universidad de león

UNIVERSITY OF LEÓN

DEPARTMENT OF ELECTRICAL, SYSTEMS AND AUTOMATIC ENGINEERING

SUPERVISED MACHINE LEARNING FOR CLASSIFICATION, MINING, AND RANKING OF ILLEGAL WEB CONTENTS

A dissertation supervised by

PROF. DR. ENRIQUE ALEGRE GUTIÉRREZ,

DR. EDUARDO FIDALGO FERNÁNDEZ

and submitted by

MHD WESAM AL-NABKI

in fulfillment of the requirements for the Degree of

PHILOSOPHIÆDOCTOR (PH.D.)

León, November 2019

universidad de león

UNIVERSIDAD DE LEÓN

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA Y DE SISTEMAS Y AUTOMÁTICA

APRENDIZAJE AUTOMÁTICO SUPERVISADO PARA LA CLASIFICACIÓN, EXTRACCIÓN DE CONOCIMIENTO Y ORDENACIÓN DE CONTENIDOS WEB ILEGALES

Tesis doctoral dirigida por

EL PROF. DR. ENRIQUE ALEGRE GUTIÉRREZ,

Y EL DR. EDUARDO FIDALGO FERNÁNDEZ

y desarrollada por

MHD WESAM AL-NABKI

a fin de optar al grado de

DOCTOR POR LA UNIVERSIDAD DE LEÓN DEL PROGRAMA DE INGENIERÍA DE PRODUCCIÓN Y COMPUTACIÓN

León, Noviembre de 2019

Abstract

In this thesis, we propose new algorithms, methods, and datasets that can be used to classify, to mine information, and rank web domains or similar resources containing text. Motivated by our joint work with INCIBE, we focus our efforts on detecting web resources which content could indicate illegal activities. Most of these textual web pages are hosted in a darknet, and, because of that, we centered our analysis in The Onion Router (Tor) Darknet, based on the common belief that this net hosts plenty of criminal activities. Additionally, we also addressed the same problem in Online Notepad Services (ONS), in particular, Pastebin service.

Several of the contributions that we present here are already incorporated in tools developed by INCIBE that help Spanish Law Enforcement Agencies (LEAs) to monitor the contents of the Tor Darknet. Our work relies on the application of machine learning, both classical and deep, using most of the time supervised learning. This approach required the creation of different datasets, naming the first of them as Darknet Usage Text Addresses (DUTA), which contained 6,831 labeled samples distributed over 26 classes. Posteriorly, we extended this dataset up to 10,367 samples, naming it as DUTA-10K.

Using DUTA, we evaluated the combination of two text representation techniques with three well-known classifiers to categorize the Tor domains. The combination of TF-IDF words representation with Logistic Regression achieved a 93.7% macro F1 score, in a subset of DUTA where eight categories of illegal activities were selected. To classify Pastebin contents, we use Active Learning to select and label only the most informative samples, reducing in this way, the cost of building a labeled dataset. Our design requires three cascade classifiers, saying the last one whether a sample belongs to one out of six categories related to criminal activities, obtaining an average class recall of 95.24% as binary, and 80.33% as multiclass.

To enrich the information that we provide to LEAs, we develop first a semi-automatic algorithm to identify emerging products in Tor marketplaces. Using Graph Theory, we build a Products Correlations Graph (PCG), in which the nodes are the markets' products, and the edges reflect the simultaneous offering of two products in the same market. Our algorithm decomposes the PCG, using the k-shell algorithm, and analyzes the connectivity of the products in the core-shell. We apply this method to drug Hidden Services (HS)

in DUTA, finding that MDMA and Ecstasy were the most emerging drug products during the analyzed period. Second, we used Named Entity Recognition (NER) to recognize rare and emerging named entities in noisy user-generated text. We overcome the use of gazetteers to incorporate external resources to neural network architectures, presenting a novel feature that we named Local Distance Neighbor (LDN), obtaining in this way the state-of-the-art F1 score on three categories of the W-NUT-2017 dataset: Group, Person, and Product. Furthermore, we present an application of NER to the domains of the Tor network by extending the samples of the W-NUT-2017 dataset with 851 manually annotated entries, naming this modified version of the dataset as Noisy User-generated Text on Tor (NUToT). Our model obtained an entity and a surface F1 scores of 52.96% and 50.57%, respectively, beating the current state-of-the-art.

Our third area of interest is to detect which of the onion domains are the most influential, both inside its category and the whole Tor Darknet. Our first solution is based on the use of ToRank, a novel algorithm based on Graph Theory, which outperforms three well-known alternatives used in the Surface Web when ranking Tor hidden services: PageRank, HITS, and Katz. ToRank creates a higher disruption to the Tor network connectivity than any other alternative, which indicates its superiority for this problem. After evaluating ToRank, we analyze DUTA-10K to reveal that 20% of the samples are related to suspicious activities, while 48% are associated with normal ones, considering the 32% left are unknown because they were inaccessible. We also discover that domains related to suspicious activities usually present multiple clones under different addresses, which could be used as an additional feature for identifying them.

Our last contribution is a proposal based on the analysis of the domain content, considering features from textual, HTML, NER, links related, and visual information. We compared three supervised ranking strategies using Learning to Rank (LtR) to detect the most influential onion domains. We represent an onion domain by 40 features, and we learn how to rank the domains using an LtR approach. We evaluate our proposal on a case-study of drug-related onion domains, founding that among the explored LtR schemes, the listwise approach outperforms the benchmarked methods with an $NDCG@10$ of 0.95. Our findings proved that the content-based ranking is superior to link-based algorithms and that features extracted from the user-visible textual content are the better choice when balancing the cost of its extraction and the precision reached.

Resumen

En esta tesis, proponemos nuevos algoritmos, métodos y conjuntos de datos que se pueden usar para clasificar, extraer información y ordenar, en base a su importancia, dominios web o recursos similares que contienen texto. Motivados por nuestro trabajo conjunto con INCIBE, centramos nuestros esfuerzos en detectar recursos web cuyo contenido podría indicar actividades ilegales. La mayoría de estas páginas web están alojadas en una darknet y, debido a eso, analizamos The Onion Router (Tor) Darknet. Adicionalmente, abordamos el mismo problema en los Servicios de Bloc de Notas en línea (ONS, del inglés Online Notepad Services), en particular, el servicio Pastebin. En este caso, antes de clasificar, necesitamos desarrollar estrategias específicas para separar primero el texto legible del código fuente y de otro tipo de contenido.

Varias de las contribuciones que presentamos aquí ya están incorporadas en herramientas desarrolladas por INCIBE que ayudan a las agencias policiales españolas (LEA) a monitorear el contenido de la Darknet Tor. Nuestro trabajo se basa en la aplicación de aprendizaje automático, tanto clásico como profundo, utilizando la mayoría de las veces aprendizaje supervisado. Este enfoque requirió la creación de diferentes conjuntos de datos, nombrando al primero de ellos como Direcciones de Texto de Uso de Darknet (DUTA), conteniendo este dataset 6,831 muestras etiquetadas distribuidas en 26 clases. Posteriormente, ampliamos este conjunto de datos hasta 10,367 muestras, nombrándolo DUTA-10K.

Usando DUTA, evaluamos la combinación de dos técnicas de representación de texto con tres clasificadores bien conocidos para clasificar los dominios Tor. La combinación de TF-IDF con Regresión logística logró un el mejor valor macro F1, 93,7 %, en un subconjunto de DUTA donde se seleccionaron ocho categorías de actividades ilegales. Para clasificar los contenidos de Pastebin, utilizamos Active Learning para seleccionar y etiquetar solo las muestras más informativas, reduciendo así el coste de construir un conjunto de datos etiquetado. Nuestro diseño propone tres clasificadores en cascada, indicando el último si una muestra pertenece a una categoría delictiva, obteniendo un recall medio por clase de 95,24 % como binario, y 80,33 % cuando asignaba la muestra a una de las seis categorías seleccionadas.

Para enriquecer la información que proporcionamos a las LEAs, desarrollamos un pri-

mer algoritmo semiautomático para identificar productos emergentes en los mercados de Tor. Utilizando la teoría de gráficos, creamos un gráfico de correlaciones de productos (PCG), en el que los nodos son los productos de los mercados, y las aristas reflejan que dos productos se ofrecen de forma simultánea en el mismo mercado. Nuestro algoritmo descompone el PCG, utilizando el algoritmo k-shell, y analiza la conectividad de los productos en el núcleo-shell. Aplicando este método a páginas de DUTA de la categoría "drogas", descubrimos que MDMA y Ecstasy fueron los drogas más emergentes durante el período analizado. En segundo lugar, utilizamos el Reconocimiento de Entidades con Nombre (NER) para reconocer entidades raras y emergentes en texto ruidoso generado por el usuario. Mejoramos el uso de gazeteers para incorporar recursos externos en las arquitecturas de redes neuronales, presentando una característica novedosa que llamamos Vecino de Distancia Local (LDN), estableciendo de este modo un valor de F1 del estado del arte en tres categorías de W-NUT-2017: Grupo, Persona y Producto. Además, presentamos una aplicación de NER a los dominios de la red Tor extendiendo las muestras del conjunto de datos W-NUT-2017 con 851 entradas anotadas manualmente, nombrando esta nueva versión como NUToT. Nuestro modelo obtuvo un valor F1 tanto de entidad como de superficie de 52,96 % y 50,57 %, respectivamente, superando de esta manera al estado actual de la técnica.

Nuestra tercer área de interés es detectar los dominios en Tor que son más influyentes, tanto dentro de su categoría como en toda la red oscura Tor. Nuestra primera solución se basa en el uso de ToRank, un algoritmo novedoso basado en la teoría de grafos, que supera a tres alternativas bien conocidas utilizadas en la web superficial: PageRank, HITS y Katz. ToRank demustra su superioridad al crear una interrupción mayor en la conectividad de la red Tor que cualquier otra alternativa. Después de evaluar ToRank, analizamos DUTA-10K para revelar que el 20 % de las muestras están relacionadas con actividades sospechosas, mientras que el 48 % están asociadas con actividades normales, considerando que los 32 % restantes son desconocidos porque eran inaccesibles. También descubrimos que los dominios relacionados con actividades sospechosas generalmente presentan múltiples clones bajo diferentes direcciones, lo que podría usarse como una característica adicional para identificarlos.

Nuestra última contribución consiste en analizar el contenido de un dominio considerando características de texto, HTML, NER, enlaces relacionados e información visual. Comparamos tres estrategias de clasificación supervisadas utilizando Learning to Rank (LtR) para detectar los dominios de cebolla más influyentes. Representamos un dominio de cebolla por 40 características, y aprendimos cómo clasificar los dominios usando un enfoque LtR. Evaluamos nuestra propuesta en un caso de estudio relacionado con drogas, descubriendo que entre los esquemas de LtR explorados, el enfoque por listas supera a los métodos de referencia con un $NDCG@10$ de 0,95. Nuestros resultados demostraron que la clasificación basada en el contenido es superior a los algoritmos basados en enlaces y que las características extraídas del contenido textual visible para el usuario son la mejor opción al equilibrar el costo de su extracción y la precisión alcanzada.

Contents

List of Figures

List of Tables

Índice general

Anexo B: Resumen de la tesis en castellano

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Dr. Enrique Alegre and Dr. Eduardo Fidalgo for their guidance, encourage, and expertise. I have no doubt that it would be impossible to complete this thesis without their tremendous support, their patience, motivation, extensive knowledge and invaluable guidance throughout the duration of this work.

I would like to express my deepest gratitude to Prof. Sarah Jane Delany who was my supervisor during my research stay in Technological University Dublin.

I also would like to thank my colleagues in Group for Vision and Intelligent Systems (GVIS), Surajit, Desiy, Rubel, Pablo, Javi, Victor, Laura, Fran, Manuel, and Alex for all their help and encouragement.

Also, I would like to acknowledge the University of León, to provide a nice research environment for my research studying, and INCIBE which made our research outcome as practical tools helping the community.

Nobody has been more important to me in the pursuit of this research than the members of my family. I would like to thank my parents, whose love and guidance are with me in whatever I pursue. They are the ultimate role models. Most importantly, I wish to thank my loving and supportive wife, Sara, and my two wonderful children, Omar and Obada, who provide unending inspiration.

Finally, and most importantly, All praise be to Allah whom facilitate to me reaching the degree, and I humbly ask Allah to accept this deed (research work) as it is sincerely dedicated for Him.

Mhd Wesam Al-Nabki
León
11th November 2019

Chapter 1

Introduction

1.1. Motivation

A famous representation of the Internet is an iceberg inside the ocean. The visible part is the *Surface Web*, and it refers to the portion of the Web that could be indexed by the traditional search engines, like Google or Yahoo, while the hidden part of the iceberg is known as the *Deep Web* (Bergman, 2001; Noor et al., 2011; Al-Nabki et al., 2017a). A study by Bergman (2001) stated startling statistics about the Deep Web. For example, only on the Deep Web, there are more than 550 billion individual documents comparing to only one billion on the Surface Web. Furthermore, the studies of Rudesill et al. (2015) and Pannu et al. (2019) emphasized on the immensity of the Deep Web, which is estimated to be 400 to 500 times bigger than the Surface Web. In the depth of the Deep Web, there is a hidden fragment, which only can be accessed through specific software applications or a proxy, named the *Dark Web* or the *Darknet* (Moore and Rid, 2016; Al-Nabki et al., 2017a) (See Figure 1.1).

The Onion Router (Tor)[1] is one of the most popular Darknet networks, and its websites are called Hidden Services (HS). These HS can be accessed through Tor Browser software[2] or a proxy, such as Tor2Web[3]. In the last few years, projects like *TOR2WEB* were developed to allow the Surface Web users access directly to Tor content through a standard browser, instead of starting an instance of a dedicated one like the Tor browser.

The concept of Darknet has existed since the establishment of the World Wide Web (WWW). However, its popularity raised in October 2013 when the FBI arrested Dread Pirate Roberts, the owner of *Silk Road* black market. The FBI estimated the sales on Silk Road to be 1,2 Billion dollars by July 2013. The trading network covered among 150,000 anonymous customers and approximately 4,000 vendors (Rudesill et al., 2015) who use Cryptocurrency for their financial transactions (Nakamoto et al., 2008; Ron and Shamir, 2014).

The privacy and anonymity offered by Tor result in an adequate shelter for journalists who are seeking freedom of speech in their dictatorship counties. Unfortunately, illegal services traders abuse the network to promote their business, creating marketplaces

[1] `www.torproject.org`
[2] `https://www.torproject.org/projects/torbrowser.html.en`
[3] `https://tor2web.org/`

Figure 1.1: Iceberg representation of the Internet

and forums for unlawful activities. These activities include, but are not limited to, illegal drugs, trading weapons, and child sexual abuse (Ling et al., 2015; Ciancaglini et al., 2013; Moore and Rid, 2016; Fidalgo et al., 2017; Gangwar et al., 2017; Norbutas, 2018; Foley et al., 2018; Fidalgo et al., 2019). The Tor metrics website[4] reported that the number of unique addresses has increased from 30K to almost 80K between April 2015 and October 2019, and sometimes it reaches to 120K addresses (see Figure 1.2). It is worth mentioning that between the years 2016 and 2017, there were two peaks in the first quarter of each year followed by a sharp decrease. However, there is no clear reason behind this as Kate Krauss, the director of communications and public policy for the Tor Project, declared (Gallagher and UTC, 2016). Nevertheless, the spikes might be explained due to political events like when the government of Uganda blocked the social network before the election in February 2016 (Duggan, 2016). Consequently, new domains were created in the Tor network and more people used this net[5].

Law Enforcement Agencies (LEAs) monitor Tor Darknet, trying to identify, manage, and address these threats. INCIBE, the Spanish National Cybersecurity Institute[6] works with Spanish LEAs, providing them automated tools and services to fight against cyber-

[4] `https://metrics.torproject.org/`

[5] Tor traffic during the Ugandan elections in 2016:
`https://metrics.torproject.org/userstats-relay-country.html?start=2015-11-22&end=2016-02-20&country=ug&events=off`

[6] In Spanish, it stands for the Instituto Nacional de Ciberseguridad de España

Figure 1.2: The number of live Tor HS between April-2015 and October-2019. The horizontal axis represents the years, while the vertical axis corresponds to the number of unique Tor addresses.

crime. These tools allow LEAs to automatically monitor Tor HS, saving them from manually explore thousands of domains daily. Also, to deal with the fast proliferation of Tor hidden services, analyzing their content to capture meaningful information that will assist in fighting against cybercrimes. An example of the work that they do is the classification of Tor HS into different categories, and hence, the information can be provided to the department of the LEA that is specialized in that cybercrime.

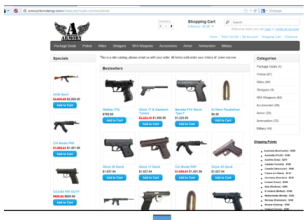
We collaborate with INCIBE, developing solutions based on Artificial Intelligence for these tools and services oriented to Cybersecurity. More specifically, in this work, we have created several solutions for INCIBE's tools and services based on Natural Language Processing (NLP). Our research and solutions for INCIBE can be divided into three principal components or units (see Figure 1.3):

1. *Text Classification Unit (TCU):* given the textual content of an onion domain, the TCU classifies an HS based on its textual content.

2. *Text Mining Unit (TMU):* given the textual content of an onion domain, the TMU extracts a set of features that helps in evaluating the importance or relevance of that domain in the Tor network.

3. *Influence Detection Unit (IDU):* given a list of onion domains and their previously extracted features, IDU assigns a rank value for each domain such that the top-ranked ones are the most influential onion domains in the Tor Darknet.

Currently, the results of this work are being used by the Spanish Police Forces to mon-



Domain Address	Armory34omvkkgr.onion
Category	Illegal
Activity Type	Weapons
Detected Entities	**Products**: Walther P22, Glock 17... **Locations**: United States, Spain... **Person**: Jack Morgan **Organizations**: ARMORY **Groups**: None **Creative-work**: None
Emerging Products	Glock 32 Gen4
Global Rank	23
Weapons Rank	2

Figure 1.3: Onion domains monitoring pipeline

itor the Dark Web of the Tor network. However, they could be generalized into any platform with textual content, such as, but not limited to, Social network, forums. We limit our research to NLP tasks. Therefore, locating or de-anonymizing the IP address of an HS (Biryukov et al., 2013; Jansen et al., 2014; Kwon et al., 2015; Matic et al., 2015) is out of the scope of this thesis dissertation.

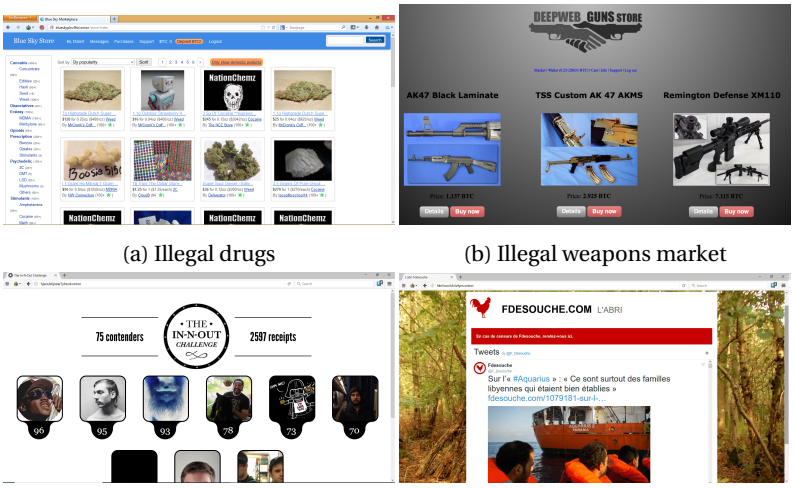
In the following, we present the motivation of each unit.

1.1.1. Text Classification Unit

The objective of the Text Classification Unit (TCU) is to *classify* the suspicious domains into different categories (Al-Nabki et al., 2017a). Previous studies (Intelliagg, 2015; Moore and Rid, 2016; Ling et al., 2015) described different types of contents hosted on the Tor network domains, regardless of their possible legality. However, Spanish LEAs are mainly interested in those domains which host suspicious activities, since they could represent a potential threat to world security.

Monitoring thousands of domains manually, and on a daily basis, is an impractical strategy in terms of time and human resources. Motivated by the large number of hidden services in Tor and the sensitivity of their content, Spanish LEAs demand tools and services that allow them to monitor Tor Darknet efficiently. The isolation of Tor domains

that host suspicious activities is one of its main features. In our work, we denote these activities as *Illegal/ Suspicious Activities*. On the contrary, the rest of the activities in Tor will be denoted as *Legal/ Normal Activities* (see Figure 1.4).



(a) Illegal drugs (b) Illegal weapons market

(c) Personal domain (d) News domain

Figure 1.4: Samples cropped from the Tor hidden services. Figures (a) and (b) refer to two samples of the illegal activities (above), while Figures (c) and (d) refer to samples of the legal activities (below).

1.1.2. Text Mining Unit

After having the onion domains classified, the Text Mining Unit (TMU) is triggered to extract potentially useful information from the text. The text mining is carried out through two approaches: Emerging Products Detection (EPD) and Named Entity Recognition (NER).

The short lifespan characterizes the onion domains, or so-called Hidden Services (HSs), of the Tor Darknet (Sanchez-Rola et al., 2017). Daily, new HSs are introduced while others are seized or shut down by the authorities. New or existing domains come with brand-new products or they might resort to pseudonyms (aliases) for well-known products, such as «Nuggets» or «Candy» to refer «Cocaine» drug[7]. Hence, the products diminish and emerge following several factors related to the market needs and demand. Usually, to trace the products' trends, statistical studies are conducted on the purchase transaction logs (Minelli et al., 2012; Suchacka and Chodak, 2017), which is, however, no possible in the Darknet markets due to the lack of access to the purchases logs (Buxton and Bingham, 2015). A conventional procedure to recognize the emerging products is the manual inspection of the products in the markets. However, it represents a significant

[7] https://americanaddictioncenters.org/cocaine-treatment/slang-names

amount of time in the work schedule of a single person, and it requires specific knowledge of the monitored subject. These problems could be solved by automatically visiting HSs marketplaces, collect and process their textual content, and detects their emerging products, which is the purpose of our Emerging Products Detection (EPD) framework (Al-Nabki et al., 2017b).

Besides the EPD algorithm, we address the NER approach, which is an NLP technique dedicated to locate and classify unknown words as a specific type of entity, such as locations, persons, or product names, among others. Those words are denominated Named Entities (NE), and several applications of NLP use them, such as text summarization (Aramaki et al., 2009), life events detection (Khodabakhsh et al., 2018), contents filtering and monitoring (Hidalgo et al., 2005), trends detection (Kontostathis et al., 2004), or even mining the Surface Web and the social networks (Ritter et al., 2011a; Whitelaw et al., 2008). In addition to the uses mentioned above, decision support systems exploit NER systems to identify fake news (Shaori Al-Ash and Wibowo, 2018), analyze crime patterns (Das and Das, 2018), detect criminal networks (Silvestre Castillo, 2019), and for detecting the influential members of a criminal organization (Taha and Yoo, 2016). NEs are considered as a cornerstone for building a knowledge graph that depicts the relationships between the recognized entities (Usbeck et al., 2014; Hasegawa et al., 2004). However, to the best of our knowledge, NER has not been used to analyze the onion domains of the Tor network, and our research comes to fill in this gap (Al-Nabki et al., 2019d).

