DDoS2Vec: Flow-Level Characterisation of Volumetric DDoS Attacks at Scale

RAVJOT SINGH SAMRA, University of Waikato, New Zealand
MARINHO BARCELLOS, University of Waikato, New Zealand

Volumetric Distributed Denial of Service (DDoS) attacks have been a severe threat to the Internet for more than two decades. Some success in mitigation has been achieved based on numerous defensive techniques created by the research community, implemented by the industry, and deployed by network operators. However, evolution is not a privilege of mitigations, and DDoS attackers have found better strategies and continue to cause harm. A key challenge in winning this race is understanding the various characteristics of DDoS attacks in network traffic at scale and in a realistic manner.

In this paper, we propose DDoS2Vec, a novel approach to characterise DDoS attacks in real-world Internet traffic using Natural Language Processing (NLP) techniques. DDoS2Vec is a domain-specific application of Latent Semantic Analysis that learns vector representations of potential DDoS attacks. We look into the link between natural language and computer network communication in a way that has not been previously studied. Our approach is evaluated on a large-scale dataset of flow samples collected from an Internet eXchange Point (IXP) in one year. We evaluate the performance of DDoS2Vec via multi-label classification in a Machine Learning (ML) scenario. DDoS2Vec characterises DDoS attacks more clearly than other baselines — including NLP-based approaches inspired by recent networks research and a basic non-NLP solution.

CCS Concepts: • **Security and privacy → Denial-of-service attacks**; • **Networks → Denial-of-service attacks**; **Network monitoring**.

Additional Key Words and Phrases: Distributed Denial of Service; Machine Learning; Natural Language Processing

ACM Reference Format:
Ravjot Singh Samra and Marinho Barcellos. 2023. DDoS2Vec: Flow-Level Characterisation of Volumetric DDoS Attacks at Scale. *Proc. ACM Netw.* 1, CoNEXT3, Article 13 (December 2023), 25 pages. https://doi.org/10.1145/3629135

1 INTRODUCTION

DDoS is a class of attack that aims to cause disruption [32]. Volumetric DDoS attacks [44, 15, 20, 21, 35] do severe damage every year; companies are obliged to use substantial capital to pay for defensive systems [59, 60]. Detecting and mitigating the effects of DDoS attacks has been a continuous battle since the early days of the Internet [36, 26, 22]. Despite all the efforts made by research, industry, and legal authorities, it would be naïve to hope to eradicate these attacks for good or create a perfect protection system against them. Attacks have been evolving regarding techniques, exploitation vectors, actors, etc. Instead, we must continuously advance our understanding of attacks and improve defences.

Authors' addresses: Ravjot Singh Samra, University of Waikato, Gate 1, Knighton Road, Hamilton, Waikato, New Zealand, 3240; Marinho Barcellos, marinho.barcellos@waikato.ac.nz, University of Waikato, Gate 1, Knighton Road, Hamilton, Waikato, New Zealand, 3240.

2834-5509/2023/12-ART13 $15.00
https://doi.org/10.1145/3629135

Unlike prior work, we focus on *characterising* DDoS attacks, i.e. establishing the features that distinguish one attack from another attack. Such features include the amplification attack protocols, prominent packet sizes, variance in an attack over time, and many other nuances in real-world DDoS attacks. DDoS attack characterisation opens the path for attack similarity analysis, the discovery of new attacks, attack history investigations, recent attack trends, etc., which can be beneficial besides immediate attack mitigation.

However, characterisation is challenging as DDoS attacks can be diverse [15], and ground truth is key for evaluation [56]. Prior proposals to analysing traffic in general for security purposes have involved packet payload content (complete packet captures) [17, 2]. Most ignore real-world data constraints, such as resource limitations, i.e. computational and storage costs [4], and encrypted payloads [49]. In contrast, we follow [56, 54, 47, 35, 20, 10] and rely on *sampled flow records* captured at line rates either at the core of the Internet (e.g. IXPs) or at the edges (ingress points of potential victims). The flow records are collected, sampled, and stored using standard methods, such as sFlow [37] and NetFlow [6], which network operators widely employ [16]. Unfortunately, having sampled flow records as input significantly reduces the information available for analysis, which makes DDoS characterisation even more challenging.

In this paper, we propose *DDoS2Vec*, a novel yet pragmatic approach towards flow-level UDP-based DDoS attack characterisation at scale with a main focus on amplification attacks. DDoS2Vec uses NLP techniques to extract potential attack characteristics based on traffic patterns and sampled flow record proximities over long periods. The intuition of exploring NLP for this problem is based on recent networks research [12, 7, 41]. DDoS2Vec learns a vector representation of a potential attack toward an IP address in a given timespan, which can then be contrasted with other vectors to identify similarities and differences. Our results show that DDoS2Vec can characterise DDoS attacks more clearly than our baseline approaches in challenging circumstances, where all common or rare characteristics of potential attacks must be identified. The contributions of this paper can be summarised as follows:

- We examine the state of publicly available DDoS datasets regarding DDoS research (§3).
- We provide a fresh perspective on DDoS attack characterisation by applying NLP concepts to network flow records. To do that, we consider the similarities between flow records and text sentences/documents (§4), and do so at scale.
- We investigate multiple NLP techniques (along with a non-NLP baseline) for DDoS attack characterisation and evaluate using real-world IXP traffic combined with suitable ground truth (§5). We assess the trade-offs between Word2Vec [31], Doc2Vec [24], and Latent Semantic Analysis [11], the last of which is used in DDoS2Vec.
- We evaluate to which extent characteristics learnt by a DDoS2Vec model are applicable ("transfer") to other periods (§6). We cover a whole year's worth of IXP flow samples for the longitudinal analysis.

2 RELATED WORK

A large body of literature exists on traffic analysis for network security purposes, including surveys (e.g. Aljuhani [1]) and prior proposals for DDoS attack detection and mitigation. Much less research has been done on characterising DDoS attacks at scale. We now review research inspiring our approach and other closely related work.

Prior work has shown that benign and malicious network traffic typically have different statistical distributions [36, 22, 26]. While these approaches can detect the *presence* of DDoS attack traffic by comparing it to a normal traffic profile, they do not provide a more significant analysis of attack characteristics beyond mere abnormality.

Approaches based on *distance metrics and clustering* have also been explored for network hosts and flows themselves [30]. Xu et al. [58] showed that it is possible to learn general characteristics from Internet infrastructure traffic. That work was used by Coull et al. [8] to show the importance of capturing two relevant properties in one unified metric space: (i) *spatial* properties, as in capturing the semantic relationship (if any) between, e.g. two different port numbers; and (ii) *temporal* properties, as in capturing similarities beyond small windows of time.

Ring et al. [41] introduced IP2Vec, a method for learning vector representations of IP addresses, ports, and protocols — all effectively represented as words. They showed that an approach inspired by Word2Vec [31] can be used to learn the contextual appearances of those flow record fields and distributed into a single vector space based on IP2Vec custom context. IP2Vec was used to cluster network hosts, and Ring et al. [41] found it to outperform the graph-based metric approach in [8]. That was an initial sign that techniques closely related to NLP could be used to learn interesting behaviours.

Using Word2Vec directly, Cohen et al. [7] introduced DANTE, a specialised network security analysis framework that used Word2Vec to extract sequences from collected darknet traffic (unsolicited traffic to IP prefixes without any services). DANTE detected previously unreported attacks because of a downstream cluster analysis, subsequently performed on an embedding generated by Word2Vec. Following DANTE, DarkVec [12] outperforms IP2Vec and DANTE for darknet traffic analysis in both supervised and unsupervised scenarios. DarkVec uses Word2Vec and forms a corpus out of darknet traffic by placing IP addresses alongside each other in sentences based on domain knowledge about services commonly found on computer networks, such as HTTP, SSH, DNS, *etc.* DarkVec is the direct inspiration for our work, as we aim to characterise DDoS attacks at scale using a similar approach.

The results obtained with Word2Vec in the networking domain led us to explore other NLP techniques related to our work (§5). Doc2Vec [24] is an extension of Word2Vec that can learn vector representations of documents. Older NLP techniques associated with information retrieval and topic modelling are also still relevant today, such as Latent Semantic Analysis (LSA) [11]. We elaborately discuss LSA as a part of DDoS2Vec itself (§4).

Very few approaches are similar to our proposed solution in the context of DDoS attacks — especially amplification attacks. They are more generally involved with intrusion detection, but we mention them here for completeness. Goodman et al. [13] used Word2Vec on packet payloads to detect malware. Shima [50] used TF–IDF alone to analyse unusual port accesses based on packets captured from a darknet. Lassez et al. [23] used LSA to detect intrusions by reducing the dimensions of their dataset and its specific features. Wu et al. [57] used a probabilistic version of LSA (pLSA) to compare aggregated flow profiles for maliciousness. These approaches generally differ from our work in three major ways aside from the focus on general intrusion detection:

- Despite using techniques associated with the NLP field, they did not closely investigate the relationship between natural language and computer network communication.
- Evaluation was held back to binary classification (detection) where ML was involved, meaning they did not evaluate characterisation abilities.
- The datasets used were extremely limited or incomparable in environment and scale, which we discuss in greater detail (regarding the datasets) during the next section.

3 THE STATE OF DDOS DATASETS

To evaluate the characterisation performance of an approach, we need a dataset that includes a large and diverse amount of attack traffic. Obtaining such datasets is challenging (as previously explained by Wichtlhuber et al. [56]) due to the sensitive nature of the attack traffic. Besides, only *labelled* datasets indicate which flows belong to an attack and to which attack type. Publicly

Table 1. CIC-DDoS2019 top 5 IPv4 UDP ports for "DrDoS_DNS" (5,069,515 flows).

Most common source ports	Most common ports pairs	Most common destination ports
564 (13.311%)	53 → 7001 (0.001%)	63461 (0.004%)
634 (0.401%)	564 → 19862 (0.001%)	41966 (0.004%)
900 (0.195%)	626 → 15257 (0.001%)	11468 (0.004%)
530 (0.195%)	53 → 9 (0.001%)	18095 (0.004%)
512 (0.182%)	947 → 24314 (0.001%)	55369 (0.003%)

available labelled datasets exist, but we cannot use them (§3.1). Instead, we later describe a private dataset (§3.2) collected at an IXP and outline the alternative we followed to label it (§3.3).

3.1 An Examination of a Publicly Available Labelled Dataset

Several publicly available datasets contain labelled DDoS attack traffic. Such datasets include CIC-DDoS2019 [48], UNSW-NB15 [33], NF-UQ-NIDS [45], KDD Cup 1999 [19], and DARPA Intrusion Detection Evaluation datasets (1998–2000) [9]. A typical intended use case for these datasets is training ML models to detect DDoS attacks; however, their usefulness is primarily limited by the dataset network environment and the size/diversity of the observations. As an example of such limitations, we summarise CIC-DDoS2019 since it has attack-specific labels (e.g. with and without reflection/amplification) and is one of the more recent datasets.

Unrealistic port usage. As flow records do not contain packet payloads, we can only infer the type of traffic from the source and destination ports indicated in flow record fields. That is backed by the fact that most popular protocols over UDP use well-known ports [53], such as DNS servers listening on port 53. Knowing the protocol is mandatory for characterising DDoS amplification attacks at a flow level, which means we expect to see, e.g. DNS responses sent by an amplifier to use source port 53 generally. Successful research on amplification attacks at Internet infrastructure has used that premise [44, 20, 35]. Table 1 shows the port make-up for the flows labelled as "DrDoS_DNS" in CIC-DDoS2019, where source port 53 is not as common as we would expect. That trait is shared by other protocols, e.g. NTP, where source port 123 is not typical for flows labelled as "DrDoS_NTP". These elements of the CIC-DDoS2019 dataset are specific to a particular network environment and are not representative of real-world traffic.

