Computer Networks 198 (2021) 108357

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Natural language processing for web browsing analytics: Challenges, lessons learned, and opportunities

Daniel Perdices[a,*], Javier Ramos[a], José L. García-Dorado[a], Iván González[a,b], Jorge E. López de Vergara[a,b]

[a] *Department of Electronic and Communication Technologies, School of Engineering, Universidad Autónoma de Madrid, Spain*
[b] *Naudit High Performance Computing and Networking, Spain*

ARTICLE INFO

Keywords:
Web browsing
Users analytics
Natural language processing
Deep learning
Traffic monetization
Internet monitoring

ABSTRACT

In an Internet arena where the search engines and other digital marketing firms' revenues peak, other actors still have open opportunities to monetize their users' data. After the convenient anonymization, aggregation, and agreement, the set of websites users visit may result in exploitable data for ISPs. Uses cover from assessing the scope of advertising campaigns to reinforcing user fidelity among other marketing approaches, as well as security issues. However, sniffers based on HTTP, DNS, TLS or flow features do not suffice for this task. Modern websites are designed for preloading and prefetching some contents in addition to embedding banners, social networks' links, images, and scripts from other websites. This self-triggered traffic makes it confusing to assess which websites users visited on purpose. Moreover, DNS caches prevent some queries of actively visited websites to be even sent. On this limited input, we propose to handle such domains as words and the sequences of domains as documents. This way, it is possible to identify the visited websites by translating this problem to a text classification context and applying the most promising techniques of the natural language processing and neural networks fields. After applying different representation methods such as TF–IDF, Word2vec, Doc2vec, and custom neural networks in diverse scenarios and with several datasets, we can state websites visited on purpose with accuracy figures over 90%, with peaks close to 100%, being processes that are fully automated and free of any human parametrization.

1. Introduction

The different actors in the Internet arena observe how users interact both with service providers and between them while they browse, chat, download a file or watch a video, among other activities. Such interactions generate large amounts of data at different points of measurement and scales. Some of these actors are exploiting this data with evident success, whereas others are still in the early stages. For instance, search engines such as Google and Bing exploit the search queries of their users, generating profits of billions of dollars [1]. Also, marketing companies like SimilarWeb or Amazon's Alexa [2] sell browsing statistics gathered through plugins and toolbars. While these actors are monetizing their data, others such as ISPs, or DNS proxies/providers have still open opportunities to take advantage of the value of their data.

Actually, both research and industry have already paid attention to profile users' browsing patterns, such as the websites they visit, as a valued piece of information [3]. As an example, it becomes apparent that the identification of a competitor attracting the interest among the customers of a given company is of paramount interest to such a company, so it can react accordingly. Another example is the impact that a certain advertising campaign achieves in a specific territory or geographical region, i.e., the set of users in this territory that has accessed a certain web after being exposed to a marketing campaign. Besides, and shifting to security issues, the popularity rankings of websites or the number of visits a website receives can be considered as a useful detector of anomaly behavior. That is, when unpopular or non-existing websites tend to be the most visited in a given network. In this scenario, ISPs have the perfect occasion to place in the web-analytics marketing arena.

Needless to say, user privacy rights are of paramount importance, and traffic encryption mechanisms developed in recent times are key tools to protect users' data from a wide range of attacks [4]. In pursuit of a balance between the provision of a service and data collecting, some governments and agencies, such as the United Nations or European Union, have promoted laws to regulate the quantity and nature of

* Corresponding author.
E-mail addresses: daniel.perdices@uam.es (D. Perdices), javier.ramos@uam.es (J. Ramos), jl.garcia@uam.es (J.L. García-Dorado), ivan.gonzalez@uam.es (I. González), jorge.lopez_vergara@uam.es (J.E. López de Vergara).

https://doi.org/10.1016/j.comnet.2021.108357
Received 26 February 2021; Received in revised form 7 June 2021; Accepted 21 July 2021
Available online 27 July 2021

the data that is gathered from users [5]. In addition to this, some ISPs follow a policy of monitoring only those users that voluntarily want to be part of commercial studies through online forms. In this way, users that expressly do not desire to be part of studies are not considered.

In this light, ISPs are collecting anonymized and aggregated data, which can be used to reconstruct the browsing activity of the users. To this end, HTTP traffic was traditionally used to infer the sites visited. However, very soon the research community realized that the trivial approach of an HTTP sniffer does not suffice for this problem, as traffic encryption was already becoming a common practice [6]. The first answer to this limitation by the research community was to turn its attention to DNS traffic or TLS' Server Name Indication (SNI) field. Such a field contains the domain name of the host that the client is connecting to, and servers use it to negotiate the certificate used in each session. As another alternative, the research community focused on how extended Netflow characteristics are useful to label connections to websites.

Unfortunately, this revealed other significant problems. Modern websites are designed [7] in such a way that once a user actively visits a web, many other HTTP and DNS connections are triggered in the background (e.g., banners, ads, social networks' links, Javascript scripts, prefetched and preloaded links). This makes it difficult to assess whether a website was visited on purpose or not. Note that this is key to create browsing statistics as HTTP or DNS traffic from no actively visited websites must not be considered. In addition to this, the DNS cache clouds the global vision of the sites visited by users as browsers may access sites using cached addresses this way not generating DNS queries when websites are actually visited.

In this light, let us remark that the common factor of these input sources is the same: a partial list of domains (including not only the actively visited website but also others) obtained while the users browse the Internet. Therefore, we propose to use any of these sources of the same information according to the particularities of each specific environment, but avoiding non-common information or parameters. For example, a potentially interesting field such as the DNS's TTL, which stands for the time that a name and its resolved address is valid, is not considered for the sake of homogeneity. In fact, TTL values can be manipulated [8].

Facing this scenario of limited inputs, as a novel approach, we realized that domains are nothing but words likewise the lists of domains can be seen as text documents. In this way, we propose to translate the problem of users' browsing profiling into a text classification context. Certainly, with a high number of classes (the possible visited websites). In particular, some of the most promising text classification algorithms and our problem, user browsing profiling, share multiple points in common. Both use incomplete information and while they aim at learning word associations to suggest additional words or synonyms for a partial sentence (e.g., Google's autocomplete function), we propose to search for relationships between the list of occurrences of website domains, our corpus of text, and the visited websites to predict. The observation of this link between problems and the particularities of the limited input of our problem allows us to approach it with a novel and promising perspective. Therefore, all the full potential behind areas such as Natural Language Processing (NLP) and Artificial Neural Networks (ANNs) have been reviewed for a problem that intuitively does not seem related.

The results shown throughout this paper confirms the usefulness of the approach. We have learned that there is no perfect model for the problem, but diverse approaches depending on the availability of processing time, the use of high-performance resources such as GPUs or Tensor Processing Units (TPUs) for training models, and the number of domains under study (i.e., all the Internet or a few domains of interest, for example, only TV show websites to measure popularity in a marketing campaign). In particular, we highlight the result of techniques, such as Term Frequency Inverse Document Frequency (TF–IDF) [9], Word2Vec [10], Doc2vec [11] and a custom neural network model with weighted accuracy over 90%, often close to 100% for diverse scenarios and data, being processes fully automated and free of any human parametrization and interaction.

The rest of the paper is organized as follows: in Section 2, we review the state of the art putting into perspective our contributions. Next, Section 3 defines the problem formally while Section 4 presents the methodologies used to address it. Afterward, Section 5 covers the data acquisition, and Section 6 studies the performance of the set of approaches in such data. Later, Section 7 discusses the main lessons and contributions of this work in relation to the results. Finally, Section 8 concludes the paper and provides some future lines of work.

2. State of the art

We first present the challenges that the Internet community faces in the task of extracting website visits from traffic measurements. Then, we focus on how novel approaches from the machine learning field can be useful in this task.

2.1. Traffic measurements for browsing analytic

The inspection of HTTP traffic and, specifically, its field HOST was the natural approach to relate traffic to visited websites. For instance, authors in [12] classify and identify the traffic using density-based spatial clustering of applications with noise (DBSCAN) clustering algorithm over the URLs, building coarse categories depending on the service such as advertising or video streaming.

However, the advent of HTTPS rendered this approach useless without Man-In-The-Middle proxies, which are unfeasible in many deployments due to privacy concerns. As an alternative, the monitoring community proposed focusing on the DNS protocol to reveal the traffic behind an HTTPS flow [13,14] and then, perform a correlation between DNS and HTTP traffic [7,15]. Nevertheless, a fraction of users can choose other DNS servers than those provided by their ISP and the use of DNS encryption and DNS over HTTPS (DoH) is gaining popularity [16,17]. Moreover, most DNS resolvers and clients implement a cache for DNS traffic where the associated IP address of a given domain name is temporarily stored. As Time To Live (TTL) for the cache entry can be long [8], chances are that a point of presence monitoring traffic cannot see clients' DNS queries although they are effectively visiting a website.

In this scenario, the Internet community's attention turned into the inspection of some fields of HTTPS. Monitoring HTTPS traffic can provide some insights on the browsed sites by means of Server Name Indication (SNI) field of TLS protocol. This field contains the name of the host that the client is connecting to and serves to negotiate the certificate used in each connection. This information is presented before the TLS handshake and allows for the coexistence of multiple HTTPS sites using the same IP address or addresses. Such a scenario is commonly found on Content Delivery Networks (CDNs) and cloud and hosting services.

However, the use of SNI is not mandatory and although it is infrequent may not be used in the HTTPS connection. Moreover, recently some companies such as Cloudflare and Mozilla[1] are promoting the use of encrypted SNIS (ESNIs) [18] and Encrypted Client Hello (ECH) TLS messages which avoids exposing this kind of information.

As a last bump in the road, the combination of TCP and TLS is progressively being substituted by UDP and QUIC [19], which has become the standard transport mechanism for HTTP/3. Additionally to the problems presented by TLS, QUIC provides full encryption for all traffic and presents the 0-RTT mechanism whereby previously established connections may remain cached for a time period, avoiding

[1] https://blog.cloudflare.com/encrypted-client-hello/.

initial handshake in case a connection is reactivated in a similar way that DNS cache does.

