POLITECNICO DI TORINO

Dipartimento di Elettronica e delle Telecomunicazioni
Master of Science in ”ICT for Smart Societies”

Master Thesis

Automatic Detection of Coordinated Events in Darknet Traffic

Supervisors:
Dr. Luca Vassio
Prof. Idilio Drago
MSc. Francesca Soro

Candidate:
Luca Gioacchini
s257076

April 2021

Ai miei genitori

Contents

List of Figures

List of Tables

Acronyms

AMI Adjusted Mutual Information.

BMU Best Matching Unit.

DBSCAN Density-Based Spatial Clustering of Applications with Noise.

DDoS Distributed Denial of Service.

ECDF Empirical Cumulative Distribution Function.

GMA Greedy Modularity Algorithm.

GT Ground Truth.

HAC Hierarchical Agglomerative Clustering.

kNN k-Nearest Neighbors.

LOO Leave-One-Out.

ML Machine Learning.

PCA Principal Component Analysis.

rDNS Reverse DNS Lookup.

RQ Research Question.

Sh Silhouette.

SOM Self-Organizing Maps.

t.u. time unit.

tSNE t-distributed Stochastic Neighbor Embedding.

Abstract

Darknets, or network telescopes, are network monitoring tools composed by sets of IP addresses announced in routing protocols, but without hosting any services. They constantly listen to incoming traffic and record it. The received packets are thus unsolicited and represent a privileged source of information to network security and debugging activities. Indeed, the lack of any production traffic in darknet makes it easier to detect possible threats like internal scans, brute-force attempts against services, etc.

Darknets however still receive a lot of traffic from thousands of sources. Large botnets in particular are massively used to scan for vulnerable services online. Such large-scale events follow diverse patterns, with multiple hosts belonging to a single botnet eventually contacting addresses of the darknet in the search for vulnerable services. Detecting and evaluating such coordinated events in darknet traffic is an important step to fully exploit the darknet monitoring potential. Indeed it could reduce the amount of data to be evaluated by security analysts as well as provide a richer picture about ongoing attacks on the Internet.

Given the huge amount of source IP addresses constantly targeting darknets, a manual analysis on the received traffic is impractical. Moreover, there is a lack of comprehensive ground truth that could be used to learn patterns on darknet traffic. In this thesis I evaluate the use of different methodologies based on unsupervised data mining for automatically detecting coordination among groups of source IPs contacting darknets. In particular, I investigate two hypotheses that could characterize the coordination: i) as sources sending traffic are usually controlled by a single entity (e.g., a bot master), the level of traffic activity reaching a darknet from coordinated hosts should be similar. Therefore, I study whether traffic intensity could indicate coordination; ii) traffic from different but coordinated sources should reach the darknet with some temporal correlation. I hence study whether coordinated sources are observed in the darknet simultaneously.

I develop a complete machine learning pipeline to test these hypotheses. Considering the traffic intensity case, I design and evaluate a set of features that could represent traffic intensity and study several non-supervised algorithms to group IP addresses based on the engineered features. For studying the temporal relationship

among sources, I employ the word2vec algorithm, an approach used in text processing to find words that frequently occur nearby in sentences and documents.

To overcome the lack of general ground truth and provide a sound validation of results, I rely on domain knowledge to build a dataset of coordinated IP addresses belonging to well-known Internet scanners, such as security search engines. While these IP addresses are not malicious, they present a coordinated behavior that is clearly visible on the darknet.

My results suggest that the generated features allow to highlight both the traffic intensity and temporal coordination. Indeed, since the used features sets represent the source IPs in a N-dimensional space, when evaluating the neighborhood, or the points closest to ones of the ground truth classes, it results that that points belonging to the same class fall in the same neighborhood with an average accuracy of 98%. The achieved accuracy is maintained over time when temporal relationship features are used, but when considering the traffic intensity ones, the accuracy rapidly decrease below 30%. Furthermore, the unsupervised algorithms are able to group together IP addresses exhibiting similar behaviors within a day of darknet traffic, but the cluster membership is not maintained over time. I observe for example that clusters built over different days of traffic are significantly dissimilar (0.5 of adjusted mutual information), which seems to be explained by changes in behavior of the whole set of coordinated IP addresses.

The approach exploiting the temporal relationships slightly overcomes the intensity-based method, especially in the long run. Indeed, even if the similarity between clusters membership over different days decrease, the trend is smoother resulting in the same adjusted mutual information value of the traffic intensity case by taking one day more.

All in all, my results show that the approaches can reduce the amount of data points to be analyzed by security analysts, putting together IP addresses that share a common behavior. Yet, algorithms must be applied to short time ranges to maximize the chances that real coordination is identified.

Chapter 1

Introduction and Problem Definition

1.1 Overview

Darknets, or network telescopes are sets of passive IP addresses which do not host any services and just listen for incoming traffic. Because of this lack in hosted services, all the received traffic is unsolicited, thus anomalous by definition. Furthermore, darknets achieve a privileged point of view in traffic analysis thanks to the absence of production traffic. In this way they can be used as support tool for network analysts in detecting services misconfigurations, benign Internet scans from research projects, crawlers, etc., Distributed Denial of Service (DDoS) attacks with spoofed IP address, malicious probing, etc. Several examples can be found in literature like authors of [1]

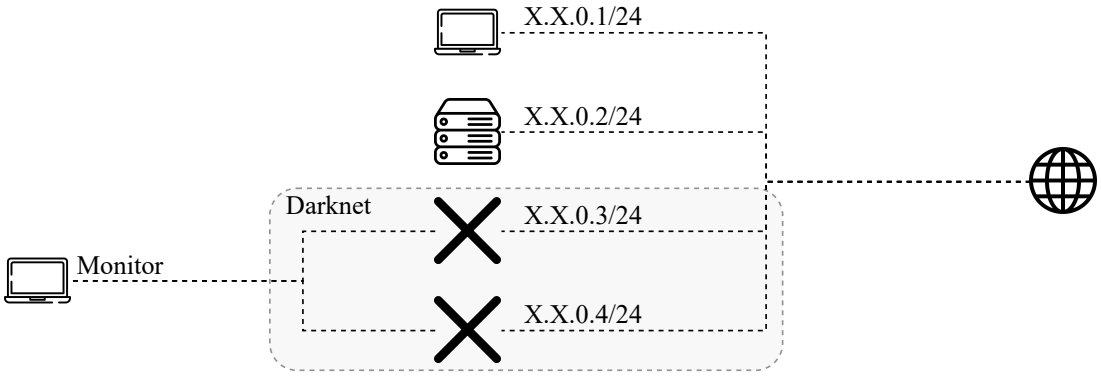


Figure 1.1: Darknet overview

applying comparative analysis on two darknet sensors characterizing the received TCP and UDP traffic or authors of [2] confirming the prevalence of benign and malicious scans in darknet traffic.

Darknets receive a lot of traffic from thousands of sources. Many of them consist of large botnets. They are made of a set of machines infected by malwares acting under the control of a master node. Because of this paradigm called Command and Control (C&C), it is highly likely that nodes belonging to a botnet exhibit a certain degree of coordination performing distributed tasks, like search for vulnerable services, or attacks, like the one run by the botnet Mirai [3], responsible of the 2016 Dyn cyberattack, a series of DDoS attack performed by more than 300000 infected devices spread in 164 countries [4].

The detection of large-scale coordinated events in darknet traffic is fundamental to reduce the amount of data evaluated by security analysts and to deepen the knowledge of the dynamics behind attacks on the Internet.

Given the huge amount of source IP addresses constantly targeting darknets, a manual analysis on the received traffic is impractical. Thus, according to the diffusion and improvement of data science and machine learning techniques, in recent years researchers have been focused on applying such techniques to darknet traffic analysis, like works reported in [5, 6].

1.2 Research Questions

In light of the described context, in this thesis I propose and investigate a methodology based on unsupervised clustering algorithms and data mining techniques applied to darknet traffic in order to identify groups of source IPs acting in a coordinated way. The main Research Question (RQ) driving the project can be formalized as:

RQ1. Is it possible to automatically detect coordination among source IPs through darknet traffic analysis? According to the importance of detecting coordination among source IP addresses, I exploit the darknets monitoring potential investigating if unsupervised data mining techniques are able to spot groups of source IP addresses acting in coordination within an active darknet of Politecnico di Torino (3 non-contiguous /24 IP addresses)

The methodology relies on a full machine learning pipeline involving unsupervised clustering and community detection algorithms (more specifically DBSCAN, Hierarchical Agglomerative Clustering, k-Means and Greedy Modularity Algorithm). Being unsupervised techniques I overcome the problem of the lack of comprehensive ground truth building a list of IPs belonging to known research projects performing benign scans whose coordination is known a-priori.

Regarding the coordination, I provide two definitions relying on two hypotheses. The first is based on traffic volumes, the second based on temporal correlation. They can be formalized in the following RQs derived from the main RQ1:

RQ1.2. Is it possible to exploit the amount and type of traffic generated toward darknets to detect coordination among source IPs? The first hypothesis is that in botnet scenarios, source IPs controlled by a single host like the bot master are expected to follow some pattern in the traffic volume (e.g., sending the same amount of traffic) or traffic type (e.g., common targeted destination ports).

For studying such coordination I design and evaluate a set of features representing traffic intensity, then I investigate the coordination of clusters found by the algorithms applied to the engineered features.

RQ1.3. Is it possible to exploit the temporal relationships among source IPs active in darknets to detect coordination? The second hypothesis is that source IP addresses acting coordinately are expected to exhibit a certain temporal correlation in their activeness within the darknet (e.g., groups of IPs active in the same temporal windows with a certain periodicity).

For studying nodes temporal relationships I adopt the word2vec algorithm used for frequency analysis in Natural Language Processing. It allows to generate real-valued embedding for each source IP whose values are related to the IP activeness within the darknet. By applying the unsupervised algorithms on the embeddings I evaluate if found clusters exhibit any coordination.

RQ1.4. If the developed model is able to identify coordination, is it maintained over time? After the model development and outcomes analysis, I investigate the consistency of results repeating the experiments over time and studying the evolution

of the IPs membership to found clusters. Such analysis allows not only to provide a richer picture of the methodology, but also to investigate the dynamics behind detected coordinated groups.

1.3 Thesis Organization

The thesis is organized as follows:
Chapter 2 provides an overview on the state of the art.
Chapter 3 describes in detail the adopted methodology from the ground truth construction and features engineering stages, up to the tested algorithms.
Chapter 4 provides a characterization of the collected darknet traffic used as dataset and an overview of the built ground truth.
In Chapter 5 I answer to RQ1.2 reporting and discussing the results of the model development and application when the traffic volume information is exploited. Furthermore, in the last section I answer to RQ1.4 in the considered scenario.
Chapter 6 I answer to RQ1.3 from the reported outcomes of the methodology applied to the temporal relationships case. Again, I answer to RQ1.4 in the considered scenario.
Chapter 7 provides the final conclusions answering to the main question RQ1 and the statement of the future works.

Chapter 2

State of the Art

Many examples of darknets involvement in traffic analysis can be found in literature.

First of all, authors of [7] propose darknets as tools for profiling attacks strategies. Authors of [8] exploit darknets traffic for observing worm attacks. They characterize the traffic as generated mainly by few source IPs whose packets interarrival time exhibit short-range dependence. Furthermore they note that during a DDoS attack the white noise traffic signature changed to Brownian motion.

In [9], authors adopt darknets for cybersecurity analysis. They focus on the statistical properties of the darknets data timeseries collected during attacks and highlight the analysis limitation due to small darknets.

Authors of [10] use a darknet prototype to analyze the backscattering caused by DDoS attacks with spoofed IP addresses. Through the network telescope they are able to detect 12000 attacks against more than 5000 targets over three weeks of traffic observation.

A similar example is provided in work [11] where authors investigate the capability of five darknets of detecting malicious event. In their work they success in detecting both an existing threat and two emerging ones.

To provide another example of the full darknet monitoring potential, authors of [12] show how from darknet traffic it is possible to detect singularities not only during Internet-related-events like attacks or scans, but also political and geophysical events like earthquakes.

Complex networks analysis for modelling darknets collected traffic is widely used in literature. For examples, authors of [13] adopt a bipartite graph approach for rep-

resenting darknet traffic and apply community detection technique like the Louvain algorithm for obtaining groups of autonomous systems characterized by similar behavior in terms of destination ports. A similar approach is adopted in [14, 15], where authors model the darknet traffic as a graph in order to successfully detect Internet ports co-targeted by scanners.

Another example of graph mining techniques applied to darknets is proposed in [16], where authors use an undirected graph for tracking attacks among sets of honeypots. The attack propagation is modelled as Markov Chain. Through the proposed methodology, they analyze 167 million attacks and are able to spot the main honeypots responsible of propagating the most of the attacks.

However, according to the huge computational requirements of graph mining techniques and to the limited scalability of the models, I focus the research on clustering algorithms. A first attempt of cluster analysis applied to darknet traffic is proposed by authors of [17]. They extract 17 traffic features from 60 days of darknet traffic collection. Then they model the points density in space as a Cauchy distribution and apply evolving Cauchy possibilistic clustering (eCauchy) algorithm. In this way they are able to detect with 98% accuracy TCP DDoS backscatter and wth 72.8% accuracy UDP non-DDoS backscattering.

Similar performance is obtained in [18] where authors obtain an error rate $< 1\%$ when investigating the possibility of forecasting the trend of ongoing DDoS attack. More specifically they exploit known attacks information like traffic intensity, rate, number of infected machines executing the attack.

Another clustering example can be found in [19], where authors adopted an approach based on i) anomaly detection techniques applied to darknet scanning targets and on ii) Detrended Fluctuation Analysis (DFA) applied to timeseries in order to detect temporal correlation among machines performing scans through the same technique. In this way they claim to successfully detect 4215 scans without zero negatives requiring 1 second of training for the algorithm stabilization. The same authors adopt the timeseries approach for analyzing 330GB of darknet traffic in work [20]. More precisely they try to infer scanning campaigns by defining the problem as timeseries forecasting through trigonometric interpolation when missing values occur.

Finally, one of the most interesting work proposed in literature is [21]. Authors proposed a framework for mining darknet traffic through the Natural Language Pro-

cessing algorithm Word2Vec. In this way they exploit the temporal sequence of darknet ports contacted by source IPs and generate embeddings accordingly. Thus each IP is identified by the average embeddings of the contacted ports. With the proposed methodology, they are able to discover 1177 new threats together with tracing the existing ones over time.

Chapter 3

Methodology

Recalling that the aim of the thesis is to detect coordination among groups of source IP addresses reaching the darknet, my work relies on two definitions of coordination:

1. **Intensity-based coordination**. By considering a general botnet scenario, the first definition of coordination is based on the assumption that groups of coordinated hosts acting under the control of a bot master should generate similar amount of traffic intensity;

2. **Time-based coordination**. Darknet traffic generated by coordinated nodes should follow temporal patterns exhibiting a certain level of temporal correlation.

Under these hypotheses I develop a machine learning pipeline to investigate if the two types of coordination can be spotted in daily darknet traffic. Figure 3.1 provides an overview of the methodology stages: the first stages consist of a preliminary collection and characterization of the received traffic and the construction of the ground truth. This latter step is fundamental to overcome the lack of a set of source IP addresses whose coordination is known a priori. Afterwards a features engineering stage allows to generate and extract nodes attributes identifying the IPs uniquely. All the generated features are evaluated and the most informative ones are selected through the features selection stage. The selected ones will be used in the proposed models. Since the investigated approach relies on nodes spatial projection, I define a distance metric able to express the degree of coordination among the IPs and evaluate the consistency of

the projection through a k-Nearest Neighbors (kNN) classifier. Regarding the unsupervised model, I implement several clustering algorithms and evaluate the temporal evolution of the found clusters membership.

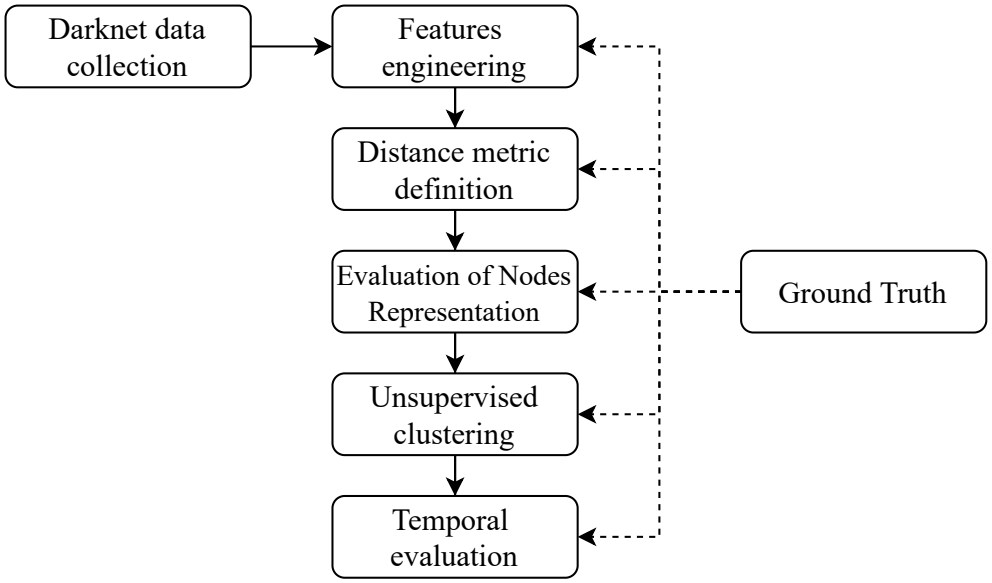


Figure 3.1: Adopted methodology flowchart

3.1 Ground Truth Construction

To evaluate the proposed methodology, obtaining groups of source IPs whose coordination is known a priori is fundamental. Indeed, after the development of each stage, by applying it to the aforementioned group, it is possible to verify if the coordination is detected at least for them. One of the simplest example of such groups is represented by research projects. Indeed several organizations and universities exploit sets of IP addresses to periodically scan the Internet for research purposes like searching for vulnerable services. Most of these projects make the list of used IP addresses public, thus, by collecting this information, a preliminary ground truth can be constructed. I further enrich the Ground Truth (GT) by means of the Reverse DNS Lookup (rDNS).

