# Unearthing and Exploiting Latent Semantics behind DNS Domains for Deep Network Traffic Analysis

**Franck Le, Mudhakar Srivatsa, Dinesh Verma**
IBM T.J. Watson Research Center
{fle, msrivats,dverma}@us.ibm.com.com

## Abstract

Machine learning has been applied to a broad range of network analysis tasks including device classification, device type identification, or abnormal behavior detection. However, existing solutions often require tedious and fragile manual engineered features. In addition, existing solutions may face additional challenges as the fraction of encrypted traffic is increasing. This paper proposes a novel approach that relies on the latent semantics behind the DNS names to discover endpoints' properties. First, we introduce the concept of DNS embeddings, which consist of dense representation of DNS names in the form of numeric vectors that capture the semantic relationships behind them. Second, we present a novel algorithm, dns2vec, to create DNS embeddings from DNS traffic. We evaluate it on actual network traffic, and show that dns2vec can unearth the semantics behind DNS names, e.g., revealing the close similarity between *newyorker.com* and *nytimes.com*, *sharelatex.com* and *overleaf.com*, or *sinovision.net* and *asiancc.net*. Finally, we demonstrate that these DNS embeddings can significantly improve the performance of network traffic analysis tasks. We implement a multilayer perceptron which takes as inputs DNS embeddings to identify IoT devices, and show that the error rate is reduced by one order of magnitude compared to a traditional Naive Bayes classifier.

## 1 Introduction

Through passive traffic monitoring, could network administrators detect IoT devices, recognize their device type (e.g., Withing Aura sleep sensor versus Netatmo weather station), identify infected endpoints (e.g., botnets), or detect abnormal behaviors? Researchers have attempted to address these questions, and more generally better understand properties of endpoints through network traffic analysis. However, given the difficulty of the tasks, existing solutions often require tedious and fragile manual engineered features (e.g., average packet size, mean rate, NTP interval, etc.) [1, 13, 18, 23, 26, 27]. In addition, existing solutions may face additional challenges as the fraction of encrypted traffic is increasing, and the majority of the Internet traffic is now encrypted [5].

In this paper, we propose a novel approach that relies on the semantics behind the DNS names to discover endpoints' properties. The approach does not require manual engineered features. In addition, by focusing on DNS traffic which are sent in cleartext, the analysis remains valid even when the actual connections are encrypted. More specifically, we introduce the concept of *DNS embeddings*, which consist of dense representation of DNS names in the form of numeric vectors that capture the semantic relationships behind them. Embeddings are inspired from the field of Natural Language Processing (NLP), where word embeddings [3, 12, 19, 20] have been a recent major breakthrough [28]. Word embeddings are dense numeric representations of words that capture the semantics behind the words. For example, the embeddings for the words "puppy" and "dog" are similar, reflecting the close semantic relationships between these two words. Word embeddings can capture semantic relationships between not only words, but also sentences, and documents: For example, word embeddings allow the sentences "*Obama speaks to the media in Illinois*", and "*The President greets the press in Chicago*" to be identified as similar although they do not share many words [14]. Similarly, intuitively, the semantics behind DNS could reflect the nature of the content and endpoints. For example, the embeddings for *cnn.com* and *nytimes.com* might be similar reflecting the common news related nature of these websites, and because DNS names reflect who a host communicates with, DNS embeddings could be further exploited to better understand the nature of clients issuing such requests.

However, creating DNS embeddings pose new additional challenges, and existing NLP state-of-the-art algorithms (e.g., CBOW, Skip-Gram) [19] are not adequate: First, the DNS name space can be large as DNS names can extend up to 255 characters, and comprise arbitrary strings as illustrated in the following observed DNS queries: e.g., r3—sn-p5qlsnsr.go–oglevideo.com, r1—sn-p5qlsnez.googlevideo.com, psd76–c8–982.pubnub.com, or ps5fdb–d67b.pubnub.com. Second, certain DNS names may appear rarely while state-of-the-art techniques to create word embeddings require large datasets, and frequent occurrences of a word to derive its embedding. This is because state-of-the-art techniques rely on neural networks which given the surrounding words to a target word in a sliding window, are trained to predict the target word, and vice-versa. The weight between the hidden layer and the output layer is then taken as the word vector representation of the word. Because these methods are based on neural networks, they require large corpus of documents, and frequent

occurrences of words. However, a DNS query (e.g., r3—sn-p5qlsnsr.google–video.com) may be queried and observed only once in a network packet trace. How can we therefore derive meaningful embedding for such DNS names?