Traditional methods of recognizing NEs use a predefined list of keywords, whose creation is time-consuming and hard to maintain. Also, these lists become outdated shortly due to the presence of new emerging terms or rare entities. Furthermore, the problem becomes even more complicated when the keywords lists need to be maintained in different languages. Therefore, it is desired an automatic NER system adapted to dynamic environments, like HSs or social networks.

NER task becomes complex in the Tor network, mainly due to two factors. On the one hand, the limited amount of supervised training samples and the difficulty of reaching online HS. On the other hand, the low quality of the input text, which it has been demonstrated that has an impact on the performance of any NLP based system (Ritter et al., 2011b; Lin et al., 2017). For example, well-structured text quoted from a newspaper is easier to understand than short text from an online HS. In the first case, the capital letters and the punctuation marks are carefully reviewed, while short text from Tor, or other similar sources, frequently contains syntax mistakes and improper grammatical structures. In addition to slang words, informal abbreviations and expressions are used to refer to illicit entities and, in particular, to products. For example, the *Cocaine* has more than 20 street names such as *candy*, *Mama coca*, *smoking gun* or even *sweet stuff*[8]. These NE are subject to change and evolve over time, which leads to emerging terms or unusual entities, as described above. Therefore, employing NER to such a dynamic environment,

[8] https://www.thetreatmentcenter.com/resources/drug-slang/

where hundreds of domains are daily introduced, would ease the task of monitoring Tor, when trying to detect current or emerging products. The NER algorithm presented in this thesis is designed to recognize six different types of named entities in the text: people names, product brands, organizations, groups, creative-work, and locations (Al-Nabki et al., 2019b) (see Figure 1.5).

Figure 1.5: Named Entity Recognition algorithm applied to a drug onion domain. The rectangles refer to the recognized entities in the domain text. The colors, yellow, blue, and red, denote entities of types: organization, product/ brand, and location/ address, respectively. A product underlined with a red line indicates an emerging product.

1.1.3. Influence Detection Unit

Given the fast-growing population of the Tor domains, the Influence Detection Unit (IDU) is responsible for *ranking* the onion domains and for *detecting* the most influential ones. The IDU can provide LEAs with answers to questions like *Who are the market leaders in each field of activities in the Tor network?* and *What are the most attractive onion domains in a specific area of activities?* Answering these questions could improve the capability of LEAs in keeping a close eye on the influential suspicious hidden services

by concentrating their efforts in monitoring them rather than the less influential ones. Moreover, in the case the LEA could take a suspicious HS down, the proposed ranking module would recognize it again even if it were hosted under a new address but it kept the same content. Similarly, when a new domain is introduced to the network for the first time, and it hosts suspicious content similar to an HS that was previously nominated as influential, our ranking module could capture it before becoming popular to the public. Therefore, LEAs will have the possibility to strike the suspicious domains preemptively.

A straightforward strategy to detect the influential onion domains is to sort them by the number of clients' requests that they receive, i.e., analyzing the traffic of the network. However, the design of the Tor network is dedicated to preventing this behavior (Arma, 2019). Chaabane et al. (2010) conducted a deep analysis of the traffic of the Tor network by establishing six *exit nodes* distributed over the world with the default exit policy. However, this approach can not assess the traffic of onion domains that are not reachable through these exit nodes. Furthermore, it could be risky because the Tor network users could reach any onion domain, regardless of its legality, through the IP addresses of these machines. Biryukov et al. (2014) tried to exploit the concept of *entry guard nodes* (Elahi et al., 2012) to de-anonymize clients of a Tor hidden service. However, this proposal will not be feasible as soon as the vulnerability is fixed.

IDU is initialized with a list of onion domains that were classified as illegal - using TCU - along with their extracted features - using TMU - to assign each onion domain a rank value that reflects its popularity among the rest. In our work, we explore two ranking approaches: link-based (Al-Nabki et al., 2019c) and content-based (Al-Nabki et al., 2019a).

On the one hand, the link-based seeks to represent the Tor network by a directed graph where nodes represent the onion domains, and the edges refer to the hyperlinks. This approach addresses analyzing edges of the graph to rank the nodes and to recognize the influential ones — the main advantage of a link-based ranking technique is that it does not require training data. However, if an influential but isolated domain exists in the network, this technique can not recognize it as an essential item. On the other hand, the content-based ranking incorporates features that are extracted from the onion domain to a Learning to Rank (LtR) schema (Li, 2011b). In our work, we extracted 40 features from five resources: 1) user-visible textual, 2) textual named entities, 3) the HTML markup code, 4) the visual content, and 5) features drawn from the topology of the Tor network. These features are used to train a supervised LtR schema (Cao et al., 2007). Our ranking algorithms provide a supplementary and valuable resource for the LEAs in leveraging the use of resources as it recommends where to focus their localization and monitoring efforts.

1.1.4. Online Notepad Services Classification: Pastebin

In addition to investigating the Tor network, we explore a portion of the Surface Web, called Online Notepad Services (ONS). It was designed to conserve text notes on the cloud

and to be accessed later via a unique Uniform Resource Locator (URL). The text can be publicly accessible or secured with a password. This facility has made text sharing irrespective of length, more convenient even in social networks, like Twitter, that limit the posts to a maximum number of characters. One of the most popular Online Notepad Services (ONS) is Pastebin[9](Squire and Smith, 2015). It was launched in September 2002 to facilitate saving and sharing code snippets and configuration files among programmers; however, Pastebin welcomes any further legal use (Pastebin.com, 2018).

Recently, several researchers have reported a significant number of suspicious posts despite that the Terms of Service clearly state the prohibition of publishing illegal content, or *pastes*, as posts are called in the Pastebin community, (see Figure 1.6). These suspicious pastes could hold leaked confidential information, hacking attacks, or even propaganda for illicit activities that are taking place in the Darknet, like the Tor network (Matic et al., 2012; Mirante and Cappos, 2013; Herath, 2017; Kashiwazaki, 2018; Brian, 2019). Early detection of suspicious contents in the ONS would allow the responsible authorities to block such content before it spread to the public. Due to the large number of pastes posted daily (57,600 pastes per day on average[10]), there is a need for an automatic classifier to efficiently recognize whether the content of an entry is suspicious and should potentially be blocked.

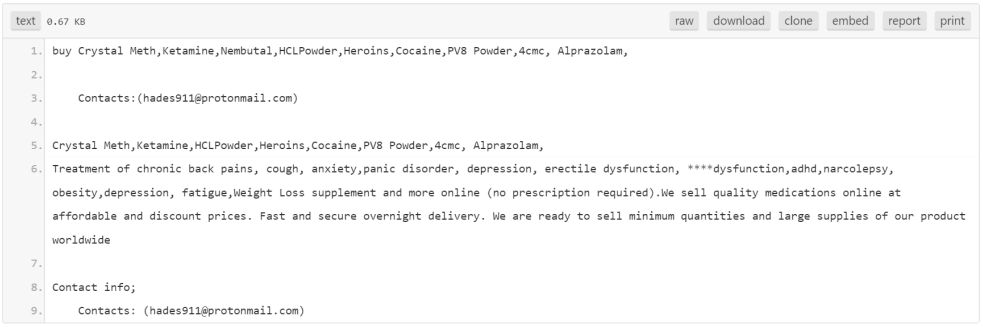
The traditional solutions, such as using a predefined list of keywords or designed using regular expressions, can solve the problem partially, however they suffer from low recall rates (Herath, 2017). In contrast, training a supervised classifier requires a sufficient amount of manually annotated samples, which is extremely labor-intensive. Furthermore, randomly collected and labeled examples may result in an insufficient number of training samples of minor but critical categories. For instance, when Riesco et al. (2019) crawled Pastebin and randomly labeled 804 pastes, only 8% of the obtained samples were associated with suspicious activities, which is insufficient for training.

A popular approach to minimize the manual labeling task is to use *Active Learning* (AL). AL is an technique that can facilitate building a labeled dataset out of an extensive collection of unlabeled samples with a minimum manual labeling effort. Only sampled selected by the AL algorithm are annotated manually and used for training (Settles, 2009).

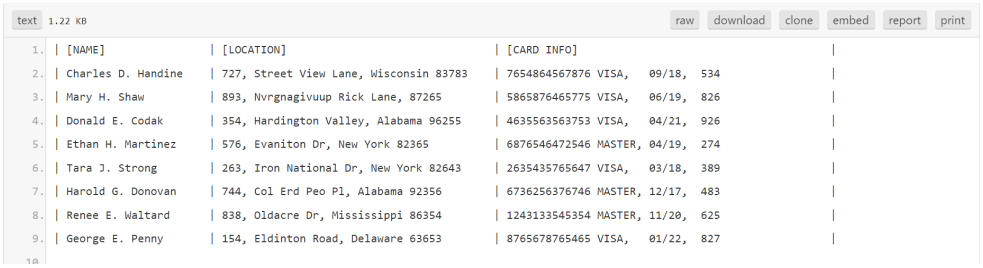
The problem addressed in this work differs slightly from regular text classification. First, the posting freedom in Pastebin results in a wide variety of text typologies ranging from binary files of executable files, logs, code snippets, configuration files, chat logs, to regular articles and books. This variety requires first designing an adequate text representation technique adapted to the content's structure. Second, there are pastes written in different languages, which makes it challenging to use a single model for language representation. And third, we have to keep in mind that the whole picture of the potential activities in Pastebin is unknown. Hence, one objective of this work is to design and build

[9] `https://pastebin.com`

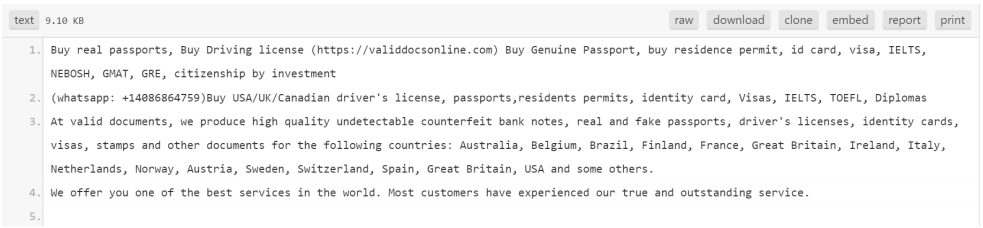
[10] This number was identified by our Pastebin scraping as we were able to collect 10 new pastes per request on average, with a 15 second delay between consecutive requests.



(a) Illegal drugs



(b) Counterfeit credit cards



(c) Counterfeit personal identification

Figure 1.6: Three pastes from the Pastebin online notepad service.

an automatic classifier to detect and categorize the suspicious activities in Pastebin ONS.

The main difference from the Tor network is the amount of the obtained samples. Pastebin, through their scrapping API[11], allows users to access millions of pastes: having much more accessible data than in the Tor Darknet, which required us to use alternative techniques to solve the problem of classifying Pastebin activities. Furthermore, the lack of literature that explores the activities in Pastebin prevented us from having a precise and comprehensive definition of the potential activities there.

[11] https://Pastebin.com/doc_scraping_api

1.2. Objectives

The main goal of this dissertation is to propose new methods and solutions, based on machine learning, to monitor suspicious content of online resources, mainly the contents present in the Tor network and Pastebin service. To that end, we explore and propose different techniques based on Machine Learning (ML), Deep Learning (DL), and Graph Theory. All our approaches have been designed to be used in real tools and services.

Based on the previous general goal, we defined the following particular objectives:

1. To build a robust text classifier capable of detecting illegal content.

2. To provide an automatic solution for mining the text to detect emerging products and to recognize textual named entities.

3. To rank and to detect the most influential onion domains in Tor by exploring several distinct ranking approaches.

1.3. Main Contributions

The main contributions of this dissertation may be summarized as follows:

1. We (Al-Nabki et al., 2017a) present the first publicly available dataset called **D**arknet **U**sage **T**ext **A**ddresses (DUTA)[12], which holds 6,831 manually labeled onion domains extracted from the Tor network. DUTA contains 26 categories that cover a wide variety of activities monitored on the Tor Darknet during our sampling period. (Section 3.1.1)

2. We (Al-Nabki et al., 2017a) build a text classifier to categorize the domains that host suspicious activities on Tor onion domains. We compared two different encoding techniques with three well-known classifiers, identifying the key stages that have a strong influence on the method performance (Section3.1.2).

3. We (Al-Nabki et al., 2017a) set a baseline methodology by fixing the elements of text classification, which allows the scientific community to compare their future research with this baseline under the defined pipeline (Section 3.1.2).

4. We discover that the classification of text content in Pastebin requires a cascade of three different classifiers, because of the different typology of the textual content present in the pastes. We train and evaluate the three models to detect content related to criminal activities (Section 3.2.3).

[12] `http://gvis.unileon.es/dataset/duta-darknet-usage-text-addresses/`

5. We demonstrate how the use of Active Learning algorithms, employing both exploration and exploitation strategies, allows the selection of the most informative samples from the examples space, reducing the labeling effort (Section 3.2.3).

6. We (Riesco et al., 2019) illustrate the superiority of using Active Learning over random sample selection in reducing the effort required to create a labeled dataset (Section 3.2.5).

7. We (Al-Nabki et al., 2017b) create a semi-automatic graph-based method that depends on the K-shell graph decomposition algorithm (Carmi et al., 2007) to detect the most famous and the emerging products in the Tor network. Also, we recognize the association rules between the products to identify the most frequent pairs in the marketplace of the Tor network (4.1).

8. We (Al-Nabki et al., 2019d,b), present a neural network architecture to recognize emerging and rare named entities in noisy user-generated text. The proposed model is inspired by Aguilar et al. (2017) but with two major differences. First, our network does not depend on any external knowledge resource like a gazetteer, which is costly to build and domain-dependent. Second, we introduce a novel feature, that we call *Local Distance Neighbor* (LDN), to substitute the use of any external knowledge resource. (4.2.2)

9. We (Al-Nabki et al., 2019d,b), present an application of the proposed NER model to recognize the named entities within the Tor hidden services (HS) that practice suspicious activities, such as weapons trading and drugs abuse.

10. We (Al-Nabki et al., 2019c) introduce ToRank, a link-based ranking algorithm for onion domains, that detects the most influential ones (Section 5.1.3).

11. We (Al-Nabki et al., 2019c) extend DUTA dataset up to 10,367 manually labeled samples, releasing DUTA-10K. The dataset introduces *CryptoLocker*, a new category which has spread widely, especially after *WannaCry* virus Mitchell (2017) (Section 5.1.1).

12. We (Al-Nabki et al., 2019c) carry out and present a statistical analysis of DUTA-10K regarding the distribution of the conducted activities in the Tor domains, the hidden services that have content replicated, and the distribution of the languages in the analyzed pages (Section 5.1.2).

13. We (Al-Nabki et al., 2019a) propose a novel supervised learning algorithm based on Learning to Rank (LtR) schema to *rank* the onion domains and to *detect* the influential ones. Our strategy exploits five groups of features structured from the Tor network topology and the content of the hidden services and incorporates them into an LtR schema. Our approach outperforms well-known link-based ranking

algorithms, such as PageRank, HITS, and Katz, when tested on samples of onion domains marketing suspicious drugs (Section 5.2.1).

14. To select the best LtR schema, we (Al-Nabki et al., 2019a) explore and compare three popular architectures: pointwise, pairwise, and listwise concluding that the latter is the most suitable for this problem. Also, we create a manually ranked dataset, that plays the role of ground truth for testing the LtR framework (Section 5.2.3).

1.4. Chapter Structure

This section describes the structure of this doctoral thesis. This first introductory chapter has been focused on motivating the work presented in this dissertation, its main objectives, and original contributions. The remaining chapters of this thesis are organized as follows.

- Chapter 2 reviews the literature for the techniques and algorithms related to the three secondary objectives presented above. More precisely, it analyzes published methods that deal with the detection and classification of suspicious activities in the Darknet. Furthermore, it reviews the state-of-the-art techniques used to classify Online Notepad Services, like Pastebin. Next, this chapter references the most recent approaches for recognizing emerging product entities in the Tor marketplaces. Alongside this, it explores the current algorithms used for recognizing named entities in noisy user-generated text. And finally, it reviews the state-of-the are methods dedicated to ranking and detecting the most influential entities among the rest.

- Chapter 3 addresses the problem of classifying onion domains in the Tor network and detecting suspicious activities there. More specifically, it presents DUTA dataset and examines six supervised text classifications pipelines. Additionally, it tackles a different aspect of text classification by employing Active Learning to classify Online Notepad Services, like Pastebin.

- Chapter 4 presents a semi-automatic algorithm for detecting emerging products on the Darknet marketplaces. It analyzes the relationships between the offered products in the markets rather than studying the purchasing transaction logs. This chapter also introduces a neural network model for Named Entity Recognition in a noisy user-generated text as well as in suspicious onion domains. In particular, it introduces a novel feature, called Local Distance Neighbor (LDN), that replaces the use of any external knowledge resource, like Gazetteers.

- Chapter 5 studies two approaches of ranking onion domains of the Tor Darknet for the purpose of detecting the most influential ones. Concerning the link-based method, it presents ToRank algorithm and its application to the Tor network. Alongside the Link-based approach, this chapter also presents a content-based algorithm

based on Learning to Rank (LtR) using 40 features extracted from different kinds of contents present in Tor domains. Finally, an extension of DUTA dataset is presented, analyzing the languages, clones, and activity legality of its domains.

- Chapter 6 contains a summary of the conducted work with the conclusions of this thesis and gives an outlook of possible future research lines.
- Chapter 7 presents a summary of the doctoral dissertation in Spanish to fulfill the regulations about the Ph.D. studies at the University of León.

1.5. Publications and Research Results

This section presents the research results obtained during the completion of this doctoral thesis.

1.5.1. Publications related to this manuscript

- Al-Nabki, M. W., Fidalgo, E., Alegre, E. and de Paz, I. (2017). Classifying Illegal Activities on Tor Network Based on Web Textual Contents. *In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL2017)*, 1, 35–43.
- Al-Nabki, M. W., Fidalgo, E., Alegre, E. and González-Castro, V. (2017). Detecting Emerging Products in Tor Network Based on K-shell Graph Decomposition, *III Jornadas Nacionales de Investigación en Ciberseguridad (JNIC2017)*, 1, 24–30.
- Al-Nabki, M.W., Fidalgo, E., Alegre, E. and Fernández-Robles, L. (2019). ToRank: Identifying the Most Influential Suspicious Domains in the Tor Network, *Expert Systems with Applications* 123, 212–226. Elsevier.
- Al-Nabki, M. W., Fidalgo, E. and Velasco Mata, J. (2019). DarkNER: a Platform for Named Entity Recognition in Tor Darknet, *Jornadas Nacionales de Investigación en Ciberseguridad (JNIC2019)*, 1, 279-280.
- Riesco, A., Fidalgo, E., Al-Nabkib, M.W., Jáñez-Martino, F. and Alegre, E. (2019), Classifying Pastebin Content Through the Generation of PasteCC Labeled Dataset, *14th International Conference on Hybrid Artificial Intelligent Systems (HAIS)*, 456-467.
- Al-Nabki, M. W., Fidalgo, E., Alegre, E. and Fernández-Robles, L. (2019a), Improving Named Entity Recognition in Noisy User-Generated Text with Local Distance Neighbor Features, *Neurocomputing* [Accepted but not published yet].
- Al-Nabki, M. W., Fidalgo, E., Alegre, E. and Chaves, D. (2019). Content-based Features to Rank Influential Hidden Services of the Tor Darknet, *arXiv preprint*

arXiv:1907.11692. 1-12, [Submitted to IEEE Transactions on Services Computing, under revision].

1.5.2. Other publications related to the thesis subject

- Paz Centeno, I. D., Fidalgo, E., Alegre, E., and Al-Nabki, M. W. (2017). Oculus-Crawl, a Software Tool for Building Datasets for Computer Vision Tasks. *Actas de las XXXVIII Jornadas de Automática*. 991-998.
- Joshi, A., Fidalgo, E., Alegre, E. and Al-Nabki, M. W. (2018). Extractive Text Summarization in Dark Web: A Preliminary Study, *International Conference of Applications of Intelligent Systems*.
- Matilla, D., González Castro, V., Fernández Robles, L., Fidalgo, E., and Al-Nabki, M. W. (2018). Color SIFT Descriptors to Categorize Illegal Activities in Images of Onion Domains. *Actas de las XXXIX Jornadas de Automática*.
- Blanco, P., Fidalgo, E., Alegre, E., and Al-Nabki, M. W. (2018). Detecting Textual Information in Images from Onion Domains Using Text Spotting. *Actas de las XXXIX Jornadas de Automática*.
- Blanco Medina, P., Fidalgo, E., Alegre, E., Al-Nabki, M. W., and Chaves, D. (2019). Enhancing Text Recognition on Tor Darknet images. *XL Jornadas de Automática: libro de actas* 828-835.

1.5.3. Research projects

- "Acuerdo de Colaboración para la puesta en marcha de un equipo de investigación aplicada en visión artificial y reconocimiento de patrones". Addendum 22 to the Framework Agreement between INCIBE (Spanish National Cybersecurity Institute) and the University of Leon.
- "Acuerdo de Colaboración para la continuidad de los trabajos de un equipo de investigación aplicada en visión artificial y aprendizaje automático". Addendum 01 to the Framework Agreement between INCIBE (Spanish National Cybersecurity Institute) and the University of Leon.

1.5.4. Attended conferences

- Interpol Workshop 2016 in Madrid about the Machine Learning applications to the Cybersecurity
- European Chapter of the Association for Computational Linguistics (EACL2017) in Valencia, 3-7 April 2017

- Artificial Intelligence International Conference – A2IC 2018 (A2IC2018), Barcelona, Spain; September 22, 2018
- 14th International Conference on Hybrid Artificial Intelligent Systems (HAIS), Léon, Spain; September 4-6, 2019

1.5.5. Patents and Intellectual Properties

Patents

- Al-Nabki, M.W., Fidalgo, E., Alegre, E. and Fernández-Robles, L. An algorithm to detect the most influential onion domains in the Tor network. Application # P201831145, submission date: November 26, 2018[13]

Intellectual Properties

- Al-Nabki, M.W., Fidalgo, E., Alegre, E. and Fernández-Robles, L. Application for the classification of illegal activities in the Tor network analyzing its textual content. Application# Le-156-17, grant date: December 12, 2017 [13].
- Al-Nabki, M.W., Fidalgo, E., Alegre, E. and Fernández-Robles, L. Application for ranking the onion domains in the Tor network and detecting the most influential ones. Application# Le-155-17, grant date: December 12, 2017 [13].

1.5.6. Other Activities

Teaching Activities

Co-supervisor of final degree projects

- Jañez Martino, F., *Use of Natural Language Processing to identify illegal content in Tor darknet*, directors: Fidalgo, E. and Al-Nabki, M.W. 2018, Computer Engineering department/ University of León
- Riesco Valbuena, A., *Creating a Pastebin Tagging dataset using Natural Language Processing*, directors: Fidalgo, E. and Al-Nabki, M. W. 2018, Computer Engineering department/ University of León

International Mobility

- Research stay at Dublin Institute of Technology (DIT), Dublin, Ireland, January - April 2019.