3.1.1 Reverse DNS Lookup (rDNS)

Reverse DNS Lookup is a querying technique of the Domain Name Service for retrieving a domain name from the associated IP address. Linux provides rDNS queries through the `dig` command. I report a sample output in Listing 3.1.

```
$ dig -x 8.8.8.8

; <<>> DiG 9.16.1-Ubuntu <<>> -x 8.8.8.8
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 33202
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;8.8.8.8.in-addr.arpa.    IN  PTR

;; ANSWER SECTION:
8.8.8.8.in-addr.arpa. 84974 IN  PTR dns.google.

;; Query time: 36 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Thu Feb 18 15:56:09 CET 2021
;; MSG SIZE  rcvd: 73
```

Listing 3.1: Sample rDNS query through the `dig` command

By providing the source IP address `8.8.8.8`, the `dig` command returns a set of information including the domain name `dns.google.`. To build the ground truth through rDNS I select a set of darknet traffic traces, then sort the source IPs according to the total number of generated flows. I then run the `dig` command with the top-N IPs to get their domain name. After that, by manually analyzing the obtained information I build a list of IPs associated to the respective domain name.

3.2 Features Engineering

Features engineering is the process of generating, extracting and selecting data attributes uniquely describing each of the data samples. Thus in Section 3.2.1 I describe the preprocessing stage, in Section 3.2.2 I provide an overview of the data mining techniques adopted to generate features starting from the preprocessing data and in Section 3.2.3 I describe the algorithm used for selecting the best set of generated features.

3.2.1 Preprocessing

Data preprocessing is the first phase of features engineering, since it allows to filter, aggregate and rearrange raw data in a way useful to highlight data properties involved in the features generation process.

The hourly traces collected by the darknet in the .pcap format are processed by the libpcap[1] tool able to generate a human-readable log file. Every instance is characterized by the raw features reported in Table 3.1.

Field	Type	Description
ts	float	Timestamp of the flow arrival
ethtype	str	Specifies the data link layer protocol encapsulated in the payload
src_ip	str	Source IP address reaching the darknet
src_port	str	Internet port from which the flow is generated
dst_ip	str	Destination IP address of the darknet reached by the flow
dst_port	str	Internet port at which the flow is reached
proto	int	Specifies the transport layer protocol through which the flow is sent
pck_len	int	Bytes length of the received flow
tcp_flags	int	Specifies the TCP flags set in the TCP header, if any

Table 3.1: Raw data structures after the libpcap processing of the traces

[1] http://www.pluto.it/sites/default/files/ildp/lfs/blfs/6.0/general/libpcap.html

Field	Identifier	Description	Decimal
Protocol	TCP	Transmission Control Protocol	6
	UDP	User Datagram Protocol	17
	ICMP	Internet Control Message Protocol	1
	GRE	Generic Routing Encapsulation protocol	47
Flag	SYN	SYN TCP flag	2
	SYN-ACK	SYN and ACK TCP flags	18
	SYN-ACK-RST	SYN, ACK and RST TCP flags	22
	SYN-ACK-FIN	SYN, ACK and FIN TCP flags	19
	OTH	All the other TCP flags	*

Table 3.2: Protocols and TCP flags classification

The raw features must be rearranged not only to reduce the amount of data, but also to make the features generation process manageable. The first step consists of making the transmission layer protocols (*proto* field in Table 3.1) and the TCP flags (*tcp_flags* field) readable, converting the decimal representation of bytes in the packet headers into string as shown in Table 3.2. After that, I grouped the destination port (*dst_port* field) and used protocol (*proto* field) by classes of service. For example, the Telnet application protocol accepts connections from the internet port number 23 through the TCP protocol, thus the 23/tcp pair is classified as Telnet. In this way, instead of considering 131072 categorical variables, one for each internet port when using TCP (65536) and other 65536 for UDP protocol, I limit them choosing only the most commonly used. Furthermore, for the sake of simplicity, not all the classified services are referred to a single port/protocol pair like the Telnet case (e.g. The MongoDB server listening to port 27017/tcp and the Microsoft SQL server listening to ports 1433/tcp, 1433/udp, 1434/tcp and 1434/udp are classified as 'Database'). The full overview of the service classification is provided in Table 3.3.

Service	Internet Port/Protocol
Telnet	22/tcp, 992/tcp
SSH	23/tcp
Kerberos	88/tcp, 88/udp, 543/tcp, 544/tcp, 749/tcp, 7004/tcp, 750/udp, 750/tcp, 751/tcp, 752/udp, 754/tcp, 464/udp, 464/tcp
HTTP	80/tcp, 443/tcp, 8080/tcp
Proxy	1080/tcp, 6446/tcp, 2121/tcp, 8081/tcp, 57000/tcp
Mail	25/tcp, 143/tcp, 174/tcp, 209/tcp, 465/tcp, 587/tcp, 110/tcp, 995/tcp, 993/tcp
Database	210/tcp, 5432/tcp, 775/tcp, 1433/tcp, 1433/udp, 1434/tcp, 1434/udp, 3306/tcp, 27017/tcp, 27018/tcp, 27019/tcp, 3050/tcp, 3351/tcp, 1583/tcp
DNS	853/tcp, 853/udp , 5353/udp , 53/tcp, 53/udp
Netbios	137/tcp, 137/udp, 138/tcp, 138/udp, 139/tcp, 139/udp
Netbios-SMB	445/tcp
P2P	119/tcp, 375/tcp, 425/tcp, 1214/tcp, 412/tcp, 1412/tcp, 2412/tcp, 4662/tcp, 12155/udp, 6771/udp, 6881/udp, 6882/udp, 6883/udp, 6884/udp, 6885/udp, 6886/udp, 6887/udp, 6881/tcp, 6882/tcp, 6883/tcp, 6884/tcp, 6885/tcp, 6886/tcp, 6887/tcp, 6969/tcp, 7000/tcp, 9000/tcp, 9091/tcp, 6346/tcp, 6346/udp, 6347/tcp, 6347/udp
FTP	20/tcp, 21/tcp, 69/udp, 989/tcp, 990/tcp, 2431/udp, 2433/udp, 2811/tcp, 8021/tcp
Unknown System	All ports in the [0, 1023] range not classified as before
Unknown User	All ports in the [1024, 49151] range not classified as before
Unknown Ephemeral	All ports in the [49152, 65535] range not classified as before

Table 3.3: Class of service classification

3.2.2 Features Generation

In the features generation stage all the preprocessed data are used to obtain information expressing certain nodes properties. According to the provided problem definition, it is necessary to rearrange and organize the traffic volume and temporal relationships information in a way that uniquely describe the behavior of each source IP address in the darknet. Since the network telescope is a passive tool not responding to the received flows, the nodes characterization can be based on three main aspects:

1. Packet frame information such as the used protocol, the reached darknet port, the set of TCP flags, etc.

2. Amount of traffic generated by each source IP per type of flow;

3. Temporal relationships among nodes like the temporal sequence of source IPs as they reach the darknet.

The generated features can be grouped in two macro-categories: *traffic intensity* ones covering aspects 1 and 2; and *temporal* ones covering aspect 3. To generate features of the first category I exploit the domain knowledge, setting an observation window of 24 hours and considering for each IP address the sum of daily flows sent with each protocol and TCP flag of Table 3.2 set and classes of service of Table 3.3 set. Moreover, two general information features expressing the total amount of daily flow per IP and the number of hours of activity within the observation window are obtained. Furthermore two features extraction techniques are applied to the classes of service features to reduce the dimensionality: Principal Component Analysis (PCA) and Self-Organizing Maps (SOM).

Regarding temporal features, they are assumed to express temporal relationships among source IP addresses highlighting coordination (e.g., nodes always active together within an observation window should have similar features values). To extract such features I use the word2vec algorithm, a Natural Language Processing approach to find words belonging to the same context in a text document.

Principal Component Analysis (PCA) PCA [22] is an exploratory data analysis technique which allows to project data points to a low-dimensional space. By considering a dataset $\mathbf{X} \in \mathbb{R}^{N \times F}$ where N is the number of samples and F is the number of

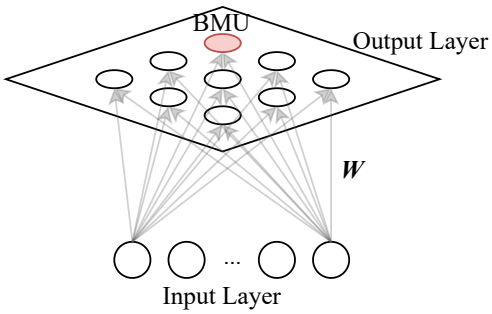


Figure 3.2: SOM layout example

features, then the decomposition of $\mathbf{X}$ in $L \leq F$ principal components is given by the matrix $\mathbf{T} \in \mathbb{R}^{N \times L}$ defined as

$$\mathbf{T} = \mathbf{X}\mathbf{W}_L \tag{3.1}$$

where $\mathbf{W}_L \in \mathbb{R}^{N \times L}$ is a weights matrix whose columns are the first L eigenvectors of $\mathbf{X}^T\mathbf{X}$. In this way PCA can be used as a dimensionality reduction tool generating a compressed ($L \leq F$) representation $\mathbf{T}$ of the original dataset $\mathbf{X}$. Furthermore, PCA is often used in factor analysis thanks to the *loadings*. They are defined as the product between the columns of $\mathbf{W}_L$ and the square root of the corresponding eigenvalues. Thus, the loadings of a principal component provide a quantitative measure of how much each feature is related with the considered component. In the considered case I extract the first n_{PC} principal components explaining $\geq$ 80% of the original data cumulative variance.

Self-Organizing Maps (SOM) A SOM [23] is an artificial neural network trained through unsupervised learning techniques for dimensionality reduction. As shown in Figure 3.2, it is characterized by a minimal design: one input layer is densely connected to an output one where the M output neurons, or *units*, are arranged in a square grid. Instead of relying on error-correction learning (e.g., applying the backpropagation algorithm), SOMs are based on *competitive learning*. This means that each input sample is mapped onto only one unit, or Best Matching Unit (BMU), among all the output ones. The BMU choice criterion is the minimization of the Euclidean distance between the input sample and the weight vectors of the SOM. Once the BMU is elected, the weights of the BMU and its neighborhood (i.e., the neurons close to the

BMU) are updated through the formula:

$$\mathbf{W}_v(s+1) = \mathbf{W}_v(s) + \theta(u, v, s) \cdot \alpha(s) \cdot (\mathbf{D} - \mathbf{W}_u(s)) \tag{3.2}$$

where $\mathbf{D}$ is the input sample, s is the iteration step index, $\mathbf{W}_v$ and $\mathbf{W}_u$ are the weight vectors of units u and v, θ is the neighborhood function and α is the decreasing learning coefficient. After the training, the (x, y) coordinates of the BMUs elected for all the samples can be treated as two additional features bringing topological information from the neurons grid. In the considered problem I set $M = 10 \cdot N$, where N is the number of samples.

Word2vec Embeddings Word2vec [24] is an algorithm relying on an artificial neural network to produce word embedding. It requires a collection of documents, or *corpus* and returns a set of vectors representing the semantic distribution of words, or *tokens*, in the corpus. In this way a single word is embedded in a $\mathbb{R}^{1\times E}$ array, where E is the embedding size.

One of the core elements of word2vec is the *context*. It is composed of a group of subsequent words appearing in a text document. The number of words to consider is specified by the *context window* C. The rationale behind the algorithm is to train the neural network by providing as targets the C words before and after the target one. After the training, the embedding is represented by the weight matrix $\mathbf{W} \in \mathbb{R}^{V\times E}$ of the output layer, where V is the number of tokens (or the vocabulary size).

In the considered case, a token is a distinct source IP and the vocabulary is the collection of the source IPs seen during 30 days of collected data. The corpus is defined as the sequence of source IPs as they reached the darknet splitted into a 1001 sentences containing the same number of tokens. In this way the resulting embedding should be generated according to the temporal relationship among IPs (e.g., addresses occurring always one after the other may be similar).

To choose the best hyper-parameters I perform a grid search training several models with different combinations of E and C. For each combination I run a Euclidean-distance-based k-Nearest Neighbors (kNN) classifier on the top-5 ground truth classes and compute the resulting accuracy. Since the word2vec represents each source IP in a E-dimensional space, the main idea consists on finding the best parameters such that nodes belonging to known groups are represented by points close to each other in the

space. The higher the accuracy, the closest are the points belonging to the same class, thus the best hyper-parameters are the one achieving the highest accuracy.

A complete overview of the generated features is reported in Table 3.4.

Macro	**Type**	**Subset ID**	**Description**	**# Features**
Traffic intensity	General	S0	Number of active time interval Amount of daily flow	1 1
	Protocol	S1	Amount of daily flow sent by each IP through each protocol of Table 3.2	4
	TCP flags	S2	Amount of daily flow sent by each IP with each TCP flag of Table 3.2 set	5
	Services	S3	Amount of daily flow sent by each IP to the ports used by each class of service of Table 3.3	15
	PCA	S4	n principal components extracted from Services features	n_{PC}
	SOM	S5	(x, y) coordinates of the SOM BMUs trained on Services features	2
Temporal information	Word2Vec embeddings	S6	IPs embeddings of the daily trace	n_{w2v}

Table 3.4: Generated features summary

The features informative content must be evaluated during the features selection stage. To speed up the process, the generated features are grouped into six subsets according to the type of features.

Nevertheless it is necessary to consider that the features express only nodes attributes, without explicitly defining a degree of similarity/dissimilarity between each source IP address. To obtain this information, a distance metric is defined in Section 3.3. Since during the thesis development different approaches have been tested, in Appendix A.1 a description of additional not considered features.

3.2.3 Features Selection

Features selection processes are mainly used to solve the so-called *curse of dimensionality* problem and reduce the computational requirements of the ML model execution. Considering that in most cases the features generation process introduces redundant or noisy information, performing features selection allows to increase the model capabilities by removing uninformative data.

Several features selection techniques are proposed in literature [25, 26, 27]. For the sake of simplicity it is possible to classify them in three main categories:

- **Wrappers**. Wrapper algorithms require a model to score the feature subsets. Different subsets are iteratively used to train and test a model, thus from the prediction it is possible to obtain a model error measure which can be used as feature rank;

- **Filters**. This algorithms class is independent of any ML model, indeed it allows to select and discard features on the basis of their statistical properties (e.g. Pearson's correlation) or the outcome of statistical tests (e.g. Chi-Square);

- **Embedded methods**. They combine the characteristics of both wrappers and filters and their implementation is part of the model construction process (e.g. LASSO).

Before describing the features selection process, it is necessary to specify the concept of *feature goodness*: a feature is required to embed a satisfactory informative contribution about the nodes behavior, avoiding information redundancy and preventing the overlap of the nodes spatial projection. Such feature should be able to provide a spatial separation of the points exhibiting different behaviors which can be exploited by clustering algorithms. To validate the features selection algorithm I use the 5 most active classes of ground truth (Section 3.1) and for preventing the selection to be biased, I add 1000 unlabelled points (200% of the top-5 GT addresses). Furthermore, to reduce the bias by the GT classes, the features are grouped in 6 subsets as reported in Table 3.4 (see Subset ID column). In this way I avoid to select features which may be discriminant for the considered classes, but irrelevant for unknown IP addresses. As features selection technique I propose a wrapper algorithm relying on a simple kNN

classifier[2]. The approach can be summarized as:

1. For the considered dataset generate all the possible combinations of features subsets (e.g., {S0, S2} or {S5, S6, S7}, etc.);
2. For each combination perform a Leave-One-Out kNN classifier [28, 29] on the considered dataset;
3. Rank the features combination computing the accuracy resulting from the classifier;
4. Get the features subsets combination yielding the highest rank.