To address these challenges, we introduce a novel algorithm, dns2vec, to create embeddings for DNS names. First, the algorithm does not rely on neural networks, but statistical methods to still capture the semantics of rarely occurring DNS names. Second, in contrast to state-of-the-art algorithms to derive word embedding, because dns2vec relies on statistical methoods rather neural networks, the resulting embeddings are interpretable. Third, dns2vec exploits the hierarchical DNS structure to address and limit the DNS name space. We evaluate it on the actual DNS traffic from three independent sources, and show that dns2vec can unearth the semantics behind DNS names, e.g., revealing the close similarity between *newyorker.com* and *nytimes.com*, *sharelatex.com* and *overleaf.com*, or *sinovision.net* and *asiancc.net*.

Then, we demonstrate that by capturing the semantics, DNS embeddings can significantly improve the performance of network traffic analysis tasks compared with traditional methods. We implement a multilayer perceptron which takes as inputs DNS embeddings to identify Internet of Things (IoT) devices, and compare its performance with that of a Naive Bayes classifier. The problem of detecting IoT devices has recently received significant attention [18, 26], especially given their weak security, and following a number of recent wide scale security attacks [2, 11, 22, 29]. Intuitively, domains queried by IoT devices (e.g., Amazon Echo) have different semantics (e.g., device-metrics-us.amazon.com) than those queried by users (e.g., nytimes.com). We indeed show that with DNS embeddings and a multilayer perceptron, the error rate can be reduced by one order of magnitude compared to a traditional Naive Bayes classifier. The identification of IoT devices is only one illustration of the potential applications. DNS embeddings can also help classify device types, detect abnormal behaviors, or identify infected hosts.

| Device | Date | Excerpt of Queried DNS names |
|---|---|---|
| 70:5a:0f:e4:9b:c0 (HP Printer) | 10/04/16 | 'chat.hpeprint.com', 'chat.hpeprint.com', 'chat.hpeprint.com', 'h20593.www2.hp.com', 'h20593.www2.hp.com' |
| 70:5a:0f:e4:9b:c0 (HP Printer) | 10/05/16 | 'chat.hpeprint.com', 'chat.hpeprint.com', 'h20593.www2.hp.com', 'h20593.www2.hp.com', 'ccc.hpeprint.com', ... , 'h20593.www2.hp.com', 'www.hp.com', ..., 'xmpp006.hpeprint.com', 'h10141.www1.hp.com', ... |
| 44:65:0d:56:cc:d3 (Amazon Echo) | 10/05/16 | 'www.example.com', 'www.example.com', 'www.example.net', 'www.example.net', 'www.example.org', 'www.example.org', 'pindorama.amazon.com', ..., 'device-metrics-us.amazon.com', ..., 'www.meethue.com', ... |

Figure 1: Illustration of observation instances for 24 hours time intervals

## 2 Related Work

Given the critical role of the DNS protocol, and the richness of the information carried in it, a large number of solutions rely on analysing DNS traffic to detect botnets, phishing, spam, other abnormal activities, and for classification (e.g., [1, 4, 9, 13, 15, 23, 27]). Those solutions look at different properties of the DNS names (e.g., Jaccard index of bigrams between malicious and non malicious DNS names, number of character changes needed to convert one DNS name to another). In contrast, the proposed approach captures the semantic similarities between DNS domains (e.g., *sharelatex.com*, *overleaf.com*) even when the strings exhibit little similarity between them and vice-versa.