Unknown attack configurations. The CIC-DDoS2019 dataset was generated by conducting DDoS attacks in a controlled environment over two days, with controlled attack parameters. The authors do not describe the attack configurations or the exact third-party software used to generate the attacks. For example, DNS requests sent to the DNS amplifier(s) impact the packet and byte sizes summarised by the "DrDoS_DNS" flows; hence, there are uncertainties about the experiments conducted. As another example, we observed 5,759 UDP flows labelled as "Syn" (TCP SYN flood) in the CIC-DDoS2019 dataset, which indicates potential labelling errors or configuration mistakes. The lack of information on the attack configurations used is a significant limitation. We cannot know how representative this dataset is of real-world attacks outside of attack taxonomy.

Small private networks are not reflective of critical Internet infrastructure. Outside of attack configurations, CIC-DDoS2019 was generated in a controlled environment with a small private network. There are 21 unique IP addresses within the dataset's flows when only considering IPv4 UDP flows, seemingly with 15 of them as the source of benign traffic and 20 as the destination of benign data. Only a few IP addresses are responsible for sending and receiving DDoS-related traffic, which indicates a severe lack of attack distribution in the dataset. In total, roughly 74 thousand UDP flows are labelled benign, while over 63 million UDP flows are labelled as belonging to an attack type. Since flow sampling is also absent, we can assuredly state that the dataset was not collected under general Internet-level conditions.

The suitability of such datasets for ground truth. Based on our brief examination of CIC-DDoS2019, we find that it is insufficient as ground truth for characterising DDoS attacks at scale and cannot be used to evaluate DDoS2Vec adequately without giving false confidence. Other datasets have similar limitations:

- UNSW-NB15 [33] is a general intrusion detection dataset which only contains 527 UDP flows labelled as "DoS" (attack type unknown) against 1.3 million benign UDP flows.
- NF-UQ-NIDS [45] is a large dataset created by merging several other datasets (such as UNSW-NB15). Questionably, it mixes flows from different computer networks into a new artificial one.
- KDD Cup 1999 [19] and the DARPA Intrusion Detection Evaluation datasets (1998–2000) [9] are outdated since they predate many newer techniques of conducting DDoS attacks, e.g. amplification.

Creating a realistic dataset is exceptionally challenging, as it requires collecting real-world DDoS attacks observed in the wild or generating realistic attacks in an extensive yet controlled environment — all while carefully covering a wide range of attack types. In light of this discussion, we describe the next best alternative.

3.2 Real-World IXP Flows

Our primary dataset of flow samples was from a medium-sized IXP with over 200 networks. The flow records are in the unidirectional NetFlow format [6] and sampled at a ratio of 1:4096. Roughly 20 million IPv4 UDP flow samples were collected at the IXP daily, with over 7.2 billion IPv4 UDP flow samples in total. Collection occurred without pause between mid-January and the end of December for the year 2019 (visualised in Fig. 6 as a part of the appendix). Even though these flow samples are a few years old, access to such a dataset is uncommon, and it allows us to assess the value of DDoS2Vec without inheriting flaws from unrealistic alternatives.

Ethics statement. This dataset is private, as potential Personally Identifiable Information (PII) is present in the dataset of flow samples due to IP addresses. We do not use IP addresses to recognise or identify users associated with those IP addresses at any time; they are only used to act as unique sources and destinations of other flow-level information. The dataset is private, so we cannot and have not shared it publicly, and no PII is present or used anywhere within our work.

Unavoidable information loss. Due to the nature of flow samples, we may miss specific traffic that we would ideally observe [16]. Furthermore, because of route asymmetry, the vantage point (the IXP) may see traffic from IP_1 to IP_2 but miss traffic from IP_2 to IP_1, as the traffic of IP_2 could take a different path on the Internet to reach IP_1 that does not cross the IXP. For example, we might see a DNS server sending DNS responses to an IP address but miss the IP address sending potential DNS requests to that DNS server. If so, we cannot determine from the IP level interactions alone whether the IP address is a DNS client or a victim of an amplification attack. Because of the scale, we cannot realistically rely on packet payloads.

The lack of ground truth. Evaluating the goodness of a DDoS attack characterisation approach depends upon some form of ground-truth labels. Traffic at IXPs is seldom labelled with DDoS attack information since it requires an elaborate process [56]. To account for this, previous DDoS attack studies regarding IXP traffic have used basic heuristics without label information to detect DDoS attacks [20] or have relied on auxiliary information from honeypots [35]. In essence, there are three options to obtain the ground truth desired: (i) manually labelling the traffic; (ii) learning characteristics from labelled datasets; or (iii) taking ground truth learned from other real-world IXPs and similar Internet infrastructure. Because (i) is time-consuming at a massive scale and more prone to human error/bias, and (ii) is unreliable (§3.1), we explore (iii).

3.3 Obtaining Ground Truth

To help assess DDoS2Vec, we adopt IXP Scrubber [56] as a pragmatic approach to obtaining ground truth. An important contribution of their work was a set of filtering rules made publicly available[1] that can be used to identify packet headers belonging to DDoS attacks in IXP traffic. We now briefly elaborate on how these filtering rules were generated, how they can act as ground truth, and how to understand their usefulness in our work's context of characterisation as opposed to detection.

Wichtlhuber et al. [56] presented an ML approach to "scrubbing" DDoS attack traffic at IXPs. The second step in the approach involves aggregating tagged flows into destination address records, where they then classify the aggregated records as benign or malicious (binary classification) based on the tagged flow(s) that contributed to the said record. We are interested in the first step of their approach, where they construct a set of human-interpretable tagging rules that can be used for filtering to address the lack of a suitably labelled dataset. They proposed a method to generate extensive amounts of labelled data by taking advantage of blackholed traffic. To validate their approach, they launched a small-scale controlled DDoS attack against their own network and captured flow samples at an IXP.

IXP Scrubber's filtering rules are human-interpretable descriptions of a flow likely to be blackholed, i.e. linked to a DDoS attack. An example of a rule is given in §A.1. Note that characterisation was not included in Wichtlhuber et al. [56]'s scope; thus, we aim to use the rules for evaluating characterisation in this work. We consider each rule as defining a characteristic and/or summarising less relevant characteristics (depending on the rule). We mark each flow in our IXP dataset as matching the highest confidence rule (if one of the rules matched the flow).

Alternative rule sets. We acknowledge that other rule sets for acting as ground truth exist, such as the Snort [43] Community Rules [51] and various rule sets for Suricata [52]. However, using these rule sets for our work is challenging, mainly because: (i) they are not specific to volumetric UDP-based DDoS attacks but are general Intrusion Detection System (IDS) rules; (ii) numerous DDoS-related rules match on packet payloads (i.e. for the Snort Community Rules, alert "PROTOCOL-DNS DNS query amplification attempt" looks into the DNS header), which we do not have access to; and (iii) the IDS rules are less tailored to our IXP environment, as they were not formulated under IXP circumstances. Proceeding with the IXP Scrubber filtering rules is the most pragmatic approach.

4 DDOS2VEC

In essence, DDoS2Vec attempts to view network traffic as a group of natural language documents. It uses Latent Semantic Analysis (LSA) to transform each document into a vector, allowing a comparison of network traffic characteristics. We first explain our process for generating a *textual corpus* from flow records (§4.1), which allows the usage of NLP techniques. Then, we show how we used LSA to turn the documents into a single vector space (§4.2), allowing us to compare network traffic characteristics.

4.1 Flow Corpus Generation

Since NLP techniques cannot directly use flow records, we must transform them into a format similar to a natural language. DDoS2Vec achieves this by creating *documents* that are lists of *words*, which can be seen as lists of *flow records*, as a DDoS2Vec word represents some information about an individual flow record. Each document has a *tag*, which identifies what the document represents. A *corpus* is a collection of tagged documents, and a *vocabulary* is the set of all words in a corpus. Before we explain our process of creating a corpus, we initially discuss how one can design a flow

[1] https://github.com/DE-CIX/ripe84-learning-acls

corpus generation process and why it is a crucial step. There are three major design elements to consider: (i) which underlying network semantics will words represent in the corpus vocabulary; (ii) how to handle non-discrete numeric values; and (iii) what tags to use for the documents.

Defining the core vocabulary. A vocabulary for a flow corpus can represent any information about flow records; however, as we are primarily concerned about DDoS attacks, we aim to represent information that can help identify them. We can achieve that by defining a vocabulary representing the key network semantics of DDoS attacks while considering the limited information in flow records. The source and destination ports are an obvious choice of information to represent at a fundamental level. A source port can indicate an amplification protocol used in an attack [44, 21, 20], while a destination port can indicate the specific port being targeted. Source ports are more important than destination ones as there is some knowledge about which ports are commonly used in amplification attacks, e.g. 53 for DNS [53].

Table 2. Example of domain knowledge of behaviours/services.

Name	Source Ports	Destination Ports	Packet Size Ranges	Packet Size Interval
"CLDAP DDoS"	389	$[0, 65535]$	$\in [0, 150], > 150$	–
"CharGEN"	19	$[0, 65535]$	> 0	–
"DNS DrDoS"	53, 853, 5353	$[0, 65535]$	> 550	–
"DNS"	53, 853, 5353	$[0, 65535]$	–	150
"Kerboros"	88	$[0, 65535]$	–	32
"Memcached DDoS"	11211	$[0, 65535]$	> 255	–
"NTP DDoS"	123	$[0, 65535]$	> 99	–
"NTP"	123	$[0, 65535]$	–	–
"NetBIOS"	137, 138, 139	$[0, 65535]$	–	100
"SNMP DDoS"	161	$[0, 65535]$	> 150	–
"TFTP"	69	$[0, 65535]$	–	300
"Generic (System-to-System)"	$[0, 1023]$	$[0, 1023]$	–	100
"Generic (System-to-User)"	$[0, 1023]$	$[1024, 49151]$	–	100
"Generic (System-to-Dynamic)"	$[0, 1023]$	$[49152, 65535]$	–	100
"Generic (User-to-System)"	$[1024, 49151]$	$[0, 1023]$	–	100
"Generic (User-to-User)"	$[1024, 49151]$	$[1024, 49151]$	–	100
"Generic (User-to-Dynamic)"	$[1024, 49151]$	$[49152, 65535]$	–	100
"Generic (Dynamic-to-System)"	$[49152, 65535]$	$[0, 1023]$	–	100
"Generic (Dynamic-to-User)"	$[49152, 65535]$	$[1024, 49151]$	–	100
"Generic (Dynamic-to-Dynamic)"	$[49152, 65535]$	$[49152, 65535]$	–	100

Representing domain knowledge. We represent/encode generally understood behaviours as words in a vocabulary. Gioacchini et al. [12] demonstrated that forming corpora based on domain knowledge improves learning performance by helping expose relevant information in the underlying dataset. Table 2 provides an example of domain knowledge, where each behaviour has its own formation rules. We adopt Table 2 throughout the rest of this paper as the source of domain knowledge for forming vocabularies to demonstrate how domain knowledge can be incorporated into a corpus vocabulary[2].

Discretisation. In many NLP techniques, numbers are simply treated as words (e.g. "1" and "2" are as different as "1" and "300"). That works for ports but not for quantities, such as packet and byte counts, which are better expressed as numeric intervals. DDoS2Vec transforms continuous values into discrete ones. The intervals and thresholds are set such that the vocabulary size is manageable, particularly regarding behaviours we consider generic or unknown. We illustrate that in Table 2, which shows where each interval or threshold yields a word in the vocabulary (different ranges indicate a variation of a behaviour). Like IXP Scrubber [56] (see §3.3), we adopt 100-byte intervals for generic behaviours.