3.3 Distance Metric Definition

Since the coordination is referred between source IPs, choosing the best distance metric is necessary to represent the degree of similarity in their behavior. The generated features are variables $\in \mathbb{R}$, thus I chose the distance metric accordingly. Among the wide set of known metrics, I test four of them, the *Canberra* distance [30, 31], the *Cosine* distance, the *Bray-Curtis* distance [32] and the *Euclidean* distance [33]. Each metric is used to generate a matrix distance $\mathbf{D} \in \mathbb{R}^{N \times N}$ such that the entry $D_{i,j}$ is the distance applied to the (i, j) nodes pair. $\mathbf{D}$ will be the input of the clustering algorithms. Given two vectors of F nodes features $\mathbf{n_1} = [n_{1,1}, n_{1,2}, \ldots, n_{1,F}]$, $\mathbf{n_2} = [n_{2,1}, n_{2,2}, \ldots, n_{2,F}]$, I define the following distance metrics as:

- Canberra distance:
$$d_{Ca}(\mathbf{n_1}, \mathbf{n_2}) = \sum_{i=1}^{F} \frac{|n_{1,i} - n_{2,i}|}{|n_{1,i}| + |n_{2,i}|} \tag{3.3}$$
- Cosine distance:
$$d_{Co}(\mathbf{n_1}, \mathbf{n_2}) = \frac{\mathbf{n_1}\mathbf{n_2}^T}{||\mathbf{n_1}||\ ||\mathbf{n_2}||} \tag{3.4}$$

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html`

- Bray-Curtis distance:

$$d_{BC}(\mathbf{n_1}, \mathbf{n_2}) = \frac{\sum_{i=1}^{F} |n_{1,i} - n_{2,i}|}{\sum_{i=1}^{F} |n_{1,i} + n_{2,i}|} \tag{3.5}$$

- Euclidean distance:

$$d_{E}(\mathbf{n_1}, \mathbf{n_2}) = \sqrt{\sum_{i=1}^{F} |n_{1,i} - n_{2,i}|} \tag{3.6}$$

After having defined the distance metrics, it is necessary to explain the best metric choice criterion. An adequate distance metric for clustering algorithms must satisfy the following three properties:

1. **Clusters separation**. The nodes coordination characterizing a found cluster must be unique, thus the distance metric must be able to express a certain degree of dissimilarity between samples in order to provide a satisfactory spatial separation between different behaviors;

2. **Robustness against noisy features**. As already reported in Section 3.2.3, grouping the features into subsets allows to prevent the feature selection process to be biased on the ground truth. Nevertheless, depending on the considered dataset, some uninformative features can still be retained (e.g. services which are never reached in a certain time range). A robust distance metric must be able to reduce the contribution of such features to the overall pairwise distance;

3. **Robustness against unknown instances**. Despite the fact that the GT described in Section 3.1 can be used during the model development, the previously two properties must be still satisfied even when considering a more realistic scenario in which nodes coordination is not known a priori.

To evaluate the metrics I use a Leave-One-Out (LOO) kNN classifier on the ground truth dataset. The metric resulting in the highest accuracy is the one that will be used for testing the chosen models. The first property is evaluated using only the ground truth dataset, then the second property satisfaction is investigated by considering both the features passed the selection and the complete set. Finally the third property is evaluated by adding unknown points to the used ground truth.

3.4 Evaluation of Nodes Representation

Once the features have been selected and the best distance metric has been identified, I investigate the spatial representation of nodes. The goal of this stage is to understand if the IP addresses whose coordination is known a priori are placed close to each other with respect to the unknown points in the features space. Indeed, if coordination exists between pairs of nodes, they are supposed to have similar features value ending up in the same neighborhood. In this stage I apply a LOO kNN classifier estimating the classes of the full ground truth set of a given day with respect to the complete dataset of the same day. The resulting accuracy expresses the goodness of the nodes proximity. Nevertheless, if the proximity of nodes belonging to the same ground truth class is achieved only for a single day, the performance is not satisfactory since the nodes placement should be consistent with time. Thus I extend the temporal horizon up to 10 days and perform a twofold analysis according to the features macro-categories. First of all, I introduce the notation t_0 indicating the reference day and t_i with $i \in [1, \ldots, 9]$ indicating the days in the extended 10-days horizon. Then for the two features macro-types:

- Traffic intensity features. For each day I run the kNN classifier on the full ground truth of t_i on the basis of the classes of t_0 as shown in Figure 3.3.

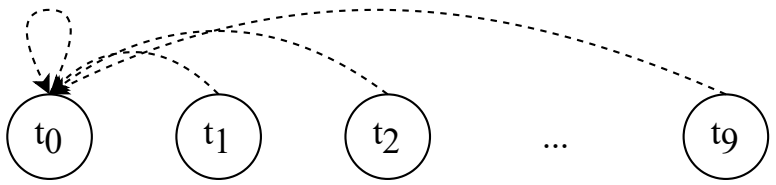
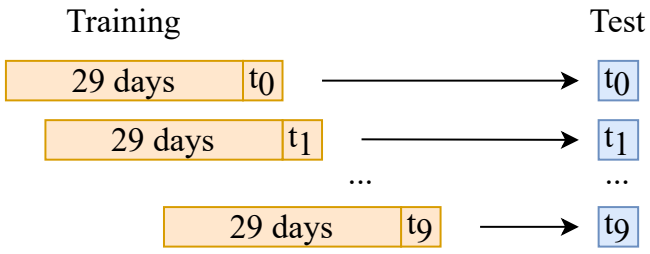


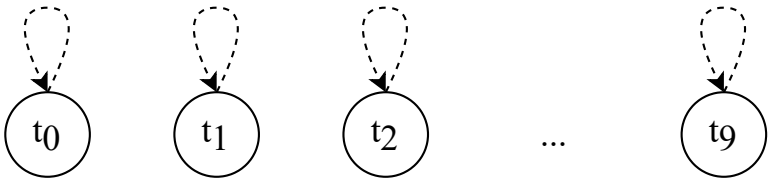
Figure 3.3: Days comparison in temporal evolution analysis of nodes placement. Traffic intensity features

- Temporal relationship features. Since the embeddings generation for an IP never seen before requires a new word2vec model training, the approach used for traffic intensity features is no longer applicable. Indeed, after the each model update, the embedding of an IP active in t_0 is different from the one of the same IP active in t_i. Thus different days embeddings are not directly comparable. To solve this problem I iteratively update the model training it with traffic collected during a

30-days sliding observation window and generate the embedding for the last day like shown in Figure 3.4a, then I run a LOO-kNN classifier on the full ground truth evaluating the accuracy achieved for each day. The day comparison is summarized in Figure 3.4b. In this way, it is possible to evaluate if, even with different embeddings values, the IPs belonging to the same GT class fall in the same neighborhood each day.



(a) Sliding window for word2vec training



(b) Days comparison in temporal evolution analysis of nodes placement. Temporal relationship features

Figure 3.4: Adopted approach for evaluating the word2vec nodes representation

3.5 Clustering Algorithms

Clustering is the process of grouping a set of objects in subsets such that objects belonging to the same group, i.e., a cluster, are more similar between each other than to objects belonging to other clusters. In this phase, the aim is to cluster the source IP addresses reaching the darknet. Each observation is described by the set of attributes defined in Section 3.2.2, thus the complete dataset is $\mathbf{X} \in \mathbb{R}^{N \times F}$ where N is the number of objects and F is the number of features. Each cluster includes a set of IPs exhibiting a possible coordinated behavior. Such behavior must be unique and different from the one characterizing other clusters. The choice of the algorithm is a crucial issue in clustering problems, since according to the data structure the performance and the de-

tected clusters may drastically change. I test four different algorithms with the relative heuristics for determining the hyper-parameters:

1. Density-Based Spatial Clustering of Applications with Noise (DBSCAN)[3] based on spatial points density which does not require the desired number of clusters in input;

2. Hierarchical Agglomerative Clustering (HAC)[4] which builds a clusters hierarchy requiring the desired number of clusters in input;

3. k-Means[5] which clusters data according to the distance between the observation and the cluster centers (or centroids). It requires the desired number of clusters k in input;

4. Greedy Modularity Algorithm (GMA)[6] well suited for directed and undirected graph structured data.

3.5.1 Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

DBSCAN [34, 35] is a density-based clustering algorithm. Clusters are defined as areas of high point density, whereas observations in sparse areas are treated as noise. Being a clustering algorithm, a distance metric is required.

Before providing an overview of DBSCAN functioning it is necessary to define the two core parameters:

- ϵ. It is called the *neighborhood radius* and states the maximum distance between two samples to be considered as neighbors.

- *minPts*. It specifies the minimum number of points required to have a cluster

Basically a point i is called a *core* point if there are at least *minPts* points within a ϵ distance range. All the points $j \neq i$ satisfying this condition are called *directly*

[3] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html
[4] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html
[5] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
[6] https://python-louvain.readthedocs.io/en/latest/api.html

reachable from i and they belong to the same cluster C_i. However, a point $j \neq i$ can belong to C_i if it exists a path of core points between i and j. In this case, j is *reachable* from i. A point which does not satisfy none of the above conditions is treated as noise.

The main functioning of DBSCAN can be summarized in three phases. First of all for each point it is identified its ϵ neighborhood and the core points are defined. Then all the core points paths are found and finally each non-core point is assigned to the nearest cluster if it is an ϵ neighbor of it, otherwise the point is considered as noise.

See Appendix A.3 for the hyper-parameters heuristics.

3.5.2 Hierarchical Agglomerative Clustering (HAC)

HAC is a cluster analysis technique aiming at building a hierarchy of clusters in a greedy manner. The rationale behind the algorithm consists on starting by treating each observation as a separate cluster, then, according to a *distance* metric and a *linkage* criterion, it iteratively merges the nearest or most similar clusters in a new one until a unique cluster containing all the samples is not obtained.

The core of the algorithms consists on the distance metric, which expresses a similarity degree among the observations, and the linkage, which determines the distance between set of samples as a function of the pairwise distances between points. One of the most used linkage is the *complete* one, in which the distance between a pair of newly formed clusters C_i and C_j is defined as:

$$dist(C_i, C_j) = \max(d(i, j) : i \in C_i . j \in C_j) \tag{3.7}$$

where $d(\cdot, \cdot)$ is the distance metric between samples (e.g. one of those described in Section 3.3).

See Appendix A.3 for the hyper-parameters heuristics.

3.5.3 k-Means

Given a dataset $\mathbf{X} \in \mathbb{R}^{N \times F}$, where N is the number of samples and F is the number of features, k-Means [36] groups the N samples into $k \leq N$ clusters $C = C_0, ..., C_k$ in order to minimize the within-cluster sum of squares, or *inertia*, through the objective function:

$$\arg\min_{C} \sum_{i=0}^{k} \sum_{\mathbf{x} \in C_i} ||\mathbf{x} - \mu_\mathbf{i}||^2 \tag{3.8}$$

where $\mu_{\mathbf{i}}$ is the centroid of the cluster C_i and $\mathbf{x}$ is one of the samples $\in \mathbf{X}$.

See Appendix A.3 for the hyper-parameters heuristics.

3.5.4 Greedy Modularity Algorithm (GMA)

Greedy Modularity Algorithm [37] is not properly a *clustering* algorithm, indeed, being applied to directed or undirected graphs, it is defined as a *community* detection method based on the modularity optimization. Before investigating its functioning it is necessary to define the modularity Q. By considering an undirected graph $G = (V, E)$, where V is the set of nodes, or vertices, and E is the set of edges, or links, a community is a group of nodes densely connected internally, whereas the modularity is a quantitative measure of the edges density inside a community with respect to the one outside it. The formal definition of modularity is

$$Q = \frac{1}{2m} \sum_{i,j} \left[E_{i,j} - \frac{k_i k_j}{2m} \right] \delta(C_i, C_j) \tag{3.9}$$

where $E_{i,j}$ is the edge between nodes i and j, k_i is the sum of the edges linked to node i, m is the sum of all the graph edges, C_i is the community the node i belongs to and δ is the Kronecker delta function equal to 1 if $C_i = C_j$, 0 otherwise. The GMA is based on two phases iteratively repeated. In the first stage it starts by considering each node as a unique member of a different community, then each node i is removed from its current community and it is moved into the one of each neighbor j of i. This will cause a change in the overall modularity which is determined as

$$\Delta Q = \left[\frac{\sum_{in} + 2k_{i,in}}{2m} - \left(\frac{\sum_{tot} + k_i}{2m} \right)^2 \right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m} \right)^2 - \left(\frac{k_i}{2m} \right)^2 \right] \tag{3.10}$$

where, by assuming that node i is moving in the community C_j, then $\sum_{in}$ is the sum of the weight of the links inside C_j, $\sum_{tot}$ is the sum of the weight of the links *to* nodes of C_j, k_i is the weighted degree of node i, $k_{i,in}$ is the sum of the weights of the links between node i and other nodes belonging to C_i and m is the sum of all the graph edges. In the second phase, it is generated a new graph whose nodes are the communities found in the first phase. The intra-community edges are now represented by self-loops on the community node with the relative weights set accordingly.

Being a graph-based algorithm, the GMA application to the considered dataset requires the distance matrix **D**. The used distance metric is the one resulting from the analysis of Section 3.3. The matrix is converted into a similarity one **S** according to practical implementation constraints, thanks to the following transformation:

$$\mathbf{S} = \max(\mathbf{D}) - \mathbf{D} \tag{3.11}$$

This new matrix $\in \mathbb{R}^{N \times N}$, where N is the number of samples, embeds the similarity degree between each pair of nodes and can be used to build an undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of edges.
At this point the GMA is applied to G and the outcome is the set of nodes labelled according to the found communities.

3.5.5 Algorithms Comparison

Similarly to the optimal distance metric definition, it is necessary to test the different clustering algorithms under certain conditions. After performing the features selection process apply the four algorithms on the considered dataset obtaining the detected clusters/communities. To evaluate the chosen algorithms and to select the best one I focus on two quality metrics:

1. **Average silhouette** *sh* (Appendix A.2.1). The more *sh* is close to 1, the better the clustering;

2. **Number of clusters**. Exploiting the domain knowledge, a reasonable number of clusters must be found (e.g., algorithms grouping N points belonging to 5 ground truth classes and some unknown points into $\leq$ 5 clusters are not acceptable. Indeed, because of the ground truth whose coordination is known a-priori, at least 5 different clusters characterizing one ground truth class each should be found);

Even if the silhouette is one of the most common quality metrics for clustering, I introduce the other metrics because of the type of data. Indeed the silhouette is a metric well suited for convex clusters, but this hypotheses is rarely verified when treating data collected from real scenarios. Furthermore, because of the definition of silhouette, by considering a scenario in which only two clusters are found, the first with the 99% of

data, the latter with the remaining 1%, will probably result in a high accuracy, but, in this case, the clustering performance is not satisfying. For this reason, the chosen clustering algorithm is the one resulting in the best trade off between the two metrics.

3.6 Temporal Consistency of Clusters Membership

The approach described so far is useful for developing and testing the adopted methodology. On a single-day case study the results allow to verify if the approach is promising and improve the knowledge about the dynamics of the identified coordinated behavior. The consistency of the model should be further verified over time. For example, it is reasonable to assume that the ground truth IPs whose coordination is known a priori exhibit coordinated behaviors independently from the considered day. From the model point of view, this means that source IPs belonging to cluster C_i in a certain day t_0 should fall in an analogous cluster even in day t_j with $j > 0$.

To evaluate the temporal consistency I consider a w-days time window of darknet traffic. Then:

1. The first day t_0 is considered as reference. I generate the dataset $\mathbf{X}_0 \in \mathbb{R}^{N_0 \times F}$, where N_0 is the number of distinct source IPs active in t_0 and F is the number of selected features.

2. Clustering algorithm is applied and the resulting clusters $\mathbf{C}_{0,\mathbf{X}_0}$ are stored.

3. The process is repeated for each day $i \in 1, ..., w$, obtaining the dataset $\mathbf{X}_i \in \mathbb{R}^{N_i \times F}$, where N_i is the number of distinct source IPs active in t_i, and the relative clusters $\mathbf{C}_{i,\mathbf{X}_i}$.

4. For each day, only the source IPs active in both t_0 and t_1 are kept obtaining the intersection dataset $\hat{\mathbf{X}}$.

5. To evaluate the consistency of the groups membership, I compute the Adjusted Mutual Information (AMI) (see Appendix A.2.2) between $\mathbf{C}_{0,\hat{\mathbf{X}}}$ and $\mathbf{C}_{i,\hat{\mathbf{X}}}$, where $\mathbf{C}_{0,\hat{\mathbf{X}}}$ are the clusters of t_0 the source IPs of $\hat{\mathbf{X}}$ belongs to, whereas $\mathbf{C}_{i,\hat{\mathbf{X}}}$ are the clusters of t_i the source IPs of $\hat{\mathbf{X}}$ belongs to.

The process is repeated for each day in the time window. The adopted methodology is summarized in Algorithm 1.

Algorithm 1: Temporal consistency of clusters membership

Input: w-days time window $W = t_0, t_1, ..., t_w$

Output: set of daily AMI A

$A \leftarrow []$;
train w2v model on 30 days;
train SOM on the first day t_0;
$\mathbf{X}_0 \leftarrow$ selected features of day t_0;
run clustering algorithm on the first day $\mathbf{X}_0$;
$\mathbf{C}_{0,\mathbf{X}_0} \leftarrow$ found clusters;
for $i \leftarrow$ *1 to w* **do**
    $\mathbf{X}_i \leftarrow$ selected features of day t_i;
    run clustering algorithm on t_i;
    $\mathbf{C}_{i,\mathbf{X}_i} \leftarrow$ found clusters;
    $\hat{\mathbf{X}} \leftarrow \mathbf{X}_0 \cap \mathbf{X}_i$;
    $A \leftarrow A + AMI(\mathbf{C}_{0,\hat{\mathbf{X}}}, \mathbf{C}_{i,\hat{\mathbf{X}}})$
end

Chapter 4

Dataset

In this chapter I report an overview on the raw data. First of all I randomly choose 4 days of traffic and get some general information reported in Table 4.1.