In terms of embeddings, representation of words as numeric vectors that capture the semantic relationships behind them has a long history [12, 24]. However, it is Mikolov et al. [19] who brought word embedding to the fore through two algorithms (CBOW, and Skip-Gram) which were shown to capture multiple different degrees of similarity between words, and enabled the training and use of pre-trained embeddings. Since then, the concept of embedding has been extended and applied to other fields including paragraphs [16], and graphs [8]. Recently, Lopez *et al.* applied Skip-Gram to Internet Domain Names to create embeddings [17]. In contrast, the proposed approach creates embeddings that are interpretable, and is also applicable to DNS domains that appear rarely.

## 3 DNS Embeddings

### 3.1 Definitions, and Terminology

A DNS name is case-insensitive, and consists of multiple components – called *labels* – that are delimited by dots (e.g., device-metrics-us.amazon.com). The right-most label reflects the top-level domain, and the hierarchy of domains further descends from right to left. A DNS name may have up to 127 labels, with each one being up to 63 characters, but the complete DNS name cannot exceed 255 characters [21].

Because the DNS name space can be particularly large, we focus on a subset $V$ – called *dictionary* – of DNS names. A dictionary may consist of DNS names that have been observed at least N times, or truncated DNS names to a given domain level. Given a device, and an observation time interval (e.g., 06/12/18 00:00 to 06/12/18 23:59), we define an observation instance $\mathcal{I}$ as the sequence of DNS names in $V$ queried by that device during the observation period. We call $\mathcal{C}$ the set of observation instances, or corpus of observation instances. For example, considering the publicly available network packet trace from UNSW [26], which consists of the

**Algorithm 1** dns2vec

**INPUT:** $V$ (Vocabulary), $\mathcal{C}$ (Corpus)
**OUTPUT:** $\mathcal{D}$ (Embedding matrix)

```
for every observation instance I in C do
    for every DNS name dns_i from V in I do
        count[dns_i]++
    for every DNS name pair dns_i, dns_j from V in I do
        count[dns_i, dns_j]++
// Compute Embedding Matrix D
for i ∈ [0, |V|] do
    for j ∈ [0, |V|] do
        D[i, j] = count[dns_i, dns_j] / count[dns_i]
Return D
```

| | Dataset 1 | Dataset 2 | Dataset 3 |
|---|---|---|---|
| Time | January 2016 | September 2016 | May 2018 |
| Days | 485 | 21 | 12 |
| IoT | 100 | 22 | 0 |
| Source | Private | UNSW [26] | Private |

Figure 2: Datasets

network traffic from 30 devices, captured over 21 days starting from September 23 2016, Figure 1 illustrates excerpts of three observation instances. Finally, we define the function $e() : V \rightarrow \mathbb{R}^d$, $d \in \mathbb{N}$ to map words from the vocabulary to their embeddings, which are vectors of real numbers.

### 3.2 Dns2vec

This section presents an algorithm, dns2vec, to define and implement a version of the function $e()$. The algorithm dns2vec builds on the intuition that DNS names with similar semantics tend to be queried by a same device, close in time, and with similar contexts: For example, shortly after browsing nytimes.com, a user may go to cnn.com, to read the news.

Algorithm 1 presents the pseudo-code for dns2vec. The algorithm takes two arguments as inputs: a vocabulary $V$, and a corpus of observation instances $\mathcal{C}$. It then returns the embeddings for the DNS names in $V$ in the form of a matrix.

Dns2vec computes the conditional probabilities $P(e_j|e_i)$ with $e_k$, ($k \in [1, |V|]$), representing the event that the DNS name $dns_k$ is observed in an observation instance of $\mathcal{I}$ from $\mathcal{C}$, for every pair of DNS names $dns_j$, $dns_i$ in the input vocabulary $V$. Then, dns2vec derives the embedding matrix $\mathcal{D}$, with $\mathcal{D}[i, j] = P(e_j|e_i)$. Finally, the embedding for DNS name $dns_i$ is provided by the $i^th$ row in $\mathcal{D}$. In other words, $e()$ is defined as follows: $\forall\ dns_i \in V$ ($i \in [1, |V|]$), $e(dns_i)$ = $[P(e_j|e_i) : j \in [1, |V|]]$. The $j^th$ dimension of the vector $e(dns_i)$ is the conditional probability $P(e_j|e_i)$.