[13] Spanish Patent and Trademark Office, published in Spanish

Chapter 2

State of the art

In the last decades, there has been substantial work in adapting Natural Language Processing (NLP) techniques to fight against cybercrimes. This chapter presents a review of the literature related to (i) text classification, (ii) text mining, and (iii) influence detection.

2.1. Text Classification

Text classification is a fundamental task in Natural Language Processing (NLP) in which its objective is to categorize a piece of text into one of a predefined list of classes (Sachan et al., 2018). Nowadays, a considerable number of research lines and applications require accurate text classification. For example, spam emails detection (Mohamad and Selamat, 2015; Wang et al., 2011), abusive text detection (Chen et al., 2017), sentiment analysis (Pang et al., 2008), detecting suspicious activities in the Darknet network (Al-Nabki et al., 2017a), and classifying illegal content in the Online Notepad Services (ONS) (Riesco et al., 2019). However, several factors collaborate to make this task challenging, such as the possibility of access to a sufficient number of labeled examples (Ohana et al., 2012; Al-Nabki et al., 2017a), and the quality and the length of the addressed text (Li et al., 2016b).

In this section, we review the literature related to text classification to detect suspicious content. A special effort has been made to correctly identify illegal content in both the Darknet of The Onion Router (Tor) network and the Online Notepad Services (ONS). Hence, the conducted literature review covers supervised text classification from two perspectives: 1) classifying the onion domains of the Tor network, which suffers from a limited number of accessible unlabeled samples, and 2) the case of the ONS, which is an inexhaustible stream of text with a wide variety of text typologies. For each scenario, we propose an adequate classification technique; for the Tor network, we review literature related to Web classification based on supervised text, while for the ONS, we address the Active Learning (AL) technique.

2.1.1. Supervised Classification Techniques

Before the rise of the Darknet, several researchers investigated the classification of the Surface Web (Dumais and Chen, 2000; Sun et al., 2002; Kan, 2004; Kan and Thi, 2005). Sun

et al. (2002) proposed a supervised text classification approach to classify the Web pages into a predefined list of categories. To represent a Web page, they did not extract textual features from the content only, but also they took advantage of its hyperlinks and HTML tags, using these features to train a Support Vector Machine (SVM) classifier. Furthermore, Kaur (2014) introduced an interesting survey covering several algorithms to classify web content, paying attention to its importance in the field of data mining. In contrast to Sun et al. (2002), Kaur (2014) pre-processed the text by eliminating the HTML tags, and punctuation marks, along with text stemming. Furthermore, they explored the use of Uniform Resource Locators (URL) in web classification by extracting features through parsing and segmentation (Kan, 2004; Kan and Thi, 2005). Another possible strategy for feature extraction is the work of Onan (2016). They explored four feature selection techniques: correlation, consistency, information gain and chi-square-based along with four different ensemble learning methods: Boosting, Bagging, Dagging and Random Subspace based on four classifiers: Naive Bayes (NB), K-nearest neighbor (KNN), C4.5 and FURIA.

Later, the growing risk of the suspicious activities practiced on the Darknet called the attention of researchers to dive into this network and explore its content (Graczyk and Kinningham, 2015; Moore and Rid, 2016; Al-Nabki et al., 2017a; Park et al., 2018; Dalins et al., 2018; Dong et al., 2018; Takaaki and Atsuo, 2019; Al-Nabki et al., 2019c). Concerning the Deep Web, Xu et al. (2007) presented a supervised text classification framework and used Information Gain (IG) to extract features from the text. To weight these features, they used a customized version of Term Frequency - Inverse Document Frequency (TF-IDF), called Deep Web Term Frequency (DWTF). Likewise, Barbosa et al. (2007) incorporated the TF-IDF technique with the K-means classifier, resulting in an unsupervised learning approach, a clustering pipeline. Noor et al. (2011) explored conventional techniques used for extracting content from the Deep Web: *Query Probing*, which is commonly used for supervised learning algorithms, and *Visible Form Feature* (Xian et al., 2009).

Regarding the Darknet, Graczyk and Kinningham (2015) proposed an algorithm to classify the products of a famous black market on Darknet, called *Agora*, into 12 classes with 79% of accuracy. Their pipeline architecture uses the TF-IDF for feature extraction, the Principal Component Analysis (PCA) for feature selection, and the SVM for classification. Additionally, Moore and Rid (2016) presented a new study based on Tor hidden services to analyze and classify Tor Darknet. Initially, they collected 5K samples of Tor onion pages and classified them into 12 classes using an SVM classifier. Sabbah et al. (2016) focused on detecting terrorist activities on Darknet domains using supervised learning. The authors hybridized feature extraction model that combines five terms weighing techniques, such as TF-IDF, Glasgow, and Entropy, and examined five popular classifiers, like SVM, Logistic Regression (LR), and NB. Furthermore, Al-Nabki et al. (2017a) provided the first public Darknet dataset, called Darknet Usage Text Addresses (DUTA), and holds 6,831 manually labeled instances. They divided the dataset into 26 activities, varying between legal and illegal ones. Using DUTA, they examined two popular text representation techniques: TF-IDF and Bag of Words (BoW), along with three popular supervised classifiers:

SVM, NB, and LR. By Comparing these classification models on the DUTA dataset, they discovered that the combination of TF-IDF with LR obtains the best performances. Lately, Al-Nabki et al. (2019c) released a new version of DUTA, called DUTA-10K, that extends DUTA to 10,367 labeled samples. Recently, Dalins et al. (2018) presented a novel model, named Tor-use Motivation Model (TMM), to classify the hidden services of the Tor network with greater granularity. Park et al. (2018) build a parallel Tor browsers framework to improve the crawling process of the Tor domains. He et al. (2019) proposed a supervised classifier for the illegal activities in the Darknet of the Tor network using the TF-IDF technique and NB classifier.

Several attempts in literature have been proposed to detect illegal activities whether on the World Wide Web (WWW) network (Biryukov et al., 2014; Graczyk and Kinningham, 2015; Moore and Rid, 2016), peer-to-peer networks (P2P) (Latapy et al., 2013; Peersman et al., 2014) and in chatting messaging systems (Morris and Hirst, 2012). Latapy et al. (2013) investigated eDonkey P2P systems to quantify the paedophile activity by building a tool to detect child-pornography queries by performing a series of lexical text processing. They found that 0.25% of entered queries are related to the pedophilia context, which means that 0.2% of eDonkey network users are entering such queries. However, this method is based on a predefined list of keywords that can not detect new or previously unknown words.

2.1.2. Online Notepad Services classification

Suspicious activities are not only taking place in the Darknet, but also exist in the Surface Web. In this section, we address a portion of the Surface Web called Online Notepad Services (ONS). Mounting concerns over publishing sensitive information through ONS have motivated researchers to design algorithms and tools for early detection of suspicious activities. In the following, we review traditional monitoring methods that depend on handcrafted features and advanced techniques that employ machine learning (ML).

The traditional techniques consider the input text against a set of predefined rules. For example, CIRCL AIL (ALI, 2019)[1] proposed a framework to analyze pastes based on a sequence of regular expressions. Another example is the work of Alvarez (2019), who presented an open-source tool for malware identification and classification, called YARA[2] allows users to build customized-rules to capture particular malware or data leak patterns in the text. These rules can be created using wild-cards, regular expressions and special operators, and therefore, can be easily adapted to ONS, such as Pastebin.

To overcome the drawback of the traditional techniques, other researchers have presented solutions based on supervised machine learning. Herath (2017) introduced a framework called *LeakHawk* that consists of a set of classifiers, which ranges between keywords based matching and supervised learning classifiers. However, building and ex-

[1] `https://github.com/CIRCL/AIL-framework`
[2] `https://github.com/virustotal/yara`

tending this platform requires a significant number of labeled samples for training, which can be expensive in terms of time and effort. Furthermore, Riesco et al. (2019) presented an algorithm to build a labeled dataset to train a supervised classifier. They introduced a dataset, called Pastes Content Classification 17K (PasteCC_17K), which holds 17K samples that are distributed over 15 classes. Using the PasteCC_17K dataset, they benchmarked three popular supervised machine learning classifiers: Support Vector Machine (SVM), Naive Bayes (NB), and Logistic Regression (LR), with two text representation techniques: Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF). However, the dataset is strongly biased as the majority of the samples are related to code snippets and media sharing files, whereas the minority represents suspicious activities.

The discussed supervised ML approaches could solve the problem but require an enormous number of labeled samples. Active Learning (AL) is an algorithm to reduce the total labeling cost as it selects only the most informative samples to be annotated by an expert. AL has been used in many fields, including, but not limited to image classification (Joshi et al., 2009; Wang et al., 2016, 2017d), information retrieval and ranking (Rubens et al., 2015; O'Neill et al., 2016), named entity recognition (Chen et al., 2015), and text classification (Yang et al., 2009; Goudjil et al., 2018; Reyes et al., 2018). Goudjil et al. (2018) used the posterior probabilities provided by a set of multi-class SVM classifiers as informative samples selection strategy, and experts label these samples for training. Haldenwang et al. (2018) conducted a comprehensive study of uncertainty-based AL approaches for sentiment analysis on Twitter. They illustrated, experimentally, that AL far outweighs randomly selected samples in supervised learning tasks. Furthermore, AL is not limited to single-label classification problems; it can be further extended to multi-label, which is more difficult as it requires experts to label each example several times since it belongs to various categories (Reyes et al., 2018; Xue and Hauskrecht, 2019; Cherman et al., 2019). The effectiveness of AL in reducing the manual annotation cost makes it suitable for the ONS classification problem.

2.2. Text Mining

In this section, we review the literature related to two fundamental and challenging tasks in text mining: Emerging Products Detection (EPD) and Named Entity Recognition (NER). These two techniques form the main components of the Text Mining Unit (TMU), which is the second stage of the Tor monitoring pipeline presented in Chapter 1.

The marketplace of the Tor network is a dynamic and competitive environment. Continuously new domains are introduced to the network, while others get seized or shut down (Sanchez-Rola et al., 2017). These changes result in presenting new products or at least new names of already exist products. Furthermore, following the market need, some products emerge and become more popular while others diminish. The proposed EPD is concerned with identifying the emerging products and mining the association rules

between them, which is usually extracted from transaction logs. Since these logs are not accessible, the conducted research in this thesis dissertation provides a practical alternative approach.

Apart from detecting emerging products, we tackle the problem of recognizing named entities in text, and in particular, in noisy users-generated text. The Named Entities Recognition (NER) task is dedicated to identify and categorize unknown words in a given text as a specific type of entity, such as locations, persons, or product names, among others. Those words are denominated by Named Entities (NE). Applying NER to a high-quality text, in which the capital letters and the punctuation marks are carefully reviewed, has achieved promising results. Nevertheless, it becomes more challenging on short text with syntax mistakes and improper grammatical structures and abbreviations. In the Tor network or other similar sources, it frequently contains text from the latter type. In the following, we review the state of the art techniques and algorithms of NER applied to regular and noisy text.

2.2.1. Emerging Products Detection

Several researchers have investigated the problem of Emerging Term Detection (ETD) in a given text corpus. The ETD process has two broad strategies (Kontostathis et al., 2004): on the one hand, the semi-automatic receives two inputs: a predefined list of terms and a text corpus, and the ETD algorithm processes the corpus to identify the emerging terms out of all the list of terms it received (Davidson et al., 1998; Porter and Detampel, 1995; Swan and Allan, 2000; Tucker and Kim, 2011; Schubert et al., 2014). On the other hand, the fully-automatic receives only a text corpus as an input and the algorithm automatically finds the emerging terms (Glance et al., 2004; Abe and Tsumoto, 2010) in that corpus.

Glance et al. (2004) proposed a temporal analysis for the weblogs services to detect trending topics, people, and content based on handcrafted features. Kontostathis et al. (2004) used a fully-automatic approach to discover the trends in a timestamped dataset by creating a matrix that models the relationships between documents and terms. The matrix dimensions were reduced using the Singular Value Decomposition (SVD) and used a similarity function to cluster the noun phrases which were closely associated.

The Emerging Products Detection (EPD) task is a particular case of the ETD, whereas the terms refer to the products' names. Also, the EPD is distinguished by using the purchases transaction logs as text corpus. Raeder and Chawla (2011) presented a framework for market basket analysis by modeling the purchase transactions as a product network, and then they clustered the network using a community detection algorithm. Rodrigues et al. (2012) used Social Network Analysis techniques to identify hidden association rules between products in marketplaces, which is a step towards detecting emerging products.

In the marketplaces of the Tor network, the purchases transaction logs are not available, which prevents using any of the above techniques. Our EPD algorithm employs

graph theory to represent the products of Tor marketplaces by an undirected graph of nodes and edges. The nodes refer to the marketplace products, and the edges express the presence of two products within the same marketplace (Al-Nabki et al., 2017b). Using the K-shell algorithm (Carmi et al., 2007), we decompose the graph into shells according to the connectivity between the products in the graph. By analyzing the products in the core-shell, the algorithm identifies the emerging ones.

2.2.2. Named Entity Recognition

Several researchers have addressed the problem of recognizing textual entities within an input text (Zheng et al., 2017; Kadari et al., 2018). The automatic feature extraction using deep learning techniques helped the researchers to propose advanced models (Kwon et al., 2019). Lample et al. (2016) presented a neural network model using Recurrent Neural Networks (RNN), specifically, a Bi-directional Long Short-Term Memory (Bi-LSTM) to capture features of the input tokens, and another one for their characters sequences. Next, the output is fed into a Conditional Random Field (CRF) layer to take advantage of the sequential constraints in the input text, resulting in a model with an F1 score of 90.94%[3]. Greenberg et al. (2018) built a NER system, on top of the Lample's model, and used multiple CRFs to recognize biomedical Named Entities (NEs). Ma and Hovy (2016) used a neural network architecture similar to Lample, but instead of employing Bi-LSTM for the characters sequences, they used a Convolutional Neural Network (CNN) and had an F1 score of 91.21%[3]. However, even though Ma's model had a higher F1 score than Lample's, Reimers and Gurevych (2017) reported that by re-training their models multiple times but with different random seeds, the latter achieved an average F1 score higher than Ma's model. Lastly, Yang et al. (2017) used Transfer Learning (TL) technique and achieved an F1 score of 91.26%[3]. To the best of our knowledge at the time this dissertation is being written, the pooled contextualized embedding model proposed by Akbik et al. (2019) obtained the state of the art performance of 93.18%[3].

The CoNLL-2003 dataset is cropped from Reuters news stories (Tjong Kim Sang and De Meulder, 2003), which means that the capital letters and the punctuation marks are carefully reviewed. Despite the competitive results obtained by the previously commented methods on CoNLL-2003, the performance drops when we apply them to datasets of user-generated text. This kind of text is found in users' posts and comments on social networks and online forums or messages in chat applications (Han et al., 2017; Phan and Sun, 2019).

In 2017, the Workshop of Noisy User-generated Text (W-NUT) (Le et al., 2016) announced a challenging task that targets emerging and rare entities in a noisy user-generated text, cropped from Twitter (Derczynski et al., 2017), and attached it in a dataset called the W-NUT-2017[4]. During the last few years, researchers have improved the F1

[3] CoNLL-2003 dataset

[4] https://noisy-text.github.io/2017/emerging-rare-entities.html

score on this dataset using various techniques, ranging from using Bi-LSTM and CRF, a model similar to the design of Lample et al., and achieved 40.42% (Lin et al., 2017), incorporating the Transfer Learning (TL) approach that achieved an F1 score of 40.78%, to the model of Aguilar et al. (2017) who boosted their model with an extra feature extracted from an external data resource, a gazetteer, and they scored an entity and surface F1 scores of 41.86% and 40.24%, respectively. Several researchers incorporated gazetteers to capture further features from the input text (Mishra and Diesner, 2016; Aguilar et al., 2017; Štravs and Zupančič, 2019; Dey and Prukayastha, 2013; Ding et al., 2019). Nevertheless, its usage is considered a limitation due to the difficulties of building and maintaining it up-to-date to cope with new terms and entities.

Recently, several researchers proposed contextualized embedding approaches that showed promising performance on various NLP tasks Akbik et al. (2018); Peters et al. (2018); Devlin et al. (2019). Akbik et al. (2018) presented a novel approach for text representation using a contextualized character-level word embedding, which surpassed the model of Aguilar et al. on downstream tasks, such as NER. They obtained an F1 score of 49.49%[5] on the W-NUT-2017 dataset. Furthermore, Akbik et al. (2019) improved their model by introducing a pooled contextualized embedding technique, which achieved a state-of-the-art F1 score on the W-NUT-2017 datasets with an F1 score of 49.59%.

To the best of our knowledge, none of the previously commented models were applied to recognize NE in the Tor network, and our work, apart from fixing the limitation of the gazetteer, tries to fill this gap (Al-Nabki et al., 2019d,b).

2.3. Influence Detection

Recently, the literature has become rich with previous works concerning the suspicious activities in the Tor Darknet (Al-Nabki et al., 2019c,a), including, but not limited to, illicit drugs markets (Dolliver, 2015; Broséus et al., 2016; Barratt and Aldridge, 2016), terrorist activities (Weimann, 2016; Chen et al., 2008), arms smuggling, and violence (Fidalgo et al., 2017), in addition to cybercrime (Ciancaglini et al., 2013; Al-Nabki et al., 2017a; Nunes et al., 2016). However, a few of them have addressed the problem of mining the Darknet networks to detect the most influential domains. This task is performed either through analyzing the relationship between its entities, called link-based (Bidoki et al., 2010a) or by evaluating the content of these entities, known as content-based (Bidoki and Yazdani, 2008; Bidoki et al., 2010b; Derhami et al., 2013). Both approaches could be merged into a hybrid one by extracting features from the network and utilizing the content of the entities (Anwar and Abulaish, 2015).

[5]This result was obtained after using the training and the development set to train the model. However, with the training set only, the result drops to 45.38%.

2.3.1. Link-based Ranking

Few works have addressed the problem of ranking the Hidden Services (HS) in the Tor network. Biryukov et al. (2014) proposed a solution that exploits the concept of *entry guard nodes* (Elahi et al., 2012) to de-anonymize clients of a Tor HS. They estimated the popularity of onion domains in the Tor network by examining incoming traffic to those domains, whose weakness is that the analysis can be blocked if this vulnerability is fixed.

In our work, we follow a different approach. We propose to measure a Hidden Service (HS) relevance without measuring the incoming traffic, and using concepts used in the Surface Web to sort search engines results. We represent the Tor network as a directed graph, and we rely upon graph theory for first ranking and later detecting the most influential onion domains.

Graph data structures have been used widely to represent a set of entities with their connections, including, but not limited to, social network analysis (Scott, 1988; Backstrom and Kleinberg, 2014; Ji et al., 2016), and data mining (Al-Nabki et al., 2017b). Henni et al. (2018) used a graph-based approach to build an unsupervised feature selection method, whereas nodes correspond to features, and the graph edges captured the relationship between those features. Next, to assign an importance score for each feature, they addressed several graph centrality measures, and, in particular, PageRank algorithm. Hasan et al. (2013) also used graph theory to build a trust relationship graph between the network users, represented by nodes, whereas the edges reflect binary trust relationships between them.

The link-based ranking algorithms have been studied widely and employed to solve several problems (Borodin et al., 2005). Xu et al. (2009) proposed an algorithm that incorporates a link-based ranking algorithm with a Support Vector Machine classifier to determine the eligibility of applicants for a credit card or loans to banks. Furthermore, Fronzetti Colladon and Remondi (2017) explored the use of social network metrics, such as in-degree, out-degree, closeness, and betweenness centrality to combat money-laundering offenses. Chen (2011) conducted a comprehensive analysis for terrorist organizations to examine the robustness of their networks against attacks. In particular, these attacks were simulated by the removal of the items featured by either their high in-degree or betweenness scores (Wasserman and Faust, 1994). Zhang et al. (2015) proposed a method to identify the influential nodes using two-node centrality techniques, the betweenness (Freeman, 1978), and Katz (Katz, 1953) centralities. Another study by Nouh and Nurse (2015) focused on identifying the nodes that refer to key players in a Facebook group using social network analysis metrics, namely, the eigenvector centrality and the betweenness centrality (Ruhnau, 2000). Moreover, Hu et al. (2016b) studied the GitHub repositories network to detect the influential ones using a graph and built a star relation graph between the repositories. Then, they assessed the performance of the weighted HITS (Kleinberg, 1999) and PageRank (Page et al., 1999). Taha and Yoo (2017) presented an algorithm using the Minimum Spanning Tree (MST) to build a network of

criminals. Each node was assigned a score that was proportional to the number of nodes whose existence depend on the existence of the targeted node. To personalized PageRank algorithm, Hu et al. (2018) proposed UserRank algorithm, which is dedicated to the GitHub developers network.

The interpretation of what *influence* means varies according to the pursued goal. In viral marketing, the opinion leaders who can convince their audience with a point of view, regarding a product, a service, or even an idea, are considered as influential (Gohari and Mohammadi, 2014). While in the field of terrorist networks, the influence could refer to detecting people who have connectivity with the majority of the network users in a decentralized network, such as the financial managers (Berzinji et al., 2012). Eliacik and Erdogan (2018) analyzed social networks and micro-blogging communities to recognize users who can change the decisions of the others via a sentiment analysis algorithm. In such a context, the cluster of those social actors represents the influential bloc. Anger and Kittl (2011) evaluated the users' influence by their social networking potential. They proposed a score that captures the interaction between users and their followers, regarding all the published tweets. Taha and Yoo (2017) employed the existence dependency concept on a network of criminals.

In the Tor network, as in the Surface Web (Cockbuen and Mckenzie, 2001; Levene, 2011), a user surfs the network moving from a domain to another following the hyperlinks connecting the web pages, until the user reaches to a market-leading domain who is identified as influential in this context. In this work, we employ social network analysis techniques and algorithms to establish a link-based approach that analyzes hyperlinks among the onion domains in order to rank each onion domain and to recognize the influential ones. However, to the best of authors' knowledge, there is no ground truth rank to judge the correctness of a given order. Alternatively, and for consistency with previous studies (Booker, 2012; Fronzetti Colladon and Gloor, 2018; Fronzetti Colladon and Vagaggini, 2017), the influence of a given domain is interpreted by the amount of disruption that can be caused to the connectivity of the network by removing that domain. This criterion is correlated with network robustness, which is defined by the ability of a network to retain its system structure intact despite being exposed to perturbations, i.e., removing the influential nodes (Holmgren, 2007). This intent is backed up by a considerable amount of literature which already proposed algorithms and techniques to effectively attach a network robustness (Chaurasia and Tiwari, 2013; Duijn et al., 2014; Zhang et al., 2015; Memon and Larsen, 2006).

2.3.2. Content-based Ranking

Despite the effectiveness of the link-based approach, it would fail in evaluating the isolated nodes which do not have connections to the majority of the community. Therefore, the content-based approach can overcome this problem, and even surpasses link-based techniques.

Anwar and Abulaish (2015) presented an algorithm to detect the influential leaders of radical groups in the Darknet forums. They mined the content of the user's profiles and their historical posts to extract textual features to reflect their radicalness. Then, they incorporate these features with a link-based ranking algorithm, PageRank (Page et al., 1999), to build a ranked list of radically influential users. Cossu et al. (2015) proposed an algorithm to detect influencers on Twitter. They used a directed graph to represent the network, in which the nodes refer to the users and the edges capture the relation between them. Each Twitter user was described by features extracted from their profile and posts, such as the tweets characteristics, and the constructed graph properties like the degree (Seidman, 1983) and the betweenness centrality (Freeman, 1978). Additionally, MacAvaney et al. (2019) proposed a content-based ranking algorithm by extracting features from the content using BM25 algorithm (Robertson and Zaragoza, 2009).