An alternative approach to the inspection of HTTP or DNS traffic for extracting visits has been analyzing network flow characteristics. This is the case of works such as [3,18,20–25]. Measures such as packet-size frequencies, total-transmission time, and sizes, among other features, are exploited to correlate flows and websites to identify visits and security vulnerabilities. In addition, features related to DNS traffic such as location, resolver, platform were useful to classify. On these flow features, the authors apply different Machine Learning techniques, such as Support Vector Machine, k-Nearest Neighbors algorithms or Random Forest. Moreover, Deep Learning techniques such as convolutional neural networks (CNN) are considered. Mechanisms based on flow features on fully encrypted traffic tend to give less accurate results. For example, an F-Score higher than 0.8 for 80% of considered domains according to the authors in [18]. Several are the downsides of this approach. First, it is characterizing how web servers and communications work, not the websites themselves. This way, a change in the server network, software version, or transport-layer protocol may have an impact on the model. Second, they need to collect much traffic, which entails scale and computational problems. Third, they are especially sensitive to missing or delayed packets. Finally, such flow-based approaches require flows to expire before any analysis is carried out.

While the application of all the above-introduced ideas circumvent the encryption problem at different levels and return a list of the domains present in the traffic, the final target of obtaining the websites that a user intentionally visited is not an immediate task. Modern websites are currently designed in such a way that include a set of external resources such as images, styles, banners, ads, or Javascript scripts that generate both DNS and HTTP traffic without being specifically requested by the user. With a similar impact, browsers preload content and prefetch links in order to speed up browsing which, in turn, generates non-requested traffic. This phenomenon has been referred to as the tangled web problem [15].

In sum, whether for one cause or another, it is wrong to consider that a user has visited a website, simply, by having found DNS, HTTP traffic, or equivalent flows from such a website or domain. While this dysphoria between the purposefulness of a visit may not be significant for other issues, it is insufficient for providing precise results for the monetization of the data.

To address this problem, the authors in [26] proposed the idea of building weighted footprints. A weighted footprint is the set of domains that are requested upon site loading ordered by relevance. By searching such a list of domains in traffic, or a fraction of them according to the observed TTLs, the effectively visited pages were inferred. Although this mechanism has proven to be successful, three problems arise nowadays. First, ISPs and other DNS providers are manipulating TTL values by means of DNS Transparent Proxies [8]. Also regarding TTL, the use of DNS encryption techniques at the same time that choosing non-ISP's servers render TTL unavailable. Moreover, the emulation of the DNS cache or other user-level browsing structures comes together with a high demand for memory and resources that scales with the number of users of the network and a complex parametrization. These problems reduced the scope of application of such footprint-based proposals.

As a conclusion and for the sake of the independence of a specific scenario, we consider that the general input that a website users' profiling tool may expect is the set of domains that DNS, QUIC/TLS' SNI or mechanisms based on extended flows can extract. Over such a common piece of information, we lay the foundations of our approach: the observation of the fact that domains can be considered as words and the set of domains as a document. In this way, we are proposing to translate inputs into a text classification problem and exploiting the advances in the natural language processing and artificial neural networks fields to determine intentional visits to websites.

2.2. Natural language processing techniques

NLP is a centenary field that still attracts attention, where the most novel machine learning techniques are significantly contributing. Since the early attempts at the first half of the XX century, the first setback was the problem of having categorical variables with a potentially unlimited number of values, the well-known *curse of dimensionality* phenomenon. Soon, the research community agreed that the key was to build some kind of smart document representation, e.g., a low-dimensional vector of characteristics, on which, subsequently, apply the most diverse set of methods to solve a classification or regression problem [27]. In this context, classical approaches such as Bag of Words [28] (BoW) representation arise, in which documents are represented just as a set of words along with the frequencies of each word in the sentence. Similarly, TF–IDF is built on top of the same representation; it only considers the frequency of the words and not the order, but with a more complex weighting procedure whereby the concept of corpus and collection of documents were introduced. This idea was particularly useful for designing Information Retrieval (IR) systems such as search engines where documents have to be found and ranked with respect to the relevance of the document [29]. Lately, this proved to be a useful and simple approach but not powerful enough to fully capture the meaning of a sentence, where order and context play an important role.

Modern representations, known as embeddings, aim at capturing both this idea of relevance and other important facts such as the order and the context of a word. Word2vec [10] is one of the most prominent unsupervised algorithms that creates a vector representation using the context of a word. The idea of the authors was in fact that if two words can fit in the same place of a sentence, they must be similar in some sense. As a further refinement, Doc2vec [11] was proposed as is an extension towards documents. This way, this concept focuses on creating vector representations of sentences, paragraphs, or documents, rather than lists of ungrouped sentences. This very same idea has been extended to very different fields such as graphs [30] or user modeling [31].

To incorporate order into such embedding, recurrent neural networks emerged as an option. In particular, Long Short-Term Memory cells [32] are the usual approach to process sequences of texts or time series. These kinds of models employ feedback connection to retain a memory or state that is trained to capture the structure of sequences, which is significantly useful for text generation [33], automatic translation [34], or sentiment analysis [35]. All the above-introduced mechanisms have been considered in the process of this work and given their relevance, they will be further explained in our particular context of utilization in the next section.

Finally, we remark that NLP and related fields have recently begun to be applied in the area of communications. As some examples, the authors in [36] searched for anomalous patterns in HTTP by means of a word2vec approach. Similarly, in [37] is studied how to recognize identical users across different social media platforms. NLP approaches were put into practice to find relationships between the words, categories, and users. In [38], the authors explore a method for detecting abnormal comments in e-commerce and review sites. Focusing on DNS-based applications, the authors in [39] have applied embeddings to separate malicious DNS queries from regular ones. In particular, they aim at detecting botnets by querying for pseudo-random domains to bypass black-list security mechanisms in data exfiltration scenarios.

3. Problem statement

The problem that we address is to determine if a user has visited a given website on purpose in a given time interval by exploiting the sources of information available on the heterogeneous Internet traffic. In such a statement, two key points stand out: the available information and the intentionality of the website visits.

Regarding the former, given the previously described state-of-the-art challenges, the following sources of information for gathering website domains are viable:

1. The field HOST of HTTP when it is not encrypted or HTTP proxy data is available.
2. The QNAME fields of DNS both question and answer when they are not encrypted or the logs of the ISPs' DNS server are available.
3. The TLS/QUIC field SNI when certificates are negotiated in each connection providing that it is not encrypted.
4. After constructing extended-features flows on the full traffic aggregate, to use them to label connections and domains by inference with a given precision.

We note that the diversity of these sources will provide significant audience coverage, while the specific number will depend on users' own configuration. That is, the set of users that after changing ISP's DNS server and encrypt this traffic and choosing a browser that encrypts ESNI field (e.g., Tor) visits an HTTPS website assuming that traffic aggregate can be gathered (for computational or storage reasons, for example) will comprise the uncovered audience. However, while some users may both encrypt DNS connections and pick a DNS server different from the ISP's one, the majority of users do not. Similarly, the possibility of finding encrypted SNI is low today. According to [18], ESNI fields were mostly absent in their measurement campaign. Anyhow, some limitations in audience coverage can be simply equivalent to the impact that those search engines that guarantee not to exploit users' data, such as DuckDuckGo [40], have on Google search engine's coverage. Even more, coverage can be directly limited simply due to the fact that operators allow users to demand not to be monitored.

This way, regardless of the particularities, the common input data on which the methods can be applied is a sequence of domain names such as:

$$\{\text{abs.twimg.com, video.twimg.com, twitter.com, } \dots\}, \tag{1}$$

which is the one a user generates when Twitter is visited.

Formally, let D be our vocabulary, i.e., the set of all domains. In addition, all the possible sequences may have many more elements outside D, all these words, which are essentially strings gathered in the traffic, will be called $\mathbb{S}$. The first issue is the dimension of D. Typically, classification problems consider only a few classes, but, in this case, the number of classes is potentially infinite. Furthermore, the different subdomains we use as predictors are also potentially infinite and they come in sequence with no fixed length. Here is an example of what our model f should do

$$\{\text{abs.twimg.com, video.twimg.com, twitter.com, } \dots\} \xrightarrow{f} \text{twitter.com} \tag{2}$$

In order to properly specify the problem, we will call the sequence of subdomains S_d, where $S_d = \{s_i\}_{i=1}^{N(d)}$, the domain is $d \in D$, and $N(d)$ is the length of the sequence, which depends on the domain. Although this sequence S_d may seem constant, we will see cases where there is some random behavior, mainly two cases: first, the domains are not queried due to cache effects, persistent connections or 0-RTT mechanism, such as what happens in this example

$$\{\text{abs.twimg.com, twitter.com, } \dots\} \xrightarrow{f} \text{twitter.com} \tag{3}$$

and, second, some parts of the domain names might be random, like

$$\underbrace{\text{ipv4-c070-mad001-ix.1}}_{\text{random part}} . \underbrace{\text{oca.nflxvideo.net}}_{\text{relevant part}}, \tag{4}$$

where the first piece is just a random string related to the current local CDN that we are connecting to, and the rest totally identifies the traffic since it is a Netflix domain.

It may be thought to be an easy problem as the main domain d is likely part of S_d, but the problem is that, for other domain d', it may also happen that $d \in S_{d'}$. Here is emerging the second key point of the posed problem, not all the gathered traffic comes from deliberate visits.

For instance, Facebook appears on many other pages since it is always referenced whenever a button of "Sign in with Facebook" is placed without an intentional visit by the user. This makes the problem way more difficult and, unfortunately, is more common than expected as the previous section stated. That is, unsolicited traffic is usually triggered by banners, ads, Javascript scripts as well as by prefetched and preloaded links techniques in most of the current websites.

To sum up, we look forward to some classifier

$$\begin{aligned} f : \mathbb{S}^N &\to [0,1]^{|D|} \\ S = \{s_1, \dots, s_N\} &\to f(S) = [P(d|S_d = S)]_{d\in D} \end{aligned} \tag{5}$$

where S is the input sequence, such as {abs.twimg.com, video.twimg.com, twitter.com, …}, N is the maximum length of the sequence, $|D|$ the number of domains we consider and $P(d|S_d = S)$ the probability that the user was browsing site d given that the sequence was S.

4. Methodologies

This section describes several methodologies to solve the problem of (2) and (5), which is essentially a text classification problem. Nowadays, NLP fueled the state of the art of text processing and classification based on ANN, so this section will cover both classical techniques and modern approaches to deal with this.