Sample day	Distinct source IPs	Received flows	Log files size [MB]
2020/04/05	101008	1920058	27.52
2020/04/06	104887	2050633	28.68
2020/05/05	97589	3350378	45.45
2020/05/06	97913	3325177	44.90

Table 4.1: Preliminary information of randomly sampled data

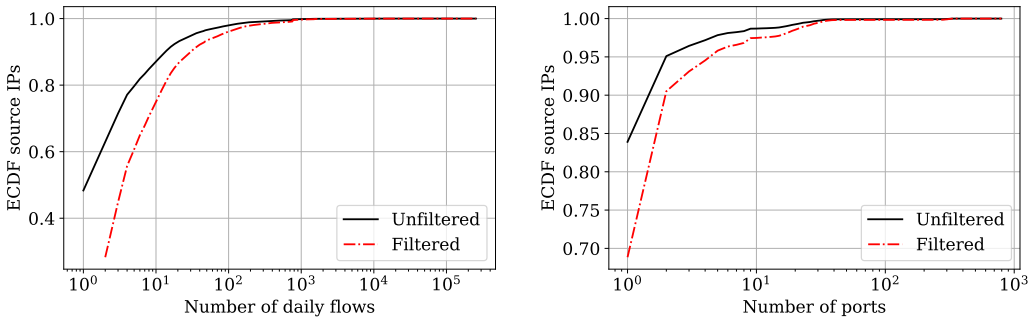
The number of distinct source IP addresses is very similar within the considered day, along with the number of received flows. The file size changes accordingly to the received packets. According to this preliminary information I consider the first sampled day 2020/04/05 as representative of a general day of darknet traffic.

4.1 Data Characterization

As a first attempt of understanding the general behavior of the source IPs active in the darknet, I investigate the distributions of the number of daily flows and contacted ports per source IP address. Assuming that for exhibiting coordinated behavior an IP should

send at least a number of daily flows > 1, I filter out all the nodes sending only one flow. I report the distributions for both the filtered and unfiltered cases.

In Appendix A.4 I report a filtering approach applied on the features generated when testing the multigraph-based approach of Appendix A.1



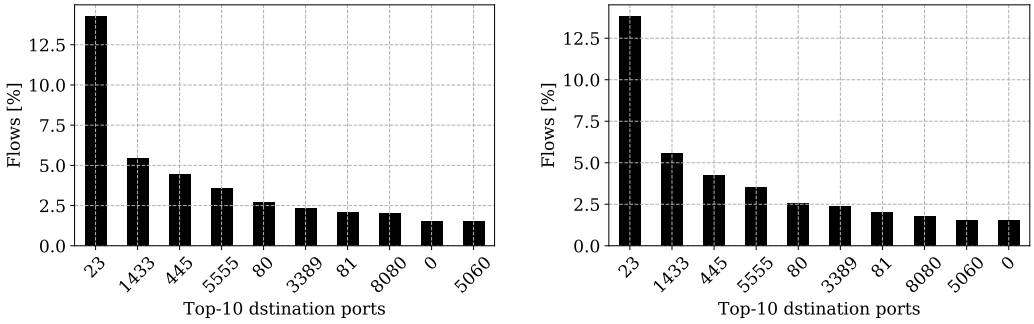
(a) Daily flows

(b) Contacted ports

Figure 4.1: Per-source IP distributions

Figure 4.1a shows that 62.95% sent at least 2 flows. Through the filter it is possible to achieve a source IPs reduction of 48% decreasing the number of IPs from 101008 to 52178. When considering the distribution tail, the Empirical Cumulative Distribution Function (ECDF) shows that only 1% of the source IPs generate more than 700 flows. A similar trend is obtained when considering the number of contacted darknet ports per source IP address shown in Figure 4.1b. Indeed, the 83% of the unfiltered source IPs contact only one port, whereas after the filtering a $\approx$ 15% gap between distributions is achieved. In both the unfiltered and filtered cases, the $\approx$ 1% of IPs reach more than 40 destination ports.

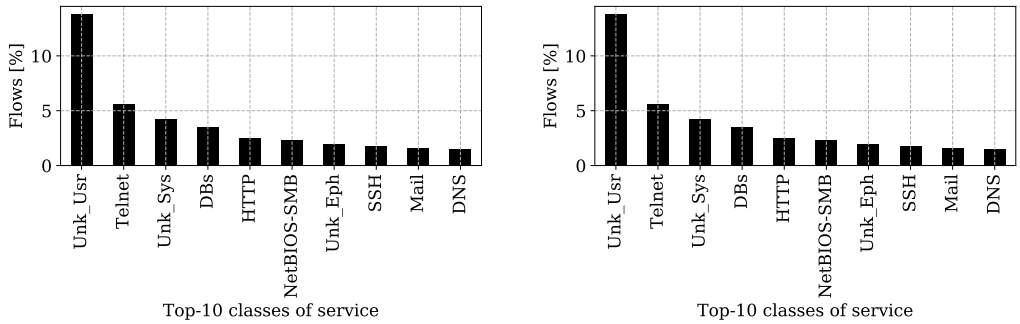
This preliminary data characterization indicates that, in a day of darknet traffic, the most of the IPs reaches few ports sending small amount of traffic, whereas few IPs generate more than 700 flows and reach more than 40 ports.



(a) Unfiltered (b) Filtered

Figure 4.2: Top-10 contacted ports. Port 0 means ICMP protocol.

Figure 4.2 shows the top-10 destination ports contacted in terms of percentage of total received flow. The top-5 ports are quite predictable, indeed port 23 is used by the Telnet service whose security breaches are well known and often exploited by hackers. Port 1433 and 445 are used by the very common services Microsoft SQL server and Server Message Block (SMB) over NetBIOS. Port 5555 is widely used as a backdoor by Trojan like NoXcape and DaoDan. Finally, port 80 is used by the Hypertext Transfer Protocol daemon (HTTP protocol), thus it is usually exploited for attempting attacks against Web servers. This preliminary analysis is coherent with the darknet definition, indeed the considered top-5 ports are commonly targeted by attackers to force widespread services whose vulnerabilities are known. Thus the most of the received traffic may represent a threat.



(a) Unfiltered (b) Filtered

Figure 4.3: Top-10 classes of service

When investigating the classes of service (Figure 4.3), it is possible to make considerations similar to the ones of Figure 4.2. The main difference is the strong presence of unknown user ports. This can be easily explained by considering that, according to Table 3.3, the most of the ports belonging to the [1024, 49151] range are not classified and they are quite likely to be contacted by source IPs according to the wide range.

4.2 Identification of Ground Truth

In this section, I restrict the characterization to the IPs beloning to the Ground Truth, obtained according to the procedure discussed in Section 3.1. The analysis of these IPs is particularly useful for tuning and validating the methodology. Among the detected ground truth classes, 8 of them reached the darknet in the considered day t_0.

Label	**# Flows**	**# Src IPs**	**# Dst Ports**	**Top-5 Ports**
Sonar [38]	16661	359	23	10443, 4567, 9000, 8984, 8899
Ipip [39]	15309	49	42	-, 53, 8000, 8888, 3128
Shodan [40]	9366	12	337	2222, 80, 6666, 88, 8888
Antlab [41]	8651	5	1	-
Engin-umich [42]	760	10	1	53
Cybercrime Cambridge [43]	693	1	2	19, 1434
Google	36	4	1	80
Heodo [44]	11	4	2	23, 445

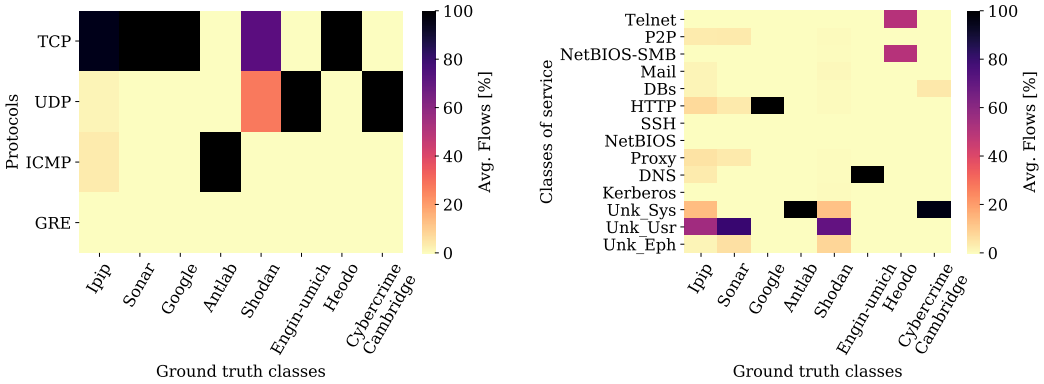
Table 4.2: Ground Truth classes active during t_0: 05/04/2020 sorted by daily flows

Table 4.2 reports the active ground truth classes and sorted by daily flows. The reported information is not filtered. When dropping the source IPs sending only one flow, the nodes remains unchanged, apart from 2 Shodan IPs. Since the IPs reported in table belong to research project performing Internet scans, already from this first analysis two different types of coordination emerge:

- **Vertical scans**: a group of IP addresses targeting the same darknet ports.

- **Horizontal scans**: a group of IP addresses targeting a wider set of ports with possible specific temporal patterns detected through a manual analysis.

According to the two provided definitions, it is reasonable to assume that IPs sets like Sonar, Ipip and Shodan are performing horizontal scans, since they reach a wide set of destination ports. On the other hand, the Antlab and Engin-umich behaviors are more similar to a vertical scan targeting a unique port.



(a) Average percentage of daily flows per protocol normalized by column

(b) Average percentage of daily flows per class of service normalized by column

Figure 4.4: Ground truth classes characterization

The heatmaps in Figure 4.4 provides some more details on the activity of each class. 100% of the Antlab IPs generates ICMP traffic. The Engin-umich set direct the 100% of the flows the port 53/UDP used by the DNS class of service. The addresses belonging to the Ipip class contact a wider set of heterogeneous services, confirming that they are more prone to horizontal scans, as it is visible in , Figure 4.4b. The same assumption holds for Shodan and Sonar.

4.3 Datasets Organization

For developing the models and run the described experiments I use three different categories of datasets:

- Model development dataset: referred to the 2020/04/05. It is used for analyzing the generated features, tuning the model parameters, and comparing the different algorithms. To speed up the processes I downsample data by selecting the top-5 GT classes in terms of generated flows (433 distinct source IPs). To make the methodology unbiased on the considered dataset, 1000 unknown sources are added achieving a dataset with 1433 samples;

- Single day dataset: daily darknet traffic of 2020/04/05. I use it to evaluate the performance of the developed model deepening the knowledge about the possible coordination exhibited in a single day;

- Subsequent day dataset: sequence of daily darknet traffic in the [2020/04/05, 2020/04/15] range. I use it to evaluate the temporal evolution of the found clusters and the consistency of the nodes spatial representation.

All the described datasets are filtered dropping nodes generating only 1 flow.

According to the generated features described in Section 3.2.2, all the experiments are performed considering three different scenarios:

1. Traffic intensity features: I consider only the features related to the traffic intensity (S0, S1, S2, S3, S4, S5);

2. Temporal features: I consider only the 30-days-trained word2vec embeddings (S4);

3. Mixed features: I consider both the features related to traffic intensity and time.

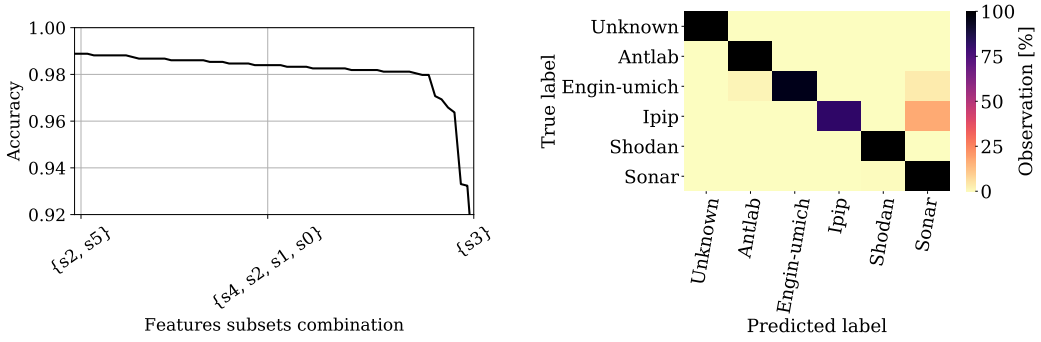
In this way it is possible to deeper investigate the contribution of each features macro-type identifying the best one.

Chapter 5

Results: Traffic Intensity Features

5.1 Features Engineering

The first stage of the model development consists on selecting the best features subsets.



(a) Traffic intensity features

(b) Traffic intensity features. Accuracy: 98.88%

Figure 5.1: Features selection results for the traffic intensity features. Model development dataset

Figure 5.1a reports the trend of the subsets rank made of the accuracy of the kNN classifier during the features selection stage. Since different subsets results in the same higher accuracy, in order to limit the curse of dimensionality effect, under the same accuracy I choose the subset with the lower cardinality.

When considering the traffic intensity features, the subset containing the amount of

daily flow per TCP flag (S2) and the coordinate of the SOM (S5) is the chosen one and 7 single features are considered.

When evaluating the confusion matrix resulting from the kNN LOO classification (Figure 5.1b), a high accuracy of 98.88% is achieved for the considered dataset. For the most of the ground truth classes the most of the source IPs belongs to the same neighborhood a part from 20% of the Ipip group that falls into the Sonar neighborhood. This is reasonable when recalling the similar behavior of the two classes (Figure 4.4b) for which the packets are spread among different services.

5.2 Distance Metric Definition

According to the described methodology I run three different experiments to identify the best distance metric for the proposed problem. For the sake of simplicity I introduce a new notation referred to the experiments:

- exp0: top-5 ground truth classes and 1000 unknown IP addresses. Only selected features;
- exp1: only ground truth classes and selected features;
- exp2: only ground truth classes and all the traffic intensity features sets.

	exp0	**exp1**	**exp2**
Euclidean	**99.31**	98.88	98.85
Bray Curtis	99.08	**98.95**	**99.54**
Cosine	99.08	98.88	99.08
Canberra	**99.31**	98.74	99.31

Table 5.1: kNN classifier accuracy [%] for different distances. Traffic intensity features

The bold results of Table 5.1 indicate the best outcome for each experiments. The Bray Curtis distance (d_{BC}) is the best one for exp1 and exp2, whereas the difference between it and the Euclidean distance (d_E) is quite small (0.07% and 0.69% respectively). Furthermore the Euclidean distance is the best one for exp0. Thanks to the

negligible accuracy difference between d_{BC} and d_E and by considering that the used `scikit-learn` library implementing some of the tested clustering algorithms is supported only for the Euclidean distance, I decide to choose d_E as the best distance metric when running traffic intensity experiments.

5.3 Clustering Algorithms

After having identified both the distance metric and the best features subsets I proceed with the clustering algorithm comparison.

	# Clusters	**Avg. Sh**
DBSCAN	57	0.28
GMA	2	0.31
HAC	2	0.94
k-Means	9	0.47

Table 5.2: Clustering quality metrics when comparing 4 different algorithms and testing the 3 features subsets combinations

Table 5.2 reports the clusters quality metrics obtained from the experiments. Regarding the number of clusters, the model development dataset is composed by 5 ground truth classes and an 'unknown' one containing unlabelled source IP addresses, thus the number of found clusters should be at least 6 meaning that the 6 classes have been identified. However, since the coordination of the 'unknown' class is not known a priori, more clusters may be found.

DBSCAN results 57 number of clusters with an average silhouette of 0.28. This means that the found clusters tends to overlap achieving bad performance. This is mainly due to the limitation of the ϵ parameter of DBSCAN. Indeed, the tested algorithm perform satisfactory when there is a clear boundary between clusters and this generally happens for convex clusters. This condition is rarely satisfied in real scenarios like the considered case, thus the algorithm is not well suited for the proposed problem.

Different results are obtained when applying Greedy Modularity Algorithm and

Hierarchical Agglomerative Clustering. Indeed both of them detect only two clusters, which is below the lower bound of 6 clusters. Furthermore, on one hand, GMA results in an average silhouette tending to 0 meaning overlapping clusters, on the other hand, HAC achieves a silhouette close to 1, but from a manual investigation of the points per clusters, one of the two groups contain only one point. According to these considerations, neither the GMA or HAC can be used in the methodology.

The only algorithm suitable for the proposed problem is the k-Means. Regarding the number of clusters, it satisfies the lower bound condition, regarding the average silhouette, even if it is not acceptable (satisfactory values are general ≥ 0.8), it is greater than the DBSCAN one.

In light of the above, I chose k-Means as clustering algorithm to embed in the methodology.

5.4 Evaluation of Nodes Representation

After having identified the best parameters and algorithms to use in the proposed methodology I proceed with the evaluation of nodes representation. The rationale behind it is to verify if nodes belonging to the same ground truth class are in the same neighborhood.

As a first visual evaluation I adopt the t-distributed Stochastic Neighbor Embedding (tSNE) technique. It reduces data dimensionality by projecting each data point to a 2D space. The relative position of the points is proportional to their similarity, thus similar nodes are placed close to each others. Nevertheless, the obtained projection is an approximation, thus it must not be considered as a final representation.[1]

Figure 5.2 shows the tSNE projection of the traffic intensity dataset. For the sake of simplicity I randomly downsample all the unknown keeping only 3000 of them. From this first evaluation it is possible to spot some cluster structures among the IPs not belonging to the ground truth. On the other hand, when evaluating the GT points, it is not possible to draw a unique considerations among all the classes. Indeed, classes like Heodo which perform mostly Telnet scans tends to be confused with the unlabelled

[1]During the intermediate developing stages I test an alternative method for visualizing the nodes placement by exploiting the software Gephi. In Appendix A.5 I provide an overview of the proposed approach and the reasons why I discard it.