Document tags. The next step is to place words into the documents which form the corpus. DDoS2Vec tags each document with a word (a document tag) that is unique, representing the

[2]Note that we avoid any claims about the completeness/accuracy of the example behaviours in Table 2.

Table 3. An example of three UDP flow records.

Timestamp	Source IP	Destination IP	Source Port	Destination Port	Packet Count	Byte Count
1648468800	192.168.1.40	192.168.1.50	11211	60000	2	2230
1648468902	192.168.1.50	192.168.1.60	49284	837	1	186
1648469003	192.168.1.1	192.168.1.70	53	58394	1	371

Table 4. Example corpus based on Table 2, Table 3, and Algorithm 1.

Document Tag	Document Words
"192.168.1.50"	"Memcached DDoS", "> 255", "11211->", "->60000", "Memcached DDoS", "> 255", "11211->", "->60000"
"192.168.1.60"	"Generic (Dynamic-to-System)", "[100,199]", "49284->", "->837"
"192.168.1.70"	"DNS", "[300,399]", "53->", "->58394"

document contents, and works as a title or identifier for the document itself. A unique tag will identify the vector representation of each textual document. A major decision is how to tag documents, as the document tag is the only (easy) way to identify a document after it is transformed into a vector during later stages.

Tagging documents with IP addresses. So far, we have not discussed the application of IP addresses to the corpora that DDoS2Vec generates using flow records as input data. IXPs may observe many IP addresses, which can further increase during DDoS attacks, thus inflating the vocabulary size. Including IP addresses in the vocabulary may also emphasise benign IP-to-IP relationships [41, 7, 12] over other flow record characteristics. For example, two attacks may have identical flow-level characteristics in all other manners; however, those attacks will not be as similar if the IP addresses are merely different (despite identical attack patterns). Additionally, source IP addresses can be spoofed, and the deployment of countermeasures is incomplete [29, 34, 28].

Destination IP addresses as document tags. In a DDoS attack that utilises reflection, the destination IP addresses indicated in attack flow records effectively involve the reflector's IP address and the victim's IP address. During *post*-reflection, the victim will receive notably different attack traffic than other destination IP addresses, including the reflector's IP address during *pre*-reflection. The flows received by reflectors and victims will be noticeably different — especially during amplification attacks due to amplification factors [44]. DDoS2Vec thus tags each document with the IP address of the flow record's destination, enabling the comparison of traffic directed to different destination IP addresses, i.e. victims.

Corpus generation. We present the algorithm to transform flow records into a DDoS2Vec corpus in Algorithm 1 (see comments in the pseudocode). The algorithm takes two inputs: (i) flow records, sorted by the earliest timestamp; and (ii) behaviours, with the most specific ones first. If there is a match between a flow record and a behaviour (L9), the behaviour's name (L10) and matching interval (L11) are used as words. The interval is calculated based on the average packet count, as flow records lack per-packet byte count information. We repeat the word sequence by the packet count (L15–L18) so word occurrence matches that of packet count. Note that we use the notation " . " to indicate the concatenation of two tuples, i.e. the right-hand tuple is appended to the left-hand tuple to create a new tuple. Algorithm 1 outputs the corpus C (effectively a list of lists summarising flow record information). An example corpus is shown in Table 4 using the example flow records in Table 3. Note that the words do not necessarily have to be human-readable and can be given any arbitrary appearance, but they must be consistently formed from network semantics. The corpus specified by C is directly used and transformed in the following stages of DDoS2Vec.

Algorithm 1 Flow corpus generation pseudocode.

Require: $F = (R_1, \ldots, R_n)$ ▷ F denotes a tuple of flow records.
Require: $K = (B_1, \ldots, B_n)$ ▷ K denotes a tuple of behaviours based on domain knowledge.
$T \leftarrow \emptyset$ ▷ T denotes a set of tags (special words that identify documents).
$C \leftarrow \emptyset$ ▷ C denotes a multiset (a set allowing duplicate elements) of documents (tuples of words).
$m\colon T \rightarrow C$ ▷ m denotes a function that maps tags to documents for the purpose of identification.
for all $R \in F$ **do** ▷ The flow records are sorted by timestamp.
  $t \leftarrow$ StringIfy($R_{destination\ address}$) ▷ t denotes a document tag.
  $s \leftarrow$ StringIfy($R_{source\ port}$) + "->" ▷ s denotes a word representing a source port.
  $d \leftarrow$ "->" + StringIfy($R_{destination\ port}$) ▷ d denotes a word representing a destination port.
  for all $B \in K$ **do** ▷ The most specific behaviours are checked first.
    if BehaviourMatch(R, B) **then** ▷ If the flow record matches the behaviour's criteria...
      $n \leftarrow$ BehaviourName(B) ▷ Use the behaviour's name as a word.
      $i \leftarrow$ BehaviourInterval(B, $\frac{R_{byte\ count}}{R_{packet\ count}}$) ▷ The average packet size is used for the interval.
      break ▷ The most specific behaviour matched, so break the loop.
  ▷ Assert that n and i are defined, as a behaviour must have matched — however generic it may be. ◁
  $D \leftarrow \emptyset$ ▷ D denotes an empty tuple. It represents the document for the tag t.
  $c \leftarrow 0$ ▷ We repeat words as necessary to add extra weight based on the packet count.
  while $c < R_{packet\ count}$ **do** ▷ The packet count will always be above zero.
    $D \leftarrow D\ .\ (n, i, s, d)$ ▷ Concatenate the tuples (extend the document D).
    $c \leftarrow c + 1$ ▷ Increment the counter.
  if $t \in T$ **then** ▷ If this tag has been seen before...
    $O \leftarrow m(t)$ ▷ Retrieve the older document with its older words.
    $D \leftarrow O\ .\ D$ ▷ Effectively prepend the older document to the newly-formed document.
    $C \leftarrow C \setminus \{O\}$ ▷ Now remove the older document from the corpus, as we will replace it next.
  else ▷ Otherwise...
    $T \leftarrow T \cup \{t\}$ ▷ Add the new tag to the set of tags.
  $C \leftarrow C \cup \{D\}$ ▷ Add the new or updated document (words representing what t received) to the corpus.
  $m(t) = D$ ▷ t of T is now mapped to D of C.
output T, C, m ▷ C is the corpus required for DDoS2Vec's next stage (LSA).

4.2 Latent Semantic Analysis (LSA)

LSA is a classic NLP technique for comparing the similarity of documents based on their word frequencies [11]. It is a loosely defined technique generally consisting of two significant steps: Term Frequency – Inverse Document Frequency (TF–IDF) and a truncated version of Singular Value Decomposition (SVD).

TF–IDF. In order to transform a document into a vector representation, we use TF–IDF to create a document-term matrix (a matrix of word frequencies in each document). TF–IDF is a popular NLP and information retrieval technique [42, 14, 61]. There are many variations of TF–IDF, so for DDoS2Vec, we use and explain one implemented in the popular scikit-learn library [38]. As indicated by its name, TF–IDF combines Term Frequency (TF) and Inverse Document Frequency (IDF). The goal of TF–IDF is to weigh how important a word is to a document both locally within a document and globally across the corpus. For brevity here, we explain the exact equation we use in the appendix (§A.2). Since TF–IDF produces a very wide matrix, LSA involves a dimensionality reduction technique to avoid the well-known curse of dimensionality.

Truncated SVD. We use the truncated SVD of the document-term matrix as the primary output for DDoS2Vec. SVD, which is a well-known linear algebra matrix factorisation technique, essentially decomposes a matrix into three matrices. The three matrices form the original matrix when multiplied together, and a truncated version allows for a low-rank approximation of the original matrix. Each row in that matrix for a DDoS2Vec corpus represents a potential DDoS attack. We provide a deeper overview of SVD in the appendix (§A.3).

Alternative techniques. Together, TF–IDF and truncated SVD represent a typical implementation of LSA; however, there are numerous alternatives that we do not explore in this paper. Instead of TF–IDF, we could use different document vectorisation techniques, i.e. entropy as the global weighting scheme instead of IDF. Instead of truncated SVD, other options for dimensionality reduction include Non-negative Matrix Factorisation (NMF) [25], sparse random projection [27], and Latent Dirichlet Allocation (LDA) [11].

Explainability. DDoS2Vec works because it can learn the semantics of network behaviour from the training corpus via word occurrences and then use that knowledge to compare the similarity of new documents (new flow records belonging to a potential attack). As DDoS2Vec is relatively simple, it is *explainable* [18] compared to other more complex machine learning techniques, such as deep learning black-box models. That is because DDoS2Vec requires very few steps compared to more complex techniques, and the steps themselves are not sophisticated. DDoS2Vec is also relatively interpretable because we can inspect the words in the corpus, and the resulting vectors by the truncated SVD step via a standard calculation (§A.3) are also created without randomness.

Attacks against DDoS2Vec. An attacker could attempt to poison the corpus by purposefully changing their attack's characteristics from other known and labelled DDoS attack characteristics. This would require noticeable effort on the attacker's part, as they would need to change the characteristics of their traffic's flow records towards victims to prevent the victim's document vector from being similar to known DDoS attack victims. We do not study this further, but note that it is a possible avenue for an attack against DDoS2Vec's characterisation abilities.

5 COMPARING DDOS CHARACTERISATION APPROACHES

To provide an overview of DDoS2Vec's ability to characterise potential attacks, we compare it against other approaches for characterising DDoS attacks at a flow level. The baseline NLP approaches are Doc2Vec [24] and Word2Vec [31], which use the same corpus as DDoS2Vec. We create a baseline non-NLP approach for comparison, which we call *Counter*. For brevity here, we elaborate on the approaches in §A.5 instead. We first explain the evaluation setup (§5.1), the metrics used (§5.2), and finally, present the results (§5.3).

5.1 Setup

Preparing the dataset. We apply the rules indicated in §3.3 to all flow samples in the dataset. We then take all flow samples from November 2019 (a relatively busy month of traffic) to train and test the solutions. Each rule has its identifier (§A.1) act as a label. As the dataset is highly imbalanced with a vastly different number of occurrences per rule (support), we reduce the occurrence of the null rule (no match) to meet the highest occurring rule's support (this does not apply to the Word2Vec run, which we explain later). We also remove all rules with fewer than 10 occurrences. While that does not balance the dataset, it prevents the null rule from dominating the others. Fig. 1 shows the class distribution after the trimming.

Training and testing. Each solution is trained and tested on the same dataset split into a training set and a test set via 5-fold cross-validation with shuffling and iterative stratification [46]. Note that the split is done after transforming flow records but only before an approach's training stage to avoid data leakage; however, that does not apply to Word2Vec, where we generate one embedding over the whole dataset instead of an embedding for all five training sets individually. We do this for Word2Vec to avoid Out-Of-Vocabulary (OOV) issues, as the vocabulary is comprised of IP addresses that are potential victims (unlike a vocabulary for the document approaches). We acknowledge this as a case of "data snooping" [3], as the train-test split is done after embedding generation, causing a *data leakage*. Regardless, this still allows us to gauge the distribution of the vectors in the embedding space and is more similar to DarkVec's evaluation [12] since the classifier

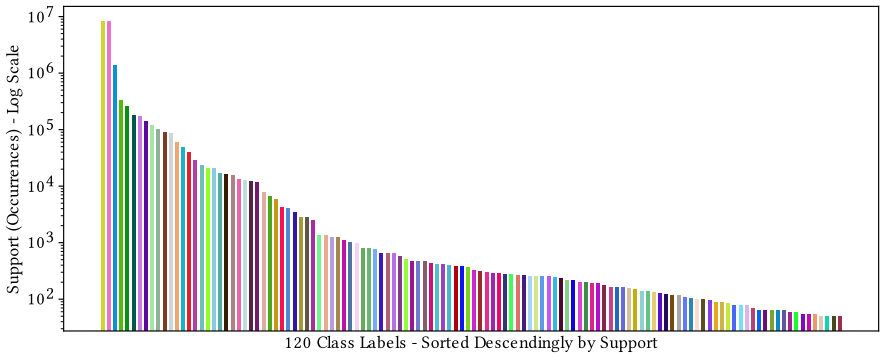

Fig. 1. Label distribution of November 2019 (null match rule trimmed).

is still only trained post-split. Additionally, no trimming of the majority class label is performed, meaning we generate a Word2Vec embedding for the whole unprocessed month.