The approach proposed in this work is entirely different. At the moment of writing this Ph.D. dissertation, there are no works that have addressed the Tor network from a content-analysis perspective. We adopt a supervised framework that automatically learns how to order items following predefined ranking criteria. Concretely, we employ a Learning to Rank (LtR) algorithm that captures characteristics of a given ranked list and maps the learned rank into a new unsorted list of items.

LtR framework has been used widely in the field of Information Retrieval (IR) (Liu et al., 2009; Jiang et al., 2016). Li et al. (2018) proposed an algorithm to help software developers in dealing with unfamiliar Application Programming Interface (API) by offering software documentation recommendations. They train an LtR model with 22 features extracted from four resources. Agichtein et al. (2006) employed RankNet algorithm to leverage search engine results by incorporating user behavior. Wang et al. (2017c) presented LtR-based framework to rank input parameters values of online forms. They used six categories of features extracted from user contexts and patterns of user inputs. Moreover, LtR has been used for mining social networks (Li et al., 2012, 2016a; Duan et al., 2010). Li et al. (2012) proposed an LtR to detect and rank critical events in Twitter social network.

Chapter 3

Text Classification

This chapter addresses the text classification problem in two different contexts: the Darknet of the Tor network and the Pastebin service. Given the different typologies of the text contained in Tor and Pastebin, in the next sections, we describe the novel approaches designed and followed to solve the classification problem for each environment.

3.1. Tor Hidden Services Classification

The Darknet of the Tor network provides a high level of anonymity and privacy for its users. These characteristics establish a secure environment where online criminal activities proliferate rapidly. Those activities range from child sexual abuse and drug smuggling to weapons trading, counterfeit personal identification and credit cards. Motivated by this specific use of the Tor onion domains, we focused our research on designing and building models to identify and to categorize the suspicious activities in Tor.

Text classification is the cornerstone in many Natural Language Processing (NLP) applications. These systems fall under two broad classes, either supervised or unsupervised (Kelleher et al., 2015). In the first case, the training data is made up of pairs of input samples and their corresponding correct labels. A *supervised learning* algorithm models this relationship between an input sample and its label. A trained model will be able to predict a label for a new data sample. In the latter case, there are no labels available for the training data, therefore the algorithm mines for patterns in the dataset looking for any kind of similarity. The primary purpose of *unsupervised learning* is to cluster the data by finding and grouping similar samples. One possible variation to improve unsupervised learning is initiate with a small portion of labeled samples to learn latter the patterns from a large number of an unlabeled dataset. This approach is a hybrid of supervised and unsupervised learning, being designated as *Semi-supervised learning*.

Our work on suspicious activity classification in the Tor network is a multi-class classification problem, where the classifier has to predict whether an input sample is similar to any of the predefined suspicious activities or is not. As we target particular suspicious classes, the supervised learning approach is suitable for the designated classifier, while the other methods are out of the scope of this thesis dissertation.

A supervised learning system needs a pre-labeled dataset for training. A classical supervised text classifier consists of four main components (Dalal and Zaveri, 2011): 1) text

pre-processing, 2) feature selection/ extraction, 3) selection of the model to be used, and finally, 4) training and testing the model. Therefore, to be capable of providing a solution to the problem, our first task was to create a dataset of labeled onion domains, also known as Hidden Services (HSs).

3.1.1. Darknet Usage Text Addresses Dataset

Dataset Building Procedure

To best of our knowledge, before releasing Darknet Usage Text Addresses (DUTA), there was not any labeled dataset publicly available, containing text with different activities from the Tor Hidden Services (HSs) (Biryukov et al., 2014; Moore and Rid, 2016; Intelliagg, 2015). We built a customized crawler that utilizes the Tor socket to fetch onion web pages through the port 80, i.e., the HTTP protocol, as it is the most popular port in the Tor network after the 55080 one (Biryukov et al., 2014). However, we ignored the latter port because it corresponds to hidden services created on computers infected by botnet malware. The crawler has 70 worker threads in parallel that download the HTML code behind each HS. Each thread dives into the second level in depth of a given onion domain to gather as much text as possible rather than just the index page, following other works (Biryukov et al., 2014). The crawler searches the HS links retrieved from several well-known Darknet resources, like onion.city [1] and ahmia.fi [2]. We reached more than 250K HS addresses, but only 6,831 were alive[3], while the others were down or not responding. After that, we concatenated the HTML pages of every HS into a single HTML file, resulting in a single HTML file for each HS domain.

Dataset Characteristics

Several researchers analyzed the contents of the HSs and categorized them into a different number of categories. Biryukov et al. (2014) sampled 1,813 HS and detected 18 categories. Intelliagg group (Intelliagg, 2015) analyzed 1,000 HS samples and classified them into 12 categories, while Moore and Rid (2016) studied 5,615 HS examples grouped in 12 classes. Our objective was to build a multipurpose dataset, and for the sake of completeness, we classified the 6,831 crawled HSs manually. We divided the collected samples among several members of our research group, and each one labeled their designated part. If someone hesitates, we conduct an open discussion among the involved people to judge that onion domain. Finally, to check the consistency of the procedure, Wesam Al-Nabki reviewed the tags assigned to each HS by analyzing random samples of the categorization made by the others. Finally, the resulting dataset, DUTA, comprises 26 classes, which are listed in Table 3.1. During the labeling process, we noticed that some of the 26

[1] www.onion.city
[2] www.ahmia.fi
[3] The crawler was running for two months, between May and July 2016

categories were candidates for labeling sub-classes beneath them. For example, into the class *Counterfeit Personal Identification*, we detected HS related to the counterfeit of different IDs, so we created three sub-classes: *Identity Card*, *Driving License*, and *Passport*. To the best of our knowledge, DUTA dataset contains the most extent and complete classification up to date.

Main Class	**Sub-class**	**Count**	**Main Class**	**Count**
Violence	Hate	4	Art/ Music	8
	Hitman	11	Casino/ Gambling	26
	Weapons	47	Services	285
Counterfeit Personal Identification	Driving Licence	4	Cryptocurrency	586
	ID	7	Down	608
	Passport	37	Empty	1,649
Hosting and Software	File-sharing	111	Forum	104
	Folders	63	Hacking	90
	Server	95	Leak-data	12
	Software	121	Locked	435
	Directory	142	Personal	405
Drugs	Illegal	230	Politics	8
	Legal	9	Religion	6
Marketplace	Black	63	Fraud	4
	White	67	Library. Books	27
Pornography	Child-pornography	914 ([4])	Counterfeit Money	55
	General-pornography	83	Counterfeit Credit Cards	240
Social-Network	Blog	71	Human-Trafficking	2
	Chat	47		
	Email	56		
	News	32		
Total				**6,831**

Table 3.1: Classes of DUTA dataset

Counterfeit is a broad class, so we split it into three main classes: 1) *Counterfeit Personal Identification*, which is related to the falsification of government documents; 2) *Counterfeit Money*, that includes currencies forging and 3) *Counterfeit Credit Cards*, which covers cloning credit cards, hacked PayPal accounts or fake markets cards like Amazon and eBay, among others. The class *Services* contains the legal services that are provided

[4]This class includes 57 unique sample plus 857 samples that are extracted from a single forum (See Section 3.1.1)

by individuals or organizations, such as voting services[5]. The class *Down* includes the errors that were returned by web pages that were down during the crawling, e.g., an SQL error in a website database or a JavaScript exception. Furthermore, the category *Violence* is associated with criminal activities and hate crimes, while the class *Drugs* refers to drug smuggling activities and drug forums. Likewise, the *Hacking* class comprises hacking tutorials and forums.

We assigned class *Empty* to a web page when: 1) the text is short, i.e., less than 5 words, 2) it has only images with no text, 3) it contains unreadable text, like special characters, numbers, or unreadable words, or 4) the empty Cryptolockers pages, i.e., pages related to the payment due to an infection by a Ransomware(Ciancaglini et al., 2016). The class *Locked* contains the HSs that require solving a CAPTCHA or request log-in credentials. We noticed that some people love to present their works, projects, or even their personal information through an HS page, and hence, we labeled them into the category *Personal*. The onion domains that fell under more than one category were labeled based on its main content. For example, we assign the *Forum* label to the multi-topic forums, unless the whole forum is related to a single topic. e.g. a hacking forum was assigned to *Hacking* class instead of *Forum*. The class *Marketplace* was divided into *Black*, when it contained a group of illegal services like Drugs, Weapons, and Counterfeit services, and *White* when the marketplace offered legal shops like mobile phones or clothes.

We realized that some forums domains in the Tor network contain a numerous number of web pages related to a single class. For example, we found a forum about Child Sexual Exploitation Material (CSEM) that has more than 800 pages of textual content. For this type of forums, we split them up into standalone samples, and we added them to the dataset.

3.1.2. Methodology

Each classification pipeline comprises three stages: text pre-processing, feature extraction and encoding, and classification. We combine two well-known text encoding techniques with three supervised classifiers, resulting in six different classification pipelines. We optimize every pipeline to figure out what is the combination that achieved the highest performance.

Text Pre-processing

In the first stage, we initially eliminated HTML tags, but when we detected an image tag, we preserved the image name and removed the extension. Then, we filtered the training set for the non-English samples using Langdetect[6] python library and stemmed the text using Porter library from NLTK package[7]. After that, we removed special charac-

[5] http://j5lo7vgmgrz5xoi3.onion/
[6] https://pypi.python.org/PyPI/langdetect
[7] https://tartarus.org/martin/PorterStemmer/

ters and stop words with SMART stop list[8] (Salton, 1971). At this stage, we modified the stop words list by adding 100 words to make it compatible with the work domain. Finally, we mapped all emails and URLs into a <*EML*> and <*URL*>, respectively.

Feature Extraction

After *text pre-processing*, we used two text representation techniques: Bag-of-Words (BOW) (Harris, 1954) and Term Frequency-Inverse Document Frequency model (TF-IDF) (Aizawa, 2003).

The BOW is a well-known model for text representation that extracts feature from a text corpus by counting the frequency of the words. Consequently, every document is represented as a sparse feature vector where every feature corresponds to the frequency of a single word in the training corpus. TF-IDF is a statistical model that assigns weights to document words, accentuate those whose frequency is higher in a given document. At the same time, de-emphasizes words that frequently occur in many documents. Both BOW and TF-IDF do not take into consideration the order of the words, and they are simple to implement, computationally efficient, and compatible with medium size dataset.

Classifier Selection

Once the features are extracted using BOW or TF-IDF techniques, the next step is selecting a suitable classification algorithm. In this following, we explore three popular classification algorithms used in supervised machine learning.

- *Naïve Bayes (NB)* is a simple but efficient algorithm for text classification (McCallum et al., 1998). NB is a conditional probability model that depends on Bayes' theory, considering the independence assumption between features. Given an unseen sample, X, with n features ($X = x_1, x_2, ..., x_n$), Bayes' model calculates the posterior probability of predicted class $P(c|X)$ following the next equation (Equation 3.1).

$$P(c|X) = \frac{P(C)P(X|c)}{P(X)} \tag{3.1}$$

  In practice, the interesting part of the above fraction is the numerator, because the denominator, $P(X)$, is effectively constant for every data point in the training set. This fact allows eliminating the denominator from Equation 3.1.

  Naïve Bayes assumes all the features are mutually independent. Under this assumption, we rewrite Equation 3.1 in a new format, as shown in Equation 3.2.

$$P(c|X) = P(c|X_1)P(c|X_2)P(c|X_3)...P(c|X_n)P(c) \tag{3.2}$$

[8] `http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/`

- Support Vector Machine (SVM) tries to find a hyperplane or set of hyperplanes in an N-dimensional space, whereas *N* refers to the number of features, to divide the data points into distinct classes (Suykens and Vandewalle, 1999). The optimal hyperplane is the one that can maximize the distance between data points of different classes.

  SVM performs well for linearly separable data points, while for non-linearity, kernel functions are needed, such as polynomial, sigmoid, and Gaussian. The main objective of introducing these kernels is to map the non-linear separable data points into a higher dimensional feature space.

- Logistic Regression (LR) is a popular statistical model that uses a logistic function (Hosmer Jr and Lemeshow, 2004). The LR is an optimized version of a Linear Regression model but uses a sigmoid function to maps predicted data points to probabilities between zero and one.

3.1.3. Empirical Evaluation

Experimental Setting

The objective of this work is to identify onion domains where criminal activities take place. To that end, we selected a subset of eight categories of DUTA, trying to cover the most representative illegal activities, as shown in Table 3.2.

We established two conditions for selecting a category on this subset. On the one hand, each class in the subset must be mono-topic, i.e., related to a single category. For example, we excluded *Black-Market* from the training set, because its contents are related to more than one class at a single time, and we wanted the classifier to learn from pure patterns. On the other hand, each class must have a minimum of samples to ensure that the category has enough variety for training and testing. We set this threshold empirically to 40.

The rest of the DUTA categories were assigned to a ninth category, which we called *Others*. Moreover, when a sample contains relevant images but an irrelevant text or without any textual information, we excluded it from the dataset. Therefore, we had 5,635 samples distributed over nine classes, i.e., the eight classes related to illegal activities plus the *Others*. Details about each category can be seen in Table 3.2. After the *text pre-processing* stage, we obtained 5,002 onion domains, which we later split into a training and test set of 3,501 and 1,501 samples respectively, i.e., 70/30.

The dataset is highly unbalanced, since the most abundant class has 3,513 samples, while the smallest one has 40. We solved the skew in the dataset thanks to the *class-weight* parameter in Scikit-Learn library[9] which assigns a weight for each class proportional to

[9] http://scikit-learn.org/

Experiment Main Class	Count
Pornography	963
Cryptocurrency	578
Counterfeit Credit Cards	209
Drugs	169
Violence	60
Hacking	57
Counterfeit Money	46
Counterfeit Personal Identification (Driving-License, ID, Passport)	40
Others	3,513

Table 3.2: DUTA subset of eight suspicious categories plus a ninth one we named others. These samples are used to train the text classifier)

the number of samples (Hauck, 2014). In addition to adjusting the weights of classes, we split and sort forums by the discussion page (See Section 3.1.1).

To tune the models, we applied a grid search over different combinations of parameters with 10-fold cross-validation. The best combination, which corresponds to the selected classification pipeline, is the one that can achieve the highest value of an averaged F1 score metric and accuracy over the 10 folds.

We used Python3 with Scikit-Learn library for the pipeline implementation. We modified the parameters that have a critical influence on the performance of the models. For the BOW dictionary, we set it to 30,000 words with a minimum word frequency of three, and we left the rest of the parameters to default. Regarding the TF-IDF, we set the maximum feature vectors length to 10,000 and the minimum to three. For the parameters of the classifier, we kept the default setting for the NB. In contrast, for LR, we set the regularization parameter $C = 10$, and we activated the balanced class-weight flag. For the SVM classifier, we set the decision function parameter to one-vs-rest (OVR), kernel to Radial Basis Function (RBF), $C = 10e5$, balanced classes weights, and the rest by default.

3.1.4. Results and Discussion

It has been proved that the accuracy metric is inaccurate when the dataset is imbalanced (Chen et al., 2017). Table 3.3 shows the F1 score per class and reports three methods of averaging the F1 score macro, micro, and weighted, of each classification pipeline. We found that the TF-IDF combined with LR achieves the highest macro F1 score and cross-validation accuracy, 93.7% and 96.6%, respectively. In contrast, the combination of TF-IDF with the NB classifier was the worst, with 46.0% and 86.3%. Our finding is compatible with the conclusion of Ng and Jordan (2002) that the discriminative model, like LR, performs better than the generative model, like NB.

Figure 3.1 shows F1 score of each classification pipelines per category. We can see

Metrics/ Methods		Average (macro %)	Average (micro %)	Average (weighted %)	CV Accuracy %
BOW LR	P	95.2	96.5	96.5	95.8 +/- 0.010
	R	88.9	96.5	96.5	
	F1	91.6	96.5	94.6	
TF-IDF LR	P	98.2	97.4	97.5	**96.6 +/- 0.010**
	R	90.2	97.4	97.4	
	F1	**93.7**	**97.4**	**97.4**	
BOW SVM	P	87.7	94.1	94.2	93.2 +/- 0.013
	R	87.5	94.1	94.1	
	F1	87.4	94.1	94.1	
TF-IDF SVM	P	98.3	97.1	97.2	96.0 +/- 0.009
	R	88.2	97.1	97.1	
	F1	92.4	97.1	97.0	
BOW NB	P	86.5	94.1	94.3	92.4 +/- 0.009
	R	97.0	94.1	94.1	
	F1	81.2	94.1	94.0	
TF-IDF NB	P	53.0	88.5	88.5	86.3 +/- 0.012
	R	42.5	88.5	88.5	
	F1	46.0	88.5	86.0	

Table 3.3: Comparison between the classification pipelines with respect to 10 folds cross-validation accuracy (CV), precision (P), recall (R) and F1 score metrics for micro, macro and weighted averaging.

that the classes which have distinct keywords, such as the *Pornography* and the *Cryptocurrency* classes, obtain relatively high F1 score over all the classification pipelines. In contrast, the classes whose keywords might intersect, like *Counterfeit Credit Cards* and *Hacking*, suffer from a low F1 score. For example, both classes could use phrases such as «We hack credit card» or «Hacked PayPal account for sale». Another factor that could influence the performance negatively is the lack of training samples, e.g., class *Violence* has 60 samples only.

As shown in Figure 3.2, the learning curve of the best classifier - TF-IDF and LR combination - is learning correctly. The validation accuracy curve has the optimal behavior as it is rising, and classification accuracy improves by introducing more samples. Conversely, the training accuracy curve is starting to decrease slightly. Nevertheless, we acknowledge that the classifier suffers from overfitting, which is due to the difficulty of reaching more samples of onion hidden services, and especially from classes like Counterfeiting Personal Identification and Violence. However, this high accuracy archived will help to build a robust model that will be able to detect suspicious activity on Tor Darknet.

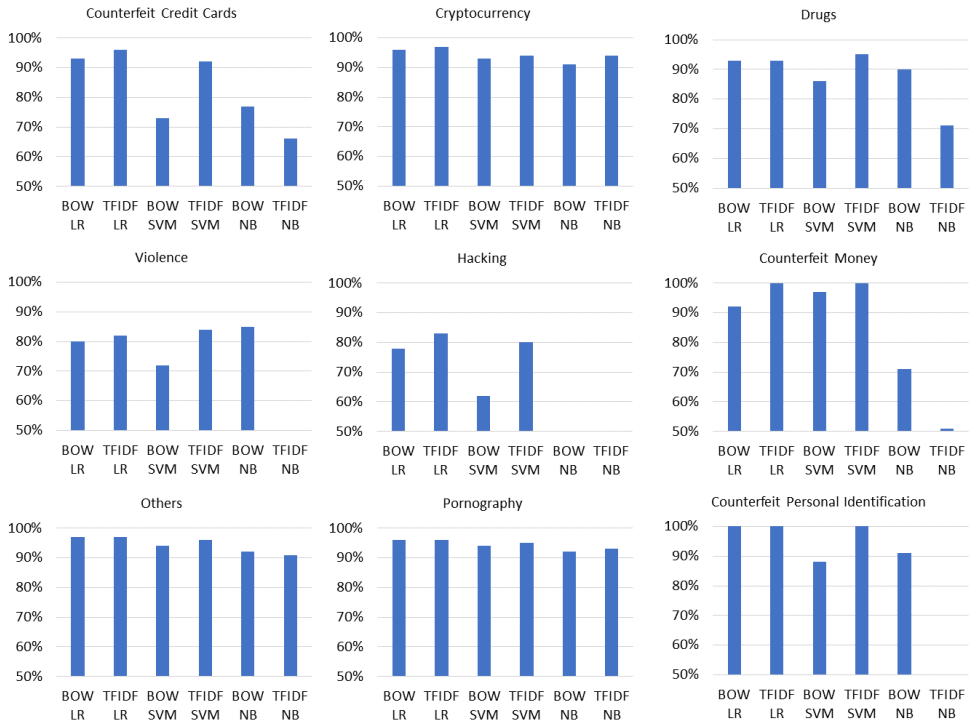


Figure 3.1: F1 score comparison per class of the classification pipelines. When a bar is not shown, it means that its value is zero.

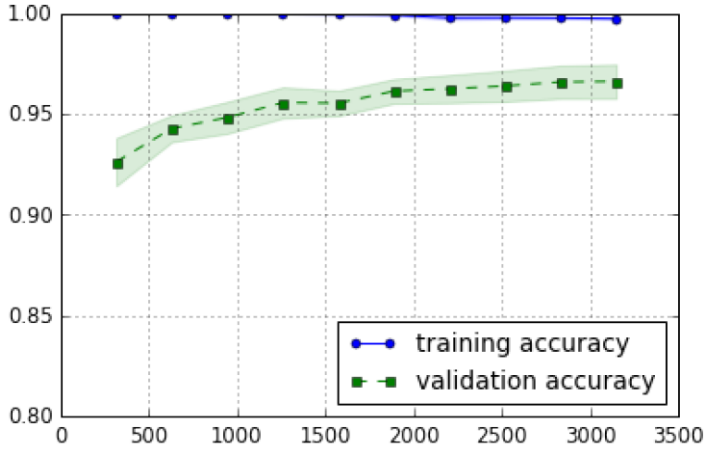


Figure 3.2: Learning Curve of TF-IDF with LR classifier. The X-axis refers to the number of the training samples, while the Y-axis shows reports the obtained accuracy.

3.2. Active Learning for Text Classification

The Online Notepad Service (ONS) allows users to publish unstructured text anonymously. This feature facilitates the propagation of criminal content, such as Child Sexual Abuse (CSA) and drug markets, or even hyperlinks to illegal activities on the Darknet of the Tor network. Pastebin, a popular example of the ONS, is an inexhaustible source of user-generated text that the Law Enforcement Agencies (LEAs) are interested in monitoring.

Building a supervised classifier to identify automatically suspicious content would be the ideal solution to combat cybercrimes. However, the bottleneck of supervised systems is the availability of labeled datasets. Active Learning (AL) is a technique that eases building a labeled dataset out of an extensive collection of unlabeled samples, reducing the manual labeling effort. Only the samples nominated by the AL algorithm are annotated manually and used for training (Settles, 2009).

3.2.1. Exploring Active Learning Strategies

The Active Learning (AL) literature presents three approaches for a learner to query a sample from a given corpus to be labeled by an oracle, typically an expert. These approaches are: (i) Membership Query Synthesis, where the annotator selects a sample or generates a new sample (Angluin, 1988), (ii) Stream-Based Selective Sampling assumes that the cost of a new unlabeled sample is free, i.e., it can be collected easily, and the samples are queried one by one and exposed to the annotator who decides whether to consider a sample or to reject it (Loy et al., 2012), and (iii) Pool-Based Active Learning which assumes that there is a large pool of unlabeled samples, and according to a particular selection strategy, a batch of samples are drawn from the pool to be labeled by the annotator (Hu et al., 2010a, 2016a). Settles (2009) presented a comprehensive survey of these approaches. However, in this work, we only consider the latter one, pool-based AL, given its popularity for solving real-world problems (Settles, 2009; Gupta et al., 2018; Beluch et al., 2018) and because it corresponds exactly with our situation where we have a large dataset of unlabeled samples collected from Pastebin.