4.1. Classical approach: term frequency and inverse document frequency

In this very first approach, we want to think of this as a recommendation algorithm or a search engine [41], where we aim at providing a definition for the similarity or distance between two sequences. For that purpose, it is even more useful to have a full representation in a metric space, since we can train a classifier on this space, such as k-Nearest Neighbors (k-NN) or a Multi-Layer Perceptron (MLP). We have chosen the former as an example of a simple classifier that allows us to benchmark solely the quality of the embedding, and the other as a powerful classifier able to fit further complicated patterns. Other classifiers can also be employed and performance is expected to be somewhere in the middle between these two classifiers. For instance, Support Vector Machines (SVMs) are known to have also good performance, but they also suffer from performance problems when the number of classes is very large [42]—as in this case—since they are binary classifiers that rely on either the one-vs-one strategy or the one-vs-rest strategy.

As a training set, we use, at least, a sample sequence of each domain that we intend to identify. For each document we want to classify, we assign the class or domain of the nearest neighbor (if $k = 1$, if we have more samples, we can use $k = 3$ or $k = 5$) or the one given by the MLP classifier.

As mentioned, both need a metric space, in this case, the TF–IDF [9] methodology provides one. First, we define TF as

$$\mathrm{TF}(s, S_d) = \begin{cases} 1 + \log_2 \mathrm{freq}(s, S_d) & \text{if } \mathrm{freq}(s, S_d) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

and IDF as

$$\mathrm{IDF}(s) = \log \frac{|D| + 1}{|D_s| + 0.5} \tag{7}$$

where $D_s = \{d \in D : s \in S_d\}$, i.e. the domains whose sequences contain s. Then, for each $d \in D$, we define v_d as the vector

$$v_d = [\mathrm{TF}(s, S_d) \cdot \mathrm{IDF}(s)]_{s\in\mathbb{S}} \tag{8}$$

As we see, v_d is a vector with infinite dimension. This would be problematic, but, the number of non-zero terms is finite due to the form of $\mathrm{TF}(s, S_d)$. In order to measure the similarity between two domains, we use either the Euclidean distance or the cosine distance defined as simply the cosine of the angle between the vectors v_d and $v_{d'}$,

$$\begin{aligned}\mathrm{sim}(d, d') &= \cos(v_d, v_{d'}) \\ &= \frac{v_d \cdot v_{d'}}{|v_d||v_{d'}|} \\ &= \frac{\sum_{s\in\mathbb{S}} \mathrm{TF}(s, S_d)\mathrm{TF}(s, S_{d'})\mathrm{IDF}^2(s)}{\sqrt{\sum_{s\in\mathbb{S}} \mathrm{TF}(s, S_d)^2\mathrm{IDF}^2(s)}\sqrt{\sum_{s\in\mathbb{S}} \mathrm{TF}(s, S_{d'})^2\mathrm{IDF}^2(s)}}\end{aligned} \tag{9}$$

As we see, the sum of the scalar product and the norm does only involve the non-zero terms, so, in fact, the dimension of $\mathbb{S}$ does not impact the algorithm, but the performance depends on the length of the sequence. To use the TF–IDF representation properly, a sparse matrix is usually employed where only non-zero terms are stored. These sparse vectors can be fed into other supervised methods such as an MLP classifier.

4.2. Modern approaches: neural networks

Sequence modeling, both in time series and text processing, is one of the areas where neural networks excel. In this case, we propose to follow a similar approach to neural networks that are able to classify texts or paragraphs (in our case, S_d) into categories (in our case, the domain d). Two approaches will be covered: the first one is based on the construction of an embedding based on context and the second one is a direct approach that exposes an end-to-end neural network model working.

Prior to the models, we highlight that neural networks do not work directly with strings as TF–IDF does; they rely on building first a vocabulary. This vocabulary is a mapping of each word to a number. Due to the dimensionality of the data, normally the vocabulary is limited and less frequent words are considered as OOV (Out of Vocabulary) tokens. In addition, other tokens are usually added as the start of the sequence, end of the sequence, or padding token. These last tokens solve the problem of variable length of the sequence, since a neural network only works with inputs of fixed dimension.

4.2.1. Unsupervised embeddings: Word2Vec and Doc2Vec

First, in order to understand Word2Vec, it is necessary to understand the two techniques used to perform the algorithm: Continuous Bag of Words architecture and Skip-Gram. Both architectures rely on the same concept: an artificial target variable to train the neural network.

Continuous Bag of Words (CBoW): In this first case, we build sequences where we delete an element, for instance, in sequence $S = s_1, \dots, s_N$, we call S^i to the sequence without s_i and the idea is to train a neural network so that $f_\phi(S^i) = s_i$, using some classification loss functions such as the logarithm of the cross-entropy. Once the neural network is trained, we only need to specify the embedding. To this end, we use the same approach as the AE, a hidden layer of size K. Normally, since these networks can easily have millions of parameters ($\approx |\mathbb{S}| \times$Sequence length$\times K$), it is recommended to keep the architecture as simple as possible and usually it is just a hidden layer and an output layer with a soft-max activation function. Fig. 1 shows the architecture of the CBoW.

Skip-Gram: As before, we build an artificial target to predict. In this case, the approach is completely the opposite, just with the information of one word s_i, we try to guess the context S^i, i.e. $f_\phi(s_i) = S^i$. In terms of parameters, this problem looks heavier and, in fact, it is known to be slower in terms of convergence than CBoW but it also results in better representations. As before, a single hidden layer is usually considered to avoid an excessive number of parameters. Fig. 2 shows the architecture of the Skip-Gram network.

Once the embedding is trained and ready, for each word, we have a number of $\mathbb{R}^k$. Given this, now, the problem is just a classification

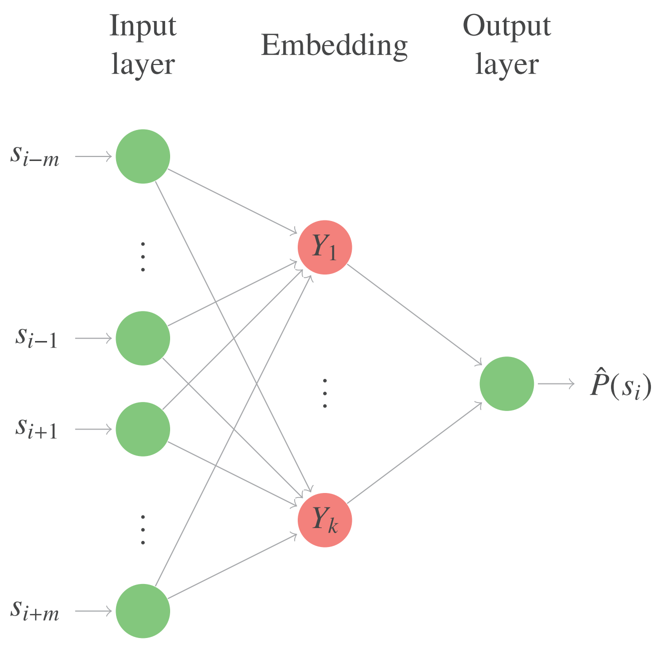


Fig. 1. Continuous bag of words architecture.

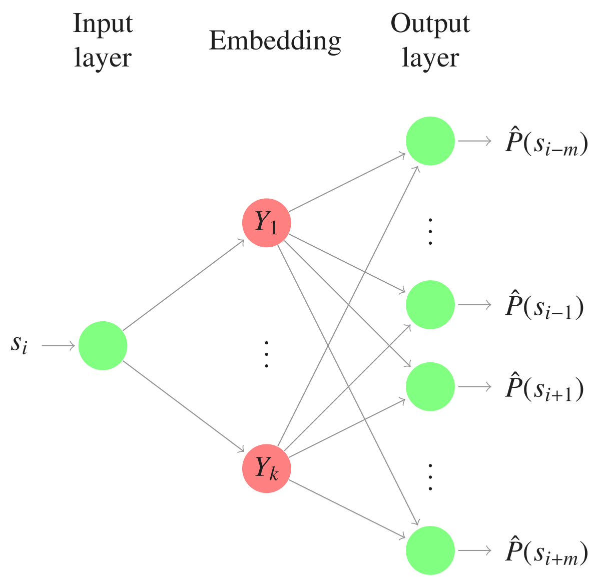


Fig. 2. Skip-Gram architecture using a context of size $2m$.

problem in $\mathbb{R}^k$ with many classes. As long as we have enough samples to train a classifier, we will be solving the problem. In order to compare the obtained result with incoming architectures, we use the same classifiers as before.

Once word2vec methods are clear, it is easier to follow the doc2vec approximation. As before, there are two possible approximations: Paragraph Vector-Distributed Memory (PV-DM) and Paragraph Vector-Distributed Bag of Words (PV-DBoW).

The first case, PV-DM, is an extension of CBoW. For each word, we compute the representation using a neural network in Fig. 3. Once all vectors are computed for every word based on some tags, the word itself, and the context, the resulting embedding is concatenated or averaged in a final vector that represents the whole paragraph. Tags act as a way of adding domain information to the network, so tags weight matrix encodes an embedding of the tags.

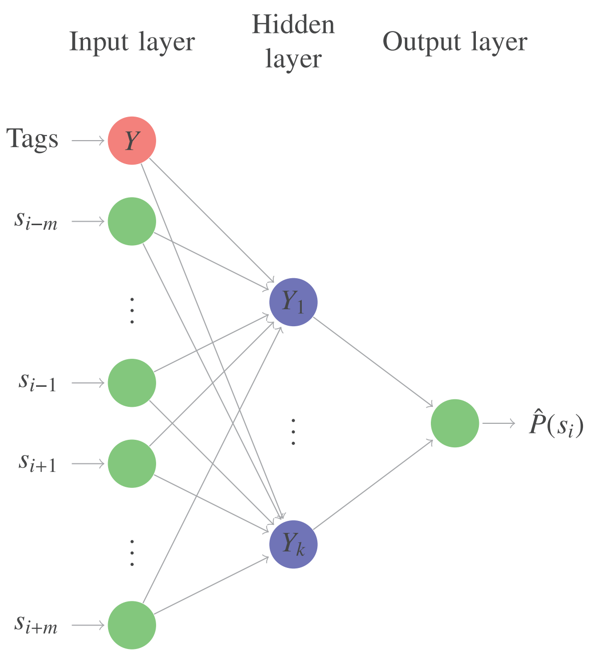


Fig. 3. Fundamental unit for the distributed memory model for paragraph vector (PV-DM).