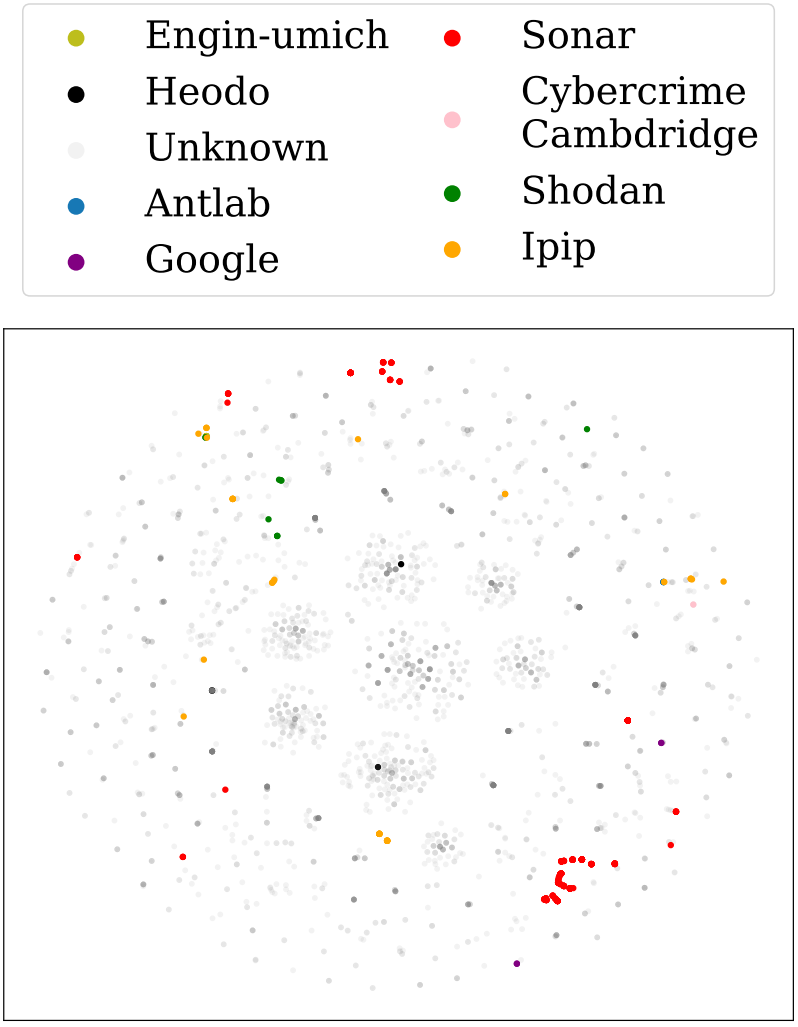


Figure 5.2: tSNE projection of traffic intensity features

points. This is mainly due to the high Telnet traffic generated by general IP addresses. When focusing on Sonar, the wide range of contacted classes of service makes them spread among the space. However, even if all the points are not in the same neighborhood, the most of them tend to overlap. Indeed, from Figure 5.2 it is possible to spot ≈ 9 different groups. This could mean that Sonar IPs tends to share the scan workload assigning the same target to a subgroup of nodes. Same considerations can be drawn for the Ipip class.

To investigate deeper the spatial projection I use a LOO-kNN classifier on the original traffic intensity dataset as described in Section 3.4.

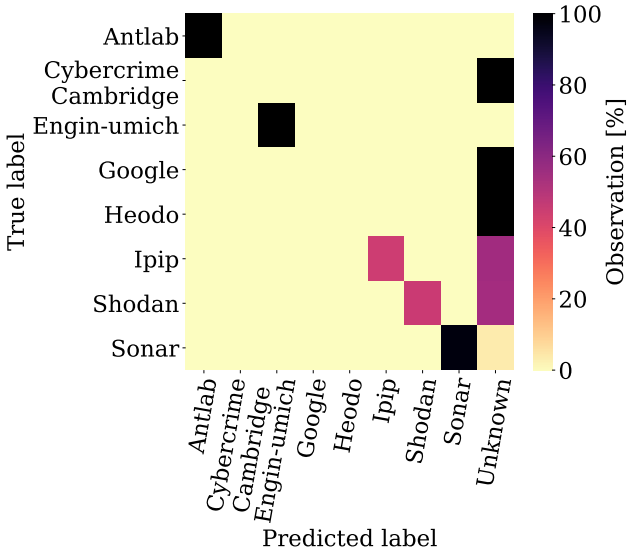


Figure 5.3: LOO-kNN confusion matrix. Traffic intensity features. Accuracy: 98.88%

Figure 5.3 reports the confusion matrix resulting from the classification of the full ground truth active in 2020/04/05. Since I set $k = 1$, the 98.88% of accuracy indicates that the nodes representation allows to place in the same neighborhoods nodes belonging to the same GT class. More specifically, the 100% of classes with a clear coordinated behavior in terms of targeted classes of service like Antlab and Engin-umich is correctly classified. By comparing this figure with Figure 5.2 in which the considered classes are not visible, it is possible to assume that the tSNE projection makes the considered groups overlap.

Despite the spread placement of Sonar nodes achieved through tSNE, the kNN classifier results in a correct classification of $\approx$ 100% of the observations. Different considerations can be drawn when considering Ipip and Shodan, for which only the 50% of the observations are correctly classified. This is in line with what has been said about their spread placement in Figure 5.2. Furthermore, by considering the spread behavior of Shodan and Ipip among the different classes of service, it is possible to assume that the considered classes does not follow any recognizable pattern with respect to the traffic intensity levels and types.

Finally, when considering the three least popular classes in terms of number of IP sources and amount of generated flows (Google, Heodo and Cybercrime Cambridge), the kNN classifier fails for the 100% of the observation. This means that with respect to the traffic intensity, the nodes representation is not able to make the considered nodes close to each other, making impossible the detection of any pattern.

The carried out analysis allows to state that for a single day of darknet traffic the selected features are able to represent source IPs in a robust way, that is the most of nodes belonging to the same ground truth class (whose coordination is known a priori) belongs also to the same neighborhood. In this way, a kNN classifier is able to achieve the 98.88% of accuracy by evaluating the first nearest neighbor of the considered points.

Even if the approach seems promising for a single day, it is necessary to evaluate if the proposed methodology is able to spot changes in traffic intensity over time.

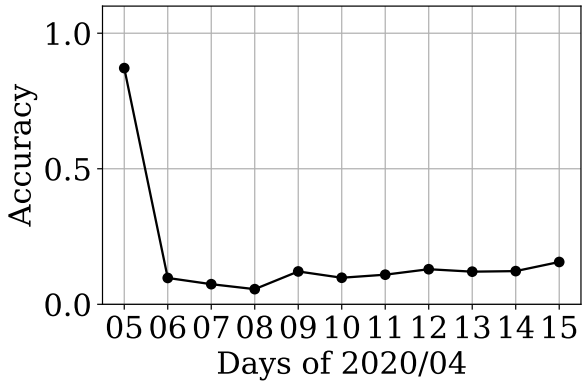


Figure 5.4: Temporal evolution of the kNN accuracy on full ground truth

Figure 5.4 shows the trend of the LOO-kNN accuracy over time. As described in Section 3.4, the day 2020/04/05 is used as reference and the classifier aims at labelling the ground truth of the next days with respect to the ground truth of the reference day.

The high accuracy achieved for the reference day discussed above is not maintained over time, indeed, after only one day it drops from 98.88% down to $< 30\%$. However, after the day 2020/04/08, the trend tends to increase. This singular behavior can be explained by considering the diverse traffic intensity and types generated by the considered ground truth.

According to the provided analysis, it is possible to state that the traffic intensity features are able to spot coordination among ground truth IPs with high accuracy as long as the considered classes consist of a sufficient number of distinct IP addresses generating a significant amount of flow. Furthermore, the satisfactory performance is achieved only if the observation window is ≤ 24 hours because of the generated traffic changes too fast over time.

5.5 Clustering on Single Day

The k-Means clustering algorithm is the chosen one according to the model development analysis. To evaluate its performance I use the dataset related to the 2020/04/05. However, according to the high data dimensionality, a visual evaluation of the found clusters is unfeasible, even by exploiting the tSNE technique which approximates the results. In light of this I report the characterization of the found clusters in terms of percentage of cluster flows per class of service and used protocol (Figure 5.5) and the distribution of the number of GT nodes per cluster (Table 5.3).

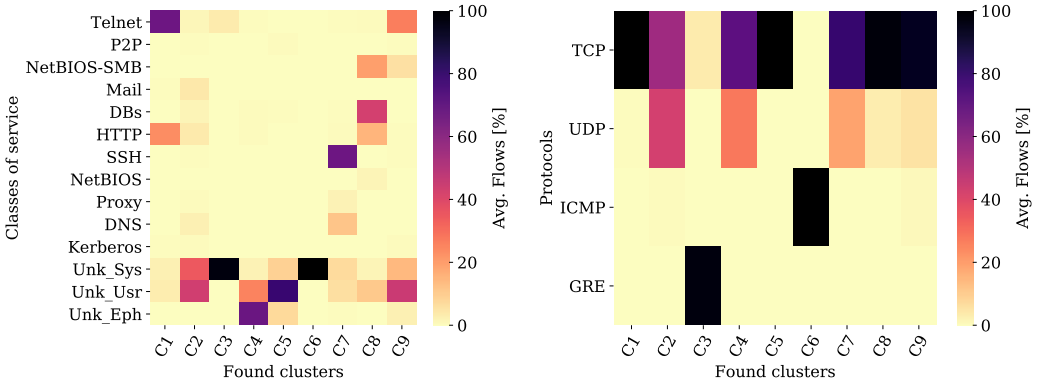
Firstly, by considering the Antlab GT class, all the IPs belongs to the same cluster C6. The cluster membership is not unique, indeed 8 unknown points and 2 Ipip ones are assigned to the same cluster. The characterization of Figure 5.5b indicates that the 100% of the C6 IPs generates ICMP traffic reflecting the Antlab class behavior.

When focusing on Engin-umich, even if all its IPs belong to the same cluster C7, other 422 nodes are assigned to the same cluster. Furthermore, by recalling that the 100% of the Engin-unich traffic is directed to the DNS class of service, in Figure 5.5a only the $\approx 20\%$ of the C7 traffic target the same class of service indicating poor performance.

Regarding the other GT classes, from Table 5.5 it is clear that most of them are assigned to the same cluster C9 containing the 79.8% of the points. This means that the clustering algorithm fails in detecting not only the possible coordination among the unknown IPs, but also the known coordination of the ground truth classes, even though they belong to the same neighborhood when evaluating their spatial representation.

	C1	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**
Unknown	21876	2	2	42	1	8	415	5084	24314
Ipip	8	-	-	1	-	2	5	6	27
Sonar	6	-	-	-	-	-	-	-	353
Antlab	-	-	-	-	-	5	-	-	-
Shodan	-	-	-	-	-	-	2	-	9
Engin-umich	-	-	-	-	-	-	10	-	-
Total	21890	2	2	43	1	15	432	5090	24703

Table 5.3: Number of observations per found cluster. Traffic intensity dataset. Average Silhouette: 0.446



(a) Classes of service

(b) Protocols

Figure 5.5: Found clusters characterization. Traffic intensity features

Furthermore, three clusters (C2, C3 and C5) contain < 3 points making uneven the distribution of nodes per found cluster. Finally, the achieve average Silhouette is

0.446, meaning that the found clusters are not well separated tending to overlap with each other.

According to the proposed analysis it is possible to state that the adopted clustering algorithm is able to spot coordination only among source IPs reaching a limited number of destination ports generating a consistent amount of flows. Regarding the other IPs, the chosen algorithm fails in detecting any kind of coordination.

5.6 Evolution of Clusters Membership

As final analysis on model performances I investigate how the membership of the found clusters evolves over time. It is necessary to recall the provided definition of $\hat{\mathbf{X}}$ containing all the IPs active both in the reference day t_0 and in t_i with $i \in [1, \ldots, 9]$. The reference day is the 2020/04/05.

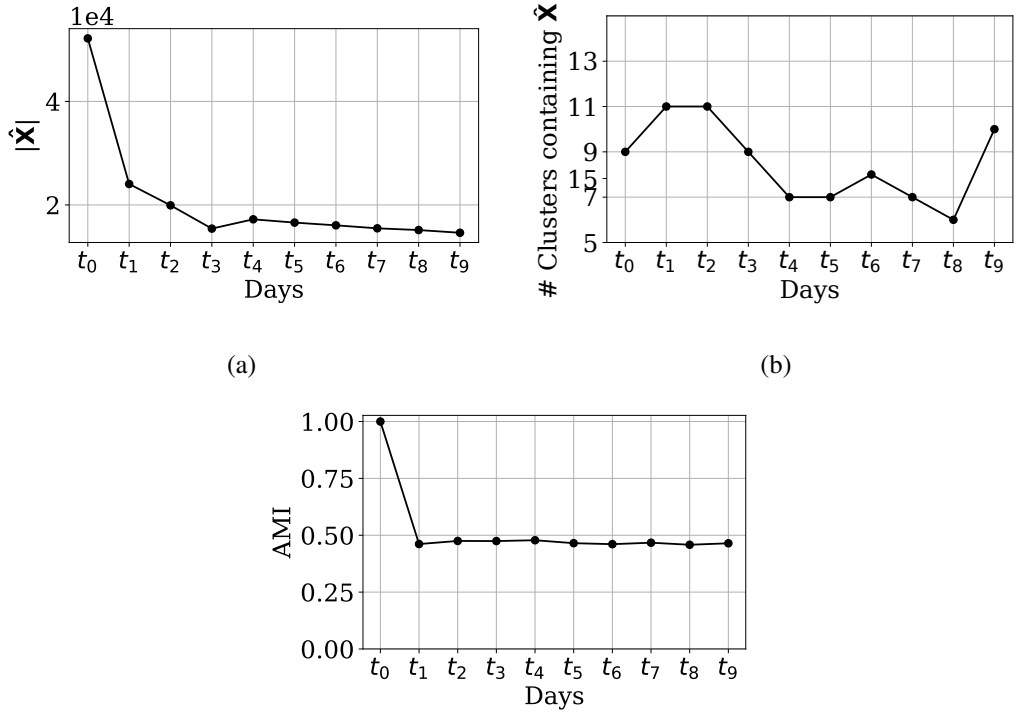


Figure 5.6: Analysis of cluster membership evolution from t_0 to t_9. Traffic intensity features. (a) Source IPs propagation from t_0 to t_9. (b) Number of found clusters containing at least one node propagating from t_0 to t_9. (c) Temporal evolution of clusters membership

Figure 5.6 shows the outcome of the conducted analysis when the traffic intensity features are used. Regarding the number of points propagating over time, in Figure 5.6a a rapid decrease from $\approx$ 50000 nodes down to $\approx$ 23000 points is shown. However, by considering that in t_0 contains all the active nodes, the initial decrease is perfectly reasonable. On the other hand, a part from the reference day, the number of nodes propagating over time oscillates between $\approx$ 23000 and $\approx$ 18000.

When evaluating the trend of the number of found clusters containing at least one propagating node shown in Figure 5.6b, it is clear that probably k-Means is not a robust algorithm to solve the proposed problem. Indeed, over 10 days of data, nodes $\in \hat{\mathbf{X}}$ are assigned to a number of clusters oscillating between 11 and 6. In the first case, where a 22% increase in the number of clusters is achieved, the propagating nodes are more

spread among the found groups. In the second case, where a 33% decrease is obtained, propagating nodes are more close between each other falling in less clusters compared with the reference day.

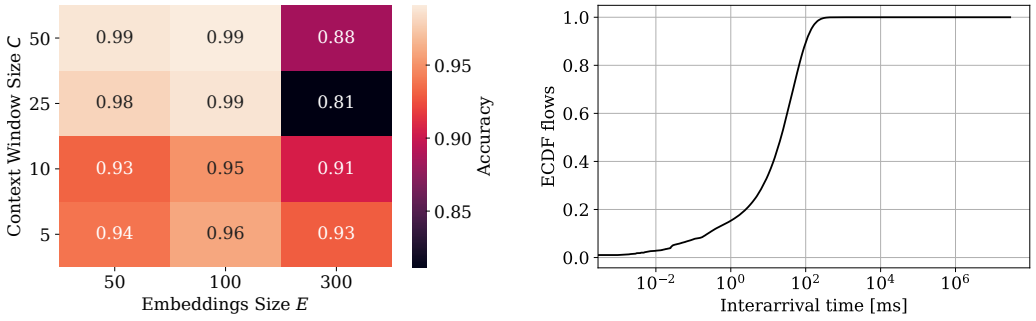
Finally, when evaluating the obtained Adjusted Mutual Information between clusters of t_0 and t_i shown in Figure 5.6c, an unsatisfactory almost constant 0.5 value is obtained (acceptable values are > 0.8).

Chapter 6

Results: Temporal Relationships Features

6.1 Word2Vec Grid Search

When considering the temporal relationships, the first stage consists of evaluating the hyper-parameters of the word2vec model. More precisely the size of the context window C and the size of the embeddings per token E.



(a) Accuracy heatmap resulting from grid search

(b) Packets interarrival time distribution

Figure 6.1: Grid search analysis of w2v models

Figure 6.1a shows the results of the grid search. The accuracy is obtained by applying a LOO-1NN classifier on the model development dataset. The 9 different models

are trained on the same 30 days. The impact of the embeddings size is not so relevant apart from $E = 300$. This can be explained by considering that I run the kNN classifier based on the Euclidean distance. Indeed the used distance is known to be affected from the *curse of dimensionality*, in which the statistical significance of data is reduced according to the high data dimensionality and the consequent increase of data sparsity. Therefore the best performance is obtained with $E = 100$ which is the value I choose for the development of this thesis.

Regarding the context window size C, when increasing it from 5 to 10 the kNN accuracy shows a decreasing trend. On one hand, when $E = 300$, the trend decrease for $C = 25$, then it is subject to a slight increase for $C = 50$. On the other hand, when setting $E = 50$ and $E = 100$, a context window size > 10 improve the performance.