## 4 Evaluation

This Section evaluates dns2vec and the quality of the derived DNS name vectors.

### 4.1 Datasets

We analyze the DNS traffic from three independent sources (Figure 2). The first source is a private US lab where commercial IoT devices are continuously added, and removed, since 2015. We focus on the traffic captured between January 1 2016, and April 30 2017. During that time period, we observe 100 IoT devices. The second source is the publicly available dataset from UNSW [26]: We focus on the DNS network traffic from the 22 IoT devices, captured over 21 days starting from September 23 2016. Finally, the third source is the DNS traffic from 1,576 non-IoT devices (e.g., smartphones, desktops, laptops) captured at a large enterprise office branch, between May 12 2018 and May 25 2018.

### 4.2 Methodology

First, we set the observation time interval to 24 hours.

Second, we observe that the three datasets comprise devices of different nature: Datasets 1, and 2, consist of IoT devices (e.g., smart speakers, sleep sensors, smart bulbs, cameras) whereas dataset 3 consists of non-IoT devices (e.g., smartphones, laptops, desktops). As such, we merge datasets 1 and 2 into a single corpus $\mathcal{C}_{IoT}$, and we treat dataset 3 as a seperate corpus $\mathcal{C}_{NonIoT}$. We apply dns2vec on both $\mathcal{C}_{IoT}$, and $\mathcal{C}_{NonIoT}$. The corpus $\mathcal{C}_{IoT}$ includes 12,727 observation instances, and 2,309 distinct DNS names. We therefore set the vocabulary to be the set of all 2,309 observed DNS names. In contrast, $\mathcal{C}_{NonIoT}$ includes 10,547 observation instances, and 265,670 distinct DNS names. Given the large size of observed DNS names, we set the vocabulary to the set of DNS names truncated to level 2, and that have been queried at least twice. We exploit the DNS hierarchical structure, and reduce the vocabulary size to 28,227 DNS names.

Third, the algorithm dns2vec maps DNS names to numerical vectors. To evaluate the quality of the derived embeddings, for each DNS name, we report the three most similar DNS names in the corpus using the cosine similarity. For two vectors $U$, $V$, the cosine similarity is defined as (with $U_i$ and $V_i$ being the $i^{th}$ dimension of vectors $U$, $V$):

$$\frac{\sum_{i=1}^{n} U_i V_i}{\sqrt{\sum_{i=1}^{n} U_i^2}\sqrt{\sum_{i=1}^{n} V_i^2}}$$

The cosine similarity ranges from 0 to 1, with 0 meaning no similarity and 1 indicating strong similarity between the vectors.

### 4.3 Results

Figure 3 illustrates results of DNS names in $\mathcal{C}_{IoT}$. For a number of DNS names, the three most similar DNS names and their cosine similarity scores are reported. First, for chat.hpeprint.com which is issued by the HP Printer (Figure 1), we observe that the most similar DNS names also belong to the same organization (HP Inc.) Second, for 0.invoxia.pool.ntp.org, the domain ntp.org is owned by the Network Time Organization which provides time servers. The Network Time Organization actually also offers vendors the option to get their own zone, with the hostnames having the form of 0.vendor.pool.ntp.org. The DNS name 0.invoxia.–pool.ntp.org suggests that the server therefore corresponds to a time server run by the Network Time Organization for Invoxia. Interestingly, the most similar results are owned in this