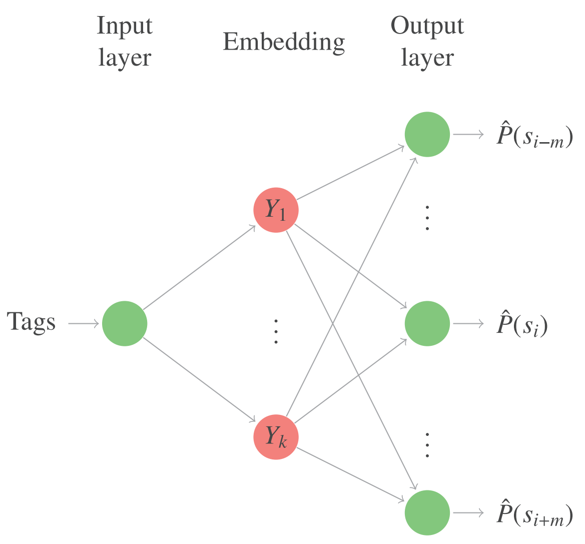


Fig. 4. Distributed bag of words model for paragraph vector (PV-DBoW).

The second case, PV-DBoW, follows an analogous procedure than skip-gram, but instead of creating the context from the center word, it is performed with the tags. Fig. 4 explains this architecture. Once the network is trained, the likelihood of the observed words can be computed to see from new samples the estimated probabilities of belonging to a class.

Implementation of all methods can be found in the Python library for topic modeling gensim [43].

4.2.2. *Custom neural networks: embedding layer and direct approach with recurrent neural networks*

Although the usual approach to text processing is unsupervised, we have categories available so we can create a classifier directly. Thus, the objective now is to create a direct model based on neural networks. The architecture is shown in Fig. 5.

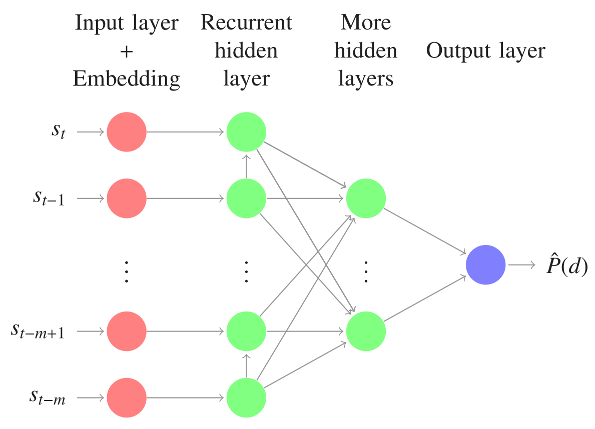


Fig. 5. Direct approach to text-classification using a RNN.

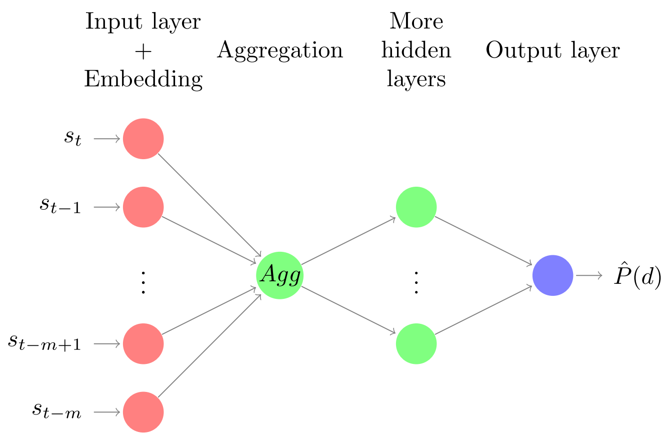


Fig. 6. Direct approach to text-classification using an ANN with custom embeddings.

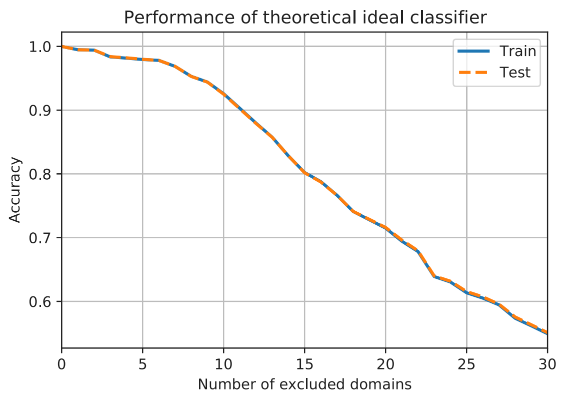
In this case, we use an embedding layer to represent the data first in a dense way, instead of a sparse representation. An embedding layer [44] of output size k is a function that converts each $s \in \mathbb{S}$ into a trainable weight, this is:

$$\begin{aligned} \mathrm{emb} : \mathbb{S} &\to \mathbb{R}^k \\ s &\to w_i \end{aligned} \tag{10}$$

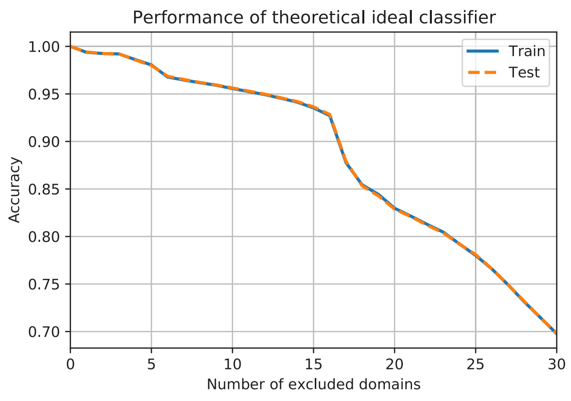
where $\{w_i\}_{i=1}^{|\mathbb{S}|}$ are trainable weights. Also, if $|\mathbb{S}|$ is too large to be practical, it is possible to use a hash function to reduce the number of parameters. This is known as the hashing trick [45]. Bear in mind that this embedding operator is exactly the same for each word, no matter the position, meaning that the number of parameters of this layer is proportional to the dimension of the input $|\mathbb{S}|$ and the desired output dimension.

Next, a recurrent layer, in particular, an LSTM layer, treats the input as a sequence. If this is not done, the input will be treated as a vector. This means that having the whole sequence of subdomains but the first one would lead to a completely different vector, whereas, with LSTM cells, neural networks can learn patterns in the sequence.

Another possibility is to eliminate the complexity of a recurrent layer and aggregate the results of each embedding. Fig. 6 presents this architecture. The aggregation can be either one that preserves the order such as concatenation or one that ignores the orders, such as the maximum or the sum. Although this model is weaker than the previous one, it is important to consider that training RNN can be extremely challenging in many environments, especially with the absence of specific hardware designed for them, such as TPUs.



(a) Top 100.



(b) Top 2500.

Fig. 7. Results for the ideal classifier that bound the best obtainable accuracy.

5. Data acquisition and preprocessing

Let us pay attention to the data to train the neural networks. For that purpose, the authors of [26] built a system that is composed of a capture engine filtering DNS and a number of web browsers that automatically access a list of objective domains simulating desktop and mobile environments (by changing, USER-AGENT field) in several operating systems.

This system, hereinafter called the "robot", is used to query the information for a list of domains, typically tops of popular websites or sets of domains of relevance for a client, e.g., stores or telecommunications companies. The robot provides a record that includes much more information than the performed DNS queries, e.g., the Time-To-Live or the server. Nevertheless, these extra attributes will not be used to feed the neural network, since we pursue to homogenize the input data for different scenarios, and not all of them can provide such extra attributes. In other words, the robot carries out measurement campaigns assuming sources of information such as sources 2 and 3, previously defined in Section 3. Also, the validity and representativeness of data have already been assessed in [26].

Once we have acquired our dataset, we need to adapt it to be fed into a neural network. The first step is to build a vocabulary. The vocabulary models $\mathbb{S}$ by adding two extra tokens or words and eliminating the least used ones. So, the vocabulary will be called $\hat{\mathbb{S}} \subset \mathbb{S} \cup \{OOV, BLK\}$, where OOV and BLK stand for Out-Of-Vocabulary token and Blank token respectively. The first one is used whenever an element in $\mathbb{S}$ is not in $\hat{\mathbb{S}}$ and the second one is used when a sequence is shorter than the maximum allowed by the neural network.

Once this is done, we map each word to a number. This can be done through a one-hot encoding (so each word is mapped to a number in $\{0,1\}^{|\hat{\mathbb{S}}|}$) or by a simple hashing (so each word is mapped to a number in $\{0,\dots,|\hat{\mathbb{S}}|\}$). Always the first approach is preferred, since distances between words in the second space are not representative whereas, in the first case, all the words are equally spaced.

Although it is clear that the first option is better, we have to consider that this means that if vocabulary size is around 100 000, we are working with an input space of dimension $100\,000m$, where m is the length of the sequence. This makes everything so expensive to compute that we have to stick to the second option in many cases.

Besides, it is possible to apply many techniques, such as splitting the domains into their subdomains or removing the repetitive parts of the domain names—e.g. `www`, and top-level domains (`com`, `net`, etc.) or country code domains (`es`, `us`, `uk`, etc.). In the latter case, these are usually called stop words in text processing. While they are sometimes removed, we decided to keep them because some regional domains, such as google.com.br or google.com.ar, are difficult to differentiate otherwise.

Then, to cope with the high dimensionality, the aforementioned embedding layer trains a linear operator and a scalar to map the input space to a fixed dimension real vector space, for instance, if we want to map the sequence of integers to a sequence of real-valued vectors. This provides a mixed solution that is usually used so that we do not exhaust the memory when creating and processing the dataset, the size of hidden layers are not excessively large and the topology of the words (that now are vectors in $\mathbb{R}^k$) is more coherent with the problem we intend to solve.

6. Results

In this section, we will evaluate the different methods explained before: TF–IDF, doc2vec, and the direct approach with neural networks. For all these methods, we will evaluate the performance with several datasets and we will model the impact of DNS caching. We recall that each time a system queries a DNS domain, the result is returned with a Time-To-Live (TTL) field. This means that as long as TTL has not expired, the device will not ask for the same domain while accessing it. As a reminder, none of the methods presented here has any information about the TTL at training, so the classifiers have no way of knowing which domains are more likely to be cached. This means that similar effects should be extendable to any caching or sampling effect, no matter whether it is related to DNS, TLS, or traffic sniffing.