By considering the interarrival time distribution of Figure 6.1b, the time between the reception of a packet and the following one is $\leq$ 80ms for the 80% of the flows. Thus, by considering this value as an arbitrary upper bound, a context window size $C = 50$ means that for each received flow, the model updates the neural network weights observing what is reached 4s before and 4s after the considered packet, whereas with $C = 25$, the current packet is centered in a 4s time window. By considering the huge number of packets received and the short interarrival times, the first case may introduce context errors, thus I choose $C = 25$.

6.2 Features Engineering

The features selection stage is not necessary in this case, since I generate only one subset of temporal features. Thus, the kNN is applied to the dataset $\mathbf{X} \in \mathbb{R}^{1433\times100}$. In this way, by exploiting the ground truth classes it is possible to investigate if the considered features are able to place points belonging to the same class close to each others.

Figure 6.2 reports the confusion matrix resulting from the classification. Temporal features result in the lowest accuracy of 98.67%. To explain the obtained value, it is possible to exploit the activeness timeseries of the ground truth classes. For example, when considering Antlab, from Figure 6.3a a temporal pattern is clearly recognizable. The word2vec model is able to detect it generating the embeddings accordingly, thus from the confusion matrix it is clear that nearest neighbors of the Antlab IPs belong

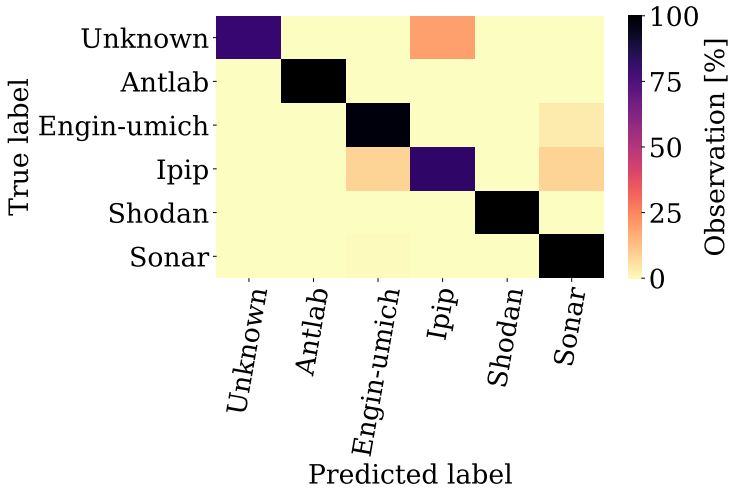
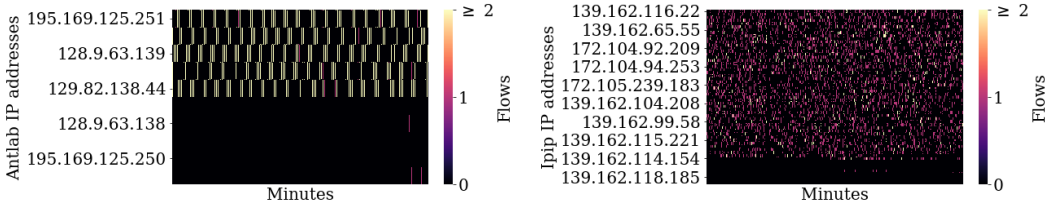


Figure 6.2: Temporal relationships features. LOO-kNN Accuracy: 98.67% traffic

to the same class. On the other hand, when considering the Ipip class of Figure 6.3b, the IPs activeness seems randomly distributed among the 2020/04 and probably the word2vec tends to confuse the ≈ 20% of the source IPs with other classes.



(a) Antlab

(b) Ipip

Figure 6.3: Timeseries of the Antlab and Ipip activeness within 2020/04

6.3 Distance Metric Definition

When working with temporal features, the same experiments names proposed in Section 5.2 are maintained. However, since I define only the subset S6 as temporal features subset, the experiment involving all the subsets is omitted. Thus the notation is:

- exp0: top-5 ground truth classes and 1000 unknown IP addresses;

- exp1: only ground truth classes.

	exp0	**exp1**
Euclidean	98.85	**98.68**
Bray Curtis	**99.31**	96.37
Cosine	**99.31**	97.42
Canberra	**99.31**	95.47

Table 6.1: kNN classifier accuracy [%] for different distances and different features types

The bold results of Table 6.1 indicate the best outcome for each experiments. The considerations about the reported outcomes are similar to the ones of Table 5.2. The Euclidean distance results in the highest classification accuracy for exp1, whereas the difference between it and the accuracy achieved with other distance metrics in exp0 is negligible (0.46%). According to the reported considerations and the implementation constraint, I choose the Euclidean distance for temporal relationships features as for the traffic intensity case.

6.4 Clustering Algorithms

As for the traffic intensity case I proceed with the comparison between the four proposed clustering algorithms to choose the best one for the considered problem.

	# Clusters	**Avg. Sh**
DBSCAN	1	-
GMA	3	0.18
HAC	2	0.80
k-Means	19	0.15

Table 6.2: Clustering quality metrics when comparing 4 different algorithms and testing the 3 features subsets combinations

Table 6.2 reports the number of clusters and the average silhouette for the conducted experiments. The number of clusters found by DBSCAN is 1, meaning that all the nodes are treated by the algorithms as noise. Generally this happens when the input dataset is sparse and characterized by high dimensionality. Since DBSCAN detects only one cluster, it is not possible to obtain the average silhouette and the algorithm can be discarded.

As in the traffic intensity case, neither the Greedy Modularity Algorithm, nor the Hierarchical Agglomerative Clustering one achieve satisfactory performances. Indeed the GMA founds three clusters almost completely overlapping (0.18 of average Silhouette), whereas the HAC results in a high average Silhouette of 0.8, but it detects only two clusters. Furthermore, from a manual investigation of the outcomes, the second cluster of HAC contains only one point.

Even in this case the k-Means algorithm seems to be the best one spotting 19 clusters, but with the lowest average Silhouette (0.15) of the four experiments. In light of the above, I chose k-Means as clustering algorithm used in the proposed methodology.

6.5 Evaluation of Nodes Representation

To visually evaluate the nodes representation I use the same tSNE technique used for the traffic intensity case by downsampling the unknown points (I keep only 3000 of them). The projection is reported in Figure 6.4.

Regarding the ground truth classes, the outcomes seems more promising with respect to the traffic intensity case. Indeed, by considering, for example, the Sonar points, they form an evident cluster in the bottom left corner. A part from few exception, it is possible to state the same for the Ipip and Shodan classes. Indeed, in the right of the figure, two small clusters containing the points of the considered classes can be spotted. However there are some single observations falling in the central cloud of unknown points.

When considering the critical classes of the traffic intensity case (Heodo, Cybercrime Cambridge and Google), it is reasonable to assume that the LOO-kNN classifier will fail the classification. Indeed the points spatial projection results in points spread all over the dimensionality-reduced space.

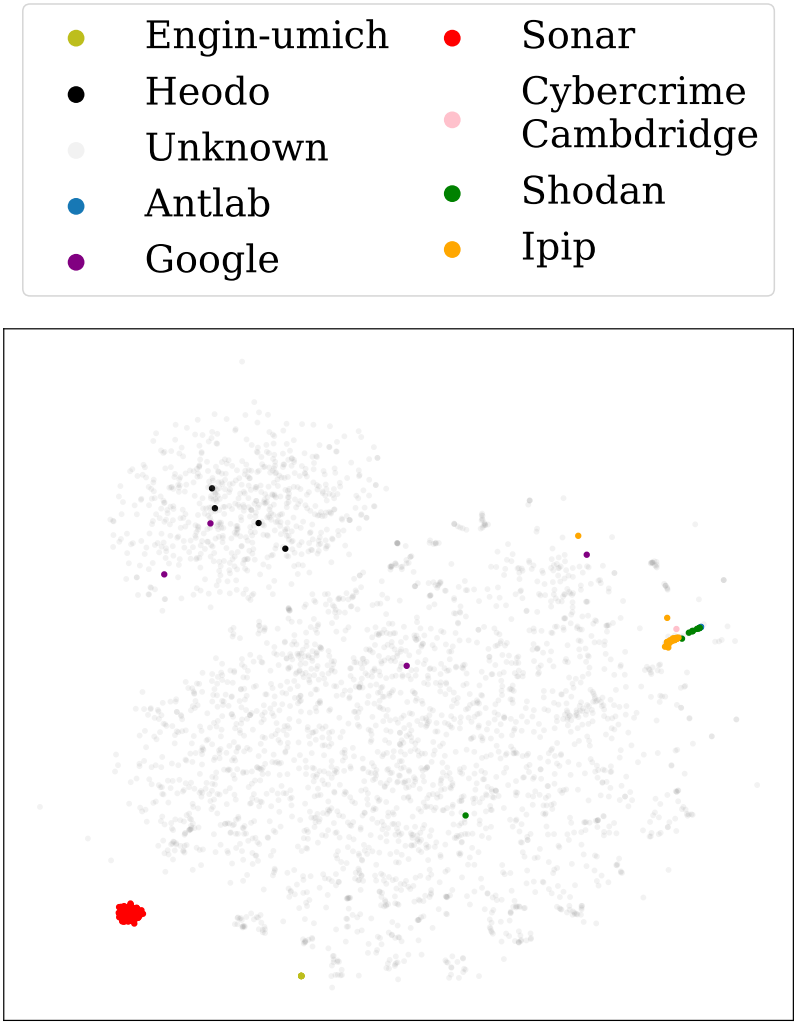


Figure 6.4: tSNE projection of temporal relationships features

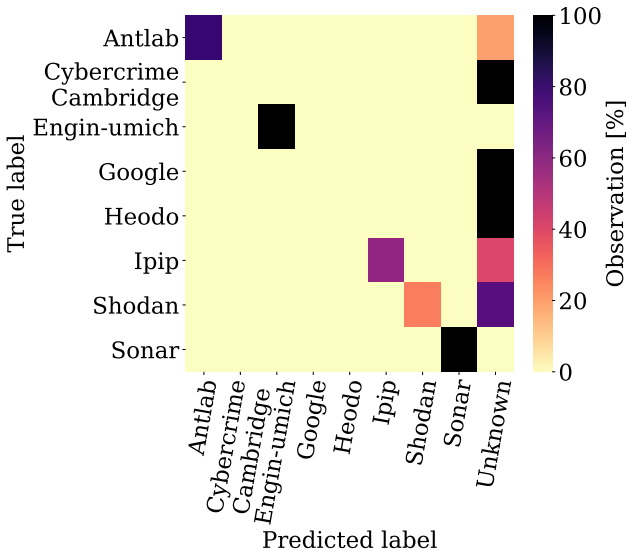


Figure 6.5: Temporal features. Accuracy: 98.67%

The above considerations are confirmed by Figure 6.5 which reports the confusion matrix resulting from the LOO-kNN classifier applied on ground truth. All the points of both Sonar and Engin-umich are correctly classified. Regarding the Antlab class, the performance is slightly worse than the traffic intensity case ($\approx$ 80% of correctly classified points compared to the 100% of the traffic intensity case). Even if it is possible to identify a temporal pattern from Figure 6.3a, there are still some IP addresses active only in the last days of the month used for word2vec training, thus probably the misclassified points fall into that case.

If the considerations about the Ipip points projection are confirmed by the $\approx$ 60% of correctly classified points, the outcomes of the Shodan class is not coherent with what has been said).

Finally, as in the traffic intensity case, the 100% of the Heodo, Google and Cybercrime Cambridge points fall in the neighborhood of the unknown points because of the small number of time units during which the IPs were active in the darknet generating traffic.

The average accuracy among all the ground truth classes active during 2020/04/05 is 98.67% indicating that the most of the a-priori-coordinated points belongs to the same neighborhood according to the GT class membership.

When evaluating the robustness of the nodes representation over time, it is necessary to recall the slightly different approach used for the temporal relationships features with respect to the traffic intensity ones. Indeed, since the word2vec model needs to be retrained any time new IPs are seen for the first time in the darknet, the embeddings generated for nodes active in more than one day are different after each re-training. Thus it is not possible to adopt the approach of Figure 3.4.

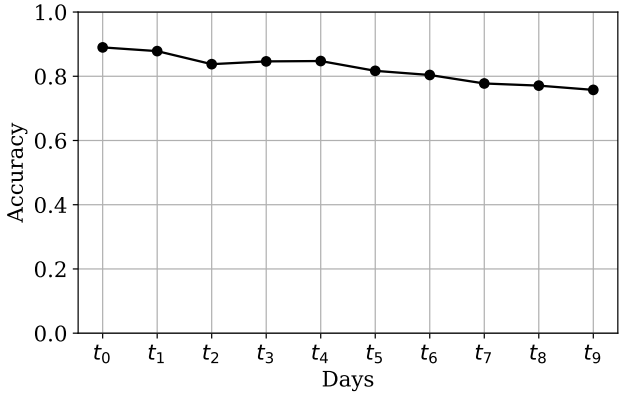


Figure 6.6: Temporal evolution of the kNN accuracy on full ground truth

Figure 6.6 shows the trend of the LOO-kNN accuracy over time applied to the full ground truth active in the day $t_i, i \in [0, 9]$.

Even the considered figure and Figure 5.4 are not directly comparable according to the different approach, it is possible to state that the temporal relationship features exhibit a satisfactory robustness. Indeed the accuracy trend tends to smoothly decay over time, but according to the different IPs activeness among different days, it increases during t_3 and t_4 reaching a minimum of $\approx 85\%$ after 10 day of traffic.

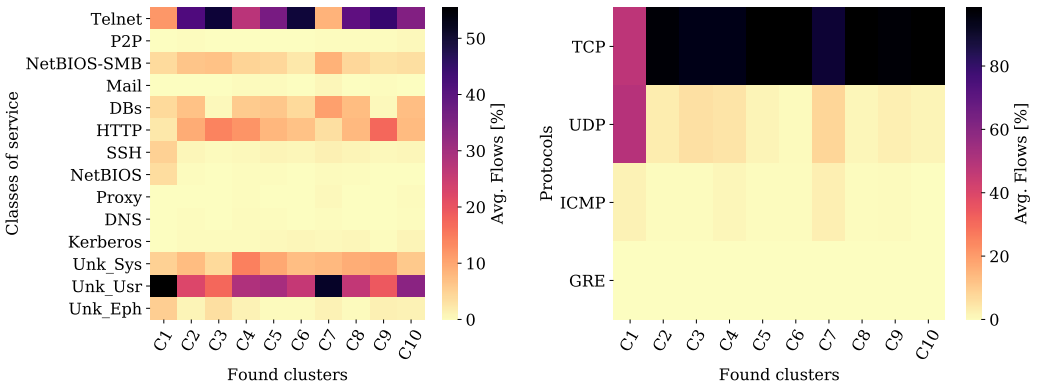
Undoubtedly the daily re-training of the word2vec model allows to generate new embeddings with fresher information keeping track of the temporal changes in source IPs temporal behavior, but the robustness of the approach is also provided by the more regular time patterns characterizing the ground truth classes under analysis.

6.6 Clustering on Single Day

To investigate if the chosen model and parameters are able to spot coordination among source IPs on the basis of their temporal relationships I run k-Means on the 2020/05/04 day.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Unknown	75	4130	10805	3111	12111	3969	313	4291	10444	2495
Ipip	-	-	-	47	-	-	2	-	-	-
Sonar	-	-	-	-	-	1	-	-	-	358
Antlab	-	-	-	5	-	-	-	-	-	-
Shodan	-	-	-	10	-	-	-	-	1	-
Engin-umich	-	10	-	-	-	-	-	-	-	-
Total	75	4140	10805	3173	12111	3970	315	4291	10445	2853

Table 6.3: Number of observations per found cluster. Temporal features dataset. Average Silhouette: 0.160



(a) Classes of service

(b) Protocols

Figure 6.7: Found clusters characterization. Traffic intensity features

Table 6.3 and Figure 6.7 report the outcome of the clustering algorithm application. Regarding the ground truth classes, Engin-umich and Sonar are well separated falling into two different clusters (C2 and C10 respectively). On the other hand, the remaining

GT classes are assigned to the same cluster C4 a part from some exceptions like C6, C7 and C9 containing mostly unknown points and < 3 ground truth ones.