We formally model the process of the pool-based approach with six parameters (U, L, S, SC, b, O) (see Figure 3.3). In the following, we explain the role of each, and we describe how they interact. U is a pool containing all the *unlabeled samples*. A crucial stage of the AL process is the *selection strategy*, S, which is responsible for assessing the *informativeness* of each instance in U. Only a batch, b, of the most informative samples are exposed to an *oracle*, O, typically an expert. The size of b is a parameter of the AL algorithm. As soon as O labels the batch manually, it is removed from U and appended to the *labeled set*, L. Then, the informativeness values of the unlabeled samples in U is updated. This process is repeated through a certain number of iterations until it reaches a *stopping criterion*, SC, such as a maximum number of iterations, a particular accuracy

value achieved on a test set, or when the pool runs out of samples (Zhu et al., 2008a; Hu et al., 2016a). The resulting labeled set, L, is used to train a supervised model, θ, which is the output of the active learning process.

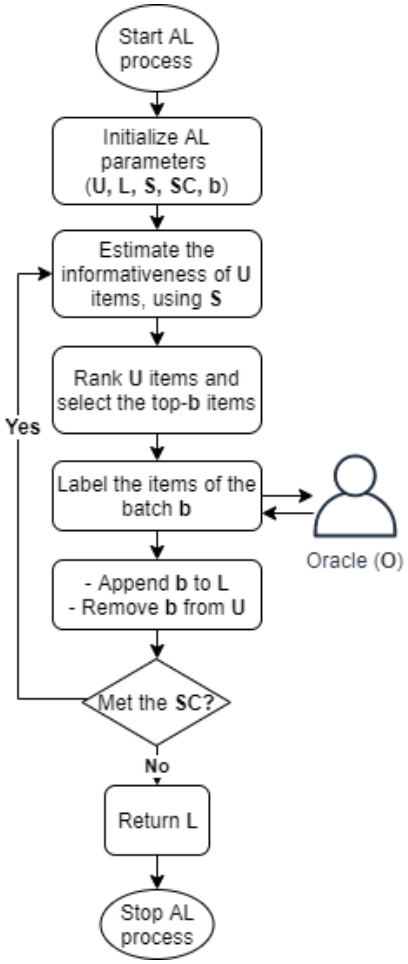


Figure 3.3: Flowchart of the pool-based active learning process

Typically, in the first cycle of the algorithm, the entries of b are selected either randomly from U, or by applying a clustering algorithm to its elements. In the latter case, the closest samples to the centers of the clusters form the first batch of b samples (Yu et al., 2008; Lughofer, 2012). It is worth mentioning that AL literature recommends using deterministic clustering techniques (Hu et al., 2010b), such as Agglomerative Hierarchical Clustering (AHC) (Voorhees, 1985).

The literature of AL categorizes the selection strategies, S, under two main classes:

exploitation-based and exploration-based, which can also be joined together into a hybrid strategy (Hu et al., 2016a).

Exploitation-based Active Learning

Exploitation-based AL is a supervised approach that attempts to refine the classification decision boundary by searching for the most uncertain instances in the feature space. The performance is highly dependent on how well the initial classification boundary was shaped (Hu et al., 2010a).

Practically, in the first cycle of the AL procedure, the initial training batch, b, is used to train a model, θ, and define the first decision boundary. This model is exploited to predict all the unlabeled examples X, in the pool U, obtaining their labels $\hat{y}$. Then, according to a selection technique (explained below), the next batch of unlabeled examples is selected from the U to be labeled by O. The new batch is appended to L to train θ again and removed from U.

Given the prediction confidence of unlabeled samples, x_i, as probability distributions, produced by the model θ, the selection strategy works as follows.

- *Least Confidence* (LC). It targets the examples which have lower confidence among their most likely labels. Equation 3.3 shows how LC is estimated for the sample x_i.

$$LC(x_i) = \arg\min P_\theta(\hat{y}_i|x_i) \tag{3.3}$$

- *Margin Sampling* (MS). The main limitation of LC is that it only considers the most probable label of x_i, while it ignores the others. The MS overcomes this limitation by incorporating both the first and second most probable labels and sorts the samples according to the smallest difference, as shown in Equation 3.4.

$$MS(x_i) = \arg\min P_\theta(\hat{y}_{(i,1)}|x_i) - (\hat{y}_{(i,2)}|x_i) \tag{3.4}$$

- *Entropy Sampling* (ES). Unlike MS, ES considers the probability $\hat{y}_{(i,c)}$ of all class labels, C, to estimate the uncertainty of the sample x_i, as shown in Equation 3.5.

$$ES(x_i) = \arg\max -\sum_{c}^{C} P_\theta(\hat{y}_{(i,c)}|x_i)\log(\hat{y}_{(i,c)}|x_i) \tag{3.5}$$

Exploration-based Active Learning

Unlike the exploitation-based approach, the exploration-based targets properties of the unlabeled samples in the pool, such as density and diversity.

- *Density.* The focus is on identifying dense regions in the space, considering that the label of a sample of each region, most likely, will be representative of the label

of its neighbors (Dasgupta and Hsu, 2008). The literature of AL presents plenty of methods to estimate the density of an instance (Dasgupta and Hsu, 2008; Settles and Craven, 2008; Zhu et al., 2008b; Settles, 2009; Hu et al., 2010a; Wang et al., 2017a; Luo et al., 2018), but Equation 3.6 expresses only the density measure used in this work.

$$\begin{aligned} density(x_i) &= \sum_{x_n \in N} cos(x_i, x_n) \\ N &= x_n \in D | cos(x_i, x_n) \geq \alpha, \end{aligned} \tag{3.6}$$

where $\alpha = \mu - 0.5 \times \gamma$; μ and γ being the mean and the standard deviation of the pair-wise similarities of all the instances in a given dataset, D, and cos refers to the cosine similarity.

- *Diversity.* In addition to density, the *diversity* of the instances in the examples space is a decisive factor to be considered. The diversity of an unlabeled sample x_i reflects its distance to the already labeled samples in the pool, U. This factor can indicate a potentially unexplored area of the space (Brinker, 2003; Hu et al., 2010a; Wang et al., 2017b). Equation 3.7 shows the diversity measure of a sample, x_i, proposed by Hu et al. (2010a), and used in this work.

$$diversity(x_i) = \frac{1}{max_{x_r \in L} Cos(x_i, x_r)} \tag{3.7}$$

It has been proved that considering any of these properties individually is prone to querying outliers (Settles and Craven, 2008; Hu et al., 2010a). Nevertheless, they can be used together, in which case the most diverse samples are selected from the dense regions of the examples space (Zhu et al., 2008b; Hu et al., 2010a). Hu et al. (2010a) presented an algorithm, called Exploration Guided Active Learning (EGAL), that adopts this strategy. EGAL has a parameter, w, to balance the influence of diversity and density while exploring the examples space.

3.2.2. Pastes Dataset

In this section, we first describe the procedure followed to collect pastes from the Pastebin website. Next, we explain how we extracted a representative subset of the collected pastes.

Pastebin crawling process

Pastebin is one of the most active Online Notepad Services (ONS), receiving more than 18 million visitors monthly (John, 2019). John (2019) claimed that there were more than

95 million active pastes in 2016. The Pastebin website has an option to create a premium account that permits the user to download 250 paste per request via their API interface[10]. The API response is represented in JSON format, whereas each element refers to a unique paste (see Figure 3.4). To download the pure text of a paste, we crawl the content behind the «scrape_url» link. The «title» and «syntax» elements are not included in our study as they are optional and, based on the pastes we reviewed manually, could be empty most of the times or hold misleading text.


```
{
    "scrape_url": "https://scrape.pastebin.com/api_scrape_item.php?i=0CeaNm8Y",
    "full_url": "https://pastebin.com/0CeaNm8Y",
    "date": "1442911802",
    "key": "0CeaNm8Y",
    "size": "890",
    "expire": "1442998159",
    "title": "Once we all know when we goto function",
    "syntax": "java",
    "user": "admin"
},
```


Figure 3.4: The response of Pastebin scraping API

Finding a Representative Subset

Pastebin ONS is an inexhaustible stream of text, and while our crawler is running, it can reach millions of pastes. To collect pastes from Pastebin, we developed a Python script that queries the last 250 pastes and downloads their content into separated text files. Between September 2018 and February 2019, we reached and downloaded 4 million unique pastes, which form the set M. Conducting exploration-based AL on this enormous number of samples at once is computationally expensive (Wei et al., 2015). We therefore resorted to identifying a subset that holds a sufficient amount of instances to represent, as much as possible, all the samples.

To select the adequate size of the representative subset, we followed the next procedure. Given the set M, which holds all the collected samples through Pastebin API, the objective is to find a subset D of size Z, which represents M entries. From the set M, we randomly sample three independent subsets $\{d_1, d2, d_3\}$, each of size Z, and we tokenize their text into words using the white spaces, resulting in three lists of words $\{l_1, l_2, l_3\}$.

[10] https://pastebin.com/api_scraping_faq

The average pair-wise overlapping percentage of these three lists indicates how much D represents M, in which the higher the overlapping, the better the representation.

3.2.3. Methodology

Overview of the System Architecture

There are various types of text in Pastebin, ranging from code snippets or binary encoded files to logs, stories or news. Motivated by the diversity of the text typology, we divided the main classification task into three levels of classification, trying to use adequate techniques to encode every paste depending on the typology of its text. Figure 3.5 depicts the flowchart of our classification framework, including classifiers: Code Snippet Classifier (CSC), Readable/ Non-Readable Classifier (RNC), and Suspicious Activities Classifiers (SAC). In the following paragraphs, we explain the motivation of each one, along with their inputs and outputs.

Since Pastebin was introduced to facilitate the sharing of code snippets between programmers, the majority of its entries are code snippets. The early exclusion of this category in our analysis, using CSC, decreases the error rate of the next classifiers, which increases the final performance in classifying suspicious activities.

The second classifier, RNC, receives all the pastes except the ones that were designated as code snippets by CSC, and separates them into *Readable Text* (RT) or *Non-readable Text* (NT), where RT refers to examples correlated with news, stories or books and NT refers to binary files, encoded text or logs. The tokens of the latter portion, NT, are rarely repeated among the other samples, and most of the time, they are unique. In this case, regular expressions can be used to match patterns, like a memory dump or binary file headers. In contrast, the tokens of the RT are usual words.

The readable portion, RT, passes to the last classifier, SAC, to detect whether it contains suspicious activities or not. The SAC has two outputs: 1) binary, to decide whether an entry is suspicious or normal, and 2) multi-class, that classifies a paste into one of the predefined suspicious categories or says that is not suspicious.

Code Snippets Classifier (CSC)

The CSC binary classifier uses GuessLang library[11] to predict whether an instance contains a piece of code or not. Given a paste, GuessLang library builds a feature vector using Bag of Words followed by feature hashing (Somda, 2019). The extracted features are used to train a machine learning model that merges a linear classifier with a with three layers neural network. It results in a multi-class classifier that predicts the distribution of a sample over 20 programming languages.

We are interested in detecting whether an instance is a code snippet or not. For that reason, we hypothesize that the *Markdown* category, which is a lightweight markup lan-

[11] `https://github.com/yoeo/guesslang`

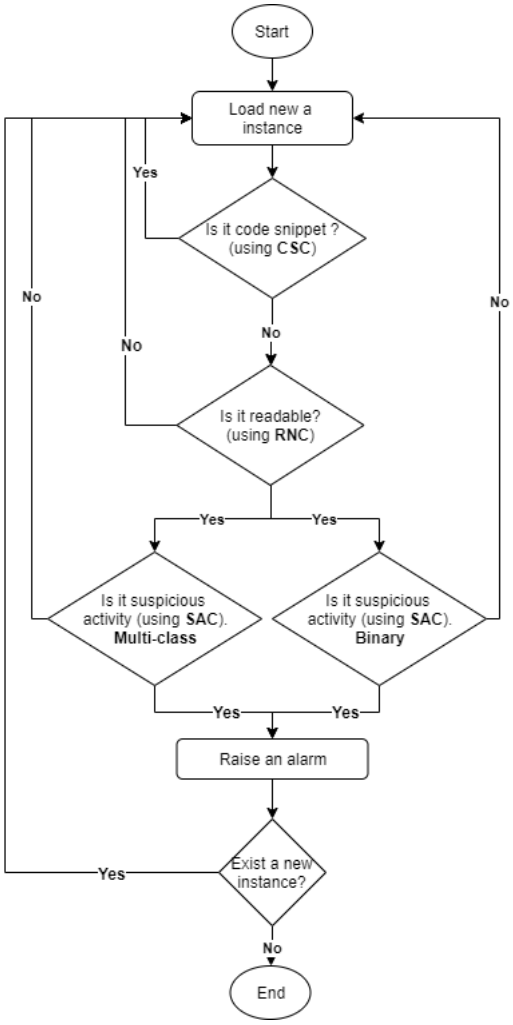


Figure 3.5: A flowchart of Pastebin classification framework.

guage that is used to format plain text into HTML, is the closest category to natural text. Hence, whenever a sample is categorized as any category, apart from Markdown, the classifier considers it as a *code snippet* sample. Otherwise, we consider it as a *not-code snippet*. We validated our hypothesize by manual inspection of 100 pastes that were randomly sampled from U, and we obtained an accuracy of 96%. Precisely, only four samples were natural text but classified as a code snippet.

Readable/ Non-Readable Classifier (RNC)

Building the RNC classifier requires exploring the patterns of RT and NT pastes from Pastebin, and constructing a labeled dataset to train the RNC. Exploration-based active learning is used to overcome this challenge. In our work, we used a variation of the EGAL selection strategy, combining both density and diversity, to explore the Pastebin space given EGAL's superiority as an exploration approach (Hu et al., 2010a). Furthermore, as recommended by EGAL's authors, we pursued the exploration-based phase, followed by another phase of exploitation-based active learning to refine the decision boundary. For the exploitation-based phase, we adopt the Margin Selection strategy, as it far outweighs other strategies, as Haldenwang et al. (2018) have shown experimentally.

To represent the text in the space, we pre-processed the text using regular expressions, replacing URLs and email addresses by fixed tokens, <URL> and <EMAIL>, respectively. Furthermore, to discard noisy tokens, we set thresholds for the minimum and the maximum term frequency. Then, we encoded the text using the TF-IDF technique with char-based 2-grams to 4-grams following the work of Chen et al. (2017). We used a Logistic Regression (LR) classifier for the exploitation-based AL. The reason behind this design - TF-IDF for text representation and LR for classification - is because in our previous work Riesco et al. (2019), it achieved the highest performance in the benchmarked pipelines.

Suspicious Activities Classifiers (SAC)

The SAC includes two classifiers: 1) a binary classifier, which separates normal and suspicious activities; and 2) a multi-class classifier, that classifies an entry into one of seven categories: six suspicious and an extra category we named it *Normal*. We initially used the exploration approach to try to discover the potential classes in the space. However, due to the variety of text patterns and the small number of suspicious activities compare to normal activities, we were unable to discover suspicious categories. Hence, we resorted to the exploitation approach of AL to build a classifier for the RT portion. Unlike the RNC, for SAC we tokenized on the word level, i.e., 1-grams along with TF-IDF and a LR classifier.

3.2.4. Empirical Evaluation

Experimental Setting

All the experiments were carried out on an Intel(R) Xeon(R) 2.4GHz CPU with 128 GB of RAM, using Python3 with Scikit-Learn library. For the LR classifier, we set the regularization parameter $C = 50$, with the balanced class-weight flag activated. Also, for TF-IDF, we set the thresholds of the minimum and the maximum term to 0.001 and 0.999, respectively. Regarding the stopping criterion (SC) of AL procedure, we set an early stopping condition, which is triggered when the performance does not increase after five consec-

utive iterations. Also, we configured the AL strategies to return a candidate set, CS, of 10 samples from the unlabeled pool, U, every iteration, i.e., $b = 10$.

Evaluation Metric

Since the objective is to detect suspicious activities in ONS, we use recall as the performance metric because it is highly sensitive to the missed samples and is suitable measure the performance when the training set is imbalanced (Chen et al., 2017). In the case of multi-class classification, we used the average class recall metric. Practically, with Sklearn library[12], the metric is calculated using *recall_score* method with *Macro* average. In our previous work (Riesco et al., 2019), we referred to this metric by Recall (Macro).

Test Set Specifications

To evaluate the performance of RNC and SAC classifiers, we used the PasteCC_17K dataset (Riesco et al., 2019), building the following test sets.

For the RNC, we created a binary test set of 1,000 elements. The Readable class (RT) has 500 samples collected from the following categories of PasteCC_17K: 217 pastes containing suspicious activities, 90 forums, 10 personal, and 183 multimedia. Likewise, to build the Non-readable class (NT), we resorted to PasteCC_17K, which has only 90 related samples: 65 log file and 25 encoded text. However, after exploring the code snippet samples identified by CSC, with EGAL, we came across new NT patterns that were not in the PasteCC_17K dataset. Hence, we added the following classes manually: 70 encoded text, 60 configuration files, 90 programming exceptions, and 90 samples contain transaction log and memory dumps. Additionally, we appended 100 samples of type «Code Snippet» to the NT category. The motivation behind this addition is to simulate the scenario when CSC fails in detecting code snippets samples and passes them to RNC, whose correct behavior is to identify them as the NT class. In total, the samples of the class NT sum up to 500 instances.

With respect to SAC, we used the readable suspicious and normal activities of PasteCC_17K as the ground truth. Table 3.4 shows the classes of the test set and the number of samples per category. For the categories that had a low number of samples, such as «Counterfeit Personal Identification», we used the search engine of Pastebin to acquire more samples. We developed a python script to download pastes that contain a particular keyword, such as «Fake passports», «U.S. residence card», and «original driving license», and we validated the label of the scraped pastes manually. Also, we introduced the category «Others» into the dataset to cover any non-suspicious paste, i.e., readable but corresponding to normal activities. Table 3.4 shows the classes of the test set and the number of samples per category.

[12] `https://scikit-learn.org`

Table 3.4: The test set specifications of the *Suspicious Activities Classifier (SAC)*. The «C» letter refers to a Counterfeit activity.

Category	# Samples
Others	1,696
C. Credit Cards	54
Hacking	75
Leaked data	103
C. Personal Identification	96
Drugs	72
Pornography	89
Total	2,185

3.2.5. Results and Discussion

In this Section, we present the performance of each component of the classification pipeline, along with a discussion of the obtained results.

Applying the Code Snippets Classifier (CSC)

The CSC recognized that 2,360,000 of M samples are code snippets, 59% of the total, while the left 1,640,000, 41%, are something else. In our previous work (Riesco et al., 2019), we identified that 70% of PasteCC_17K dataset samples were code snippets, which is higher than our current estimation. This difference could be due to the possible error introduced by CSC, who classified some samples as *Markdown* while they were code snippets. However, we overcame this potential flaw in the following classifier, RNC, which can detect any misclassified instances in CSC. The other possible source of this difference could be the low confidence threshold set by Riesco et al. (2019) to build PasteCC_17K dataset, which allowed not-code snippets samples to be classified as code snippets.

Estimating the Representative Subset Size

To estimate an adequate subset size, we followed the procedure explained in Section 3.2.2. We evaluated seven different values of Z, the size of the representative subset, ranging from 1,000 samples to 125,000, and for each subset, we reported its corresponding average overlapping percentage, as shown in Table 3.5. It can be observed that setting Z to 25,000 achieves 86.40% overlapping, while enlarging it to 125,000 introduced a slight increase to 88.90% (see Figure 3.6). This stability is explained by the existence of unique tokens, like passwords, email addresses, and user names, which are most likely unique in the dataset. Hereafter, the selected representative subset, D, will be used rather than the entire dataset, M.

In addition to the overlapping metric, we estimated the time needed to calculate the

Table 3.5: Variation of the average overlapping when increasing the size of the representative subset.

Subset Size (Z)	Average Overlapping (%)
1000	23.82
5,000	66.31
10,000	75.95
25,000	86.40
50,000	87.54
75,000	88.85
125,000	88.90

diversity and the density of each subset size[13]. Figure 3.6 shows that estimating the diversity and the density of a subset of 25,000 elements took 67 and 173 seconds, respectively. However, the time curve steadily increased when we expanded Z to 125,000 elements, and consumed 1,201 and 2,621 seconds, respectively. Hereafter, the selected representative subset, D of size $Z = 25,000$ elements, will be used rather than the entire dataset, M.

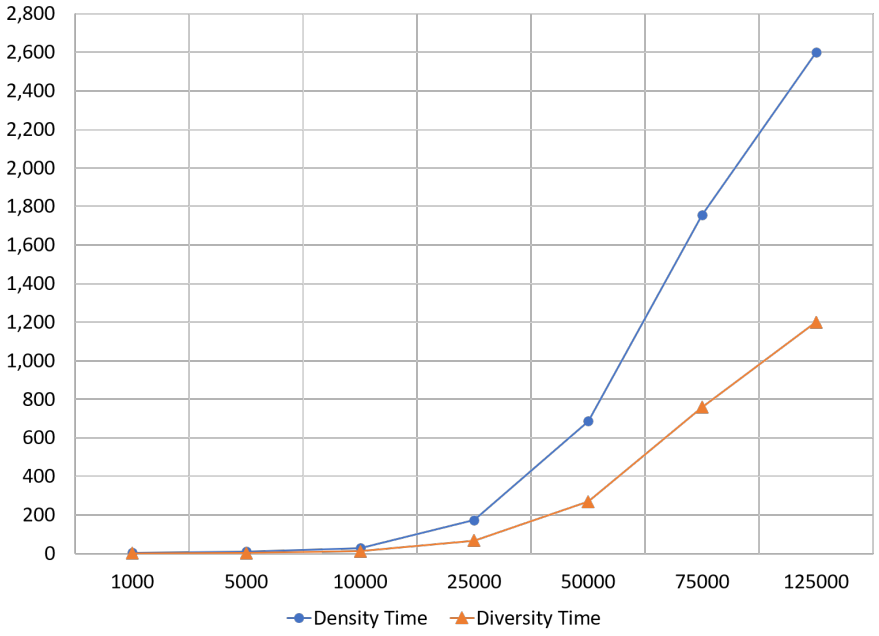


Figure 3.6: The representative subset estimation curve. The Y-axis refers to the consumed time (in seconds) to estimate the diversity and the density of a given subset, while the X-axis indicates the size of the subset.

[13]We estimated the diversity by considering that only 5% of the dataset is labeled.

Applying the Readable/ Non-Readable Classifier

To initialize the AL procedure, we use the Agglomerative Hierarchical Clustering (AHC) algorithm to group the unlabeled pool, U, into 10 clusters. The clusters' centers are labeled by the oracle O. Figure 3.7 shows that training the RNC with the initialization batch the average class recall is only 50.28%. To select the informative samples from the unlabeled pool, we compare three sample selection strategies: the exploration-based and the exploitation-based, and a hybrid one, that combines both. Below, we report the performance of each strategy.