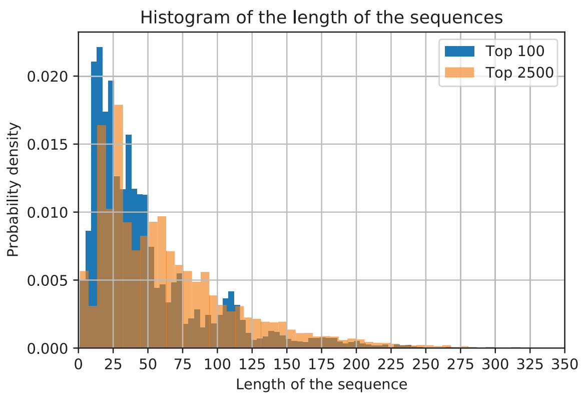
To obtain the datasets, we used the aforementioned robot to query the Top 100 and Top 2500 of worldwide most visited domains according to Alexa [46]. This is done for two web browsers: Chromium, the open-source alternative of Google Chrome, and Mozilla Firefox. In real traffic, we have observed that a set of only four domains (Google, Facebook, Apple, and Microsoft) can accumulate more than 60% of the global traffic in terms of the number of accesses. In fact, these tops follow Pareto's Law, as it happens in salary distribution or the frequency of words in English.

It is clear that the less information you have, the less precision you can achieve. Thus, firstly, we analyze the best achievable results. This way the performance of the methods is compared with respect to such best possible behavior.

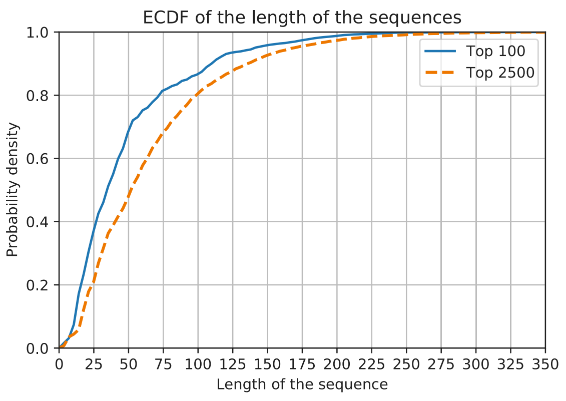
6.1. Ideal classifier

We define the ideal classifier as the one which, as long as the information is enough, always predicts the correct domain. A classifier has enough information to predict the domain d if and only if $S_d \neq S_{d'}$ for every domain $d' \in D$ different from d, i.e. as long as there is no other sequence that happens to be the same for a different domain.

When we are not considering the effect of caching, this happens with a very low probability. However, if we start missing domains



(a) Histogram.



(b) ECDF.

Fig. 8. Distribution of the length of the sequences.

in the sequence, the probability starts growing, which influences the highest possible accuracy.

Fig. 7 displays the performance as a function of the number of missing elements of the sequence for both Alexa's Top 100 and Top 2500. In the Top 100 case, we see that there is no significant loss of information after more than 10 excluded domains from the sequence. After that, it decreases linearly. With more than 25 missing elements, the performance is significantly compromised given that accuracy cannot be higher than 60%. This is because many of the sequences have less than 25 elements and become easily empty.

For the Top 2500, behavior differs significantly. An accuracy of 90% is still attainable even with more than 15 missing elements, but after that, there is a change in the slope and it ends with less than 70% for 30 elements.

These calculations show that results of accuracy should be measured in a different scale, i.e. we should modify the accuracy so that results are comparable for a different number of missing elements. Thus, we define the weighted accuracy with k missing elements as

$$\mathrm{wacc}(k) = \frac{\mathrm{acc}(k)}{\mathrm{acc}_{\mathrm{ideal}}(k)}, \tag{11}$$

where acc is the accuracy with k missing elements and $\mathrm{acc}_{\mathrm{ideal}}(k)$ is the accuracy of the ideal classifier.

Intuitively, the weighted accuracy helps us to measure the accuracy of the classifier as a percentage of the best achievable accuracy—i.e., the accuracy of the ideal classifier. However, we need extra information to completely evaluate the results. In particular, Fig. 8 displays the normalized histogram and the Empirical Cumulative Distribution Function (ECDF) of the length of the sequences for each dataset. The mean sequence length is 48.27 for the Top 100 and 65.12 for the Top 2500. In the histogram, we observe that the mode in both distributions is around 15–25. This justifies the change of behavior we have observed in Fig. 7b around this range. Furthermore, the ECDF depicts the probability of having a sequence of length higher or equal than X. This means that if you exclude 30 elements of the sequences, you would lose 40% to 45% of the dataset due to empty sequences.

6.2. Results for TF-IDF

First, we test the TF–IDF embedding. TF–IDF embedding retains a lot of information from the texts, in fact the frequencies of the words, but we expect that high-dimensional data may arise when coping with huge datasets. For that purpose, sparse matrices are used to avoid computational issues. Nevertheless, the dimensionality of the data can also affect the convergence of the algorithm, so we do not have high expectations in this method for huge datasets.

Fig. 9 shows the results in terms of the accuracy for the dataset obtained for the Top 100 of Alexa. The dataset is composed of 15000 samples of the top 100 domains in terms of visits. The training subset, in this case, is just composed of one sample per class whereas the test is the rest of the dataset. Although this split seems very aggressive, bear in mind that this representation can get highly dimensional and noise (random subdomains) can affect it. As we mentioned before, the objective of the experiments is to see the impact of the DNS caching on the results. We observed that results are not affected when 4 or 5 domains are in the cache. However, from that point on, the results are affected by an approximate ratio of 10% per 5 excluded domains.

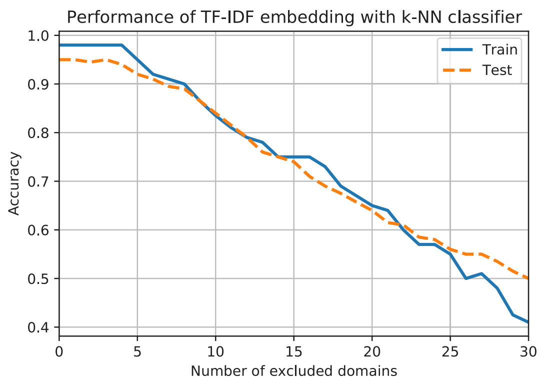
For the case of the Top 2500, we expect worse results since dimensions are much higher. In this case, the training set and test set are divided randomly with 70% of the sample for training and 30% for test. Fig. 10 displays the results for this case. In this example, we see that the behavior is almost a straight descending line. MLP classifier scores better than k-NN due to its complexity, but it cannot show results higher than 85% of accuracy for the test set. It can be observed also that in both classifiers we have overfitted the training data so the performance for the test subset is always lower.

6.3. Results for Doc2Vec

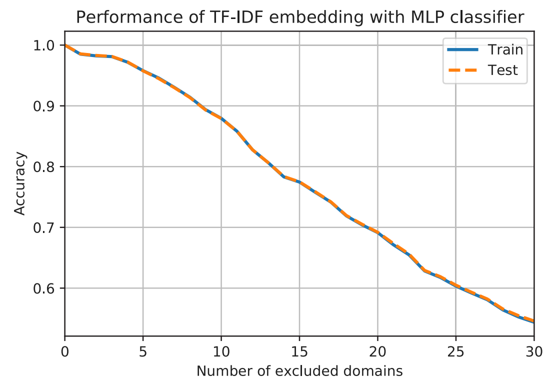
In this case, we split the dataset into a training set composed of 70% of the sample and a test set composed of 30% of the sample. Then we trained the embedding using both PV-DM and PV-DBoW algorithms with different sets of parameters. Once embeddings are trained, we use k-NN and a MLP classifier to solve the classification problem now in some real-valued space.

As before, this is done both for Top 100 and Top 2500 of Alexa. Fig. 11 shows the results for the Top 100 of Alexa. We observed that both MLP and k-NN classifiers score similarly and, again, the decay of the amount of information when eliding subdomains of the sequence is linear and more or less with a similar slope to TF–IDF. As a positive advantage, there is no overfitting in this case. About the hyperparameters, we found out that a hyperparameter was critical: whether to concatenate or to average the representations of the words. When doing concatenation, the embedding retains information of the order of the sequence. However, retaining this information causes the algorithm to overfit and not generalize really well. On the other hand, averaging provides a way of representing the words and their contexts partially ignoring the order of the sequence.

For the Top 2500, Fig. 12 represents the score for k-NN and MLP classifiers. Now, overfitting is clear and the difference between k-NN and MLP is more obvious as well, being MLP better than k-NN. It is similar also to TF–IDF with no highlighted differences.

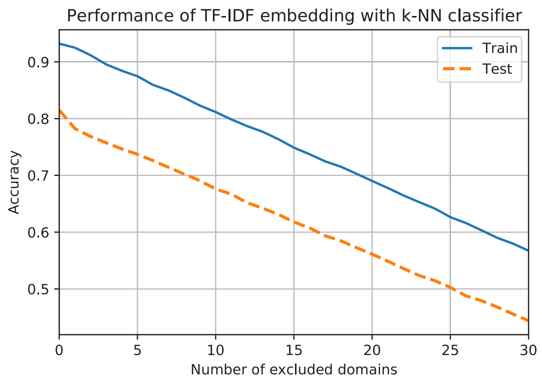


(a) k-NN classifier.

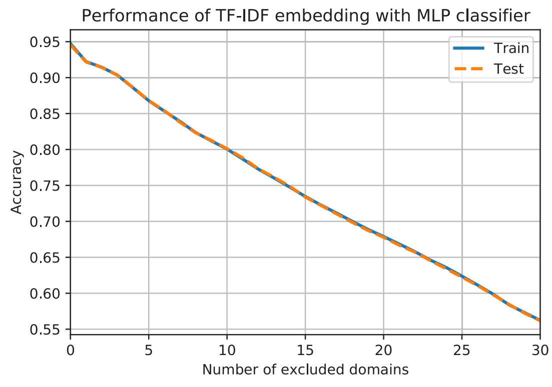


(b) MLP classifier.

Fig. 9. Results for the Top 100 of Alexa for TF–IDF embedding. The solid blue line represents the training dataset and the dashed orange line the test dataset.