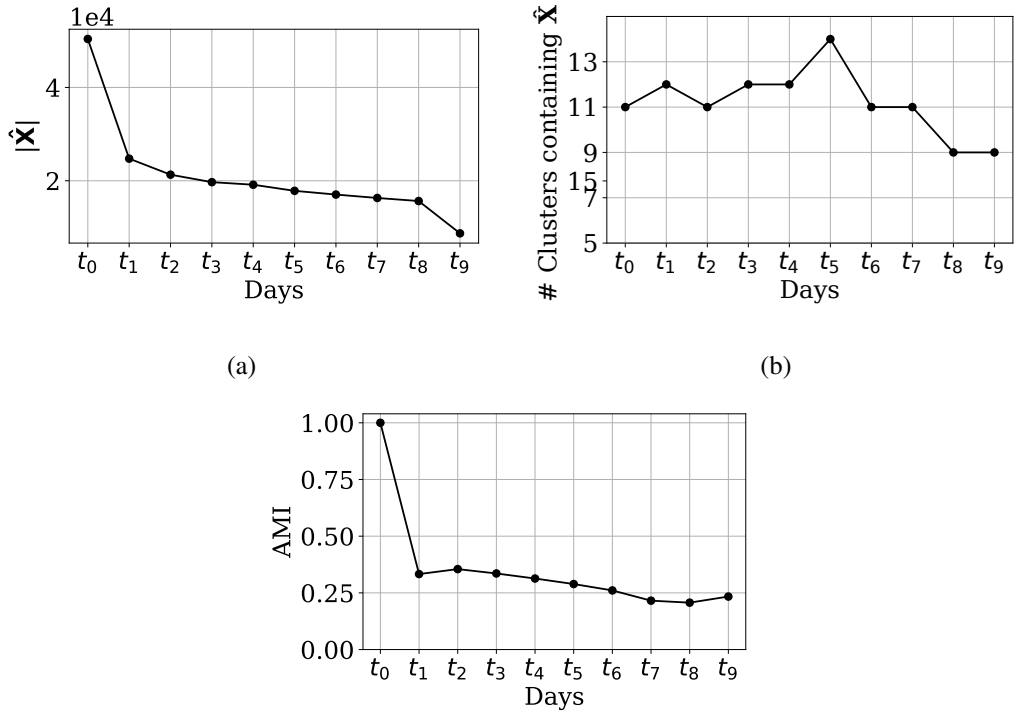
When investigating the allocation of the clusters total flow per class of service and protocols it is clear that the algorithm is not able to spot any coordination related to type of traffic. From Figure 6.7a the maximum percentage of flow directed to a particular class of service is < 60% meaning that none of the found clusters is characterized by a unique behavior. For example, C2 containing all the Engin-umich IPs is expected to direct all the traffic to the DNS class of service, but since the considered class is the 0.24% of the total C2 IPs, it results that the 40% of the cluster traffic reaches the Telnet service, whereas the DNS packets are < 5% of the total. Furthermore, when evaluating the percentage of cluster flow per protocol, the outcomes of Figure 6.7b is substantially different from the ones of Figure 5.5b. Indeed, when using the w2v embeddings, the clustering algorithm is not able to separate nodes sending packets through different protocols and, a part from C1, all the others Ci $i \in [2, 10]$ send the $\approx$ 80% of the traffic through TCP.

Though lower obtained performance with respect to the traffic intensity case is reasonable since the used features are generated according to the temporal relationship and there is no contribution of the traffic types features.

Finally, by considering that the achieved average silhouette value is 0.160 meaning that the found clusters are almost completely overlapping, it is possible to draw the consideration similar to the traffic intensity case: the clustering algorithm is not able to spot temporal coordination among source IPs although the generated features place in the same neighborhood nodes belonging to the same ground truth class.

6.7 Evolution of Clusters Membership

Like in the traffic intensity features I investigate the temporal evolution of clusters membership. The reference day t_0 is the 2020/05/01.



(a) (b)

(c)

Figure 6.8: Analysis of cluster membership evolution from t_0 to t_9. Temporal relationship features. (a) Source IPs propagation from t_0 to t_9. (b) Number of found clusters containing at least one node propagating from t_0 to t_9. (c) Temporal evolution of clusters membership

When evaluating the trend of the number of nodes active both in the reference day and in t_i with $i \in [1, \dots 9]$ shown in Figure 6.8a, again the expected drastic reduction between t_0 and t_1 is obtained. Furthermore, for $t_i > t_1$ the number of propagating points progressively decreases from ≈ 22000 to ≈ 17000. The shown decreasing trend of $|\hat{\mathbf{X}}|$ should result in more compact clusters and an increased AMI.

Despite what it would expect, the propagating points are assigned to a number of clusters (Figure 6.8b) oscillating between 14 (+27% of the reference day) and 9

(-18% of the reference day) and the obtained AMI oscillates between 0.3 and 0.24 (Figure 6.8c). By recalling the high accuracy obtained when evaluating the nodes representation, the obtained results suggest that, again, probably the chosen clustering algorithm is not well suited for the proposed problem failing in detecting coordination over time.

Chapter 7

Conclusion and Future Works

In the proposed work I define and investigate a methodology for automatically detecting coordinated source IPs among darknet traffic. The methodology relies on two hypotheses that define the concept of coordination: (i) coordinated IP addresses (e.g., belonging to a Botnet) should target both complementary and same Internet ports. (i) coordinated IP addresses should follow similar time patterns during Internet scans or attacks exhibiting strong temporal correlation between each others. From the first hypothesis the coordination is defined in terms of reached classes of services (Internet port and used protocol pairs) and generated traffic intensity.

According to the proposed methodology relying on clustering algorithm, my results highlight that the generated features are able to highlight similarity between IPs whose coordination is known a-priori. This is achieved by applying a Leave-One-Out kNN classifier to the generated ground truth dataset with both the traffic intensity and temporal relationships features obtaining an accuracy of 98.8% for both the cases. This means that, according to the source IPs representation, nodes belonging to the same ground truth class belongs also to the same neighborhood.

However the satisfactory achieved behavior is not consistent over time when using traffic intensity features. Indeed the 98.8% of accuracy achieved in the reference day, drop drastically down to $\approx$ 20% when evaluating the second day. The reason behind the described behavior is probably due to rapid changes in traffic volume and types of the IP addresses.

On the other hand, when using the embedding generated by the Word2Vec algorithm trained on 30 days of darknet traffic, a more robustness of the nodes representa-

tion is achieved with an accuracy that never goes below $\approx 75\%$ after 10 days of traffic analysis.

Even though the generated features seem promising, when applying unsupervised clustering algorithms with the relative proposed heuristics, it results that none of them is able to spot the highlighted coordination. Indeed, for both the features macro-types, values of silhouette close to 0 are achieved, meaning that the found clusters overlap between each others, or too few clusters are detected by the algorithms.

Furthermore, to evaluate if at least the nodes membership of the found clusters is maintained over time, I repeat the analysis for 10 days of darknet traffic. The obtained results are expressed in terms of AMI between the clusters found in the reference day and the ones found in the subsequent days. The outcomes reveal that when using traffic intensity features, the AMI is approximately constant around 50% with a number of found clusters floating in the [6, 11] range. On the other hand, when exploiting the temporal relationships among source IPs, the obtained AMI never reaches the 40% with a number of found clusters oscillating between 9 and 14 depending on the considered day.

The obtained results are perfectly reasonable when comparing the AMI trend with the one of the number of found clusters. Indeed, the higher AMI achieved with traffic intensity features is not an indication of good performance, since at most 11 clusters are detected among 18000 nodes. This means that if the nodes distribution among clusters was homogeneous, each cluster would be made of $\approx$ 1636 nodes on average. By considering that in the case under analysis the number of points per cluster is uneven, it is clear that biggest groups are spotted, thus it is highly likely that nodes fall in the same group.

What has been said is coherent even with temporal relationship features. Indeed the number of found clusters is greater than the the one of the traffic intensity case and, consequently, the obtained AMI decreases.

In light of the above it is possible to answer to the work RQs in the following way:

RQ1.2. Is it possible to exploit the amount and type of traffic generated toward darknets to detect coordination among source IPs? According to the obtained results it is possible to state that the amount and type of traffic characterizing the source IPs and the found group allows only to highlight strongly different behaviors constant

over time like ICMP routine scans with respect to GRE one. On the other hand, when nodes generate TCP and UDP traffic reaching multiple destination ports, the traffic volume cannot be used to detect coordination among IPs because of the rapid changes which make the proposed algorithm fail.

RQ1.3. Is it possible to exploit the temporal relationships among source IPs active in darknets to detect coordination? With respect to traffic volume features, the approach based on the Word2Vec algorithm is more promising. Indeed the performance indicators are more robust not only in a single day, but also over time highlighting the coordination among nodes belonging to the same research projects. However, even if the obtained metrics have higher values than the traffic intensity case, probably the tested clustering algorithms are not able to fully exploit the nodes distance embedded in the dataset and a deeper investigation is required.

Q1.4. If the developed model is able to identify coordination, is it maintained over time? The previous research questions partially answer to this last one. Indeed, the developed model is not robust at all when analyzing the traffic evolution over time, despite the good performance achieved from the LOO kNN classifier. Indeed, when considering the temporal relationships among nodes, the Word2Vec algorithm maintains satisfactory performance $> 70\%$ over 10 days of traffic, even though the clustering algorithm is not able to fully exploit the features potential failing in detecting consistent clusters after one day of analysis.

According to the highlighted weakness of the proposed methodology, as future works, the algorithm performance could be improved by investigating semi-supervised or unsupervised clustering techniques relying on Deep Neural Networks, like the one of [45, 46]. In this way it should be possible to overcome the limitation of the adopted algorithms which are not able to spot the coordination that Word2Vec embeddings can highlight.

Regarding the most promising technique, the Word2Vec algorithm, in this thesis I performed a simple grid search for improving the standard off-the-shelf configurations. However the embedding potential should be deeper investigated and optimized. An example of the future methodology to follow is proposed in [47], where different word embedding models are compared evaluating the consistency of the points neigh-

borhood over different model configurations.

Finally, once the new algorithms will be tested and evaluated, an interesting future development could be the implementation of a framework for constituting a self-learning knowledge-base system to track all the past threat and highlight new suspicious coordinated source IPs following the methodology proposed in [48], even though the context of application is different with respect to the discussed problem and scenario.

Appendix A

Methodology Background

A.1 Features Generation for Multigraph

Even though the thesis is based on clustering algorithms, in the development phase different approaches have been tested.

The original idea was to represent data through an undirected multigraph $G = (E, V)$, where E is the set of nodes and V is the set of edges. In this case an edge is a link between each pair of nodes expressing a certain degree of behavior similarity. Each node is represented by a set of attributes, or nodes features, which are the ones described in Table 3.4. Each feature is embedded in a layer of the multigraph and a weight is associated to each link embedding the nodes features.

Whereas in the clustering approach the nodes relationships are embedded in the distance matrix **D**, in the multigraph one it is necessary to explicitly define them as link features.

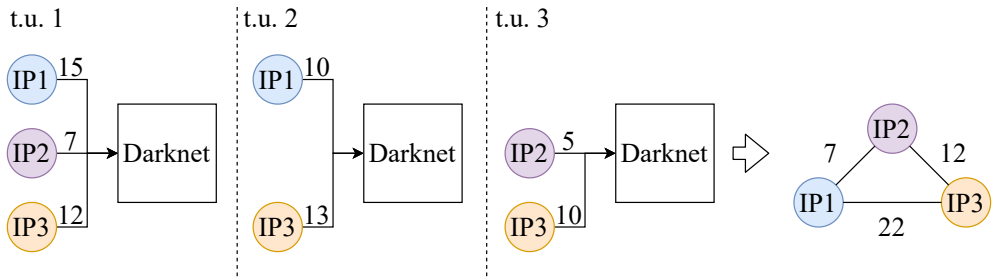


Figure A.1: Link features generation example. Minimum common flow

Figure A.1 provides an example of the minimum common flow feature generation. Firstly the daily observation window is splitted in 24 hours time units. Then, the minimum flow generated to the darknet between each pair of nodes active in the considered t.u. is obtained. The final link weight is the sum of the calculated minima.

The same rationale is applied to each nodes feature and a full overview of the link ones is provided by Table A.1.

Type	**Feature**	**Description**	**# of Features**
General	time	Number of common hours of activity between each nodes pair	1
	min_flows	Minimum common flow between each nodes pair	1
Protocol	*_min_flows	Minimum common flow between each nodes pair through each protocol of Table 3.2	4
TCP flags	*_min_flows	Minimum common flow between each nodes pair with each TCP flag of Table 3.2 set	5
Serivices	*_min_flows	Minimum common flow between each nodes pair sent to the ports used by each service of Table 3.3	15
	*_max_bytes	Maximum common bytes between each nodes pair sent to the ports used by each service of Table 3.3	15

Table A.1: Generated link features summary

According to the scalability limits of the multigraph approach and to the arbitrary choice of the criterion used in the link features generation process (maxima or minima), the clustering approach is found to be a more robust and promising one.

A.2 Clustering Quality Metrics

A.2.1 Silhouette (Sh)

By assuming that nodes have been clustered in k clusters, for each node i in the cluster C_i, the intra-cluster distance is defined as

$$a_i = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j) \tag{A.1}$$

where $d(i, j)$ is the distance between node i and node $j \in C_i$. Then the nearest-cluster distance for each node i is defined as

$$b_i = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j) \tag{A.2}$$

In this way, a_i is a measure of how well node i is assigned to a cluster in terms of distance with respect to all the other points belonging to the same cluster, whereas b_i expresses a dissimilarity of the node i with respect to all the points belonging to the cluster $C_k \neq C_i$ nearest to C_i.
Now the clustering validation metric Silhouette can be defined as

$$Sh = \frac{\overline{a} - \overline{b}}{\max(a, b)} \tag{A.3}$$

where $\overline{a}$ is the mean intra-cluster distance among all the samples and $\overline{b}$ is the mean nearest-cluster distance. In this way, $-1 \leq Sh \leq 1$. The best Silhouette value is 1, meaning that all the clusters are separated; the worst value is -1 and Sh=0 means overlapping clusters.

A.2.2 Adjusted Mutual Information (AMI)

Adjusted Mutual Information between two clusterings is an adjusted measure of their mutual dependence, or Mutual Information, to account the MI unbalancing occurring when two clustering are made of a large number of clusters.

Given a clustering U with R clusters and a clustering V with C clusters, the Adjusted Mutual Information between U and V is defined as:

$$AMI(U, V) = \frac{MI(U, V) - E\{MI(U, V)\}}{\max\{H(U), H(V)\} - E\{MI(U, V)\}} \tag{A.4}$$

where $MI(U, V)$ is the mutual information between U and V, $H(U)$ and $H(V)$ are the entropy values associated to U and V, and $E\{MI(U, V)\}$ is the adjustment for chance. The more the AMI tends to 1, the more identical the clusters

A.3 Heuristics for Hyper-parameters of Clustering

A.3.1 DBSCAN

Regarding the two DBSCAN parameters minPts and ϵ two different strategies have been used. The methodology is guided by the original DBSCAN documentation [34, 35].

- minPts. In the case study proposed by the algorithm developers, minPts have been set equal to a fix value. Nevertheless, since the dataset of the proposed thesis is characterized by a high number of samples, a slightly flexible strategy is adopted by setting

$$minPts = \lfloor ln(N) \rfloor \tag{A.5}$$

  where N is the number of samples;

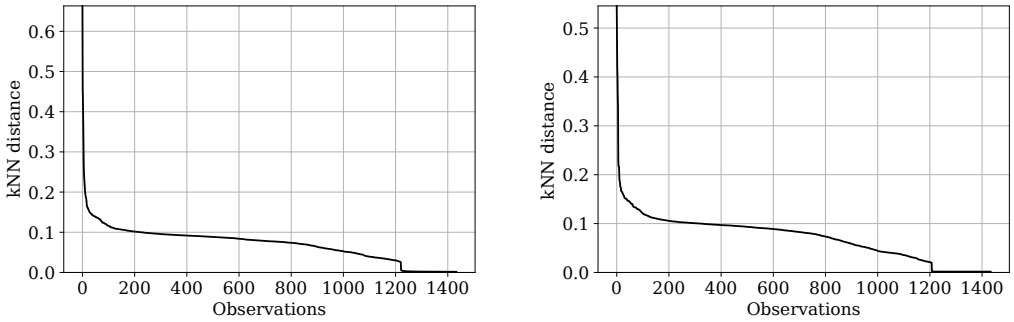
- ϵ. Regarding the neighborhood radius, the *elbow method* is used. It is based on the bservation that by considering d as the distance of a node n to its *k-th* nearest neighbor, then the d-neighborhood of n contains $k + 1$ distinct points. Thus, by considering $k = minPts$, where $minPts$ is determined through Equation A.5, the function *k-dist(n)* is defined as the array of distances between n and its k nearest neighbors sorted in ascending order. At this point, the so called *sorted k-dist graph* is obtained in the following way:

  1. Compute the *k-dist* function for all the points in the dataset;
  2. For each obtained distance vector take the last entry corresponding to the distance between n and its furthest *k-th* nearest neighbor;
  3. Sort the considered distance for each point in descending order.

  In this way the *sorted k-dist graph* provides information about the density of the distances between each dataset points and their furthest *k-th* nearest neighbor.

For a given k and a given node n, all the points that have $d \leq k\text{-}dist(n)$ are core points for the cluster n belongs to.

Since the graph is sorted in descending order, it is possible to consider the point of graph maximum curvature, or *elbow*, as a threshold indicating the maximum *k-dist* value used in the sparsest cluster. Thus the distance corresponding to the elbow point can be used as ϵ.



(a) Traffic intensity features

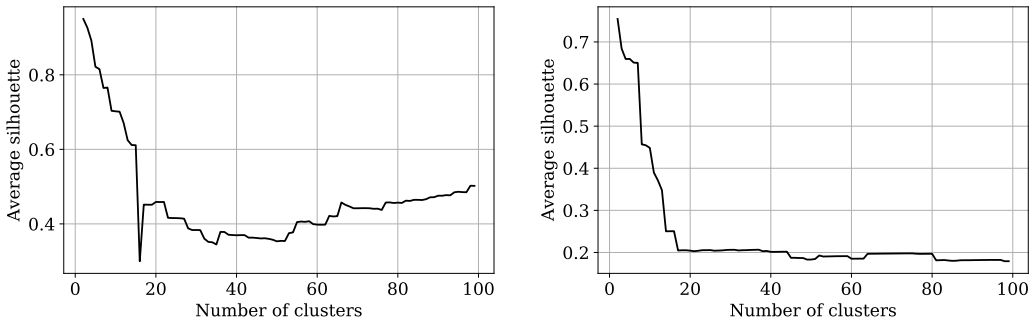
(b) Word2Vec embeddings

Figure A.2: Results of the DBSCAN heuristic based on the *k-dist* plot. The results are referred to the model development dataset

In Figure A.2 I report the results obtained through the DBSCAN heuristic based on the elbow method on the *k-dist* plot.