| DNS name | Three Most Similar DNS names (DNS name, cosine similarity) | | |
|---|---|---|---|
| chat.hpeprint.com | h20593.www2.hp.com, 0.75 | xmpp006.hpeprint.com, 0.72 | h10141.www1.hp.com, 0.68 |
| 0.invoxia.pool.ntp.org | sip.invoxia.com, 0.97 | icecast.icecast.sbs.com.au, 0.93 | ws.invoxia.io, 0.93 |
| r1—sn-p5qlsnez.googlevideo.com | r9—sn-p5qlsney.googlevideo.com, 1.0 | vassg142.ocsp.omniroot.com, 1.0 | gv.symcd.com, 0.96 |
| pscfcb6ec6.pubnub.com | pscab6d5d1.pubnub.com, 1.0 | psc3f5c69e.pubnub.com, 1.0 | psc8a67f35.pubnub.com, 1.0 |
| v4.netatmo.net | _vpn._udp.netatmo.net, 0.98 | v3.netatmo.net, 0.97 | v5.netatmo.net 0.97 |
| ntp1.glb.nist.gov | time.nist.gov.lan, 0.70 | time.nist.gov, 0.46 | time1.google.com, 0.33 |

Figure 3: Examples of semantic relationships for DNS names in $\mathcal{C}_{IoT}$

| DNS name | Three Most Similar DNS names (DNS name, cosine similarity) | | |
|---|---|---|---|
| newyorker.com | buzzfeed.com, 0.96 | nytimes.com, 0.95 | nymag.com, 0.95 |
| nba.com | vividseats.com, 0.98 | theundefeated.com, 0.98 | espnfc.us 0.98 |
| sharelatex.com | overleaf.com, 0.96 | slack-imgs.com 0.96 | slack-edge.com, 0.96 |
| sinovision.net | asiancc.net, 0.96 | hking.hk, 0.96 | uschinapress.com, 0.96 |
| 247checkers.com | cardgamesolitaire.com, 0.99 | 123freecell.com, 0.99 | solitairetime.com, 0.99 |
| akamaiedge.net | akadns.net, 0.99 | collabserv.com, 0.99 | akamai.net, 0.99 |

Figure 4: Examples of semantic relationships for DNS names in $\mathcal{C}_{NonIoT}$

case by a different organization (than ntp.org) but are still related to Invoxia which is the company designing and manufacturing the Triby Smart Speaker. The third DNS name r1—sn-p5qlsnez.googlevideo.com is actually appearing only in two instances, yet dns2vec is able to identify that this DNS name is very similar to r9—sn-p5qlsney.google–video.com. Similarly, the DNS name pscfcb6ec6.pubnub.com is appearing only once, and dns2vec points its close similarity to DNS names owned by the same organization and with very similar structure. Finally, based on the words present in the DNS names (e.g., time, ntp), the last entry in Figure 3 seems to point to time servers from different organizations (e.g., ntp.org, google.com).

Figure 4 illustrates results of DNS names in $\mathcal{C}_{NonIoT}$. We observe that newyorker.com, buzzfeed.com, nytimes.com, and nymag.com are all magazines and newspapers of reportage with an emphasis on New York City's political and cultural life. nba.com, vividseats.com, theundefeated.com, and espnfc.us are all websites related to sports. sharelatex.com and overleaf.com are both online latex editing tools. sinovision.net, asiancc.net, and uschinapress.com are related to Chinese news with a focus towards an audience in the US. 247checkers.com, cardgamesolitaire.com, 123freecell.com, and solitairetime.com are online game websites. Finally, akamaiedge.net, akadns.net, and akamai.net are all domains from Akamai while collabserv.com is a cloud based messaging and collaboration suite of services that relies on Akamai infrastructure.

# 5 Use Case Applications

Previous sections show that dns2vec can map DNS names to numerical vectors that capture the semantic relationships behind the DNS names. Such embeddings can then be used for a wide range of network traffic analysis tasks (e.g., device type classification, abnormal behavior identification, infected host detection, etc.)

Section 5.1 illustrates one example: As the problem of detecting IoT devices has recently received significant attention [18, 26, 29], we show that DNS embeddings can help to detect IoT devices with a much higher accuracy than traditional solutions. Then, Section 5.2 further discusses how DNS embeddings can also help other network analysis tasks such as IoT device type identification, and abnormal behavior detection.