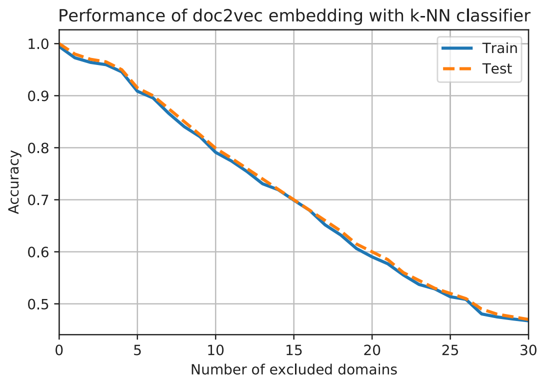


(a) k-NN classifier.

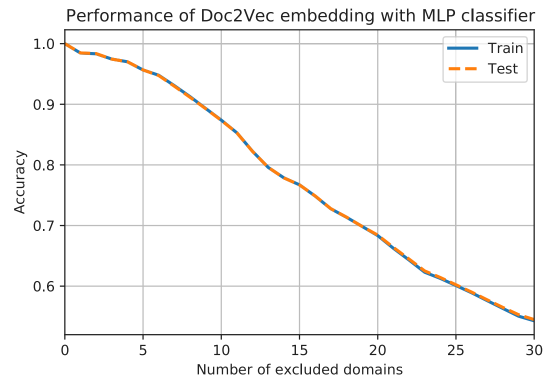


(b) MLP classifier.

Fig. 10. Results for the Top 2500 of Alexa for TF–IDF embedding. The solid blue line represents the training dataset and the dashed orange line the test dataset.



(a) k-NN classifier.



(b) MLP classifier.

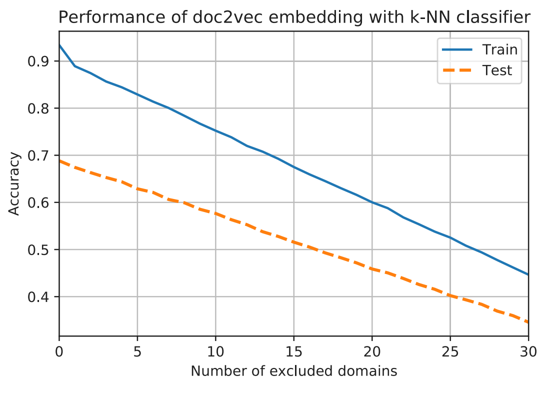
Fig. 11. Results for the Top 100 of Alexa for doc2vec embedding. The solid blue line represents the training dataset and the dashed orange line the test dataset.

6.4. *Results for RNN and ANN with custom embeddings*

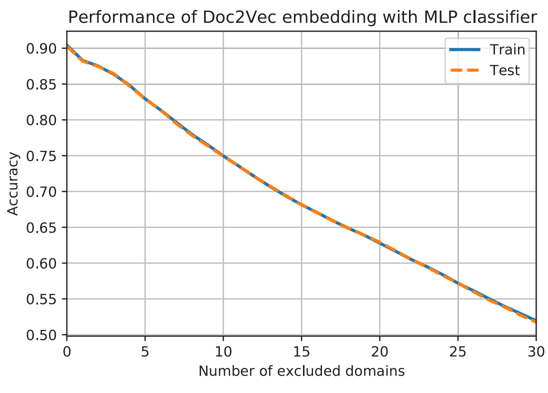
Following the previous case, for both the Top 100 and the Top 2500 datasets, we performed a train–test split with 70% of the samples for training and 30% for test. In the first case, we found out that performance is similar to other methods as we can see in Fig. 13 and the difference between training and test is negligible (which means there are no overfitting issues). However, the decay of the accuracy as a function of the number of missing subdomains is quite steep and not linear in this case. This can be due to the fact that RNN takes into account the order of the domains whereas TF–IDF and doc2vec with averaging do not, which makes the algorithm more sensible to missing elements of the sequence.

In the Top 2500 case for RNN, the achieved accuracy is much worse than TF–IDF. In this case, the decay is not so steep but, since accuracy is below 50%, it does not make sense to consider the results. This means that the data is not enough even to reliably train the RNN. As we will see next, the order plays no significant role, since ANN performance in the same conditions is significantly higher.

Since the order of the sequence is likely not important, we get rid of the recurrent layer and just aggregate all the embeddings with a sum operator. Fig. 14 shows improved results, on a par with doc2vec, both in Top 100 and in Top 2500, but with some overfitting for the Top 2500. This confirms our hypothesis that the order of the elements in the sequence is not such a relevant factor.

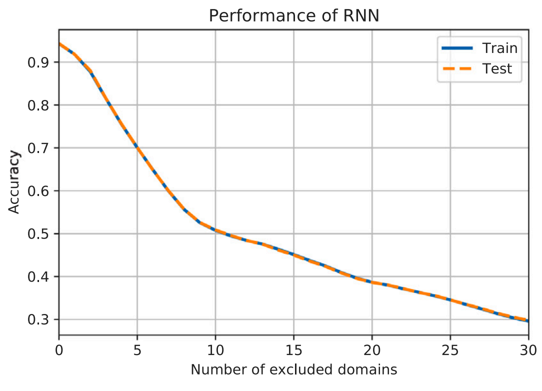


(a) k-NN classifier.

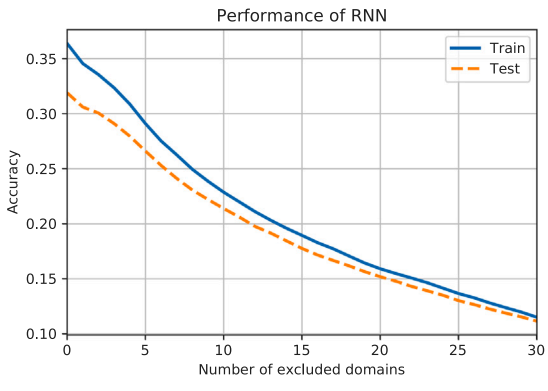


(b) MLP classifier.

Fig. 12. Results for the Top 2500 of Alexa for doc2vec embedding. The solid blue line represents the training dataset and the dashed orange line the test dataset.

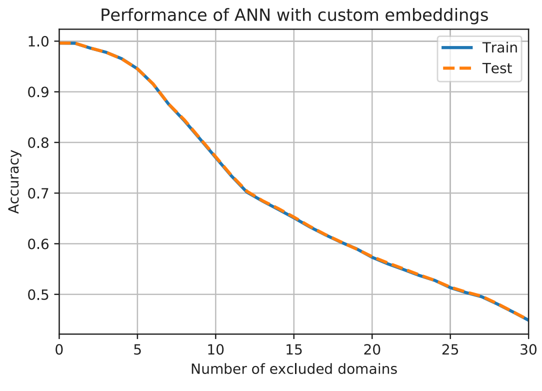


(a) Top 100.

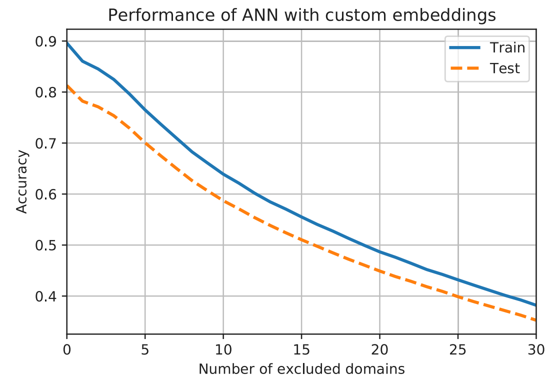


(b) Top 2500.

Fig. 13. Results for the RNN for the Top 100 and Top 2500 of Alexa. The solid blue line represents the training dataset and the dashed orange line the test dataset.



(a) Top 100.



(b) Top 2500.

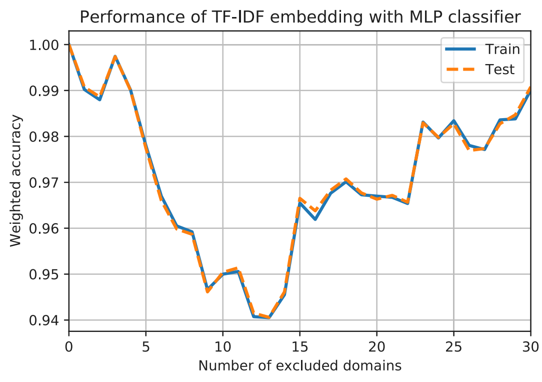
Fig. 14. Results for the ANN without recurrent layers for the Top 100 and Top 2500 of Alexa. The solid blue line represents the training dataset and the dashed orange line the test dataset.

6.5. Comparison

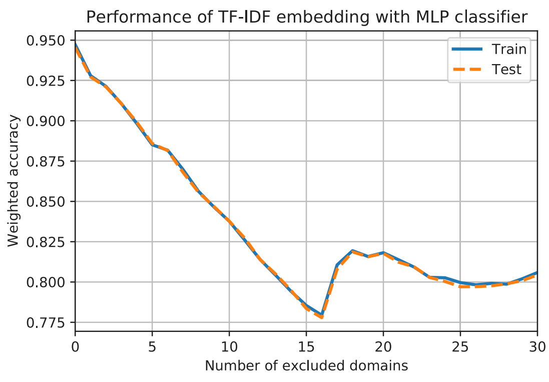
As we mentioned in the ideal classifier, accuracy is biased when the number of excluded domains grows, so in this subsection, we cover a comparative study of the performance of the different methods against the ideal classifier. For that purpose, we compare in Fig. 15 the performance in terms of the weighted accuracy. For the sake of brevity, we have chosen the best performing scenarios for each method.

All methods have similar behavior, they decrease until the number of missing elements is around 12–15 and then they either maintain the same performance or they even improve it. As we previously saw in the histogram, almost no sequences have a length less than 10 and the mode of length of the sequence is around 15, which justifies this change of behavior around 15. ANN is the only method that suffers from overfitting, given that training and test performance are not similar. TF–IDF and doc2vec perform similarly for the Top 100 domains in Alexa, but some differences arise in the Top 2500. In this case, doc2vec performs slightly better than TF–IDF, especially with more than 10 excluded domains. On the other hand, ANN follows similar behavior but with a significantly worse score.