DDoS2Vec hyperparameters. DDoS2Vec is based on LSA, meaning that it inherits its hyperparameters. We kept vector length at 100 (similar to the other approaches' hyperparameters). We generated uni-grams, bi-grams, and tri-grams as new words within each document for the DDoS2Vec corpus used in this section. We did not trim any words from documents based on minimum or maximum word counts across the entire corpus for the approach comparisons. Domain knowledge was included in the flow corpus generation process (all non-generic behaviours indicated in Table 2 during §4.1). This example configuration is unassuming, and we perform hyperparameter tuning after the approach comparisons.

Multi-label classification. We used a k-NN classifier with $k = 10$ and distance weighting for all solutions. While other classifiers commonly provide stronger classification performance, e.g. XGBoost [5, 56], we adopted k-NN due to its ubiquity. The classifier aims to predict the set of IXP Scrubber rules (in §3.3) that apply to each document. We did not give the classifier any information on how much a rule weighs for each document; instead, it predicts a binary value for each rule (indicating its presence or contribution to a document). The null match (class label "—") can also be predicted, which corresponds to a flow record that did not match any of the IXP Scrubber filtering rules, yet it still contributed to a document.

System and version information. All evaluations in this paper were conducted on a GNU/Linux system with an AMD EPYC 7702 processor with 64 cores (128 total threads) and 1 TB of memory. The system was running Ubuntu 18.04.1 with a 5.4.0-144-generic kernel. Python 3.10.9 with the following libraries was used: NumPy 1.24.3, SciPy 1.8.1, scikit-learn 1.2.1, iterative-stratification 0.1.7, and Gensim 4.3.0. The seed used for all random number generation was 463. We publicly release all code[3] used in this paper; however, we cannot release our dataset or the accompanying corpora due to their highly sensitive nature. Regardless, the open-source research artefacts we provide (as a collection of Jupyter notebooks) can be used to customise, extend, or reproduce our exact steps for all experiments on a different set of flow records.

[3]https://github.com/RavSS/DDoS2Vec

5.2 Metrics

We now explain our general criteria for selecting the best approach. This is a multi-label classification problem, and its results are quantified using classic ML metrics (§A.4) for which higher values are desirable. We use both *macro-averaged* and *micro-averaged* metrics. The macro-averaged metrics do not consider the imbalance of the rules, while the micro-averaged metrics do. The imbalance is natural, as some rules (characteristics) are simply more common than others. To understand the dataset imbalance's impact on the metrics and the least acceptable values for each metric, we include a "dummy" approach that only predicted the majority null rule ("—") for every potential attack. The "dummy" approach effectively tells us the worst-case scenario for each metric, as it always provides a meaningless characterisation of benevolence regardless of data.

Selection criteria. To summarise the importance of both rare and common characteristic prediction, we take the macro and micro values for each averaged metric and again average (arithmetic mean) them to obtain an "intermediate" value. We evaluate the goodness of an approach by its intermediate F1 score, which considers precision and recall. A high precision means the approach can correctly recognise the characteristics of attacks, while a high recall indicates that the approach can consistently recognise said characteristics. In essence, the approach that has the highest value for all metrics is the best and can provide the clearest DDoS attack characterisation.

5.3 Results

We now show how well each approach can characterise a potential DDoS attack based on the predicted IXP Scrubber rules that matched its related flow samples. First, we present the overall classification performance, followed by the classification performance for individual classes (limited to the top 15 filtering rules). Afterwards, we experiment with DDoS2Vec hyperparameters (including a small ablation study) by tweaking them to obtain better performance values.

Characterisation considering all rules. Table 5 shows the characterisation ability for each of the approaches, as expressed by different classification metrics (we highlight in blue the best values to aid visualisation). First, the worst intermediate F1 score (excluding "dummy") was from Word2Vec (0.27), even considering it observes all data prior to the train-test split and therefore has the advantage of not needing to infer vectors post-training. We hypothesise that most words in the Word2Vec corpus were not learned adequately due to the non-ideal default negative sampling hyperparameters, which most likely suit natural language corpora better than our domain-specific ones. Doc2Vec and Counter come close to each other, achieving the same intermediate precision value (0.57), but the latter has a higher recall. We expected Counter to outperform Doc2Vec because most of the IXP scrubber rules are associated with prevalent abused protocols in DDoS, such as DNS and NTP, and can be covered reasonably well with domain knowledge alone. However, Doc2Vec highlighted the benefit of NLP being a "dynamic" approach, not purely limited by domain knowledge. Finally, DDoS2Vec achieved a noticeable gap over the other approaches with an intermediate F1 score of 0.64, indicating that it can learn attack characteristics better than the other approaches. Both the macro-averaged and intermediate precision values for DDoS2Vec are noticeably higher than the other approaches, meaning DDoS2Vec avoids false and/or misleading characterisations of potential attacks much better than the other approaches. In essence, DDoS2Vec combines the best of both Doc2Vec and Counter, even with an untuned selection of hyperparameters.

Computational cost. Table 5 also shows the time taken for training and testing during cross-validation [4] on the preprocessed month of flow samples that summarise roughly 752 million packets. Note that the time for corpus generation or count transformation was done before cross-validation; hence, the time taken for those steps is not reflected in Table 5, but nonetheless they are negligible

[4]Includes fitting, transforming, and predicting — along with k-NN itself.

Table 5. Characterisation results; considering all rules.

Metric	Dummy	Doc2Vec	Word2Vec	Counter	DDoS2Vec
Macro-Averaged Precision	0.01	0.32	0.01	0.22	0.47
Macro-Averaged Recall	0.01	0.12	0.02	0.12	0.30
Macro-Averaged F1 Score	0.01	0.15	0.01	0.14	0.34
Micro-Averaged Precision	0.73	0.82	0.73	0.91	0.96
Micro-Averaged Recall	0.42	0.72	0.78	0.88	0.92
Micro-Averaged F1 Score	0.53	0.76	0.75	0.90	0.94
Intermediate Precision Value	0.37	0.57	0.37	0.57	0.71
Intermediate Recall Value	0.21	0.42	0.40	0.50	0.61
Intermediate F1 Score Value	0.27	0.46	0.38	0.52	0.64
Exact Match Ratio	0.15	0.46	0.39	0.77	0.84
Mean Training Time per Fold (H:MM:SS)	0:00:10	7:12:15	0:00:40	0:00:35	4:33:19
Mean Testing Time per Fold (H:MM:SS)	0:01:18	3:04:40	0:11:01	0:08:08	0:29:15

compared to the later steps. The slowest approach (Doc2Vec) took roughly two days to train for five embeddings. Word2Vec took a few hours before cross-validation for its single embedding (explained in §5.1). Counter was by far the fastest, taking only a few minutes, as it does not require a training state. DDoS2Vec is substantially slower than Counter during its training over five individual corpora, but the gap shortens considerably during testing (inferring new attacks), and we further shorten it overall during hyperparameter tuning. Training is not performed often, and training and testing can be adjusted for different traffic periods or in a more selective manner than considering every potential victim (destination IP address) in a month. The vocabulary size (at its upper limit here) and corpus size are the main factors affecting the computational cost of DDoS2Vec, which we discuss later in this paper. Despite the computational cost, DDoS2Vec doubles the classification performance of Counter on recognising subtle attack characteristics in traffic at scale in what is a non-urgent task (compared to live attack detection).

Focused analysis. Since not all rules are equally prominent in our dataset due to severe dataset imbalance, we zoom in on the 15 most prevalent rules. Table 6 shows to which extent each approach can classify attack characteristics through precision, recall, and F1 score metrics. Comparing row by row, we can observe that DDoS2Vec generally outperforms the other solutions (to aid visualisation, we highlight in blue the best F1 score in each row, and any values ≤ 0.60 are in red). The low recall performance of Doc2Vec indicates it falls short in consistently recognising DDoS attack characteristics, so it is inferior to Counter within this aspect. Counter performs its absolute best on the most prevalent rules due to our example of domain knowledge in §4.1 (that we based Counter upon) covering many of them. Despite that, DDoS2Vec still outperforms Counter, meaning DDoS2Vec can learn common characteristics better than it too, not just rarer characteristics.

Elaborating on the performance of DDoS2Vec. DDoS2Vec (via LSA) is essentially a dimensionality reduction technique that condenses documents (which describe potential DDoS attacks) into an embedding (a vector space). DDoS2Vec outperforms the other NLP approaches tested on the same corpus because DDoS2Vec is more sensitive to individual behaviour words (unlike Doc2Vec). On the other hand, the sentence approach of Word2Vec means that the destination IP address words (the only words in the special Word2Vec corpus) are too heavily weighted on proximity. The Counter baseline approach does not require training, but that also means it cannot emphasise unique characteristics of DDoS attacks that are not covered by domain knowledge (which may be flawed, as we cover later on). Different corpora for Doc2Vec and Word2Vec — without data leakage or built-in knowledge of the IXP Scrubber filtering rules — may increase their performance, but we do not expect them to outperform LSA as the core NLP technique for DDoS2Vec.

Table 6. Characterisation results; top 15 most prevalent rules only.

Filtering Rule		Doc2Vec			Word2Vec			Counter			DDoS2Vec		
Label	Support	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
–	8303095	0.81	0.78	0.79	0.73	0.93	0.82	0.98	0.94	0.96	0.99	0.94	0.96
66cc3d22	8302945	0.83	0.82	0.83	0.73	0.93	0.82	0.93	0.98	0.95	0.98	0.97	0.97
a4027970	1415070	0.79	0.26	0.39	0.12	0.00	0.00	0.65	0.39	0.48	0.89	0.71	0.79
be412651	336845	0.80	0.52	0.63	0.00	0.00	0.00	0.92	0.83	0.87	0.97	0.90	0.93
2be8ca2f	260205	0.90	0.55	0.68	0.00	0.00	0.00	0.87	0.75	0.81	0.88	0.84	0.86
cdcedb50	178000	0.91	0.63	0.74	0.00	0.00	0.00	0.85	0.75	0.80	0.85	0.81	0.83
cab055dc	171195	0.37	0.05	0.09	0.00	0.00	0.00	0.83	0.88	0.86	0.75	0.78	0.76
8ea23b49	142795	0.68	0.19	0.30	0.00	0.00	0.00	0.72	0.46	0.56	0.82	0.79	0.80
6333a63c	118740	0.62	0.16	0.25	0.00	0.00	0.00	0.68	0.27	0.38	0.72	0.66	0.69
6512a9da	100950	0.80	0.60	0.68	0.00	0.00	0.00	0.75	0.67	0.71	0.75	0.74	0.75
204bef79	91530	0.84	0.30	0.43	0.00	0.00	0.00	0.32	0.40	0.35	0.92	0.96	0.94
1a84f2f8	86065	0.64	0.12	0.21	0.00	0.00	0.00	0.70	0.22	0.34	0.88	0.68	0.77
c8baa04b	59560	0.58	0.13	0.22	0.00	0.00	0.00	0.58	0.18	0.28	0.71	0.57	0.63
2e4c3e69	48725	0.49	0.00	0.01	0.00	0.00	0.00	0.93	0.93	0.93	0.81	0.79	0.80
9021485b	39525	0.60	0.07	0.13	0.00	0.00	0.00	0.57	0.15	0.24	0.81	0.56	0.66

Table 7. DDoS2Vec minimum word frequency and vector length impact on intermediate F1 score values and mean evaluation time per fold (H:MM).