## 5.1 IoT Device Detection

### Methodology

**Datasets, Vocabulary, and Embeddings:** We consider the datasets presented in Section 4.1. Similarly to Section 4.2, we set the observation time period to 24 hours. Each observation instance originated from an IoT device is labelled as 1, whereas observation instances originated by non IoT devices are labelled as 0. To reduce the vocabulary size, we truncate DNS names to domains up to level-4, and we apply a stratified sampling to the non IoT observation instances: For each non IoT observation instance, we randomly select 20 DNS names. As a result, the vocabulary is reduced to 33,226 DNS names. We then apply dns2vec to derive the embeddings for each DNS name in the vocabulary. Each embedding has a dimension of $|V|$ = 33,226. To reduce the subsequent amount of computation, we further reduce the dimensionality of the embeddings through Singular-Value Decomposition [7] to 512.

**Classifiers:** We consider and compare two approaches to identify IoT devices from their DNS traffic: A multilayer perceptron that relies on DNS embeddings, and a Naive Bayes classifier that does not use DNS embeddings.

• *Multilayer perceptron*: A multilayer perceptron is a class of feedforward artificial neural network. For simplicity, we assume a multilayer percepton with a single hidden layer. As depicted in Figure 5, the input layer takes the 20 first DNS names from each observation instance. If an instance $\mathcal{I}$ has less than 20 DNS names, we padd the observation instance by generating DNS names following the observed distribution: for example, if $\mathcal{I}$ has four DNS names $\{dns_1, dns_1, dns_2, dns_3\}$, we generate 16 additional DNS names with the following probabilities $\{dns_1$:1/2, $dns_2$:1/4, $dns_3$:1/4$\}$. Each DNS name is then replaced with its embedding. As such, the input layers consists of 20 DNS names $\times$ 512 dimensions per embedding = 10,240 inputs. We consider a hidden layer

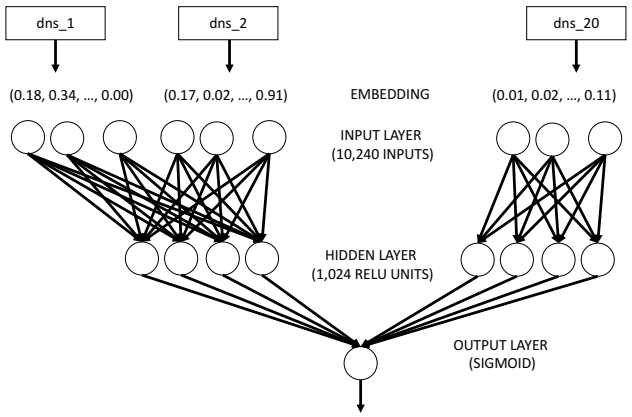


Figure 5: Illustration of multilayer perceptron architecture

| N = 4,654 | Actual Non IoT | Actual IoT | |
|---|---|---|---|
| Predicted Non IoT | TN = 2,094 | FN = 13 | 2,107 |
| Predicted IoT | FP = 6 | TP = 2,541 | 2,547 |
| | 2,100 | 2,554 | |

Figure 6: Confusion matrix of multilayer perceptron

with 1,024 rectified linear units [6]. Finally, the output layer consists of a single node implementing the sigmoid activation function [10] whose output ranges from 0 to 1, and is commonly used for binary classification. We split the datasets according to a 80:20 ratio into a training set, and test set. The training set is used to train the multilayer perceptron, while the test set is used to evaluate its performance. For the training, we set the batch size to 128, and the number of epochs to 2. For the testing, each observation instance $\mathcal{I}$ from the test set is fed into the classifier. If the output is greater than 0.5, the instance is classified as "IoT" and vice versa. We then compare the predictions with the actual labels.

• *Naive Bayes classifier*: A naive Bayes classifier is a simple probabilistic classifier based on Bayes' theorem with strong independence assumptions [25]. Similar as above, we first split the datasets according to a 80:20 ratio into a training set, and test set. Second, we use the training set to compute the probabilities $P(IoT)$, $P(\overline{IoT})$, $P(dns_k|IoT)$, $P(dns_k|\ \overline{IoT})$ for every observed DNS name $dns_k$. Then, for every observation instance $\mathcal{I}$ in the test set, we compute $P(IoT|\ dns(\mathcal{I}))$, and $P(\overline{IoT}|dns(\mathcal{I}))$, with $dns(\mathcal{I})$ representing the list of DNS names in $\mathcal{I}$. If $P(IoT|dns(\mathcal{I})) > P(\overline{IoT}|dns(\mathcal{I}))$, we classify $\mathcal{I}$ as "IoT" and vice-versa. We then compare the result with the actual labels.