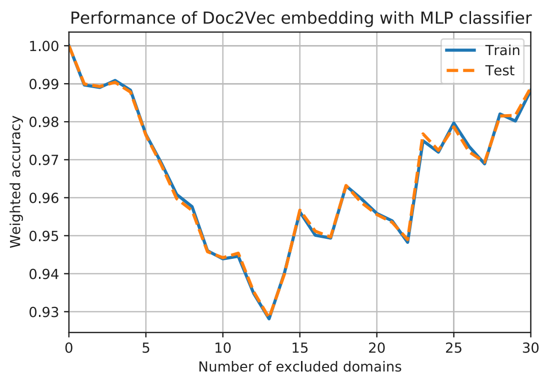
It is also important that training and prediction processes are feasible in a real environment, since models would be impractical otherwise. Table 1 shows a summary of the results along with an evaluation of the approximate memory necessity, training time, and prediction time



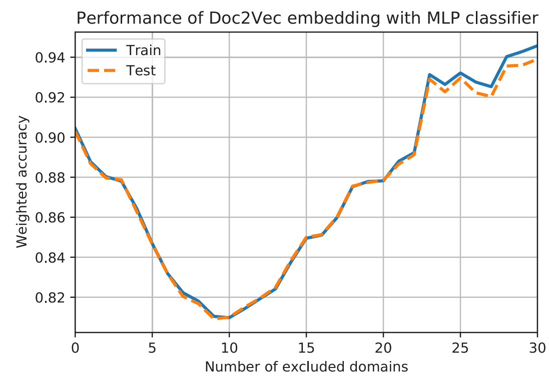
(a) Top 100 - TF-IDF.



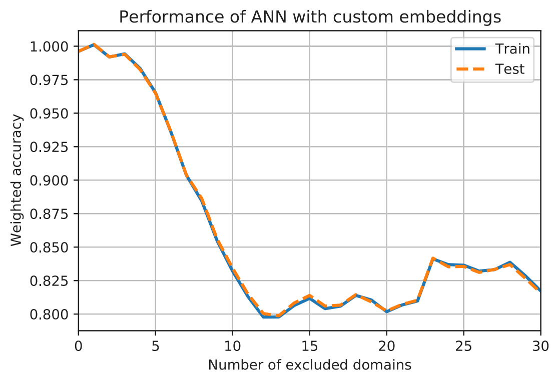
(b) Top 2500 - TF-IDF.



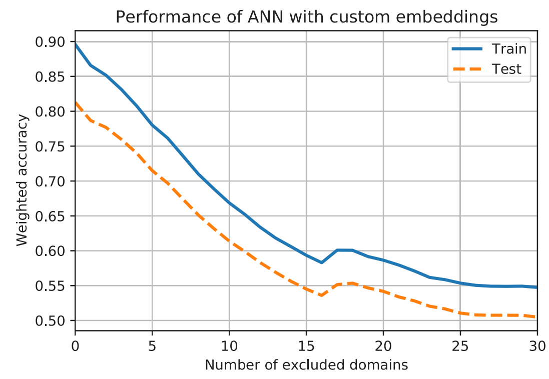
(c) Top 100 - doc2vec.



(d) Top 2500 - doc2vec.



(e) Top 100 - ANN.



(f) Top 2500 - ANN.

Fig. 15. Weighted accuracy for all the methods presented in this work.

for each algorithm. TF–IDF itself is an immediate representation of the data, but it builds extremely high dimensional data. This makes TF–IDF an appropriate option for Top 100 but completely impossible for Top 2500, where memory footprint becomes a problem. In fact, this makes training and prediction in this representation an extraordinarily expensive process that can take several hours or even days. On the other hand, ANN offers a relatively low memory consumption since most of the libraries are already prepared to handle NLP, which usually requires even larger sequences or larger vocabularies. Training takes minutes or hours, depending on the size of the network, but the prediction is always a fast process that takes only a few seconds. Doc2vec performance is somehow similar to ANN but marginally faster. To make the difference between TF–IDF and the rest more evident, these last two methods can be trained using GPUs or TPUs, reducing training time by a factor of 10 in some cases.

For the sake of reproducibility, examples of the methods and datasets have been made publicly available.[2]

7. Discussion: challenges and lessons learned

Throughout this paper, we have focused on the problem of profiling users' web browsing based on different resources such TLS records or DNS data. This objective leads us to several results and contributions about the difficulties and possibilities:

1. *Formulation of the problem in terms of NLP*: we have defined in Section 3 the problem we propose to solve. These precise and mathematical definitions allow us to foresee the difficulties that may appear as well as possible solutions based on the

[2] Available at https://github.com/hpcn-uam/nlp-web-analytics.

Table 1
Summary of results of the text classification methods for the Top 2500 domains of Alexa. Accuracy@k stands for the accuracy of the method when the number of excluded elements in each sequence is k.

Model		Memory footprint	Train time	Predict time	Accuracy	Accuracy@5	Accuracy@10
Ideal		–	–	–	>99%	>97%	>95%
TF–IDF	With k-NN	High	Minutes	Hours	>80%	>70%	>65%
	With MLP	High	Hours	Minutes	>95%	>85%	>80%
Doc2vec	With k-NN	Medium	Hours	Hours	>70%	>65%	>55%
	With MLP	Low	Hours	Minutes	>90%	>80%	>75%
ANN	With recurrence	Low	Days[a]	Minutes[a]	>35%	>25%	>20%
	Without recurrence	Low	Hours[a]	Minutes[a]	>80%	>70%	>65%

[a]Computation time using CPU. GPU/TPU might improve this result drastically.

state of the art. This means that we have translated an open problem of identifying the user's web browsing behavior from the traffic into a natural-language classification problem with a high number of classes, enabling us to employ the pre-existing state of the art of NLP to cope with this.

2. *Extensive, generic, and scalable approach with respect to alternatives*: we recognize that there are many alternatives for traffic identification and web browsing analytics extraction. In our work, we present an extensive alternative that can be used with either DNS data or with TLS/QUIC data. It could also be potentially used with flow data in combination with techniques such as [3, 18,20–25] where resolved domains are estimated through flow characteristics. In this case, the resulting performance would be the result of the combined performance of both systems. Our system is built on top of the state of the art of consolidated topics of text classification which have already been widely tested. Furthermore, it does only require a small percentage of the traffic, instead of relying on full fingerprints of a set of network packets. Finally, it can be deployed in a distributed way, scaling up the monitoring for networks of any number of users.
3. *Study of the performance with loss of information*: one of the main alternative sources as input data of our proposal is DNS data. Consequently, we highlight the performance of our methods in terms of the portion of the data that is unseen due to the effect of the local cache. This also extends beyond that and shows even a stronger result: since training data do not need any TTL or any feature indicating that a domain is more likely to be missing in the sequence, we are resistant to missing data. This can be due to cache effects, losses in capture engines, sampling techniques or any other issue that may happen in high-speed network probes.
4. *Definition of ideal classifier*: we have defined the theoretical ideal classifier that provides a bound of the best achievable performance in terms of accuracy. As we mentioned, the accuracy is biased since it is expected to decrease when the number of missing elements of the sequence rises. With this, we were able to define a brand new metric, weighted accuracy (11), and use it to make a fair comparison of the different methods in Section 6.5.
5. *There is no perfect model*: we have assessed the performance of many models with different parameters. Depending on the situation, one option outperforms the others. In this case, TF–IDF is a promising option valid for small datasets. Doc2vec is more suitable for larger datasets, whereas generic ANN with custom embeddings is also on par with doc2vec. Moreover, it is the most promising option to learn the effects of DNS caching with data augmentation.
6. *Overall accuracy*: Considering the most suitable model for every situation (availability of GPU/TPU, number of domains in the dataset, among other issues), the weighted accuracy shows figures over 90%. This means that NLP methods were able to learn the web browsing behavior successfully, proving to be good alternatives that helped to build bridges between network monitoring and NLP.

8. Conclusion

In this paper, we have approached the web browsing analytics extraction problem using NLP techniques applied over diverse traffic sources. This allows ISPs and DNS providers to exploit and monetize the data that inherently flow through their infrastructures, thus creating new business opportunities in marketing and analytics markets. Specifically, we have analyzed several text modeling techniques applied to DNS and HTTPS data or even extended flows.

As in many situations, we did not find out an ideal technique able to be used in every situation. Although TF–IDF is the most simplistic approach, it is useful even for situations where there are numerous domains and, thus, neural networks may not converge, making doc2vec and RNN worthless. However, TF–IDF has a high memory footprint and it comes with an overfitting issue that neither doc2vec nor RNN does. In general, doc2vec is better than RNN and similar to ANN. This is because the order of the sequence, in this case, provides no information, as it is shown in the performance of ANN with aggregation instead of a recurrent layer.

There are still open research lines, such as improving RNN performance through data augmentation with permutations. As another improvement, models can be trained to be resilient against several factors such as concurrent users behind a NAT router or DNS proxy or missing data due to cache effects. Although this problem cannot yet justify the search for more complex models than those we covered, attention networks [47] as well as other equivalent [48] models are emerging as promising mechanisms that will become useful in addressing the future challenges of web browsing analytics.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research has been partially funded by the Spanish State Research Agency under the project AgileMon (AEI PID2019-104451RB-C21) and by the Spanish Ministry of Science, Innovation and Universities under the program for the training of university lecturers (Grant number: FPU19/05678).

References

[1] Statista, Annual revenue of Google from 2002 to 2019, 2021, https://www.statista.com/statistics/266206/googles-annual-global-revenue.
[2] Amazon Web Services, Alexa Internet, 2021, https://www.alexa.com.
[3] G. Maciá-Fernández, Y. Wang, R.A. Rodríguez-Gómez, A. Kuzmanovic, Extracting user web browsing patterns from non-content network traces: The online advertising case study, Comput. Netw. 56 (2) (2012) 598–614.
[4] D. Sicker, P. Ohm, D. Grunwald, Legal issues surrounding monitoring during network research, in: ACM SIGCOMM Conference on Internet Measurement, 2007, pp. 141–148.