Minimum Word Frequency	Vector Length 100	200	300
10	0.658 (4:06)	0.662 (6:15)	0.662 (7:55)
1000	0.645 (2:26)	0.652 (2:50)	0.650 (2:32)
10000	0.594 (2:35)	0.596 (2:36)	0.598 (2:29)

Table 8. DDoS2Vec *n*-gram generation's performance impact.

n-grams Generated	Intermediate F1 Score Value	Mean Evaluation Time per Fold (H:MM)
Uni-grams	0.668	0:32
Uni-grams, Bi-grams	0.656	1:24
Uni-grams, Bi-grams, Tri-grams *(Table 5)*	0.635	*5:03*
Bi-grams	0.632	1:07
Bi-grams, Tri-grams	0.617	2:38
Tri-grams	0.579	1:48

Minimum word frequency and vector length hyperparameter sensitivity. For DDoS2Vec, we performed a basic grid search over the minimum word frequency and vector length hyperparameters (5-fold cross-validation included with mean averages taken across all folds). Table 7 shows the intermediate F1 scores for each combination of hyperparameters. We observe that requiring a higher minimum frequency for all unique words across the corpus has a more significant penalty on the intermediate F1 score than the vector length. Removing less common words from the vocabulary decreases the model's ability to infer potential DDoS attacks correctly and understand rarer characteristics. Also seen in Table 7 is the time impact (in parentheses). The main advantage of removing less common words and decreasing the vector length is the (often considerable) reduction in time taken for both training and testing (evaluation). For higher minimum word frequencies, we see a significant reduction in time taken for computing the truncated SVD of the document-term matrix outputted during TF–IDF vectorisation — the latter of which is not optimised for speed in the generic single-threaded scikit-learn implementation.

***n*-gram sensitivity.** We further experimented with DDoS2Vec's hyperparameters to understand the impact of word order and *n*-gram generation. Table 8 shows the performance impact of generating *n*-grams (up to tri-grams) from the corpus. Generating uni-grams alone provides the best performance, but the performance gap is insignificant until only tri-grams remain. The key outcome is that word order is not as important as the words themselves, as the *n*-grams where $n > 1$ appear

Table 9. DDoS2Vec domain knowledge's performance impact and best hyperparameter tuning.

Metric	Domain Knowledge (*Table 5*)	No Domain Knowledge	Tuned
Macro-Averaged Precision	0.47	0.50	0.60
Macro-Averaged Recall	0.30	0.33	0.38
Macro-Averaged F1 Score	0.34	0.38	0.44
Micro-Averaged Precision	0.96	0.96	0.96
Micro-Averaged Recall	0.92	0.92	0.94
Micro-Averaged F1 Score	0.94	0.94	0.95
Intermediate Precision Value	0.71	0.73	0.78
Intermediate Recall Value	0.61	0.63	0.66
Intermediate F1 Score Value	0.64	0.66	0.70
Exact Match Ratio	0.84	0.84	0.87
Mean Training Time per Fold (H:MM:SS)	4:33:19	2:26:44	0:10:25
Mean Testing Time per Fold (H:MM:SS)	0:29:15	0:17:40	0:06:01

to reduce the significance of the words (uni-grams); thus, we can save a substantial amount of time by avoiding n-gram generation. This is likely to be the case because the IXP Scrubber filtering rules identify individual flow characteristics, not flow sequence characteristics; thus, this particular outcome may potentially be different for other labelled datasets.

Non-domain-knowledge corpus. In Table 9, we experimented with a corpus that did not include domain knowledge (generic behaviours only), with all other hyperparameters remaining the same as described in §5.1. We observe a performance gap in the non-domain-knowledge corpus's favour. That indicates flaws in our example domain knowledge (which the Counter baseline wholly relied upon), meaning there is missing knowledge and/or the behaviours are too broad, so more subtle characteristics of DDoS attacks are obscured. To highlight DDoS2Vec's potential, we also provide an extra run for our best-case DDoS2Vec configuration based on the results in the previous experiments, which we indicate in Table 9 as "Tuned". It involves no domain knowledge, uni-grams only, no minimum word frequency, and a vector length of 200. We achieved a further performance increase on the rarer rules, but most importantly, there is a substantial computational cost decrease. Despite the hyperparameter tuning, we proceed with the initial hyperparameters selected in §5.1 for the longitudinal analysis in the next section to avoid a best-case evaluation only.

6 LONGITUDINAL ANALYSIS

This section analyses the *transferability* of a month of IXP traffic (2019-06-01 — 2019-07-01) over to the rest of 2019 at that IXP. Unlike the previous section, the IXP flows were not filtered, balanced, trimmed, etc., in any manner except that we again only consider IPv4 UDP flows. We first investigate the corpora generated by DDoS2Vec (§6.1), then show how a model trained on that month of IXP traffic performs on other months of IXP traffic (§6.2).

6.1 Investigating DDoS2Vec-Generated Corpora

DDoS2Vec, a domain-specific NLP technique to characterise potential DDoS attacks, generates corpora as the result of its first stage (§4.1). The later stages work on the generated corpora, and the quality of a corpus is essential for the quality of its corresponding DDoS2Vec model; therefore, we more closely investigate the link between NLP and computer network communication. For all corpora generated by DDoS2Vec for all months, we focus on two aspects: (i) word frequencies in the vocabularies and (ii) vocabulary overlap of corpora between months.

Word frequency distributions. We consider two cases: the training corpus and all testing corpora combined (counts for each word are mean averaged during the combination). There are two comparisons to be made; first, we compare DDoS2Vec corpora with natural languages. In many natural languages (e.g. English), the word frequency distribution for a sizeable corpus closely

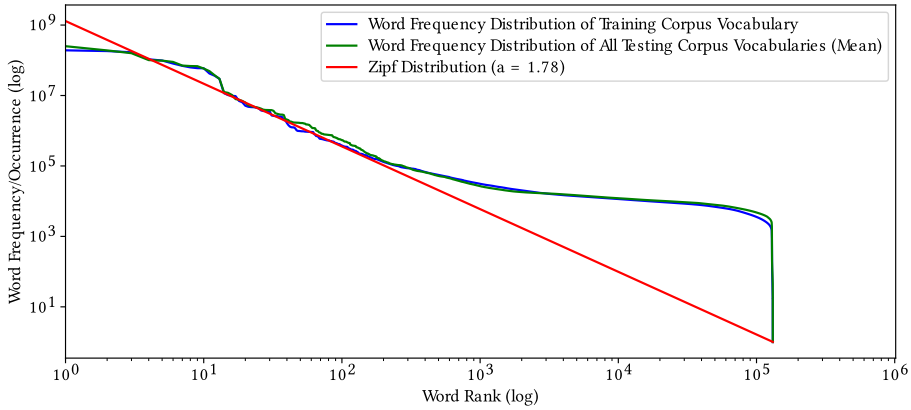

Fig. 2. Log-log plot of word frequencies against word ranks in corpus vocabularies.

follows a Zipf distribution [39]. We investigated if this also applied to the corpora generated by DDoS2Vec[5]. Fig. 2 shows (in log scale for both axes) the frequency or count of words against their popularity (rank) from the most popular to the least popular word. The word frequency distribution of the corpora does not closely fit a Zipf distribution: the vast majority of less popular words (whose rank is $\geq 10^3$) occur much more often than the less popular words of natural languages. The second (more telling) comparison is between the training corpus and testing corpora. We observe that their distributions are *almost identical*, even considering that the traffic for the test corpora was observed in completely different months, which is a promising sign of transferability.

Out-Of-Vocabulary (OOV) words. A corpus generated by DDoS2Vec has a vocabulary that is comprised of all the words that appear in the corpus's documents. As such, the make-up of the vocabulary changes based on the flow records considered. OOV words are those found in documents to be processed but absent from the training vocabulary, i.e. "new" words. They may limit the ability of a DDoS2Vec model to characterise a potential DDoS attack, as the words describing flow-level behaviour were not present in documents used for training the DDoS2Vec model. Hence, we investigated the prevalence of OOV words in documents in our IXP dataset. Fortunately, when DDos2Vec is trained on a month of IXP traffic, the presence of OOV words in other months is negligible. The highest OOV word situation happened for 2019-05, where just 6 unique words with 132 occurrences are OOV from a vocabulary of 131141 unique words with over 2.4 billion occurrences in that month. The vocabulary size for all twelve months individually ranges from 131134 to 131141 unique words, which is highly stable. For example, the Jaccard similarity between the vocabulary sets of any two months is greater than 0.99. Like the word frequency distribution similarities, the lack of OOV issues increases transferability, which we evaluate next.

6.2 Classification Performance Drift

The results of §5 showed that DDoS2Vec could learn attack characteristics within a month of IXP traffic itself. We now investigate whether a DDoS2Vec model trained in a month (2019-06) can be transferred to other months with reasonable classification performance. We evaluated backward from 2019-05 to 2019-01 and forward from 2019-07 to 2019-12. As with §5, we use an identical multi-label classification setup, except we do not remove/trim any documents, and we consider all IXP Scrubber filtering rules (§3.3) regardless of support levels. The hyperparameter configuration

[5]Not including the addition of n-gram words where $n > 1$ for the later TF–IDF stage of DDoS2Vec.

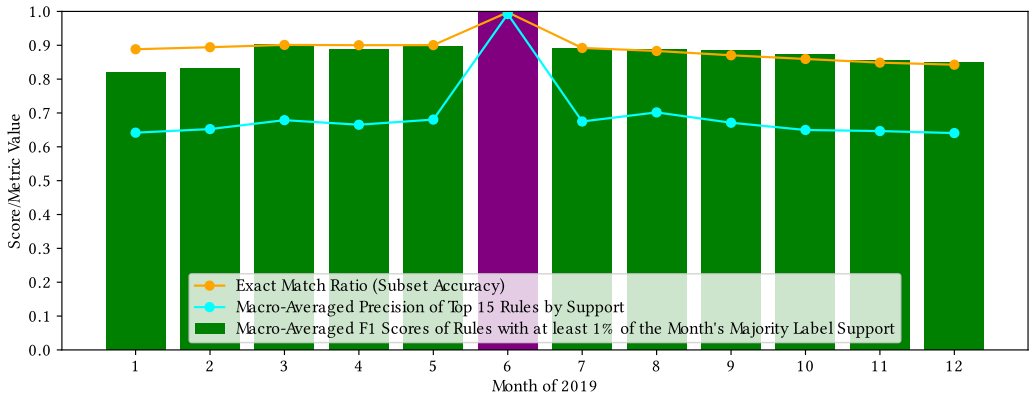

Fig. 3. Classification performance over 2019 of a DDoS2Vec embedding trained on 2019-06-01 — 2019-07-01.