- Exploration-based. In this configuration, we carried out 22 iterations of EGAL, resulting in 230 labeled samples, including the initial training batch. In Figure 3.7, the blue curve - with triangles - shows that after labeling 130 samples, the curve increases slightly with an average class recall of 65.65%. Then, by exploring 30 samples more, the recall rises sharply to reach 91.84%. We justify this sudden recall rise by the structure of the explored dataset. Figure 3.7 shows that between the point 20 to 130, the algorithm was stuck in exploring only distinct patterns of the NT class and that what we observed while labeling the returned selected samples. Next, between 130 and 160, the algorithm came across new patterns of the RT category, resulting in boosting the average class recall measure. After labeling 180 instances, the curve started oscillating around the value of 92% with a maximum of 92.84% at 180. The stability triggered the stopping condition to terminate the exploration process after five iterations and having 230 samples labeled. Hence, the first 180 samples from the labeled set, L, are the most informative instances selected using this approach, to train RNC.

- Exploitation-based. In this setting, we apply the Margin Sampling (MS) selection strategy with 14 iterations, resulting in 150 labeled samples, including the initialization batch. Figure 3.7 shows that the exploitation average recall curve - gray diamonds - increases gradually after labeling 50 samples and even surpassed the exploration-based curve with an average class recall of 71.02%. However, this increase did not last for further iterations and became almost trivial to reach 72.70% at 100 samples. Finally, the stopping criterion was triggered at point 150 with a performance of 69.17%. Thus, this approach entailed labeling only 100 samples to obtain a result of 72.70%.

- Hybrid. As recommended by the authors of EGAL, we pursued the exploration-based AL with a few iterations of the exploitation-based AL. We started exploiting the space using MS at point 180, i.e., where EGAL stopped. Figure 3.7 illustrates a slight increase in the average class recall to 93.95%, the highest value achieved at point 200. However, the next five iterations did not yield higher performance, resulting in triggering the stopping condition at point 250.

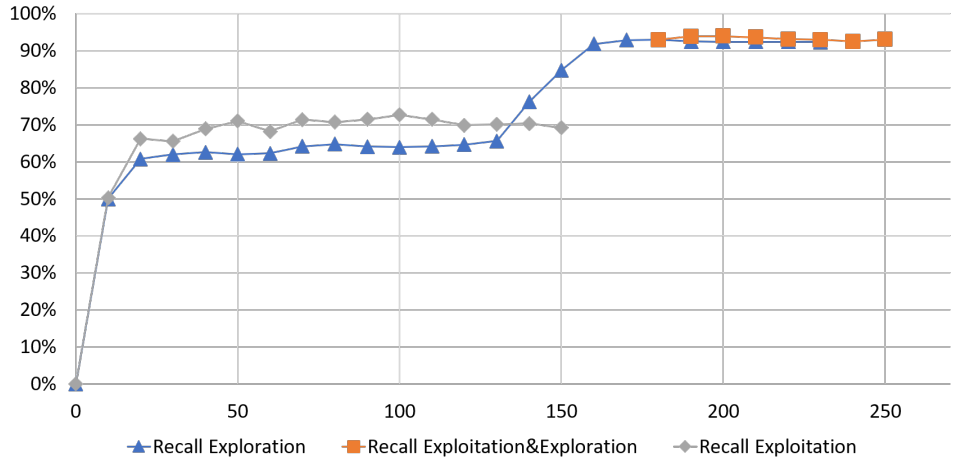


Figure 3.7: Comparison of the three AL approaches to construct Readable/ Non-Readable Classifier (RNC). The X-axis refers to the number of labeled samples, while the Y-axis presents the average class recall of the RNC.

Applying the Suspicious Activities Classifiers

In order to discover the potential suspicious activities in Pastebin, we adopted the exploration-based AL approach for building the Suspicious Activities Classifiers (SAC). Likewise, for the RNC, we used Agglomerative Hierarchical Clustering (AHC) with 10 centroids to build the initial batch and started exploring the space. Figure 3.8 shows that both the samples selected using AHC and the ones selected by Exploration Guided Active Learning (EGAL), i.e., the first 20 examples, are related to the same class «Others», i.e., normal activities. Hence, we could not estimate the performance for the first two iterations, and we set it to zero. Later, in the third cycle, after having 30 samples labeled, EGAL detected a new suspicious pattern, which incremented the number of the explored classes by one.

Furthermore, with more iterations of EGAL and having 70 instances labeled, the algorithm recognized two new patterns, summing up four the number of classes at this point. It is noticeable that the curve showing the average class recall was declining as new patterns were being discovered, as it reached 25%. The reason behind this behavior lies behind having suspicious activities as a minority comparing to the normal activities. In other words, EGAL was continuously discovering various types of patterns, but all were normal, resulting in increasing the normal training samples, while the suspicious ones were not. Thus, the training samples were sharply skewed to the «Others» classes, leading the classifier to assign this category to any new sample.

In addition to the exploration-based, we examined the exploitation-based approach. We initiated the MS selection strategy with a portion of 20% of the test set, which forms

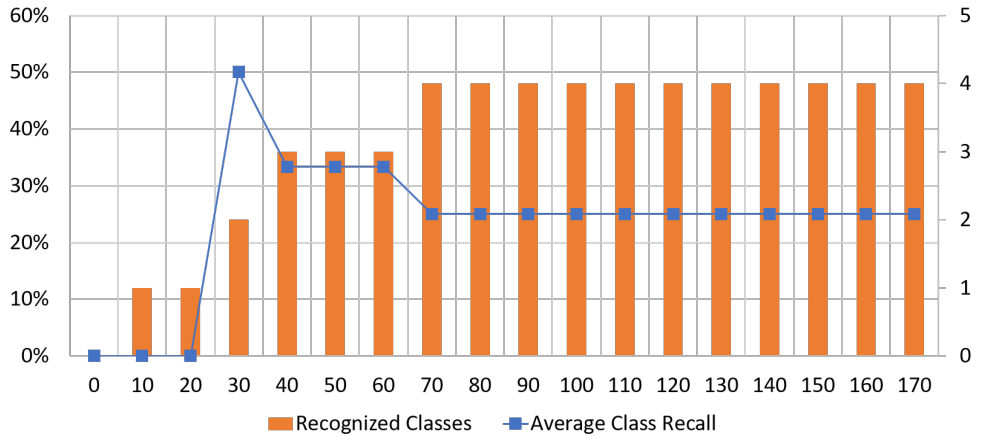


Figure 3.8: Exploration-based AL for the RT portion of Pastebin. The X-axis refers to the number of the labeled samples. The left Y-axis shows the average class recall metric and the right one indicates the number of the recognized categories at that point.

437 samples, to train SAC, and tested on the left 80% of the set. As shown in Table 3.4, the dataset has seven categories: six suspicious plus the «Others» class. Figure 3.9 shows that when the SAC classifier was trained only with the initialization batch, it obtained an average class recall of 75.97%. Then, after 16 iterations and 160 samples were labeled, the performance increased to 80.33%. It is worthy to note that after having 530 samples labeled, the curve stopped increasing, but oscillated around the value of 80%, which triggered the stopping criterion.

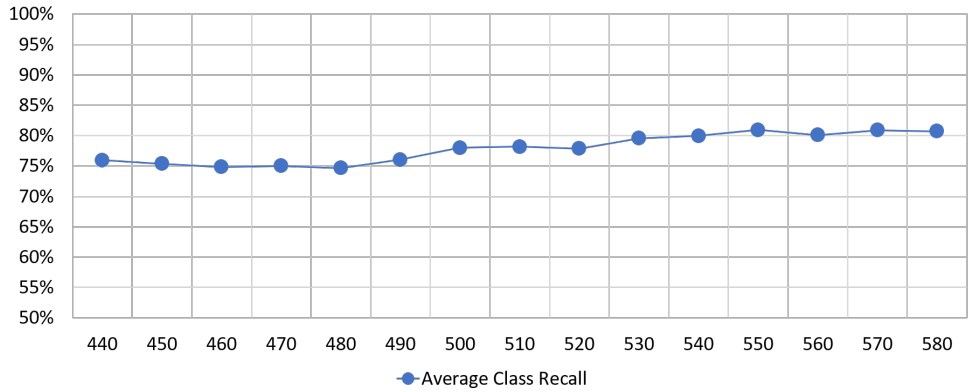


Figure 3.9: Exploitation-based AL for the RT portion of Pastebin. The X-axis refers to the number of labeled samples, while the Y-axis shows the average class recall.

By reviewing the misclassified samples, we realized the existence of distinct categor-

ies sharing common tokens. For example, a sample has the text «We sell hacked credit cards for low prices» and «cheap hacked VISA cards», while another sample that leaks critical information has the text «Leaked Master Card CVV». Such samples fall between two categories, like «Counterfeit Credit Cards» and «Hacking» for the first example, and «Counterfeit Credit Cards», and «Leaked-data» for the second one, reduced the classifier precision. Moreover, a paste might be involved in more than one activity at the same time, like a marketplace. To overcome this limitation, we decided to support the multi-class classifier with a binary one to predict whether a paste is suspicious or not. Another advantage of this design is that it could detect new suspicious activities, even if they were not predefined previously.

We followed the same procedure of the multi-class classifier to build the binary one. We initialized it with 20% of the test set, and we started the exploitation-based AL process. Figure 3.10 shows that by training the classifier with only 437 samples, it obtained an average class recall of 94.21%. Next, after 10 iteration, when 540 samples were labeled, the classifier achieved a slight increase to 95.24%, whereas the stopping criterion was triggered.

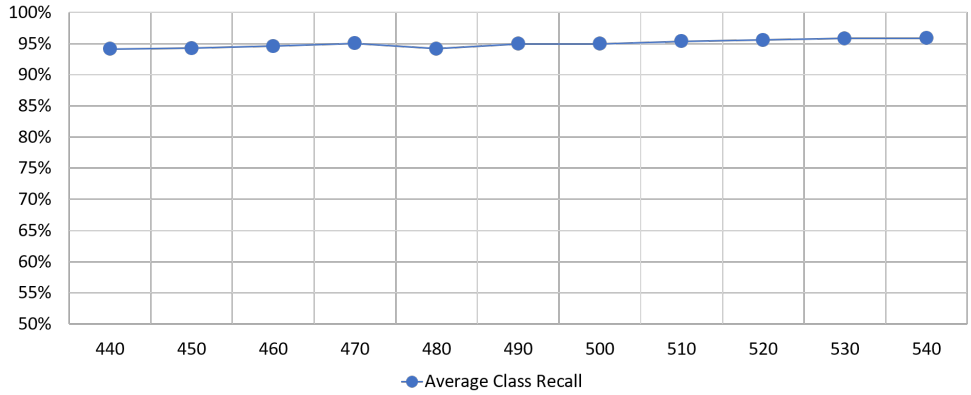


Figure 3.10: Exploration-based AL for the RT portion of Pastebin. The X-axis refers to the number of the labeled samples, while the Y-axis shows the average class recall.

Evaluation of Binary and Multi-class classification in SAC

To decide whether to use a binary, a multi-class, or their combination for the SAC, we conducted the following experiment. We used both classifiers to predict the samples left in U and carried out the next experiment on them. We compared the percentage of the suspicious and the normal activities detected by each classifier, as shown in Figure 3.11. The binary classifier recognized 6.48% of the activities as suspicious; meanwhile,

the multi-class classifier detected only 5.53%[14]. Hence, the binary classifier has a higher recall of suspicious pastes. The superiority of the binary one indicates its ability to detect more suspicious activities than the multi-class one.

We also compared the intersection ratio of the samples that were categorized as suspicious by both classifiers (see Figure 3.11). We found that 87.82% of the suspicious activities were in common between both of them. The superiority of the binary classifier and its high average class recall allowed us to strengthen our confidence in its prediction.

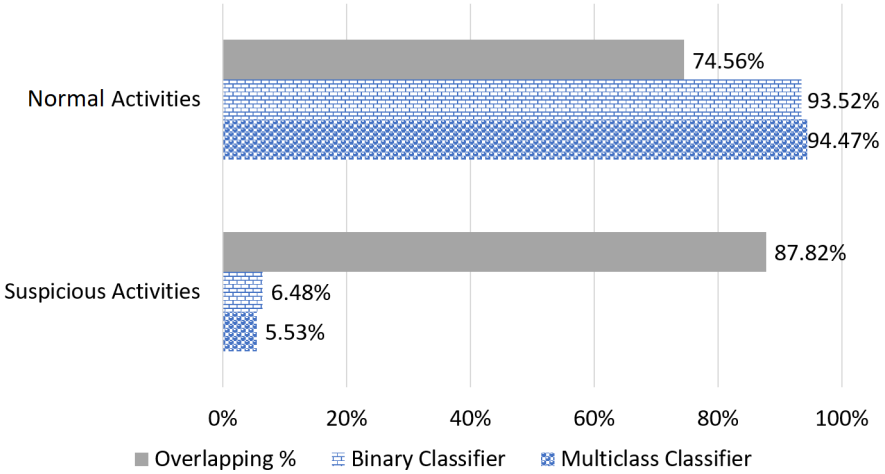


Figure 3.11: The percentage of the normal and the suspicious activities using the binary and the multi-class models after initializing the classifier with 20% of the labeled samples. The solid gray bar refers to the overlapping rate.

In the previous experiments, we used only 20% of the labeled samples to initialize the exploitation-based AL algorithm. However, in the production phase, we would use all the labeled samples for the initialization. Hence, this configuration has no test set to evaluate the performance. Figure 3.12 compares the percentages of the suspicious and the normal activities along with their overlapping using all the labeled samples for both classifiers. Again, it emphasizes the superiority of the binary classifier over the multi-class one for detecting suspicious activities. It worth mentioning that in this case, using all the labeled samples, the overlapping dropped to 59.83% because the binary classifier recognized even more examples as suspicious than the multi-class one.

Furthermore, we carried out a manual validation for both classifiers. We randomly sampled 100 pastes of the predicted pool using both models and validated them manually. For the binary one, 91% of the samples were predicted correctly, while for the multi-class one, only 78%. Moreover, by comparing the number of suspicious samples recognized

[14]To compare the multi-class classifier with the binary classifier, we considered that the activities which are not tagged as «Others» are suspicious.

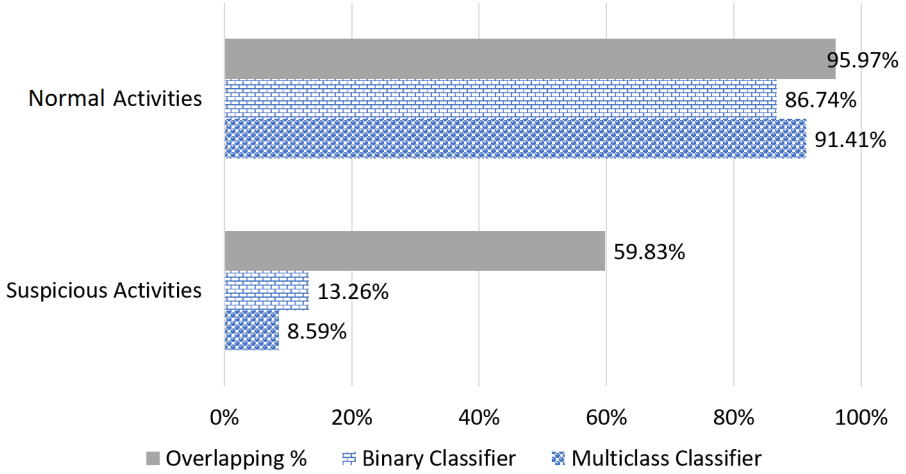


Figure 3.12: The percentage of the normal and the suspicious activities using the binary and the multi-class classifier after initializing the classifier with 100% of the labeled samples. The solid gray bar refers to the overlapping rate.

only by the binary classifier, we counted 5,163 pastes. In contrast, the suspicious samples discovered by the multi-class classifier were 1,634. Hence, the binary classifier is capable of detecting more new patterns of suspicious samples than the multi-class one.

Given the advantages of each model, and with the purpose of reducing false positives, we recommend considering a sample suspicious only when both models agree in this decision. After that, to know the specific category, the multi-class model will provide its prediction.

Active Learning versus Random Sampling

AL has proved its superiority and efficiency over random sample selection for building a training set (Hu et al., 2010a). In our previous work (Riesco et al., 2019), we trained a supervised classifier with 12,348 samples (70% of PasteCC_17K dataset). The classifier, which is similar to SAC multi-class architecture, achieved an average class recall of 72% despite the enormous number of its training set. In contrast, the presented classification pipeline obtains an average class recall of 80.33%. The classifier proposed in this work used only 437 labeled samples to train the SAC (20% of the labeled samples) and another 180 labeled samples to train the RNC, 617 samples in total, which is a training set 20 times smaller than the one used in Riesco et al. (2019). Our findings emphasize the usefulness of AL in terms of saving effort and labor time for building a labeled dataset.

3.3. Conclusions

In this chapter, we presented several contributions. The first one is a text classification pipeline to categorize the suspicious activities of Tor Darknet. In particular, we explored two text representation methods, TF-IDF and BOW, combined with three classifiers, SVM, LR, and NB. We also created a new dataset, Darknet Usage Text Addresses (DUTA), containing 6,831 samples labeled manually into 26 categories. To train a supervised model that detects suspicious activities in the Tor network, we selected eight classes of DUTA: Pornography, Cryptocurrency, Counterfeit Credit Cards, Drugs, Violence, Hacking, Counterfeit Money, and Counterfeit Personal Identification, in addition to an extra class, called *Others* to reflect other possible activities. Furthermore, we distinguished the critical aspects that affect the classification results in terms of text representation, i.e., the dictionary size and the minimum word frequency, and the regularization parameter on the LR and the SVM classifiers. We found that the combination of the TF-IDF text representation with the Logistic Regression classifier achieves 96.6% of accuracy over 10-fold cross-validation and a 93.7% macro F1 score. We noticed that our classifier suffers from overfitting due to the difficulty of reaching more samples of onion hidden services for some classes like counterfeiting personal identification or illegal drugs.

Second, we designed and presented an end-to-end supervised classification pipeline using Active Learning to recognize suspicious activities in Pastebin, probably the most used and known Online Notepad Service (ONS). Our model is made up of three sequential classification levels: 1) Code Snippets Classifier (CSC), a binary classifier to separate the pastes containing code snippets from other pastes; 2) Readable/ Non-Readable Classifier (RNC), another binary classifier to select the readable natural text; 3) and Suspicious Activities Classifiers (SAC), both a multi-class and a binary classifiers to detect suspicious activities in Pastebin. We found that using 25,000 readable pastes we are representing up to 86% of Pastebin entries. On top of this subset, we adopted two AL approaches: exploitation-based and exploration-based, to select the most informative samples for training. Selecting only 180 pastes following the exploration approach, we can obtain an average class recall of 92.84%. Furthermore, by initializing an exploitation-based AL algorithm with 20% of the labeled samples, we obtained a binary classifier and another multi-class classifier with an average class recall of 95.24% and 80.33%, respectively.

Overall, this study strengthens the use of AL as a competitive solution to classify an extensive collection of unlabeled data. The method proposed here to classify Pastebin contents could be applied to other ONS elsewhere in the web and to the Darknet networks. Despite the limitation of the exploration approach in finding a new suspicious pattern, the exploitation approach offers a highly reliable recall rate. In the future, we will examine transformer-based models, such as BERT (Luo et al., 2018), RoBERTa (Liu et al., 2019), and XLNet (Yang et al., 2019) for text classification, as it showed promising results on various NLP tasks. Also, to overcome the overfitting issue, we are planning to investigate oversampling techniques, which fits with ML classifiers (Chen et al., 2018).

Chapter 4

Text mining

In this chapter, we propose two text mining techniques. First, a novel algorithm to detect emerging products in the marketplace domains of the Tor network. Second, an algorithm to recognize named entities in noisy user-generated text.

4.1. Detecting Emerging Products in the Tor Network

In this work, we present a semi-automatic algorithm to detect emerging products in the marketplaces of the Tor Darknet. Applying data mining techniques to purchase logs of the markets can solve the problem efficiently (Raeder and Chawla, 2011). However, when these logs are inaccessible, as is the case of the marketplaces of the Tor network, alternate solutions should be sought. Our algorithm aims to overcome this shortage by using the relationship between the offered products rather than studying their purchase logs. We model the problem with an undirected graph of nodes and edges. The nodes represent the products, and an edge between two products is created to capture their existence in the same marketplace. Subsequently, we decompose the nodes of the graph into groups using the K-shell algorithm (Carmi et al., 2007), whereas the focus of our algorithm is on the core-shell, through which emerging products can be identified (see Figure 4.1). We assessed our algorithm on a set of onion domains that offer drug products extracted from the DUTA dataset (Al-Nabki et al., 2017a). The results are quantitatively justified by comparing them with the most recent reports published by international drug organizations, and visually by analyzing the products correlations graph. The proposed framework might help in identifying seasonal trends and emerging suspicious products in different personal shops and marketplaces located in the Tor network.

4.1.1. Background

Our algorithm depends on both Graph Theory and data mining techniques. This Section reviews the concepts of *degree centrality* of a node in a graph, and *item support* and *confidence* that we borrow from the data mining field. Moreover, we present an overview of the k-shell decomposition algorithm.

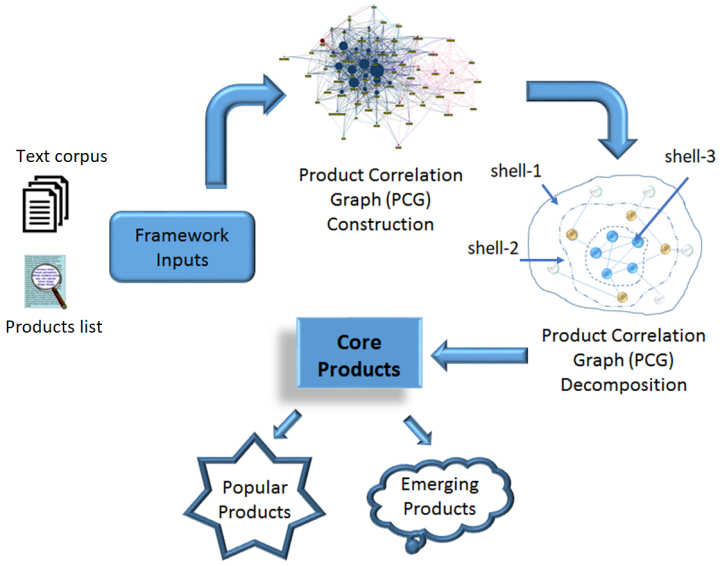


Figure 4.1: Stages of the emerging product detection algorithm. We initialize the algorithm with a corpus and a list of products. Then, we model this problem by an undirected graph, called Products Corrections Graph (PCG), which is decomposed into shells using the k-shell algorithm. Finally, the products of the core-shell are designated as popular or emerging products.

Degree centrality

For a given node in a graph, it is defined by the number of its direct neighbors, and it is calculated by counting the number of edges associated with the studied node (Seidman, 1983), as shown in Figure 4.2.

Item support and confidence

In this work, we borrow a few terms from the literature of data mining, and in the following, we explain them to pave the road for the reader (Zhang and Zhang, 2002). An *itemset* is made up of two or more items, which can be products, for example. Hence, the items of each transaction form an itemset. An *association rule* is an implicit relationship that links two independent items in the itemsets, and it consists of two parts: an antecedent *if* and a consequent *then*. Typically, these rules are drawn from transactional data, such as the purchase log of a supermarket, by searching logs for frequent if-then patterns. An example of a market basket analysis rule is: if a customer buys tea, then most probably he/she is going to buy sugar as well. To identify the most important rules, two more criteria are required, called *support* and *confidence*. The support measures how frequently the items appear in the data, while the confidence the number of times the condition if-then has been satisfied.