A.3.2 HAC

To optimize the number of clusters that the Hierarchical Agglomerative Clustering algorithm must recognize I use a heuristic based on the average silhouette (Appendix A.2.1). It consists of iteratively applying the clustering algorithm providing an increased number of cluster to be found. Then I compute the silhouette resulting from each run and generate the Sh trend. The optimal number of clusters is the one resulting in the greatest silhouette value.



(a) Traffic intensity features

(b) Word2Vec embeddings

Figure A.3: Results of the HAC heuristic based on the average silhouette trend. The results are referred to the model development dataset

In figure A.3 I report the HAC heuristic results obtained during the model development stage.

A.3.3 k-Means

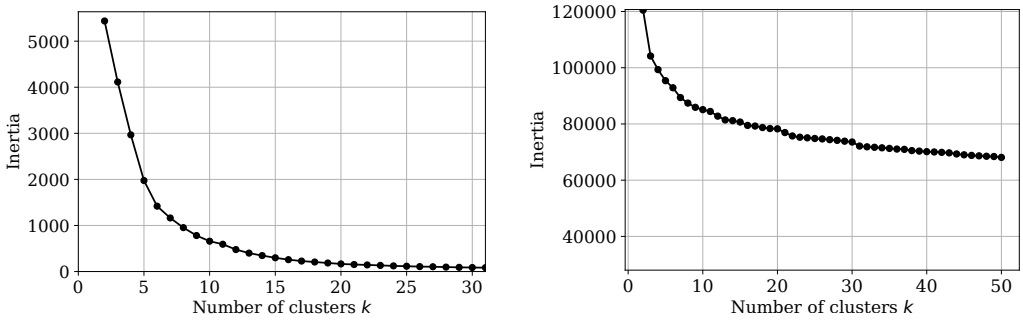
Like the algorithm, the k-Means heuristic for the optimal k is strictly related to the *inertia*, or the within-cluster sum-of-squares. Basically the algorithm is iteratively applied with an increasing number of clusters k. After each run the inertia is computed and the relative plot is generated. The final k is the one at which the elbow of the inertia plot occurs.

In figure A.4 I report the k-Means heuristic results obtained during the model development stage through the elbow method on inertia.

A.4 Discarded Filtering Attempt

As a first attempt of data filtering I adopted a simulation-based approach. The result reported here have not been included in the work since the methodology relies on graph-based features described in Appendix A.1.

The filtering approach is based on a statistical hypothesis test. Basically, after having generated the traffic features, even with the multigraph-based methodology, the data characterization leads to considerations analogous to Figure 4.1a: the most of the



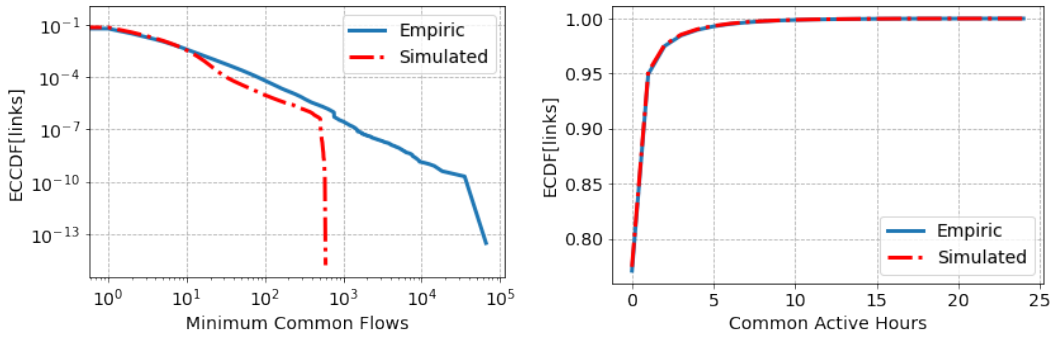
(a) Traffic intensity features

(b) Word2Vec embeddings

Figure A.4: Results of the HAC heuristic based on the inertia trend over different k. The results are referred to the model development dataset

traffic is generated by few IPs. Thus I extract the ECDF of the minimum common flows between nodes and the one of the hours within the source IP pairs were contemporaneously active in the darknet. Then I formulate the hypothesis that is all the seen source IPs act randomly with no coordination.

At this point I run the simulation by randomly assigning a number of common flows per each source IP pairs on the basis of the empirical distribution function and by calculating the new simulated distribution function. In this way, the portion of the ECDF below or overlapping with the simulated one can be filtered out, since the nodes are actually behaving randomly.



(a) Minimum common flows features

(b) Common active hours

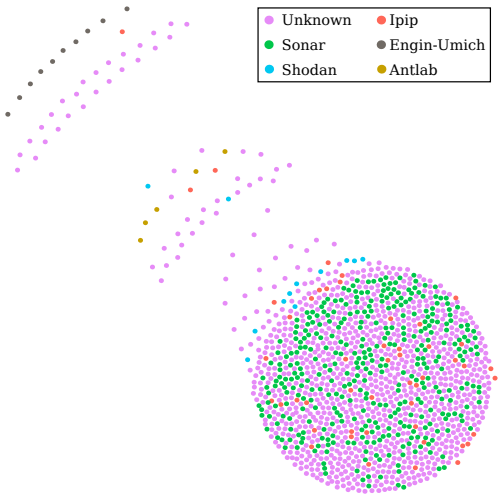
Figure A.5: Comparison between simulated and empirical ECDF for two multigraph-features used during the discarded filtering approach

Figure A.5 reports the simulated and empirical ECDFs comparison. From Figure A.5a it is reasonable to discard node pairs whose minimum number of common flows daily flows is ≤ 10, whereas the common active hours probably are not useful features, since the ECDF overlaps with the simulated one, meaning that the nodes activeness in darknet is completely random.

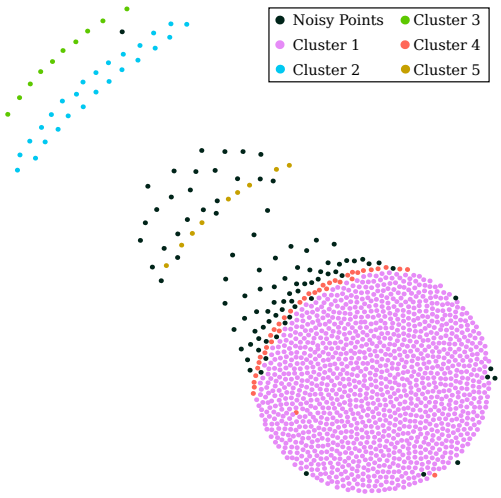
The proposed approach has been discarded according to the change in the methodology more oriented towards clustering. Even if a similar approach can be adopted in the proposed methodology, I focus more on the model development and on the results discussion by dropping only nodes generating less than two flows in a day.

A.5 Gephi Visualization Attempt

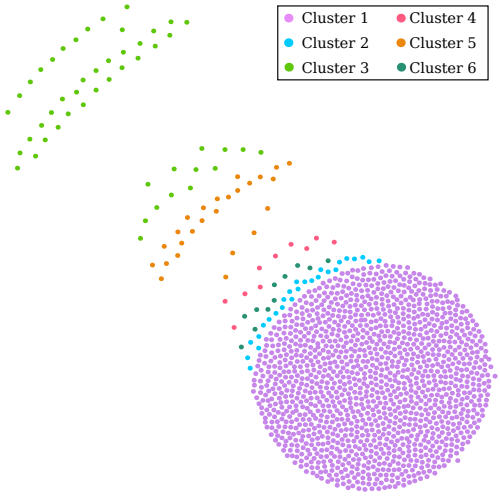
As an additional attempt of visualizing source IPs in the space before and after clustering I tried an approach based on Gephi [49]. It consists on starting from the distance matrix used in clustering algorithms and to turn it into a similarity matrix. The obtained matrix is then used to generate an undirected graph which is imported in Gephi. At this point, I obtain the nodes spatial representation through the Force Atlas 2 algorithm [50] which set the nodes relative position as proportional to their similarity (e.g. similar nodes are placed close to each other). In Figure A.6 I report an example of the proposed visualization method. The results are referred to intermediate development stages with features that are no longer included in the dataset. However I decided to discard the proposed approach adopting the aforementioned tSNE which is less time consuming and more customizable in terms of hyperparameters.



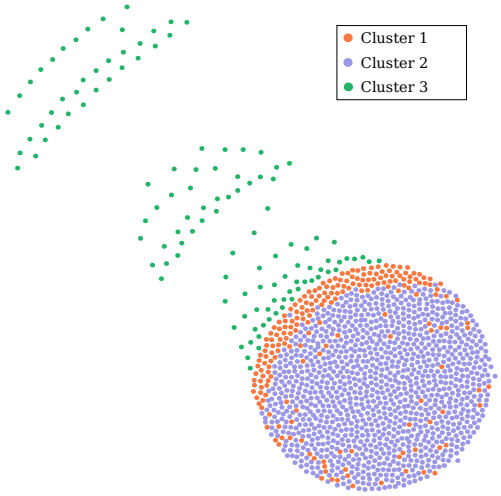
(a) Original GT classes



(b) DBSCAN clusters



(c) HAC clusters



(d) GMA clusters

Figure A.6: Nodes visualization attempt through Gephi after clustering

Bibliography

[1] F. Gadhia, J. Choi, B. Cho, and J. Song. Comparative analysis of darknet traffic characteristics between darknet sensors. In *2015 17th International Conference on Advanced Communication Technology (ICACT)*, pages 59–64, July 2015.

[2] Francesca Soro, Idilio Drago, Martino Trevisan, Marco Mellia, João Ceron, and José J Santanna. Are darknets all the same? on darknet visibility for security monitoring. In *2019 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, pages 1–6. IEEE, 2019.

[3] J. Fruhlinger. The mirai botnet explained: How teen scammers and cctv-cameras almost brought down the internet, 03 2018. `https://www.csoonline.com/article/3258748/\the-mirai-botnet-explained-\how-teen-scammers-and-cctv-cameras-almost-brought-down-the\-internet.html`.

[4] K. Nakao. Proactive cyber security response by utilizing passive monitoring technologies. In *2018 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–1, 2018.

[5] F. Shaikh, E. Bou-Harb, J. Crichigno, and N. Ghani. A machine learning model for classifying unsolicited iot devices by observing network telescopes. In *2018 14th International Wireless Communications Mobile Computing Conference (IWCMC)*, pages 938–943, 2018.

[6] T. Ban and D. Inoue. Practical darknet traffic analysis: Methods and case studies. In *2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big*

Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), pages 1–8, 2017.

[7] P. S. Joshi and H. A. Dinesha. Survey on identification of malicious activities by monitoring darknet access. In *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pages 346–350, 2020.

[8] Uli Harder, Matt W. Johnson, Jeremy T. Bradley, and William J. Knottenbelt. Observing internet worm and virus attacks with a small network telescope. *Electronic Notes in Theoretical Computer Science*, 151(3):47–59, 2006. Proceedings of the Second International Workshop on the Practical Application of Stochastic Modeling (PASM 2005).

[9] Zhenxin Zhan, Maochao Xu, and Shouhuai Xu. A characterization of cybersecurity posture from network telescope data. In Moti Yung, Liehuang Zhu, and Yanjiang Yang, editors, *Trusted Systems*, pages 105–126, Cham, 2015. Springer International Publishing.

[10] David Moore, Geoffrey M. Voelker, and Stefan Savage. Inferring internet denial-of-service activity. In *10th USENIX Security Symposium (USENIX Security 01)*, Washington, D.C., August 2001. USENIX Association.

[11] B. Irwin. A baseline study of potentially malicious activity across five network telescopes. In *2013 5th International Conference on Cyber Conflict (CYCON 2013)*, pages 1–17, 2013.

[12] Alberto Dainotti, Roman Amman, Emile Aben, and Kimberly C. Claffy. Extracting benefit from harm: Using malware pollution to analyze the impact of political and geophysical events on the internet. *SIGCOMM Comput. Commun. Rev.*, 42(1):31–39, January 2012.

[13] Francesca Soro, Mauro Allegretta, Marco Mellia, Idilio Drago, and Leandro M Bertholdo. Sensing the noise: Uncovering communities in darknet traffic. In *2020 Mediterranean Communication and Computer Networking Conference (MedComNet)*, pages 1–8. IEEE, 2020.

[14] S. Lagraa and J. François. Knowledge discovery of port scans from darknet. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 935–940, 2017.

[15] Sofiane Lagraa, Yutian Chen, and Jérôme François. Deep Mining Port Scans from Darknet. *International Journal of Network Management*, 29(3):e2065, February 2019.

[16] A. Bar, B. Shapira, L. Rokach, and M. Unger. Identifying attack propagation patterns in honeypots using markov chains modeling and complex networks analysis. In *2016 IEEE International Conference on Software Science, Technology and Engineering (SWSTE)*, pages 28–36, 2016.

[17] I. Škrjanc, S. Ozawa, D. Dovžan, B. Tao, J. Nakazato, and J. Shimamura. Evolving cauchy possibilistic clustering and its application to large-scale cyberattack monitoring. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7, 2017.

[18] C. Fachkha, E. Bou-Harb, and M. Debbabi. Towards a forecasting model for distributed denial of service activities. In *2013 IEEE 12th International Symposium on Network Computing and Applications*, pages 110–117, 2013.

[19] E. Bou-Harb, M. Debbabi, and C. Assi. On detecting and clustering distributed cyber scanning. In *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 926–933, 2013.

[20] E. Bou-Harb, M. Debbabi, and C. Assi. A time series approach for inferring orchestrated probing campaigns by analyzing darknet traffic. In *2015 10th International Conference on Availability, Reliability and Security*, pages 180–185, 2015.

[21] Dvir Cohen, Yisroel Mirsky, Manuel Kamp, Tobias Martin, Yuval Elovici, Rami Puzis, and Asaf Shabtai. Dante: A framework for mining and monitoring darknet traffic, 09 2020.

[22] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[23] Teuvo Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, 1995.

[24] Tomas Mikolov, G.s Corrado, Kai Chen, and Jeffrey Dean. Efficient estimation of word representations in vector space, 01 2013.

[25] K. Naidu, A. Dhenge, and K. Wankhade. Feature selection algorithm for improving the performance of classification: A survey. In *2014 Fourth International Conference on Communication Systems and Network Technologies*, pages 468–471, 2014.

[26] A. H. Shahana and V. Preeja. Survey on feature subset selection for high dimensional data. In *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, pages 1–4, 2016.

[27] S. Khalid, T. Khalil, and S. Nasreen. A survey of feature selection and feature extraction techniques in machine learning. In *2014 Science and Information Conference*, pages 372–378, 2014.

[28] Evelyn Fix and J. L. Hodges. Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review / Revue Internationale de Statistique*, 57(3):238–247, 1989.

[29] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[30] G. N. Lance and W. T. Williams. Computer Programs for Hierarchical Polythetic Classification ("Similarity Analyses"). *The Computer Journal*, 9(1):60–64, 05 1966.

[31] G. Lance and W. T. Williams. Mixed-data classificatory programs i - agglomerative systems. *Aust. Comput. J.*, 1:15–20, 1967.

[32] J. Roger Bray and J. T. Curtis. An ordination of the upland forest communities of southern wisconsin. *Ecological Monographs*, 27(4):325–349, 1957.

[33] ELI MAOR. *The Pythagorean Theorem: A 4,000-Year History*. Princeton University Press, 2007.

[34] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. Dbscan revisited, revisited: Why and how you should (still) use dbscan. *ACM Trans. Database Syst.*, 42(3), July 2017.

[35] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[36] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. of Berkeley Symposium on Math. Statist. and Prob.*, pages 281–297, 1967.

[37] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, oct 2008.

[38] Project sonar.
https://www.rapid7.com/research/project-sonar/.

[39] The only ip database based on real time bgp/asn data analytics.
https://en.ipip.net/.

[40] Shodan, the search engine.
https://www.shodan.io/.

[41] Antlab, advanced network technologies laboratory.
http://www.antlab.polimi.it/.

[42] Michigan engineering - university of michigan.
https://www.engin.umich.edu/.

[43] University of cambridge, computer laboratory.
https://www.cambridgecybercrime.uk/.

[44] J. Xavier. Emotet: Why did the 'most wanted' malware go on a 5-month hiatus?, 08 2020. https://www.thehindu.com/sci-tech/technology/

emotet-why-did-the-most-wanted-malware-go-on-a-5-month-hiatus/article32443939.ece.

[45] Yazhou Ren, Kangrong Hu, Xinyi Dai, Lili Pan, Steven C.H. Hoi, and Zenglin Xu. Semi-supervised deep embedded clustering. *Neurocomputing*, 325:121–130, 2019.

[46] Abu Quwsar Ohi, M.F. Mridha, Farisa Benta Safir, Md. Abdul Hamid, and Muhammad Mostafa Monowar. Autoembedder: A semi-supervised dnn embedding system for clustering. *Knowledge-Based Systems*, 204:106190, 2020.

[47] Lucas Rettenmeier. Word embeddings: Stability and semantic change. *ArXiv*, abs/2007.16006, 2020.

[48] A. Morichetta and M. Mellia. Lenta: Longitudinal exploration for network traffic analysis from passive data. *IEEE Transactions on Network and Service Management*, 16(3):814–827, 2019.

[49] Gephi, the open graph viz platform. https://gephi.org/.

[50] Mathieu Jacomy, Tommaso Venturini, Sebastien Heymann, and Mathieu Bastian. Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software. *PLOS ONE*, 9(6):1–12, 06 2014.