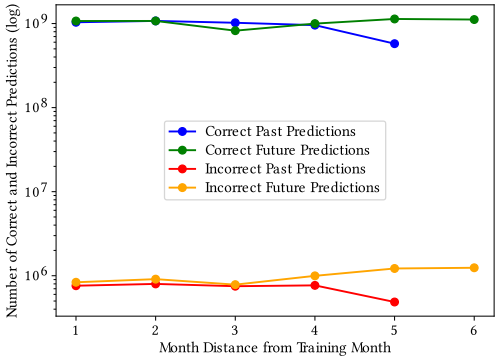

Fig. 4. Rule predictions over time in two directions.

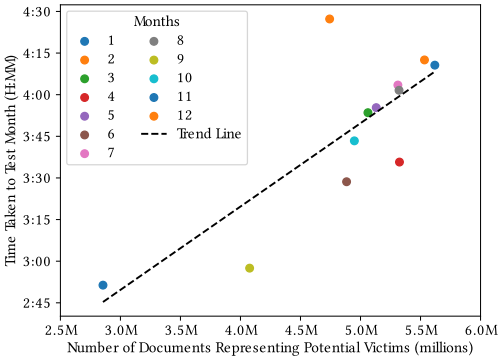

Fig. 5. Time taken for months based on corpus size.

remains the same as described in §5.1 as well. It is unlikely that all attack characteristics would be present in a single month for learning purposes, but we expect to find some characteristics shared among months.

Classification performance drift. Fig. 3 shows an overview of the classification performance over 2019. Three metrics are represented: the exact match ratio, macro-averaged F1 scores of reasonably supported rules, and macro-averaged precision of top-15 rules. As expected, there is a slight decay in classification performance for all three metrics as we move away from the training month. The exact match ratio (orange curve) is consistently high (over 0.85) and decays steadily, but it does not account for class imbalance. The average precision of the top 15 rules (cyan curve) falls more acutely to around 0.65, but it also remains relatively steady. More important, we look towards the macro-averaged F1 score where we *only* consider rules with at least 1% of the majority rule's support within a month. That allows us to focus on the rules that dominate the traffic in a month, which is more helpful to understand than scarce characteristics — especially as zero trimming of extremely rare labels (rules) was done. We can transfer a DDoS2Vec model to other months for attack characterisation, but the classification performance depends on the characteristics learnt during training. A wider variety of attack characteristics in the training set is preferable for transferability.

Comparing past vs. future predictions. Still based on Fig. 3, we observe a small difference between the backward and forward results, meaning traffic characteristics are dominant over time. Fig. 4 shows the correct (true positive and true negative) and incorrect (false positive and false negative) rule predictions over time. The training month (June) was not included. Note that the fifth data point for the past (correct and incorrect) predictions dropped overall compared to the previous months because January was truncated (seen in Fig. 5 or §3.2). We see extremely few incorrect classifications compared to correct ones, and both correct and incorrect predictions for their direction have a similar trend, indicating the difficulty of predicting obscure characteristics (a strength of DDoS2Vec over other approaches). A key takeaway is that DDoS2Vec is not distinctly biased toward the past or future, indicating a certain transferability level.

Time taken for evaluation. As we evaluated DDoS2Vec on a year of 7.2 billion flow records (§3.2), there is a concern that the time taken to obtain characteristics of potential attacks may be prohibitive. Fig. 5 shows the time taken by DDoS2Vec to evaluate all months of traffic in our full 2019 IXP dataset. The time is relatively consistent based on how many potential attacks (towards victims) DDoS2Vec is characterising, roughly taking a few hours for an entire month — considering that it is without any balancing, trimming, etc. The time performance can be further improved by discarding obscure words in the corpus vocabularies, which we showed in §5 via DDoS2Vec hyperparameter tuning for one month. In essence, the time taken depends more heavily on the number of potential attacks (corpus size) and the number of unique flow-level behaviour descriptions (corresponding vocabulary size) rather than the strict number of flow records alone due to how DDoS2Vec uses LSA (§4.2) and how LSA mathematically works (§A.2, §A.3).

7 CONCLUSIONS

Throughout this work, we have shown that NLP techniques can aid in characterising DDoS attacks within real-world Internet traffic. Initially, we demonstrated that synthetic datasets were unsuitable for this task and instead based all our work on flow samples collected from an IXP. Our approach — DDoS2Vec — is a domain-specific application of LSA that outperforms other NLP baselines and a typical attack port counting approach for learning characteristics of DDoS attacks in a multi-label classification scenario. Even if one month does not typically contain all variations of DDoS attacks, we have shown that a trained DDoS2Vec model can still be successfully applied to other months of IXP traffic. Additionally, we investigated the link between natural language and computer network communication, particularly word frequencies and occurrences when flow records are represented inside textual documents as multiple words.

Future work. This paper is a first step toward considering the application of NLP techniques on DDoS attack characterisation, which we have only shown a glimpse of in this paper yet still achieved notable results. We have avoided using more computationally expensive deep learning architectures, e.g. transformer-based models [55], because no pre-trained models are available for us in this highly domain-specific task. We have not considered online training experiments for DDoS2Vec, which are possible. Alternative methods for formulating the flow corpus generation should be investigated (§4.1) that allow for different forms of analysis, e.g. representing Autonomous Systems or IP prefixes in document tags instead of only IP addresses.

REFERENCES

[1] Ahamed Aljuhani. 2021. Machine Learning Approaches for Combating Distributed Denial of Service Attacks in Modern Networking Environments. *IEEE Access*, 9, 42236–42264. DOI: 10.1109/ACCESS.2021.3062909.

[2] Tom Anderson, Timothy Roscoe and David Wetherall. 2004. Preventing Internet Denial-of-Service with Capabilities. *SIGCOMM Comput. Commun. Rev.*, 34, 1, (Jan. 2004), 39–44. DOI: 10.1145/972374.972382.

[3] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro and Konrad Rieck. 2020. Dos and Don'ts of Machine Learning in Computer Security. In *USENIX Security Symposium*. https://arxiv.org/abs/2010.09470.

[4] Anat Bremler-Barr, Yotam Harchol, David Hay and Yaron Koral. 2014. Deep Packet Inspection as a Service. In *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies* (CoNEXT '14). Association for Computing Machinery, Sydney, Australia, 271–282. ISBN: 9781450332798. DOI: 10.1145/2674005.2674984.

[5] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (KDD '16). ACM, San Francisco, California, USA, 785–794. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939785.

[6] Benoît Claise. 2004. Cisco Systems NetFlow Services Export Version 9. RFC 3954. (Oct. 2004). DOI: 10.17487/RFC3954.

[7] Dvir Cohen, Yisroel Mirsky, Manuel Kamp, Tobias Martin, Yuval Elovici, Rami Puzis and Asaf Shabtai. 2020. DANTE: A Framework for Mining and Monitoring Darknet Traffic. In *Computer Security - ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14-18, 2020, Proceedings, Part I*. Springer-Verlag, Guildford, United Kingdom, 88–109. ISBN: 978-3-030-58950-9. DOI: 10.1007/978-3-030-58951-6_5.

[8] Scott E. Coull, Fabian Monrose and Michael Bailey. 2011. On Measuring the Similarity of Network Hosts: Pitfalls, New Metrics, and Empirical Analyses. In *Proceedings of the 2011 Network and Distributed System Security Symposium, NDSS*.

[9] [n. d.] DARPA Intrusion Detection Evaluation (1998, 1999, 2000). (). Retrieved 15th May 2023 from https://archive.ll.mit.edu/ideval/index.html.

[10] Christoph Dietzel, Matthias Wichtlhuber, Georgios Smaragdakis and Anja Feldmann. 2018. Stellar: Network Attack Mitigation Using Advanced Blackholing. In *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies* (CoNEXT '18). Association for Computing Machinery, Heraklion, Greece, 152–164. ISBN: 9781450360807. DOI: 10.1145/3281411.3281413.

[11] S. T. Dumais, G. W. Furnas, T. K. Landauer, S. Deerwester and R. Harshman. 1988. Using Latent Semantic Analysis to Improve Access to Textual Information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '88). Association for Computing Machinery, Washington, D.C., USA, 281–285. ISBN: 0201142376. DOI: 10.1145/57167.57214.

[12] Luca Gioacchini, Luca Vassio, Marco Mellia, Idilio Drago, Zied Ben Houidi and Dario Rossi. 2021. DarkVec: Automatic Analysis of Darknet Traffic with Word Embeddings. In *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies* (CoNEXT '21). Association for Computing Machinery, Virtual Event, Germany, 76–89. ISBN: 9781450390989. DOI: 10.1145/3485983.3494863.

[13] Eric L. Goodman, Chase Zimmerman and Corey Hudson. 2020. Packet2Vec: Utilizing Word2Vec for Feature Extraction in Packet Data. (2020). https://arxiv.org/abs/2004.14477.

[14] Maarten Grootendorst. 2022. BERTopic: Neural topic modeling with a class-based TF-IDF procedure. *arXiv preprint arXiv:2203.05794*.

[15] Tiago Heinrich, Rafael R. Obelheiro and Carlos A. Maziero. 2021. New Kids on the DRDoS Block: Characterizing Multiprotocol and Carpet Bombing Attacks. In *Passive and Active Measurement: 22nd International Conference, PAM 2021, Virtual Event, March 29 - April 1, 2021, Proceedings*. Springer-Verlag, Cottbus, Germany, 269–283. ISBN: 978-3-030-72581-5. DOI: 10.1007/978-3-030-72582-2_16.

[16] Rick Hofstede, Pavel Čeleda, Brian Trammell, Idilio Drago, Ramin Sadre, Anna Sperotto and Aiko Pras. 2014. Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX. *IEEE Communications Surveys & Tutorials*, 16, 4, 2037–2064. DOI: 10.1109/COMST.2014.2321898.

[17] Joel Hypolite, John Sonchack, Shlomo Hershkop, Nathan Dautenhahn, André DeHon and Jonathan M. Smith. 2020. DeepMatch: Practical Deep Packet Inspection in the Data Plane Using Network Processors. In *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies* (CoNEXT '20). Association for Computing Machinery, Barcelona, Spain, 336–350. ISBN: 9781450379489. DOI: 10.1145/3386367.3431290.

[18] Arthur S. Jacobs, Roman Beltiukov, Walter Willinger, Ronaldo A. Ferreira, Arpit Gupta and Lisandro Z. Granville. 2022. AI/ML for Network Security: The Emperor Has No Clothes. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (CCS '22). Association for Computing Machinery, Los Angeles, CA, USA, 1537–1551. ISBN: 9781450394505. DOI: 10.1145/3548606.3560609.

[19] 1999. KDD Cup 1999 Data. (Oct. 1999). Retrieved 15th May 2023 from http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[20] Daniel Kopp, Christoph Dietzel and Oliver Hohlfeld. 2021. DDoS Never Dies? An IXP Perspective on DDoS Amplification Attacks. (2021). DOI: 10.48550/ARXIV.2103.04443.

[21] Marc Kührer, Thomas Hupperich, Christian Rossow and Thorsten Holz. 2014. Exit from Hell? Reducing the Impact of Amplification DDoS Attacks. In *Proceedings of the 23rd USENIX Conference on Security Symposium* (SEC'14). USENIX Association, San Diego, CA, 111–125. ISBN: 9781931971157.

[22] Anukool Lakhina, Mark Crovella and Christophe Diot. 2005. Mining Anomalies Using Traffic Feature Distributions. *SIGCOMM Comput. Commun. Rev.*, 35, 4, (Aug. 2005), 217–228. DOI: 10.1145/1090191.1080118.

[23] Jean-Louis Lassez, Ryan Rossi, Stephen Sheel and Srinivas Mukkamala. 2008. Signature Based Intrusion Detection using Latent Semantic Analysis. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 1068–1074. DOI: 10.1109/IJCNN.2008.4633931.