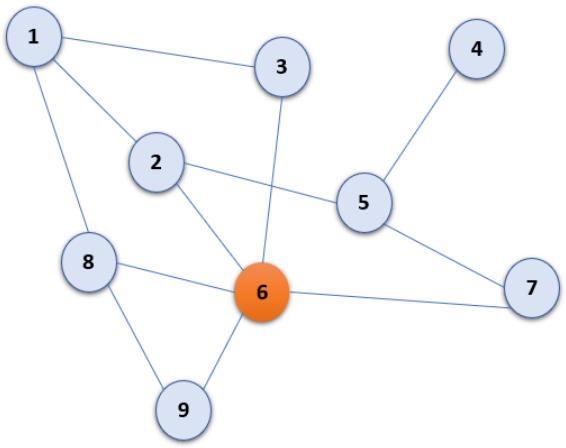


Figure 4.2: Illustration of the node degree metric in an undirected graph. The node colored in orange color (marked with number six) has a degree centrality of five.

Given a set of transactions, $T = t_1, t_2, ..., t_n$, of length n, in which each transaction forms an itemset, and being D an itemset of m items $D = d_1, d_2, ..., d_m$, the support of an item d_i is calculated using Equation. 4.1, while the confidence is estimated using Equation 4.2.

$$Support(d_i) = \frac{F(d_i)}{n}, \tag{4.1}$$

where $F(d_i)$ refers to the frequency of d_i in the itemsets, and n is the number of transactions.

$$Confidence(d_i) = \frac{Support(d_i \cap d_j)}{Support(d_i)}, \tag{4.2}$$

being $Support(d_i \cap d_j)$ is the support of having the products d_i and d_j together.

In the following, we explain how we adopted the above definitions to the context of our problem, where the transaction logs are missing. We have to remember that the objective of this work is to identify the emerging product on the onion domains which offer drug products in the Tor network, not per domain. Hence, we can consider that each onion domain is a transaction, and the products of that domain form the itemset.

Graph Decomposition Algorithm

k-Shell algorithm is based on k-core algorithm (Miorandi and De Pellegrini, 2010). Let $G = (N, E)$ be a graph where N are the nodes and E are the edges, $|N| = n$ and $|E| = e$. The

k-core of G is defined as a sub-graph $H = (C, E|C)$ induced by the subset $C \subseteq N$ where all the nodes have a degree of at least k. In other words, the sub-graph H will have an order k when the condition in Equation 4.3 is satisfied.

$$\forall N \in C : degree_H(n) \geq k \tag{4.3}$$

A practical description of how the k-shell algorithm decomposes a graph into shells is as follows (see Figure 4.3). The algorithm works in a recursive manner. First, it eliminates all nodes with degree $k = 1$, and it assigns them to the shell one. This procedure continues in a recursive manner until there are only nodes with degree $k \geq 2$ in the graph. Then, the algorithm removes all the nodes with degree $k = 2$, and assigns them to the shell two. The eliminating process continues recursively until only nodes with degree $k \geq 3$ are left in the graph, and so on. This process is repeated as long as the graph still have nodes, and when it stops, each node will be associated with an integer shell value.

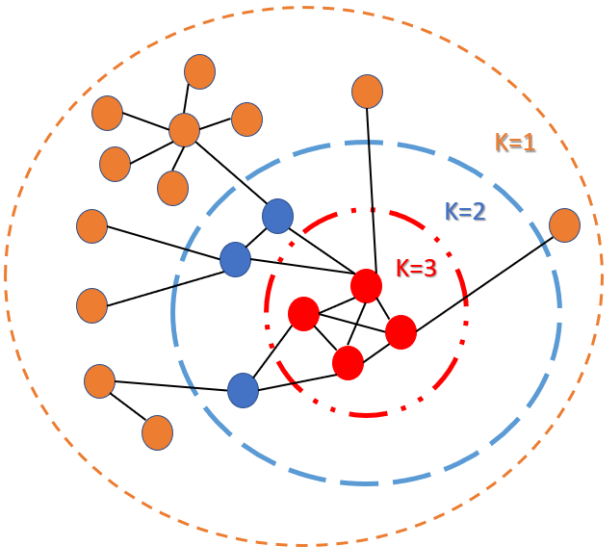


Figure 4.3: Illustration of the decomposition of a graph using k-shell algorithm. Each ring forms a shell, and the graph is decomposed into three shells.

4.1.2. Methodology

The Emerging Product Detection (EPD) has two inputs (1) text corpus and (2) a list of products related to the field of study. The EPD algorithm employs Graph Theory to build the Products Corrections Graph (PCG) and uses the k-shell algorithm to decompose it into levels (shells).

The Framework Inputs

Our algorithm is semi-automatic and receives two lists as input. The first one contains product names related to the targeted field, and it is prepared by an expert. For example, to detect emerging drug products, the algorithm needs a list of drug names, like Cocaine, Hashish, or Weed. The second input is a text corpus, such as dumps of marketplaces websites, where the algorithm is going to carry out the analysis. Then, by looking up the product names - the first input - in the corpus, we can guess the products which appear in each sample of the corpus.

Construction of the Products Correlations Graph

Once the algorithm receives the inputs, it constructs a Products Correlations Graph (PCG), a weighted undirected graph that consists of nodes and edges. A node refers to a product, and it is associated with three parameters. They are (i) the product frequency in the corpus, (ii) the degree centrality of the product node in PCG, and (iii) the support of the product, which is measured as we explained in Section 4.1.1. An edge is added between two products when they exist in the same sample of the corpus, i.e., offered together in the same marketplace, at least once. We weight an edge by its frequency, following to the number of times that the nodes of that edge are present together in the corpus.

However, the above design has an apparent glitch when facing a marketplace with an enormous number of products. For example, a leading drug marketplace, called *Dream Market*, has more than 12,000 products, and around 50% of them are drug-related (Deepwebsiteslinks, 2019). Following the above procedure, we have to create a tremendous number of edges for each pair of products, with a weight equals to one. To address this flaw, we provide the PCG model with two parameters to avoid outliers in terms of the edges and the nodes. The first one eliminates the edge whose weight is less than or equal to α, and we set it to the average weight of all the edges. The second parameter controls the presence of a node in the PCG according to its *supports*. If the support of a node is less than a threshold, β, the node is eliminated. Typically, we set β to the average support of all the nodes.

Algorithm for Detecting Emerging Products

After building the PCG, we apply the k-shell algorithm to decompose the graph into shells. The Emerging Product Detection (EPD) algorithm tackles only the core-shell, as it contains products of high degree regardless of their frequency.

The EPD algorithm separates the products of the core-shell into two groups: *popular products* and *candidates for emerging products* (see Figure 4.4). Since both categories are in the core-shell, the major difference between them is the weight of the node, i.e., its frequency. If the weight of a node is greater than a threshold, γ, we consider the product as popular; otherwise, we categorize it as a candidate to be an emerging product. We set

the threshold γ to the average weight of the nodes contained in the core-shell. Finally, we sort the products categorized as candidates according to the sum of their weighted edges with the popular products in descending order. The products at the head of the list are most likely emerging products, while the ones at the tail are not.

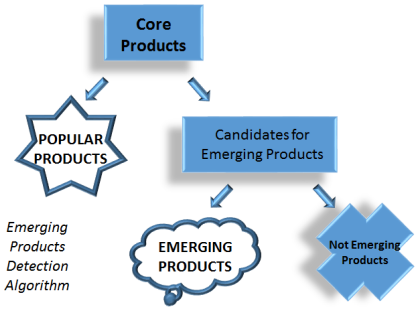


Figure 4.4: Visual representation of the emerging product detection algorithm

4.1.3. Experiments

Experimental Setting

We conducted the experiments on an Intel Core i7 PC with 32GB of RAM. For the graph construction, we used Python3 with the NetworkX library[1]. Concerning the chart visualization, we sorted to the vis.js library[2]. In this work, we identified the emerging products related to drugs. We extract the text corpus from a subset of the DUTA dataset (Al-Nabki et al., 2017a)[3]. In the experiments, we considered the drug products which fall under two categories in DUTA: Drugs - Illegal or legal - and Black Marketplaces. The selected subset has 302 unique onion domains. However, after removing the onion domains that host a duplicated content, only 197 are left, that form our corpus. The list of drug names was extracted using a Python script from the following websites: drugs.com[4] and druginfo[5]. In total, we collected a list of 862 unique entries.

4.1.4. Results

Key Findings of Drug Products

Before analyzing emergent products, we introduce some results and key findings derived from our study. First, we ordered the Products Correlations Graph (PCG) nodes,

[1] https://networkx.github.io/
[2] http://visjs.org/
[3] We conducted this analysis on an enhanced version of DUTA, version 1.1
[4] https://www.drugs.com/alpha/a1.html
[5] https://druginfo.nlm.nih.gov/drugportal/drug/names

the products, according to their weights. We found that the most popular products are MDMA, LSD and Cannabis, offered by 24%, 18% and 17% of the Tor marketplaces, respectively. Figure 4.5 depicts a list of the top-20 most popular drug offered during the crawling period when DUTA was created.

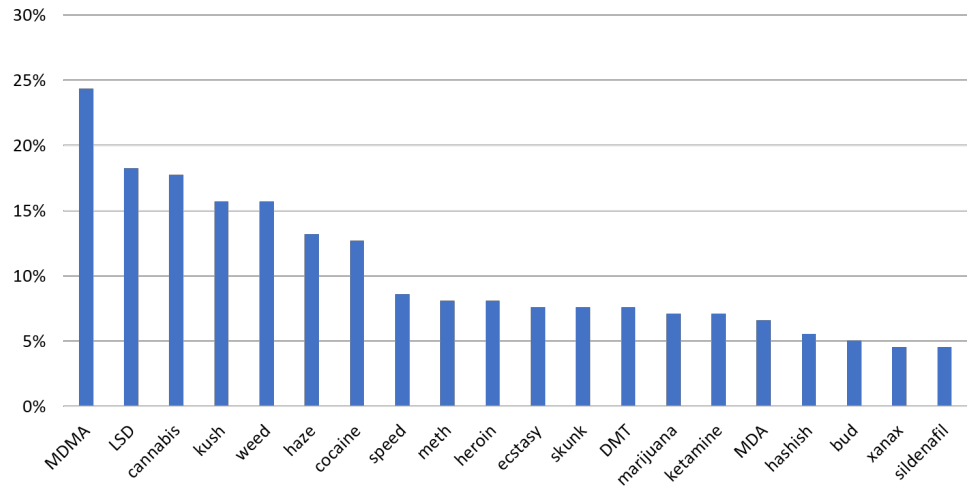


Figure 4.5: Top-20 popular drug in DUTA dataset. The X-axis refers to the products, while the Y-axis refers to the percentage of their presence in the corpus.

The previous findings are compatible with the results of the Global Drug Survey (Globaldrugsurvey.com, 2016). They indicated that these three products are the most commonly bought in the Dark Web during their analysis period (i.e., November-2015 to January-2016). Also, we discovered the tightly associated products, which appear together frequently in the markets. We sorted the edges by their weights, whereas the higher the weight, the stronger the association between the products of that edge. Figure 4.6 shows the top-20 most popular pairs of products. We can see that the pair (LSD, MDMA) is the most frequent as it is present in 12.7% of the markets, while in the second place is the (Weed, Haze) pair, which appears in 8.7%.

Moreover, we explored the confidence of the frequently associated pairs (see the definition of the confidence in Section 4.1.1). We represent the confidence of an association rule as a $(A \rightarrow B)$ relation, and this relation is interpreted as how frequent the product B is offered together with the product A. We observed that the highly frequent pairs of products have high confidence and vice versa (see Table 4.1).

Analysis of the Emerging Drugs in the Tor Network

We extracted 395 different drugs, i.e., nodes, with 38,347 mutual offering relations between them. Afterwards, we set the thresholds α and β to 1 and 0.153, respectively, as

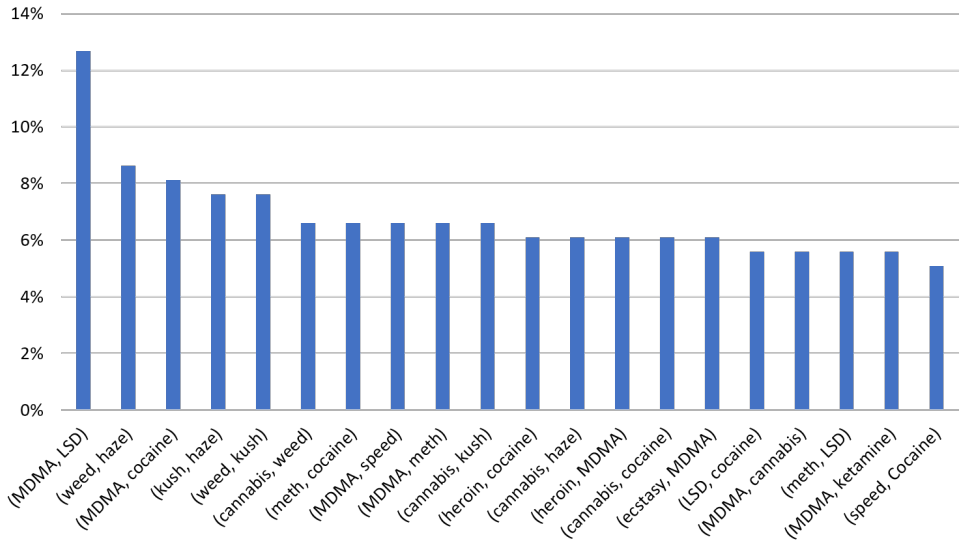


Figure 4.6: Top-20 famous pair of products in the drug marketplaces of DUTA dataset, whereas the most frequent pair is (MDMA and LSD) drug. The X-axis shows the identified pairs, while the Y-axis refers to the percentage of their presence in the dataset.

explained in Section 4.1.2. After filtering the PCG with the previously thresholds, it obtained 66 nodes and 797 edges with a density of 0.371. The k-shell algorithm decomposed the PCG into 12 shells, where the core-shell holds 27 products.

Following the pipeline detailed in Figure 4.4, we estimated the popular and the emerging products of drug-related activities in the DUTA dataset. However, since we do not have a ground truth for emerging drugs, we evaluated the results qualitatively by comparing it with reports published by international organizations. We selected reports that present statistics, approximately in the same period of time when DUTA was built. Table 4.2 shows the products of the core-shell, whereas these products are candidates to be emerging. In particular, the core-shell holds 27 products ordered by their weights. The algorithm selected nine of them as emerging products because their weights are higher than the average. The results indicate that the *Ecstasy*, which is a famous member of the MDMA drug family, is the most emerging drug during the crawling period of DUTA, and it is followed by the *Ketamine* and the *Dimethyltryptamine (DMT)*.

We validated our findings with three public studies and reports. The first one is a study published by the European Monitoring Center for Drugs and Drug Addiction (EMCDDA) (Arrigo and Goosdeel, 2016), which reported an increase in the use of Ecstasy in adults during 2016. The second one is a report from the BBC News titled *The growing popularity, and potency, of Ecstasy and MDMA* (News, 2017), which emphasized on the growing usage of the Ecstasy drugs. Also, a study established by Global Drug Survey Globaldrug-

Table 4.1: Confidence analysis of the drug products

Rules	**Confidence**	**Rules**	**Confidence**
MDMA → LSD	0.81	LSD → MDMA	0.69
Weed → Haze	0.80	Haze → Weed	0.65
MDMA → Cocaine	0.75	Cocaine → MDMA	0.64
Kush → Haze	0.72	Haze → Kush	0.75
Weed → Kush	0.69	Kush → Weed	0.75
Cannabis → Weed	0.59	Weed → Cannabis	0.75
Meth → Cocaine	0.55	Cocaine → Meth	0.70
MDMA → Speed	0.52	Speed → MDMA	0.67
MDMA → Meth	0.48	Meth → MDMA	0.65
Cannabis → Kush	0.48	Kush → Cannabis	0.60
Heroin → Cocaine	0.37	Cocaine → Heroin	0.55
Cannabis → Haze	0.37	Haze → Cannabis	0.54
Heroin → MDMA	0.34	MDMA → Heroin	0.25
Cannabis → Cocaine	0.34	Cocaine → Cannabis	0.43
Ecstasy → MDMA	0.33	MDMA → Ecstasy	0.43
LSD → Cocaine	0.31	Cocaine → LSD	0.44
MDMA → Cannabis	0.27	Cannabis → MDMA	0.28
Meth → LSD	0.27	LSD → Meth	0.31
MDMA → Ketamine	0.23	Ketamine → MDMA	0.19
Speed → Cocaine	0.23	Cocaine → Speed	0.19

survey.com (2016) discussed the raising of the DMT and the Ketamine, the second, and the third emerging drug detected through our algorithm.

Table 4.2: The emerging drug products in the core-shell for a subset of DUTA dataset

Core-Shell Products	**Candidates for Emergent**	**Selected Emerging Products**
Marijuana, DMT, Edibles, Ecstasy, XTC, Cocaine, MDMA, Speed, Heroin, Weed, Mxe, 2C-B,,Haze, Mescaline, GHB, MDA, Benzos, Cannabis, LSD, Mushrooms, GBL, Meth, Hashish, Amphetamine, Ketamine, Psychedelics, Kush	Ecstasy, Ketamine, DMT, MDA, Marijuana, Hashish, Mescaline, Psychedelics, Amphetamine, GHB, Mushrooms, 2C-B, Edibles, GBL, Mxe, Benzos, XTC	Ecstasy,Ketamine, DMT, MDA, Marijuana, Hashish, Mescaline, GHB, XTC

In addition to the quantitative analysis, we present a visualization of the Products Correlations Graph (PCG) since it might help in understanding the underlying relationships between the products in the graph. In the graph - better viewed in color -, we colored the

nodes based on their shells, while the size of the nodes depends on their frequencies in DUTA. Figure 4.7 shows the complete PCG. It illustrates that the most popular products are located in the center since they have the highest connections with the other nodes, while the lower connectivity nodes are in the graph borders. However, having a small node in the center indicates that this node tends to grow since it appears connected with the popular ones within the same shell.

For further clarification, we took a sub-graph of the PCG that contains the core-shell nodes only (see Figure 4.8). Then, we highlighted the Ecstasy drug, which occupied the first position among the emerging products. This Figure shows 10 favorite products, i.e., popular, and 17 candidate ones. By applying the EPD algorithm, we found that Ecstasy is offered with all the products contained in the popular list. The total sum of its edges with popular products was 68, while the *Benzos* drug is provided only with seven popular, and the sum of its weighted edges was 15. Therefore, the Benzos was excluded from the emerging product list. In contrast, the Global Drug Survey Globaldrugsurvey.com (2016) has reported that the 2C-B and the DMT are both emerging, and our framework detected only the DMT, but not the 2C-B. We suspect that this is due to the difference in the study sample and the followed criteria for nominating a drug as emergent. In ours, we have the drug 2C-B in the core-shell as a candidate for emerging drugs. But, in this case, it could not evolve to the final emerging drugs list because it had only six connections with the popular product, a value lower than the chosen threshold. In the case of DMT, the weight 10 is higher than the threshold we used to propose the final emerging product list, i.e., the average weight of their edges, so we nominated DMT as a formal emerging drug.

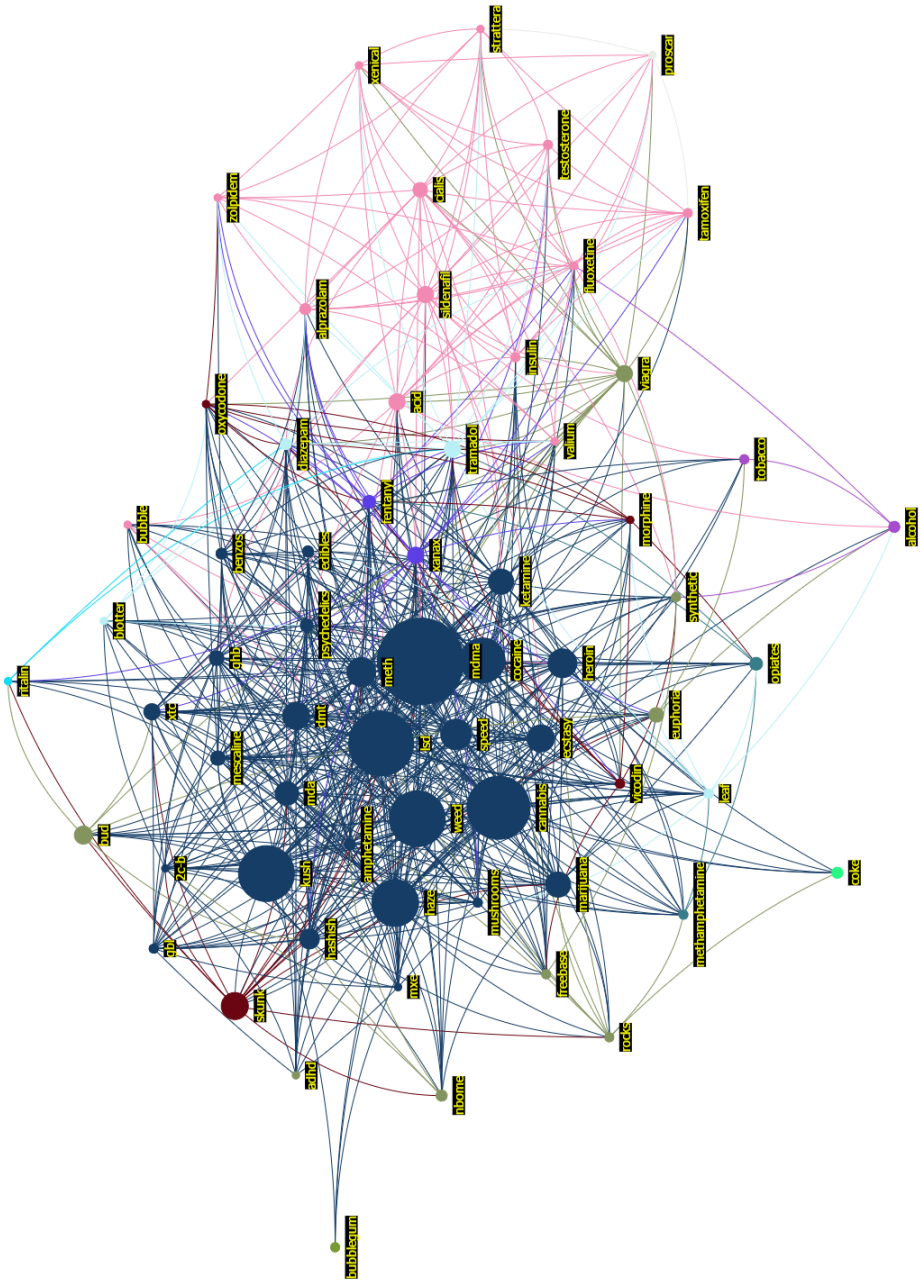


Figure 4.7: PCG of Drugs in the Tor network. The colors of the nodes refer to their shell, while the size of the nodes follows its weight. The dark blue nodes, which are located almost in the center of the graph, refer to the core-shell products.