[5] K. Claffy, Ten things lawyers should know about Internet research, Cooperative Association for Internet Data Analysis, blog series, 2008.
[6] A.P. Felt, R. Barnes, A. King, C. Palmer, C. Bentzel, P. Tabriz, Measuring HTTPS adoption on the web, in: USENIX Security Symposium, 2017, pp. 1323–1338.
[7] T. Mori, T. Inoue, A. Shimoda, K. Sato, K. Ishibashi, S. Goto, SFMap: Inferring services over encrypted web flows using dynamical domain name graphs, in: Workshop on Traffic Monitoring and Analysis, 2015, pp. 126–139.
[8] T. Hernandez-Quintanilla, E. Magaña, D. Morató, M. Izal, On the reduction of authoritative DNS cache timeouts: Detection and implications for user privacy, J. Netw. Comput. Appl. 176 (2021).
[9] A. Aizawa, An information-theoretic perspective of Tf—Idf measures, Inf. Process. Manage. 39 (1) (2003) 45–65.
[10] T. Mikolov, K. Chen, G.S. Corrado, J. Dean, Efficient estimation of word representations in vector space, in: International Conference on Learning Representations, 2013.
[11] Q. Le, T. Mikolov, Distributed representations of sentences and documents, in: International Conference on Machine Learning, vol. 32, 2014, pp. 1188–1196.
[12] A. Morichetta, M. Mellia, LENTA: Longitudinal exploration for network traffic analysis from passive data, IEEE Trans. Netw. Serv. Manag. 16 (3) (2019) 814–827.
[13] D. Plonka, P. Barford, Context-aware clustering of DNS query traffic, in: ACM SIGCOMM Conference on Internet Measurement, 2008, pp. 217–230.
[14] D. Plonka, P. Barford, Flexible traffic and host profiling via DNS rendezvous, in: Workshop on Securing and Trusting Internet Names, 2011, pp. 29–36.
[15] I.N. Bermudez, M. Mellia, M.M. Munafò, R. Keralapura, A. Nucci, DNS to the rescue: Discerning content and services in a tangled web, in: ACM SIGCOMM Conference on Internet Measurement, 2012, pp. 413–426.
[16] P.E. Hoffman, P. McManus, DNS Queries over HTTPS (DoH), in: Request for Comments, no. 8484, 2018.
[17] S. Deckelmann, Firefox continues push to bring DNS over HTTPS by default for US users, 2020, https://blog.mozilla.org.
[18] M. Trevisan, F. Soro, M. Mellia, I. Drago, R. Morla, Does domain name encryption increase users' privacy? SIGCOMM Comput. Commun. Rev. 50 (3) (2020) 16–22.
[19] J. Rüth, I. Poese, C. Dietzel, O. Hohlfeld, A first look at QUIC in the wild, in: International Conference on Passive and Active Measurement, 2018, pp. 255–268.
[20] T. Wang, X. Cai, R. Nithyanand, R. Johnson, I. Goldberg, Effective attacks and provable defenses for website fingerprinting, in: USENIX Conference on Security Symposium, 2014, pp. 143–157.
[21] V. Rimmer, D. Preuveneers, M. Juarez, T. Van Goethem, W. Joosen, Automated website fingerprinting through deep learning, in: Network and Distributed System Security Symposium, 2018.
[22] P. Sirinam, M. Imani, M. Juarez, M. Wright, Deep fingerprinting: Undermining website fingerprinting defenses with deep learning, in: ACM SIGSAC Conference on Computer and Communications Security, 2018, pp. 1928–1943.
[23] S. Bhat, D. Lu, A. Kwon, S. Devadas, Var-CNN: A data-efficient website fingerprinting attack based on deep learning, in: Privacy Enhancing Technologies, no. 4, 2019, pp. 292–310.
[24] S. Siby, M. Juarez, C. Diaz, N. Vallina-Rodriguez, C. Troncoso, Encrypted DNS privacy a traffic analysis perspective, in: Network and Distributed System Security Symposium, 2021.
[25] O. Barut, M. Grohotolski, C. DiLeo, Y. Luo, P. Li, T. Zhang, Machine learning based malware detection on encrypted traffic: A comprehensive performance study, in: International Conference on Networking, Systems and Security, 2020, pp. 45–55.
[26] J.L. García-Dorado, J. Ramos, M. Rodríguez, J. Aracil, DNS weighted footprints for web browsing analytics, J. Netw. Comput. Appl. 111 (2018) 35–48.
[27] D. Kim, D. Seo, S. Cho, P. Kang, Multi-co-training for document classification using various document representations: TF–IDF, LDA, and Doc2Vec, Inform. Sci. 477 (2019) 15–29.
[28] Z.S. Harris, Distributional structure, WORD 10 (2–3) (1954) 146–162.
[29] J. Ramos, Using TF-IDF to determine word relevance in document queries, in: First Instructional Conference on Machine Learning, vol. 242, 2003, pp. 29–48.
[30] A. Grover, J. Leskovec, Node2vec: Scalable feature learning for networks, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 855—864.
[31] J. Misztal-Radecka, B. Indurkhya, A. Smywinski-Pohl, Meta-User2Vec model for addressing the user and item cold-start problem in recommender systems, User Model. User-Adapt. Interact. 31 (2021) 261–286.
[32] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[33] I. Sutskever, J. Martens, G. Hinton, Generating text with recurrent neural networks, in: International Conference on International Conference on Machine Learning, 2011, pp. 1017–1024.
[34] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder–decoder for statistical machine translation, in: Conference on Empirical Methods in Natural Language Processing, 2014, pp. 1724–1734.
[35] J. Wang, L.-C. Yu, K.R. Lai, X. Zhang, Dimensional sentiment analysis using a regional CNN-LSTM model, in: Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, 2016, pp. 225–230.
[36] J. Li, H. Zhang, Z. Wei, The weighted Word2vec paragraph vectors for anomaly detection over HTTP traffic, IEEE Access 8 (2020) 141787–141798.
[37] A. Agarwal, D. Toshniwal, SmPFT: Social media based profile fusion technique for data enrichment, Comput. Netw. 158 (2019) 123–131.
[38] W. Chang, Z. Xu, S. Zhou, W. Cao, Research on detection methods based on Doc2vec abnormal comments, Future Gener. Comput. Syst. 86 (2018) 656–662.
[39] X. Fang, X. Sun, J. Yang, X. Liu, Domain-embeddings based DGA detection with incremental training method, in: IEEE Symposium on Computers and Communications, 2020, pp. 1–6.
[40] DuckDuckGo Inc., About DuckDuckGo, 2021, https://duckduckgo.com/about.
[41] C.D. Manning, P. Raghavan, H. Schütze, Introduction to Information Retrieval, Cambridge University Press, 2008.
[42] M. Zareapoor, P. Shamsolmoali, D. Kumar Jain, H. Wang, J. Yang, Kernelized support vector machine with deep learning: An efficient approach for extreme multiclass dataset, Pattern Recognit. Lett. 115 (2018) 4–13.
[43] R. Rehurek, P. Sojka, Software framework for topic modelling with large corpora, in: Workshop on New Challenges for NLP Frameworks, 2010, pp. 45–50.
[44] F. Chollet, Keras, 2015, https://keras.io.
[45] C.B. Freksen, L. Kamma, K. Green Larsen, Fully understanding the hashing trick, in: Advances in Neural Information Processing Systems, vol. 31, 2018.
[46] Amazon Web Services, Alexa - Top sites, 2021, https://www.alexa.com/topsites.
[47] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, vol. 30, 2017.
[48] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 1, 2019, pp. 4171–4186.

Daniel Perdices is researcher and teaching assistant at Universidad Autónoma de Madrid (Spain). He holds a FPU research grant by the Spanish Ministry of Science, Innovation and Universities. Previously, he was an R&D engineer at Naudit HPCN. He received the B.Sc. (Hons) degrees in Mathematics and in Computer Science (2018), the M.Sc. in Mathematics (2019) and the M.Sc. in Information and Communications Technologies (2020) and currently is a Ph.D. student, all at Universidad Autónoma de Madrid (Spain). He researches on statistics, mathematical modeling, machine learning, network traffic analysis and SDN.

Javier Ramos is associate professor at Universidad Autónoma de Madrid (Spain). He received the M.Sc. degree in Computer Science and the Ph.D. degree in Computer Science and telecommunications from the Universidad Autónoma de Madrid, Spain, in 2008 and 2013, respectively. He was a visiting researcher at the Fraunhofer Institute for Open Communication Systems FOKUS, Germany (2012). His research interests are in the analysis of network traffic, quality of service, software defined networks and network function virtualization.

José Luis García-Dorado received his M.Sc. and Ph.D. degrees both in Computer and Telecommunications Engineering from Universidad Autónoma de Madrid (UAM) in 2006 and 2010, respectively. He became a member of the High Performance Computing and Networking research group at UAM in 2005. Thereafter, he was awarded a four-year predoctoral fellowship by the Ministry of Education of Spain (2007), and he was a visiting scholar with the Telecommunication Networks Group at Politecnico di Torino, Italy (2010), the Internet Systems Laboratory at Purdue University, USA (2013), and the Faculty of Applied Science at Universidad Técnica del Norte, Ecuador (2014 and 2015). Currently, he is an associate professor at UAM whose research interests are in the analysis of Internet traffic: its management, modeling, and evolution.

Iván González received his M.Sc. degree in Computer Engineering in 2000 and his Ph.D. in Computer Engineering in 2006, both from UAM, Spain. From October 2002 to October 2006, he was a teaching assistant at the Computer Engineering Department of UAM. From November 2006 to January 2008, he was a postdoctoral research scientist at the HighPerformance Computing Laboratory (HPCL), Electrical & Computer Engineering Department, George Washington University (Washington, DC). He was a faculty member of the NSF Center of High-Performance Reconfigurable Computing (CHREC) at George Washington University. He is currently associate professor at UAM, where he is teaching computer-architecture-related courses. He is also partner of Naudit HPCN. His main research interests are heterogeneous computing (with GPUs, FPGAs, etc.), parallel algorithms, and performance tuning. Other interests include big data, machine learning and data analytics.

Jorge E. López de Vergara is associate professor at Universidad Autónoma de Madrid (Spain) since 2007 and is a partner of Naudit HPCN, which is a spin-off company that was founded in 2009 and is devoted to high-performance traffic monitoring and analysis. He received his M.Sc. and Ph.D. degrees in Telecommunication Engineering from Universidad Politécnica de Madrid (Spain) in 1998 and 2003, respectively, where he also held an FPU-MEC research grant. During his Ph.D., he stayed for 6 months in 2000 at HP Labs in Bristol. He studies network and service management and monitoring and has coauthored more than 100 scientific papers on this topic.