[24] Quoc V. Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. (2014). https://arxiv.org/abs/1405.4053.

[25] Daniel Lee and H. Seung. 1999. Learning the Parts of Objects by Non-Negative Matrix Factorization. *Nature*, 401, (Nov. 1999), 788–91. DOI: 10.1038/44565.

[26] Wenke Lee and Dong Xiang. 2001. Information-Theoretic Measures for Anomaly Detection. In *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001.* (May 2001), 130–143. DOI: 10.1109/SECPRI.2001.924294.

[27] Ping Li, Trevor J. Hastie and Kenneth W. Church. 2006. Very Sparse Random Projections. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (KDD '06). Association for Computing Machinery, Philadelphia, PA, USA, 287–296. ISBN: 1595933395. DOI: 10.1145/1150402.1150436.

[28] Qasim Lone, Alisa Frik, Matthew Luckie, Maciej Korczyński, Michel van Eeten and Carlos Gañán. 2022. Deployment of Source Address Validation by Network Operators: A Randomized Control Trial. In *IEEE Symposium on Security and Privacy (SP)*. (May 2022), 2361–2378. DOI: 10.1109/SP46214.2022.9833701.

[29] Matthew Luckie, Robert Beverly, Ryan Koga, Ken Keys, Joshua A. Kroll and k claffy k. 2019. Network Hygiene, Incentives, and Regulation: Deployment of Source Address Validation in the Internet. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (CCS '19). Association for Computing Machinery, London, United Kingdom, 465–480. ISBN: 9781450367479. DOI: 10.1145/3319535.3354232.

[30] Anthony McGregor, Mark Hall, Perry Lorier and James Brunskill. 2004. Flow Clustering Using Machine Learning Techniques. In *Passive and Active Network Measurement*. Chadi Barakat and Ian Pratt, editors. Springer Berlin Heidelberg, Berlin, Heidelberg, 205–214. ISBN: 978-3-540-24668-8.

[31] Tomas Mikolov, Kai Chen, Greg Corrado and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. (2013). https://arxiv.org/abs/1301.3781.

[32] Jelena Mirkovic and Peter Reiher. 2004. A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. *SIGCOMM Comput. Commun. Rev.*, 34, 2, (Apr. 2004), 39–53. DOI: 10.1145/997150.997156.

[33] Nour Moustafa and Jill Slay. 2015. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS)*. Dataset available at: https://research.unsw.edu.au/projects/unsw-nb15-dataset, 1–6. DOI: 10.1109/MilCIS.2015.7348942.

[34] Lucas Müller, Matthew Luckie, Bradley Huffaker, Kc Claffy and Marinho Barcellos. 2019. Challenges in inferring spoofed traffic at ixps. In *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies* (CoNEXT '19). Association for Computing Machinery, Orlando, Florida, 96–109. ISBN: 9781450369985. DOI: 10.1145/3359989.3365422.

[35] Marcin Nawrocki, Mattijs Jonker, Thomas C. Schmidt and Matthias Wählisch. 2021. The Far Side of DNS Amplification: Tracing the DDoS Attack Ecosystem from the Internet Core. In *Proceedings of the 21st ACM Internet Measurement Conference* (IMC '21). Association for Computing Machinery, Virtual Event, 419–434. ISBN: 9781450391290. DOI: 10.1145/3487552.3487835.

[36] George Nychis, Vyas Sekar, David G. Andersen, Hyong Kim and Hui Zhang. 2008. An Empirical Evaluation of Entropy-Based Traffic Anomaly Detection. In *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement* (IMC '08). Association for Computing Machinery, Vouliagmeni, Greece, 151–156. ISBN: 9781605583341. DOI: 10.1145/1452520.1452539.

[37] Sonia Panchen, Neil McKee and Peter Phaal. 2001. InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks. RFC 3176. (Sept. 2001). DOI: 10.17487/RFC3176.

[38] F. Pedregosa et al. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

[39] Steven Piantadosi. 2014. Zipf's word frequency law in natural language: a critical review and future directions. *Psychonomic bulletin & review*, 21, (Mar. 2014). DOI: 10.3758/s13423-014-0585-6.

[40] Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. English. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. http://is.muni.cz/publication/884893/en. ELRA, Valletta, Malta, (May 2010), 45–50.

[41] Markus Ring, Alexander Dallmann, Dieter Landes and Andreas Hotho. 2017. IP2Vec: Learning Similarities Between IP Addresses. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, 657–666. DOI: 10.1109/ICDMW.2017.93.

[42] Thomas Roelleke and Jun Wang. 2008. TF-IDF Uncovered: A Study of Theories and Probabilities. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '08). Association for Computing Machinery, Singapore, Singapore, 435–442. ISBN: 9781605581644. DOI: 10.1145/1390334.1390409.

[43] Martin Roesch. 1999. Snort - Lightweight Intrusion Detection for Networks. In *Proceedings of the 13th USENIX Conference on System Administration* (LISA '99). USENIX Association, Seattle, Washington, 229–238.

[44] Christian Rossow. 2014. Amplification Hell: Revisiting Network Protocols for DDoS Abuse. In *Proceedings of the 2014 Network and Distributed System Security Symposium, NDSS*. (Jan. 2014). ISBN: 1-891562-35-5. DOI: 10.14722/ndss.2014.23233.

[45] Mohanad Sarhan, Siamak Layeghy, Nour Moustafa and Marius Portmann. 2021. NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems. In *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Springer International Publishing, 117–135. DOI: 10.1007/978-3-030-72802-1_9.

[46] Konstantinos Sechidis, Grigorios Tsoumakas and Ioannis Vlahavas. 2011. On the Stratification of Multi-Label Data. In *Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part III* (ECML PKDD'11). Springer-Verlag, Athens, Greece, 145–158. ISBN: 9783642238079.

[47] Gao Shang, Peng Zhe, Xiao Bin, Hu Aiqun and Ren Kui. 2017. FloodDefender: Protecting data and control plane resources under SDN-aimed DoS attacks. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 1–9. DOI: 10.1109/INFOCOM.2017.8057009.

[48] Iman Sharafaldin, Arash Habibi Lashkari, Saqib Hakak and Ali A. Ghorbani. 2019. Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy. In *2019 International Carnahan Conference on Security Technology (ICCST)*. Dataset available at: https://www.unb.ca/cic/datasets/ddos-2019.html, 1–8. DOI: 10.1109/CCST.2019.8888419.

[49] Justine Sherry, Chang Lan, Raluca Ada Popa and Sylvia Ratnasamy. 2015. BlindBox: Deep Packet Inspection over Encrypted Traffic. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (SIGCOMM '15). Association for Computing Machinery, London, United Kingdom, 213–226. ISBN: 9781450335423. DOI: 10.1145/2785956.2787502.

[50] Keiichi Shima. 2021. Catching Unusual Traffic Behavior using TF–IDF-based Port Access Statistics Analysis. In *2021 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI)*, 1–5. DOI: 10.1109/CCCI52664.2021.9583212.

[51] 2023. Snort Rules and IDS Software Download. (2023). Retrieved 30th Sept. 2023 from https://www.snort.org/downloads/#rule-downloads.

[52] 2023. Suricata. (2023). Retrieved 30th Sept. 2023 from https://suricata.io.

[53] Joe Touch, Eliot Lear, Kumiko Ono, Wes Eddy, Brian Trammell, Jana Iyengar, Michael Tuexen, Eddie Kohler and Yoshifumi Nishida. 2022. Service name and Transport Protocol Port Number Registry. (Nov. 2022). Retrieved 4th Nov. 2022 from https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml.

[54] Daniël van der Steeg, Rick Hofstede, Anna Sperotto and Aiko Pras. 2015. Real-time DDoS attack detection for Cisco IOS using NetFlow. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 972–977. DOI: 10.1109/INM.2015.7140420.

[55] Julian Von der Mosel, Alexander Trautsch and Steffen Herbold. 2022. On the validity of pre-trained transformers for natural language processing in the software engineering domain. *IEEE Transactions on Software Engineering*, 1–1. DOI: 10.1109/TSE.2022.3178469.

[56] Matthias Wichtlhuber, Eric Strehle, Daniel Kopp, Lars Prepens, Stefan Stegmueller, Alina Rubina, Christoph Dietzel and Oliver Hohlfeld. 2022. IXP Scrubber: Learning from Blackholing Traffic for ML-Driven DDoS Detection at Scale. In *Proceedings of the*

ACM SIGCOMM 2022 Conference (SIGCOMM '22). Association for Computing Machinery, Amsterdam, Netherlands, 707–722. ISBN: 9781450394208. DOI: 10.1145/3544216.3544268.
[57] Junrui Wu, Wenyong Wang, Lisheng Huang and Fengjun Zhang. 2022. Intrusion Detection Technique Based on Flow Aggregation and Latent Semantic Analysis. *Applied Soft Computing*, 127, 109375. DOI: https://doi.org/10.1016/j.asoc.2022.109375.
[58] Kuai Xu, Zhi-Li Zhang and Supratik Bhattacharyya. 2005. Profiling Internet Backbone Traffic: Behavior Models and Applications. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (SIGCOMM '05). Association for Computing Machinery, Philadelphia, Pennsylvania, USA, (Aug. 2005), 169–180. ISBN: 1595930094. DOI: 10.1145/1080091.1080112.
[59] Omer Yoachimik. 2022. DDoS Attack Trends for 2022 Q1. (Apr. 2022). https://blog.cloudflare.com/ddos-attack-trends-for-2022-q1.
[60] Omer Yoachimik. 2022. DDoS Attack Trends for 2022 Q2. (July 2022). https://blog.cloudflare.com/ddos-attack-trends-for-2022-q2.
[61] Ted Tao Yuan and Zezhong Zhang. 2018. Merchandise Recommendation for Retail Events with Word Embedding Weighted Tf-Idf and Dynamic Query Expansion. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '18). Association for Computing Machinery, Ann Arbor, MI, USA, 1347–1348. ISBN: 9781450356572. DOI: 10.1145/3209978.3210202.

A APPENDIX

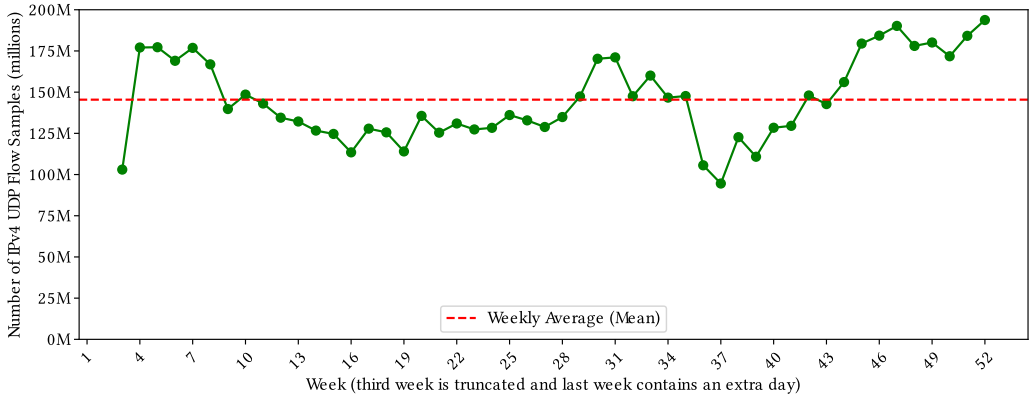

Fig. 6. IXP flow samples of 2019 per week.