To compare the two classifiers, we report the number of instances that are correctly classified, the number of false positives, and the number of false negatives.

**Results**

The multilayer perceptron incorrectly classified 19 instances (13 False Negative, 6 False Positive) (Figure 6), whereas the Naive Bayes classifier cannot classify 376 instances (e.g., $P(IoT|dns(\mathcal{I}))$ = $P(\overline{IoT}|dns(\mathcal{I}))$ = 0), and incorrectly classifies 4 instances. As such, the multilayer perceptron reduces the number of errors from 380 to 19, i.e., by a factor of 20 times.

Intuitively, to illustrate the reason for the better performance, we consider an observation instance $\mathcal{I}$ with a list of $n$ DNS names [$dns_1$, $dns_2$, ..., $dns_n$]. If any of the DNS names has not been seen in the training, the Naive Bayes classifier results in a probability of 0.00, and cannot classify the instance. In contrast, with the multilayer perceptron, even if the majority of DNS names has not been seen, and the remaining DNS names occurred very rarely in the training set, the semantics of those DNS names can still be extracted, and the instance can then be correctly classified.

## 5.2 Other applications

**IoT Device Type Classification**

Network administrators want to identify the types of IoT devices present in their network, for example, so that vulnerable IoT devices can be quarantined [18].

DNS embeddings can improve the identification of the types of IoT devices present in a network. For example, given an observation instance, one could convert the DNS names into their embeddings, apply different classification or clustering techniques, and return the device type of the most similar instances from a reference corpus.

We implemented such approach using the Word Mover's Distance [14], and showed that solutions based on embeddings outperform traditional solutions[1]. The reason is that while two IoT devices of the same type (e.g., Samsung refrigerators) may query distinct DNS names (e.g., psd76–c8982.pubnub.com, and ps5fdb–d67b.pubnub.com), their embeddings capture the fact that these two names have very similar semantics, and therefore the DNS access patterns can be identified as similar.

**Abnormal Behavior Detection**

Previous studies have pointed out that endpoints infected by botnets need to connect to their master, and may generate considerably different DNS access patterns [1, 13, 23, 27].

As such, abnormal behaviors may reveal that an endpoint got compromised. Abnormal behaviors can be detected by converting DNS names in observation instances to their embeddings, computing the cosine similarity between an observation instance and all the observation instances from the same device but corresponding to the previous days, and reporting instances where the minimum of the cosine similarities exceed a configurable threshold. This will indicate that the device is behaving differently from the previous days.

We implemented it and detected various changes in behaviors in IoT devices, including the installation of new applications on the devices, and changes in the servers where the IoT devices report their metrics or check for software updates[2]. More importantly, the results show that by taking into account

[1]Details of the experiments are omitted due to space constraints.

[2]Details of the experiments are omitted due to space constraints.

the latent semantics behind the DNS names, solutions based on DNS embeddings achieve higher accuracy than traditional approaches.

## 6 Conclusion

We introduce the concept of embeddings for network traffic, and propose dns2vec, an algorithm to train and use pre-trained DNS embeddings. We show that DNS embeddings capture different forms of semantic relationships between DNS names, and could be used for a wide range of network traffic analysis tasks. We demonstrate that embeddings reduce the error rate in IoT device classification by 20 times compared to traditional approaches (e.g., Naive Bayes classifier), and discuss how other network traffic analysis tasks could also benefit from them.

## Acknowledgments

We thank the reviewers for their useful comments and feedback. This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

## References

[1] M. Antonakakis, R. Perdisci, W. Lee, N. Vasiloglou, II, and D. Dagon. Detecting Malware Domains at the Upper DNS hierarchy. In *USENIX Conference on Security*, 2011.

