. P. C. da Silva, J. M. Almeida, Mining and modeling web trajectories from passive traces, in: 2017 IEEE International Conference on Big Data (Big Data), 2017, pp. 4016–4021. doi:10.1109/BigData.2017.8258416.

[55] B. D. Ripley, Pattern Recognition and Neural Networks, Cambridge University Press, 1996. doi:10.1017/CBO9780511812651.

[56] E. Hüllermeier, J. Fürnkranz, On minimizing the position error in label ranking, in: European Conference on Machine Learning, Springer, 2007, pp. 583–590.

[57] F. Chollet, et al., Keras, `https://github.com/fchollet/keras` (2015).

Appendix A. Parameter configurations for the classification methodologies

Table A.3 wraps up the parameters selected for the different methodologies explored in the paper. The choice of the resulting values results from the 10-fold cross-validation tuning process or our domain knowledge.

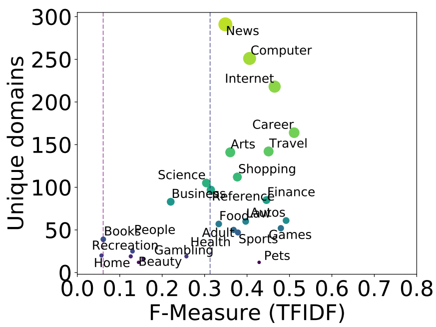
Table A.3: Employed classification methodologies and their parameters.

Method	Parameters
TFIDF-Supervised-DomainsName	n-grams in $[3-9]$
NFA-Supervised-DomainsName	n-grams in $[3-5]$
SVM-Supervised-DomainsSequence [32]	skip-gram, dimension = 100, windows = 5 SVM with linear kernel
SSDN-SemiSupervised-DomainsName	$\epsilon_N = 0.98$, n-grams in $[3-5]$
SSDS-SemiSupervised-DomainsSequence	skip-gram, dimension = 100, windows = 5 $\epsilon_N = 0.985$, $\epsilon_S = 0.47$
SSB-SemiSupervised-both	skip-gram, dimension = 100, windows = 5 $\epsilon_N = 0.985$, $\epsilon_S = 0.5$, n-grams in $[3-6]$
LSTM-Supervised-DomainsName	embedding layer: size 64 LSTM layer: 128 memory units dense output layer: 25 neurons activation function: softmax Loss function: $categorical - crossentropy$ optimizer: $Adam$
Naive-Most-Frequent	–
Naive-Uniform	–

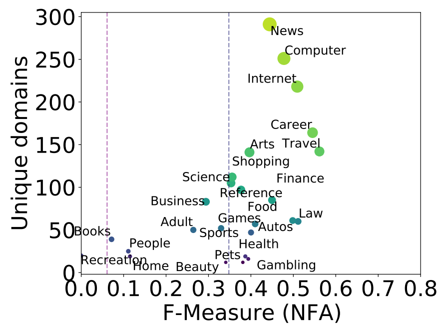
Appendix B. F-Measure distribution over the 25 SimilarWeb categories, for the analyzed algorithms

The scatter plots in Figure B.10 report the F-Measure results for the considered classifiers, correlating them with the size of the categories in terms of unique domains. Figure B.10a represents the TF-IDF approach, Figure B.10b reports NFA, Figure B.10c shows SVM results, Figure B.10d and Figure B.10e refer to SSDN and SSDS respectively. Finally, Figure B.10f reports our reference algorithm SSB.

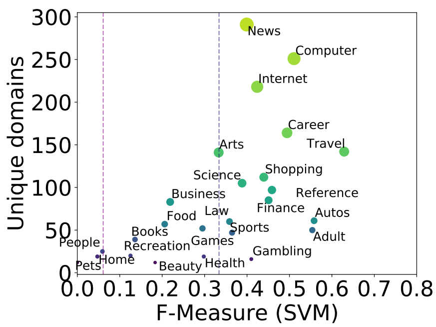
The plots show the F-Measure values on the x-axis and, on the y-axis, the number of unique domains per category. All the plots have an x-axis range going from 0.0 to 1.0 to facilitate comparability. Furthermore, there are two dashed vertical lines. The purple one shows the macro F-Measure score for the *Naive-Uniform* approach. The dark blue vertical line represents the macro F-Measure value for the depicted algorithm. Starting from the similarities, it is noticeable how all the algorithms struggle to classify rare categories correctly. In particular, "Home and Garden," "Books and Literature," and "People and Society" seem to be the classes that are the most difficult to predict. The TF-IDF method, in Figure B.10a, have all the F-Measure scores in the range $[0.0, 0.5]$. NFA does a little bit better, especially for "Travel," "Career and Education," "Law and Government," and "Internet and Telecom." The range is $[0.0, 0.6]$. Figure B.10c shows a behavior similar to NFA, but on different categories, namely "Autos and Vehicles", "Adult", and "Computer and Electronics". Interesting is the result for "Adult", that had poor scores with TF-IDF and NFA. SSDS and SSDN achieve better results. However, SSB outperforms the other techniques, with F-Measure scores shifted towards higher values.



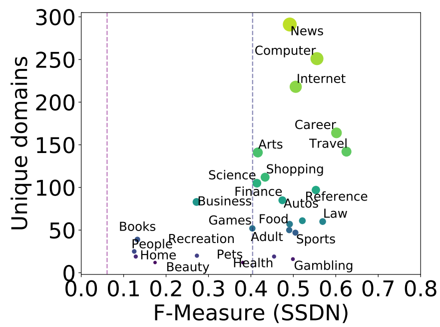
(a) TFIDF Distribution



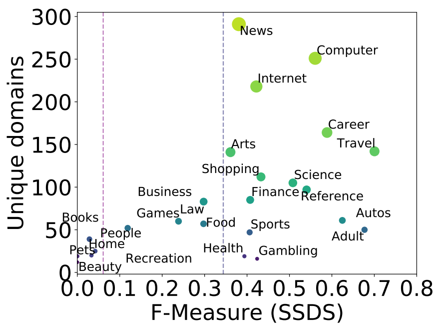
(b) NFA Distribution



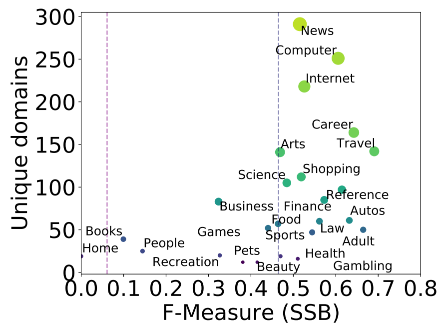
(c) SVM Distribution



(d) SSDN Distribution



(e) SSDS Distribution



(f) SSB Distribution

Figure B.10: Scatter plots of F-Measure values and the size of the considered categories in terms of unique domains, for the considered classifiers.