A.1 IXP Scrubber Filtering Rule Example

```
"20d10ae9": {
        "protocol": 17,
        "port_src": 53,
        "port_dst": 2701,
        "packet_size": "(1400,1500]",
        "confidence": 1.0,
        "antecedent population": 410966
}
```

Listing 1. IXP Scrubber Filtering Rules Excerpt

A filtering rule explained. Listing 1 shows one of the 327 rules in its original JSON format. Each rule has a unique identifier, a protocol number, source and destination port matches, a packet size range, a confidence value, and an antecedent population. In this work, we are only interested in UDP (protocol number 17) flows. Ports can be matched on a specific port number, a set of ports to *not* match, or a match on any port. The packet size interval is 100 and goes from $(0, 100]$ up to $(1400, 1500]$, with the latter being the highest packet size interval (in addition to rules which match any packet size). The confidence value is a number between 0.0 and 1.0, indicating how likely a flow matching the rule is blackholed. Wichtlhuber et al. [56] only publicly make available the rules with confidence values at and above 0.9. The antecedent population is the number of flows used to mine the rule itself.

A.2 Term Frequency – Inverse Document Frequency (TF–IDF)

Given a corpus C...

$$TF(t, D) = \frac{\text{Number of times } t \text{ occurs in } D}{|D|} \tag{1}$$

$$SublinearTF(t, D) = \begin{cases} \log(TF(t, D)) + 1 & \text{if } TF(t, D) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$DF(t) = \text{Number of documents in } C \text{ containing } t \tag{3}$$

$$IDF(t) = \log\left(\frac{|C| + 1}{DF(t) + 1}\right) + 1 \tag{4}$$

$$TF\text{–}IDF(t, D) = SublinearTF(t, D) \times IDF(t) \tag{5}$$

Fig. 7. TF–IDF variant used in DDoS2Vec's process.

Local weighting. Equation (1) is the most basic form of TF, which counts the number of times a word occurs in a document. We then account for the length of the document by dividing the count by the document's length (to avoid longer document bias). We also use a sublinear (logarithmic) scaling function indicated in Equation (2), as a word's importance to a document is unlikely to scale linearly — especially when it indicates generic flow records and is repeated heavily during Algorithm 1.

Global weighting. IDF is used to weigh how important a word is across the corpus. The intuition is that a word that occurs in many documents is less important than a word that occurs in fewer documents. Equation (4) differs from the ordinary IDF equation ($\log\left(\frac{|C|}{DF(t)}\right)$) by adding one to the value in case the word is observed in every document; else, it would be ignored due to $IDF(t) = 0, TF\text{–}IDF(t, D) = SublinearTF(t, D) \times 0 = 0$. One is also added to the numerator and denominator to avoid a zero division if an unseen word is observed, i.e. an Out-Of-Vocabulary (OOV) issue post-training. The effect of this is if a document was observed containing one of every word in the corpus's vocabulary set.

From a corpus into a document-term matrix. For a document D and a corpus C, we calculate $TF\text{–}IDF(t, D)$ where t is every word in the ordered vocabulary V of C. The order of the vocabulary must be consistent as it is used to indicate the columns of the full document-term matrix. The result is a vector representation $\vec{v}$ of D with $|V|$ dimensions. We repeat this process for every document in C to create a document-term matrix M, where M has $|C|$ rows and $|V|$ columns; thus, the value of $M_{t,D}$ is the TF–IDF value of the word t in document D. Once M is created, we conduct L2 normalisation[6] for each $\vec{v}$ in M, e.g. $||\vec{v}||_2 = \sqrt{v_1^2 + v_2^2 + v_3^2 + \cdots + v_{|V|}^2}$. That has the effect of cancelling out the document length's effect, as we are primarily interested in which potential (DDoS or otherwise) characteristics documents represent, not their lengths.

TF–IDF does not consider word order. A major limitation of TF–IDF is that it does not consider word order. This is an immediate problem for us, as the order of words is vital in our context because they represent flow record information over time (a reminder that Algorithm 1 orders document contents *if* the flow records are sorted by ascending timestamps). Newer NLP techniques, e.g. word embedding techniques (Word2Vec [31] and Doc2Vec [24]), can consider word order or at least word proximity, but TF–IDF does not consider the order of words.

Introducing n-grams into vocabularies. We resolve that issue by generating n-grams (a sequence of n words) and using them as new words, contributing to a corpus's vocabulary. For

[6]Also referred to as Euclidean normalisation.

example, in Table 4, document "192.168.1.70" contains the words "53->" and "->58394" that indicate source port and destination port information, respectively, from what is actually one flow record. TF–IDF would not fundamentally retain the context, but we can generate the bi-gram "53->->58394" and add it as a new word into that document (the insertion position is inconsequential). This step is conducted before the TF–IDF process, creating the document-term matrix with the new n-gram words in each document.

Extreme dimensionality. The dimensionality of the document-term matrix is another significant concern because the document-term matrix is $|C| \times |V|$ in size. The value of $|C|$ is the number of destination IP addresses we observe in the flow samples, while the value of $|V|$ is the number of unique words in the corpus that were generated by Algorithm 1. For example, Algorithm 1 can generate 2^{17} words to represent source and destination ports alone. Generating n-grams to resolve the word-order drawback will also drastically increase the number of unique words in the corpus.

Sparsity and dimensionality reduction. The vectors will only contain non-zero values where the word is observed (vectors mostly contain zeros); hence, the document-term matrix will be sparse, and the zeroed elements can be compressed. While it is possible to use sparse matrices for downstream tasks, they are often more convoluted to manage than dense matrices. Additionally, the dimensionality of the document-term matrix will still be too large for many downstream tasks, *e.g.* classification and clustering. To resolve that, LSA uses truncated SVD to reduce said dimensionality.

A.3 Truncated Singular Value Decomposition (SVD)

$$\mathrm{M} = \underset{m\times n}{\begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,n} \\ x_{2,1} & x_{2,2} & \dots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \dots & x_{m,n} \end{bmatrix}} \approx U_r \times \Sigma_r \times V_r^T = \begin{bmatrix} \uparrow & & \uparrow & & \uparrow \\ u_1 & \dots & u_k & \dots & u_n \\ \downarrow & & \downarrow & & \downarrow \end{bmatrix} \times \begin{bmatrix} \sigma_1 & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & & \vdots \\ 0 & 0 & \sigma_{k,k} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \sigma_{m,n} \end{bmatrix} \times \begin{bmatrix} \longleftarrow & V_1^T & \longrightarrow \\ & \vdots & \\ \longleftarrow & V_k^T & \longrightarrow \\ & \vdots & \\ \longleftarrow & V_m^T & \longrightarrow \end{bmatrix}$$

Fig. 8. Singular Value Decomposition (full rank if $r = n$ and ignoring k, or truncated if $k < n$ and $r = k$).

SVD explained. With SVD, we can decompose a document-term matrix M (with real numbers) into three matrices: U, Σ, and V^T (see Figure 8)[7]. U indicates the left singular vectors, Σ indicates the descending order singular values, and V^T indicates the right singular vectors. Σ tells us of the most (relatively) important singular vectors via larger singular values, allowing for dimensionality reduction. Principal Component Analysis (PCA) is a closely associated technique to SVD, which instead uses a covariance matrix. Both techniques require M to be centred (mean subtracted) first, which removes the sparseness of M discussed previously in §A.2.

Truncating the SVD. The truncated SVD is often denoted as $U_k \times \Sigma_k \times V_k^T$, where k is the number of singular vectors and singular values to retain. We can truncate the SVD by removing ("truncating") the least important singular vectors and singular values until k of them remain; hence, truncated SVD gives a low-rank approximation of M. It can be computed on sparse matrices, which is vital for DDoS2Vec as the document-term matrix is sparse and extremely wide. This is the final step of DDoS2Vec altogether, as it returns a dense matrix where the row is a potential attack victim, and the columns approximate the victim's received traffic.

[7] V should not be confused here with the earlier denotation of V in §A.2.

A.4 Multi-Label Classification Metrics

Throughout this paper, we evaluate DDoS2Vec (among other approaches) using multi-label classification metrics. Here, we explain the metrics we use and how we compute them.

Main metrics. The main metrics we use are *precision*, *recall*, and *F1 scores* — all of which are classic metrics in ML. Precision is the ratio of true positives to the total number of predicted positives, recall is the ratio of true positives to the total number of actual positives, and the F1 score is the harmonic mean of precision and recall. We also report the *exact match ratio* [8], which is the ratio of the number of samples where the predicted labels exactly match the true labels to the total number of samples. Unlike the other metrics, the exact match ratio is specific to multi-label classification. However simple they are, the way to compute these metrics for a classification approach is more complicated.

Averaging. In this work, we handle a highly imbalanced dataset. Because of that, we report both *micro* and *macro* averages of the metrics. Macro-averaging computes the metric independently for each label and then averages the metrics, meaning all class labels (DDoS attack characteristics) have equal weight when gauging performance. Micro-averaging computes it globally by counting the total predicted true positives, false negatives, and false positives. Macro-averaging is highly sensitive to class imbalance, but micro-averaging is not; unfortunately, micro-averaging is also sensitive to the majority classes and can give a false sense of strong classification performance. We report both as we are interested in the approach that reports high values for both macro and micro averages — a good ability to learn more subtle characteristics and learn the overall traffic, respectively. We also summarise classification performance overall with an *intermediate* value, e.g. the intermediate F1 score is the macro-averaged and micro-averaged F1 scores added together and halved. An alternative would be to use sample weights (assigning a weight to each potential attack based on the associated rules), which require a more complex scheme and is unnecessary for our purposes.

A.5 Baseline Approaches

For the sake of comparison, we adopt three baseline approaches: (i) Doc2Vec, (ii) Word2Vec, and (iii) a simple amplification port counting method. Unlike the first two, the third is a simplistic approach we formulate to compare NLP with non-NLP approaches. We perform only simple experimentation and tuning of DDoS2Vec-specific hyperparameters since a complete investigation would be an effort on its own.

Doc2Vec. Doc2Vec [24] is an extension of Word2Vec [31] that can learn document vectors. Doc2Vec requires a corpus, and we use the same process adopted by DDoS2Vec (§4.1) for fairness. We extract the document vectors from a trained model via Gensim [40] with the following configuration: a vector size of 100, a context window size of 5 with the Distributed Memory learning model, and 25 epochs. No minimum or maximum word counts are used, and the negative sampling hyperparameters and others are kept at Gensim defaults.

Word2Vec. To provide a non-document baseline, following the approach of DarkVec [12], we compare with Word2Vec [31]. We transform the document corpus (§4.1) into a series of long sentences, where each word is the potential victim (destination IP address). The sentences are based on the entries in Table 2, except the average packet size intervals are their own sentences as well. The word vectors are retrieved from a trained model via Gensim with an equivalent hyperparameter to the Doc2Vec setup, except that the context window size is 10 and the skip-gram learning model is used with 15 epochs.

[8] Also known as the *subset accuracy*.

Non-NLP baseline. NLP techniques might be inadequate for DDoS characterisation compared to non-NLP techniques. To avoid limiting the comparison, we formulate a simple baseline non-NLP approach that leverages domain knowledge, denoted as *Counter*. We count the times each amplification port is used for each potential victim (destination IP address). For each behaviour's source port in Table 2, we gather the number of flows, byte count, and packet count associated with that port when it is the source of the traffic sent to the potential victim. That creates a row where each column represents a source port and the associated flow, byte, or packet count (three columns for each source port). We do the same for the generic behaviours, but their respective columns represent a range of ports. Counter produces rows with 48 columns each. A caveat is that this process entirely relies on domain knowledge, so it is unsuitable for unsupervised analysis due to its restrictiveness.

Received June 2023; revised October 2023; accepted October 2023