[2] N. Apthorpe, D. Reissman, and N. Feamster. A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic. In *Workshop on Data and Algorithmic Transparency (DAT)*, 2016.

[3] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 2003.

[4] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi. EXPOSURE: Finding malicious domains using passive DNS analysis. In *NDSS*, 2011.

[5] K. Finley. Half the web is now encrypted. that makes everyone safer. `https://www.wired.com/2017/01/half-web-now-encrypted-makes-everyone-safer/`. [Online; accessed 01-January-2017].

[6] X. Glorot, A. Bordes, and Y. Bengio. Deep Sparse Rectifier Neural Networks. In *International Conference on Artificial Intelligence and Statistics*, 2011.

[7] G. H. Golub and C. Reinsch. Singular Value Decomposition and Least Squares Solutions. *Numer. Math.*, 1970.

[8] A. Grover and J. Leskovec. Node2Vec: Scalable Feature Learning for Networks. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[9] H. Guo and J. Heidemann. IP-Based IoT Device Detection. In *Workshop on IoT Security and Privacy*. ACM, 2018.

[10] J. Han and C. Moraga. The Influence of the Sigmoid Function Parameters on the Speed of Backpropagation Learning. In *International Workshop on Artificial Neural Networks*, 1995.

[11] L. Hautala. Why it was so easy to hack the cameras that took down the web. In *CNET Security*, October 2016.

[12] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. MIT Press, 1986.

[13] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling. Measuring and Detecting Fast-Flux Service Networks. In *Symp NDSS*, 2008.

[14] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger. From word embeddings to document distances. In *ICML*, 2015.

[15] F. Le, J. Ortiz, D. Verma, and D. Kandlur. Policy-Based Identification of IoT Devices Vendor and Type by DNS Traffic Analysis. In *Lecture Notes in Computer Science: Policy-Based Autonomic Data Governance*, 2019.

[16] Q. Le and T. Mikolov. Distributed Representations of Sentences and Documents. In *ICML*, 2014.

[17] W. Lopez, J. Merlino, and P. Rodriguez-Bocca. Vector representation of Internet Domain Names using a Word Embedding technique. In *XLIII Latin American Computer Conference (CLEI)*, 2017.

[18] M. Miettinen, S. Marchal, I. Hafeez, T. Frassetto, N. Asokan, A.-R. Sadeghi, and S. Tarkoma. Iot sentinel demo: Automated device-type identification for security enforcement in iot. In *IEEE ICDCS*, 2017.

[19] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, 2013.

[20] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[21] P. Mockapetris. *RFC 1035 Domain Names - Implementation and Specification*. Internet Engineering Task Force, 1987.

[22] D. Palmer. 175,000 IoT cameras can be remotely hacked thanks to flaw, says security researcher. In *ZDNet*, July 2017.

[23] R. Perdisci, I. Corona, D. Dagon, and W. Lee. Detecting Malicious Flux Service Networks Through Passive Analysis of Recursive DNS Traces. In *Annual Computer Security Applications Conference*, 2009.

[24] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Neurocomputing: Foundations of Research. chapter Learning Representations by Back-propagating Errors. 1988.

[25] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009.

[26] A. Sivanathan, D. Sherratt, H. H. Gharakheili, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman. Characterizing and Classifying IoT Traffic in Smart Cities and Campuses. In *IEEE Infocom Workshop Smart Cities and Urban Computing*, 2017.

[27] S. Yadav, A. K. K. Reddy, A. N. Reddy, and S. Ranjan. Detecting Algorithmically Generated Malicious Domain Names. In *IMC*, 2010.

[28] T. Young, D. Hazarika, S. Poria, and E. Cambria. Recent trends in deep learning based natural language processing. *CoRR*, 2017.

[29] T. Yu, V. Sekar, S. Seshan, Y. Agarwal, and C. Xu. Handling a Trillion (Unfixable) Flaws on a Billion Devices: Rethinking Network Security for the Internet-of-Things. In *HotNets*, 2015.