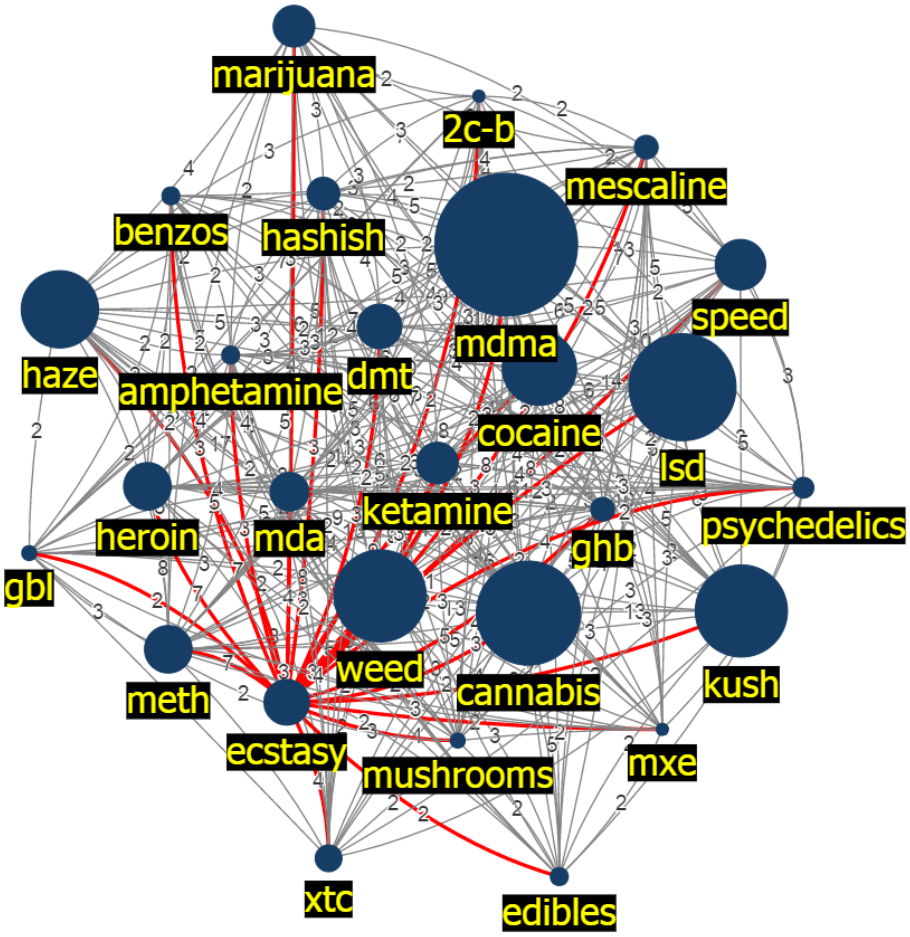


Figure 4.8: The core-shell sub-graph of the PCG. It holds 27 drug products and over each edge, it is shown its weight. The edges in red correspond to the links of Ecstasy, the most emerging product in our dataset.

4.2. Named Entity Recognition in Tor Network

Named Entity Recognition (NER) task tackles the problem of recognizing textual entities in natural text. This work employs NER on text retrieved from onion domains that were classified as suspicious - using our Text Classification Unit (TCU). The recognized Named Entities (NE) will be used as a feature for the content-based ranking - Section 5. However, NE could be used for making the Emerging Product Detection (EPD) algorithm - Section 4.1 - fully automatic, or for providing Law Enforcement Agencies (LEAs) with a list of entities for each onion domain more precise and richer than commercial tools, such as Elasticsearch NER[6] (Nafziger, 2017). In Tor Darknet, NER can return people names and nicknames, market names, shipping addresses, or even references to groups or terrorist organization names.

In this work (Al-Nabki et al., 2019b), we present a neural network architecture to recognize named entities in noisy user-generated text. The proposed model is inspired by Aguilar et al. (2017) but with two major differences. First, our network does not depend on any external knowledge resource like a gazetteer, which is costly to build and domain-dependent. Second, we introduce a novel feature, that we call *Local Distance Neighbor* (LDN), to substitute the use of any external knowledge resource. In addition to the aforementioned objectives, we present an application of the proposed model to recognize the NE within the Tor hidden services (HS) related to weapons trading and drug abuse. For this purpose, we adapted the W-NUT-2017 dataset[7] (Derczynski et al., 2017) by introducing annotated samples extracted from the commented HS domains. The additional samples were extracted from DUTA (Al-Nabki et al., 2017a), and it was used to extend the W-NUT-2017 dataset. Recently, Akbik et al. (Akbik et al., 2018, 2019) presented a novel approach for text representation using a contextualized character-level word embedding. However, although the model of Akbik et al. surpasses the model of Aguilar et al., we build on top of the latter model because our objective is to present an automatic alternative to the gazetteer, which has been used in the literature. Nevertheless, in the experiments carried out, we demonstrate that our proposal outperformed Aguilar's average F1 score and also Akbik's, but in this last case only for the Product, People, and Group categories.

4.2.1. Baseline Model Description

Given the objective of this work to propose an automatic approach that replaces gazetteers, we adopt the model of Aguilar et al. (2017) as a baseline to compare our proposal. The model involves three categories of features: character, word, and lexicons (see Figure 4.9).

- **Character**: an orthographic encoder is used to represent the characters of an input token to reduce the sparsity of the character features (Limsopatham and Col-

[6] https://github.com/bnafziger/elasticsearch-ingest-opennlp
[7] http://noisy-text.github.io/2017/emerging-rare-entities.html

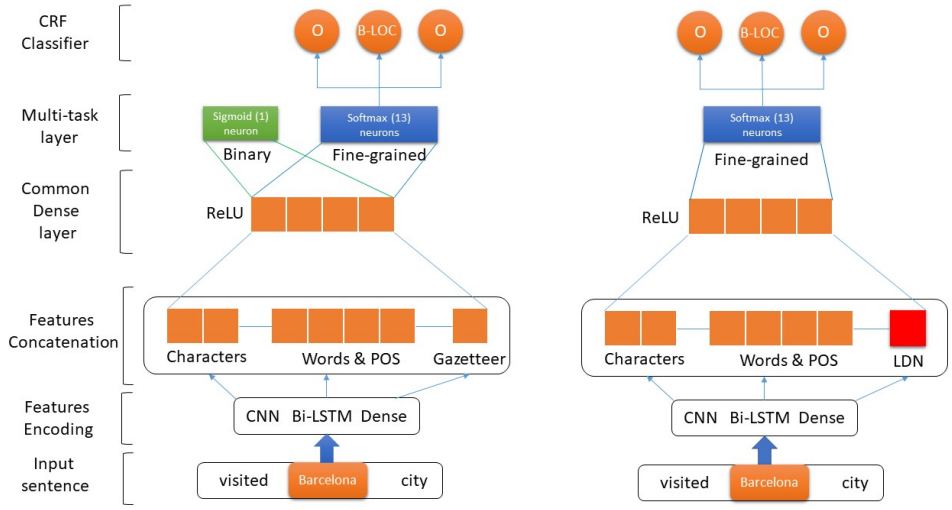


Figure 4.9: The baseline architecture proposed by Aguilar et al. (2017) (2017) (left chart), while the right one illustrates the proposed network where we introduced LDN and omitted the binary output (right chart).

lier, 2016). The encoded characters are embedded into a $\mathbb{R}^{d \times t}$ embedding space, where d is the dimension of the features per character, set to 30, and t denotes the word length threshold, set to 20, whereas the longer tokens are trimmed and the shorter ones are padded. Next, the authors apply 2-stacked convolutional layers with a global average pooling (Zhou et al., 2016). Finally, the output is passed into a fully-connected network with a Rectifier Linear Unit (ReLU) activation function (Zeiler et al., 2013).

- **Word**: two different codifications are used to represent a word. First, a pre-trained word2vec word embedding model that was trained on a dataset quoted from Twitter with a vocabulary of 3,039,345 words, each word was represented by a dense vector of 400-dimensions (Godin et al., 2015). Second, using the CMU Twitter POS tagger (Owoputi et al., 2013), part-of-speech (POS) tags are generated for each word in the dataset. Next, the authors create an embedding for the POS tags initialized randomly using a uniform distribution. The words embedding and the POS tags embedding are concatenated to form the final representation of an input word. Finally, the resulted embedding is passed into a Bidirectional Long Short-Term Memory (Bi-LSTM) (Huang et al., 2015) to learn features about the context of the input word.

- **Lexicon**: the Aguilar et al. (2017) model uses an external knowledge resource, a gazetteer, proposed by Mishra and Diesner (2016). A binary vector of M dimensions

represents each word, where M refers to the number of entity types. For the W-NUT-2017 dataset, $M = 6$, having one dimension per class. Each element of the vector is set to 1 if that word appears in the gazetteer of the corresponding class, and to 0 otherwise. Finally, the lexical vector of the input token is passed into a fully-connected layer with a ReLU activation function.

- **Multi-task network**: the characters, the word, and the lexical vectors are passed into a common dense layer of 100 neurons with a ReLU activation function. Next, it is fed into a unified model consisting of (i) NE segmentation of a single-neuron with a sigmoid activation function and (ii) NE categorization layer of 13-neurons with a softmax activation function.
- **Conditional Random Field**: to account for the sequential constraints in the input text, the authors apply a *Conditional Random Fields (CRF)* classifier (Tseng et al., 2005) over the NE categorization output of the trained neural network.

4.2.2. Methodology

Local Distance Neighbor Feature

The LDN feature is inspired by the way a human tries to figure out the meaning of an unknown term or an ambiguous word inside a text document. One possible way to do that is to use a known source of synonyms, such as a gazetteer or a dictionary of terms Donlevy (2005). Another way could be to look for tokens semantically-similar to the ambiguous term. Once these semantically-similar tokens are defined, probably the meaning of the text is more understandable for the receiver of that text. LDN tries to emulate this behavior by considering that each input token is ambiguous, i.e., does not have a tag, and turns to its tagged semantically-similar tokens to identify it.

In practice, the tagged vocabulary of a training set refers to a knowledge resource, where each training token has an embedding vector within an embedding space. We evaluate the similarity between the embedding vector of the ambiguous query token, i.e., the one with an unknown tag, and the embedding vector of every token in the training set. Next, we sort them, based on the similarity of their word embedding, in a descending order to obtain a list of tokens ordered by their semantic similarity to the query token (Chen et al., 2013). The LDN algorithm considers the top-X semantically-similar tokens, and according to the tags of the tokens, it provides an initial guess about the potential categories of the ambiguous query token.

For example, if the query token is *Barcelona*, the LDN will look for its semantically-similar tokens which can be *Madrid: Location*, *Spain: Location*, *Liverpool: Group*. Accordingly, LDN concludes that *Barcelona* tends to be a name of a Location or a Group but neither a Person name, nor a Product, and nor a Corporation. We use a vector to represent this correlation for each token, and we refer to the generated array of vectors as the LDN feature.

This LDN matrix has a shape of $(N, M+1)$ where N corresponds to the number of unique tokens in the studied dataset and M denotes the number of categories (for the W-NUT-2017 dataset, $M = 6$), whereas the additional value will be used to refer to the *Outside* tag, i.e., regular words that are not considered as named entities.

LDN Algorithm Description

The LDN algorithm consists of two phases: (i) the initialization phase, which is triggered only once to look up the embedding vectors of the training tokens, and (ii) The accumulation phase, which is called for every token to assign an initial guess about the potential tag or set of tags to the input query token. We refer to the potential tags of a token as the *trend* of that token. When training a NER system, both phases are called sequentially. In contrast, when the NER system is used for prediction or testing, only the accumulation phase is required. Figure 4.10 shows an overview of the algorithm and presents an example where the LDN vector is obtained for the query token *Cordoba*.

Initialization Phase. A summary of this phase can be found in the algorithm 1. For each input token k in the training set, we evaluate three parameters. First, $\vec{V}_k$ is the embedding vector of k within an embedding space. In the case of Out Of Vocabulary (OOV), we call a pre-processing function to remove the special characters if exist and to split the token on the capital letter characters, for example, the location entity *#EiffelTower* becomes *Eiffel Tower*. When the input token returns more than one token after the splitting, the embedding vectors of the parts are averaged Kenter et al. (2016). However, if none of them have an embedding, i.e., all the parts of the query token are also OOV, the LDN vector is assigned a 0 for each category dimension and 1 for the *Outside* dimension, meaning that this token is related only to the *Outside* category. This procedure is encapsulated under the function *embed()* (line 4) in Algorithm 1.

Algorithm 1 Computes the initialization phase for the LDN feature

```
1: function INITIALIZELDN(trainTokens_X, trainTokens_y)
2:     CT = []
3:     for (k, c) in (trainTokens_X, trainTokens_y) do
4:         V_k = embed(k)
5:         f = TokenFreq (k)
6:         CT.append([k, c, f, V_k])
7:     end forreturn CT
8: end function
```

The second parameter is the category name c of the token k, which refers to the tag of the token regardless of being a *Beginning* or an *Inside*. Finally, the frequency of the token k, when it has been tagged as c is denoted by f. The function *TokenFreq()* calculate this frequency per token (line 5). The obtained frequency f expresses the confidence of having

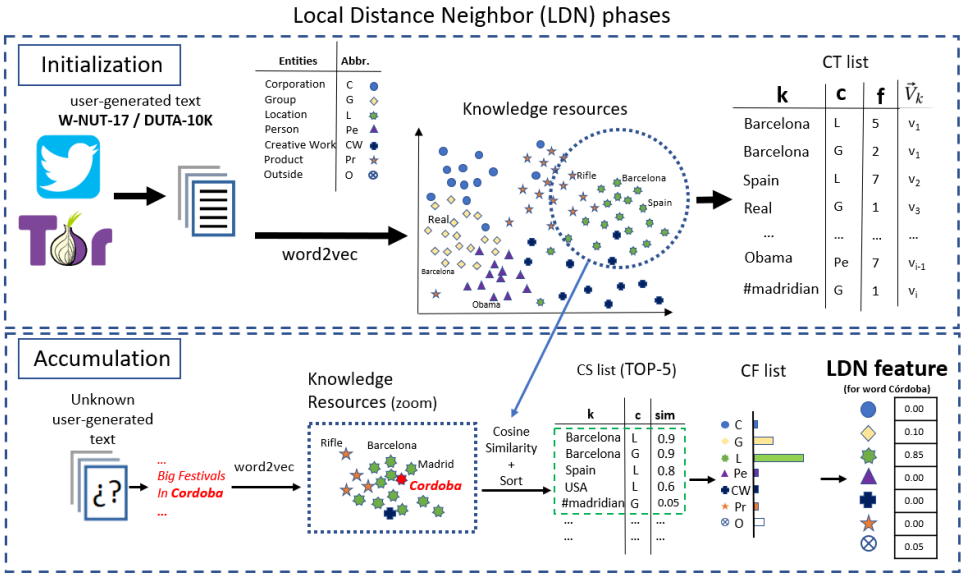


Figure 4.10: A procedure to build the Local Distance Neighbors (LDN) feature given an embedding space of the training set and a query token, e.g., *Cordoba*. First, in the *Initialization Phase*, we create the *Category-Token* (CT) list, where each training token has a category (c), a frequency (f), and an embedding vector ($\vec{V}_k$). Next, in the *Accumulation Phase*, we compute the cosine distance between the query token and every token in the training embedding space. Afterward, we sort them in descending order according to the similarity score to the query token and select the top-X neighbors. Additionally, we calculate the *Category-Frequencies* (CF) list for the top-X tokens. Finally, the gathered information is fed into the LDN formulae to evaluate the trend of the query token. In our example, we concluded that the token *Cordoba* has a probability of 0.85% to be tagged as a Location, 0.10% as a Group, and 0.05% as an Outside word.

the token k from the category c. Next, the resulting three parameters - the embedding vector $\vec{V}_k$, the category c, and the frequency f of the token k - in addition to the token k itself are appended into the *Category-Token* (CT) list (line 6) in the algorithm.

Accumulation Phase. It determines the trend of an input token with respect to $M+1$ categories as shown in Algorithm 2. Given the input query token k with an embedding vector of $\vec{V}_k$ (line 2), the cosine similarity between $\vec{V}_k$ and each vector of the training set tokens $\vec{V}_i$ is calculated using the function *sim()* (line 5) following Equation 4.4.

$$sim(i,k) = \frac{\vec{V}_k \cdot \vec{V}_i}{|\vec{V}_k| \cdot |\vec{V}_i|}. \tag{4.4}$$

Next, the embedding vectors of the tokens are sorted in descending order according to their similarity to the embedding of k using the function *Sort()* (line 8). The similarity list

is cut on the top-X using the *Cut()* function (line 9) to form a list Candidate Set (CS). The CS holds the neighbors of k, its semantically-similar tokens in the training embedding space. After that, the CS elements are grouped according to their tags to return a 1D vector of dimension $M+1$, called *Category-Frequency* (CF). It refers to the frequency of each tag in the CS. In the case a tag does not exist in the CS, it is replaced by 0. This procedure is carried out by the function *Group()* (line 10). Then, the LDN' value of token k with respect to the category c is evaluated according to Equation 4.5 (line 12).

$$LDN'_{(k,c)} = e^{sim(k,i)}(1 + CT_{(i,c)}), \tag{4.5}$$

where $CT_{(i,c)}$ corresponds to the number of times that the token i has occurred with the category c in the training set, which expresses the confidence of the tag, the term $LDN'_{(k,c)}$ represents the likelihood that the query token k belongs to the class c. Hence, LDN'_k is a likelihood vector for the query token k over the categories.

After that, we penalize LDN'_k vector according to CF, as shown in Equation 4.6 (line 14).

$$LDN''_k = LDN'_{k,c} \times CF_c \mid c \in C, \tag{4.6}$$

where the resulting LDN''_k refers to the penalized LDN'_k vector of the token k, CF_c is the frequency of the category c in the top-X neighbors, and C is the tags set.

In our experiments, we realized that the majority of the entities could be regular words, for example, the word *Just* in the name of *JustEat* corporation. To capture this fact, we added an adjustable noise parameter α to the category *Outside* only, and we set it empirically to 1, as shown in Equation 4.7 (line 15).

$$LDN'''_k = \begin{cases} LDN''_{(k,c)} + \alpha, & \text{if } c = Outside \\ LDN''_{(k,c)}, & \text{otherwise} \end{cases} \tag{4.7}$$

Finally, we normalize the LDN'''_k for a range between 0 and 1 (line 16), obtaining the LDN_k vector of the token k.

Named Entity Recognition Model

Our proposal adopts the network architecture presented by Aguilar et al. (2017), but it introduces the LDN instead of the gazetteer. The LDN of the training tokens is fed into a fully-connected layer of 32 neurons with a ReLU activation function. Next, this layer is connected to the common dense layer introduced by the original model. Another difference with Aguilar's architecture is that we omitted the segmentation task, but we kept the fine-grained one, and therefore, our proposal is a single-task network.

Algorithm 2 The algorithm of estimating the accumulation phase for the LDN feature

function ACCUMULATELDN(CT, $queryToken$, $top-X$)
  $\vec{V}_k$ = embed($queryToken$)
  CS = []
  for (i, c, _, $\vec{V}_i$) in (CT) **do**
    s = sim($\vec{V}_i$, $\vec{V}_k$)
    CS.append([i, s, c])
  end for
  CS = Sort (CS)
  CS = Cut (CS, $top-X$);
  CF = Group (CS, `unique(c)`)
  for (i, s, c) in (CS) **do**
    $LDN'_{(k,c)} = e^s\ (1 + CT_{[i,c]})$ ▷ Eq. (4.5)
  end for
  LDN''_k = Penalization(CF, LDN'_k) ▷ Eq. (4.6)
  LDN'''_k = AddNoise(LDN''_k) ▷ Eq. (4.7)
  LDN_k = Normalize(LDN'''_k)
  return LDN_k
end function

4.2.3. Adapting the W-NUT-2017 Dataset for the Tor Domains

Why to Extend the W-NUT-2017

One of the limitations of building a NER system for the Tor domains is the lack of training data that are extracted from domains practicing criminal activities in the Tor network. Considering that, we realized that extending the W-NUT-2017 dataset with samples obtained from the Tor domains might solve the problem for the following reasons:

1. To the best of our knowledge, it is the state-of-the-art dataset for noisy user-generated text.

2. Based on the experience we earned while labeling the DUTA dataset (Al-Nabki et al., 2017a), we observed that the W-NUT-2017 mimics the text of the Tor Darknet domains as both include user-generated text in slang and contains improper grammatical structures.

3. The W-NUT-2017 dataset holds a wide variety of categories including product, person, location, group, corporation, and creative-work, which are of interest Spanish LEAs, according to the feedback we have received from them.

Extending Procedure

As we already explained, *Darknet Usage Text Addresses (DUTA)* (Al-Nabki et al., 2017a) is a publicly available dataset[8] for the Tor hidden services, and holds 6,831 samples classified into 26 classes in which eight contain contents that could be considered illegal. To extend the W-NUT-2017, we accounted for samples from the *Drugs* and *Weapons* categories. Thanks to the collaboration between the Spanish LEAs with INCIBE, the latter provided us with lists of keywords related to the commented activities. In addition to online websites that provide informal definitions of drugs and common street names of weapons, such as Urbandictionary[9]. We appended to those lists expressions and terms presented in the report of the National Institutes of Health in 2012[10]. Next, we extracted the sentences that contain those keywords and labeled them manually using BIO encoding (Derczynski et al., 2017). The annotated dataset holds 851 samples extracted from DUTA, with 1,200 unique tokens and spread over the same categories of the W-NUT-2017 dataset (Table 4.3). We refer to the extended version of the W-NUT-2017 dataset as *Noisy User-generated Text on Tor* (NUToT), and it is publicly available[11].

Table 4.3: Statistics of the new Named Entities used to extend the W-NUT-2017 dataset. It shows the number of NE introduced in each category. The last row, *Total Samples*, represents the number of samples in the dataset with respect to each portion.

Category	Entities Count per Category
Corporation	64
Group	37
Location	72
Person	7
Creative-work	0
Product	1,020
Total Entities	1,200
Total Samples	851

4.2.4. Experimental Settings

The experiments were carried out on the datasets offered by the coordinators of the W-NUT-2017 shared task (Derczynski et al., 2017) (Table 4.4) and on its extended version NUToT. To evaluate the performance, we used the *entity* and *surface* F1 scores as

[8] http://gvis.unileon.es/dataset/duta-darknet-usage-text-addresses/
[9] www.urbandictionary.com
[10] https://www.drugabuse.gov/sites/default/files/cadchart_2.pdf
[11] http://gvis.unileon.es/dataset/nutot/

explained in the shared-task paper[12].

Table 4.4: W-NUT-2017 dataset specifications

Category	**#Training**	**#Development**	**#Testing**
Corporation	140	32	60
Group	231	38	141
Location	434	68	125
Person	546	399	376
Creative-work	127	100	136
Product	126	109	117
Total Entities	1604	746	955
Total Samples	3394	1009	1287

Regarding the LDN feature, we set the number of the top local neighbors empirically to five because we observed that increasing this number would produce worse results and more noisy tokens, whereas decreasing it biased the result to very few tokens. Since we adopted the model of Aguilar et al. (2017), and for a fair comparison, we left the model hyper-parameters to their default values, as described in the original paper. In particular, we used half of the development set as a validation set, and we appended the left half to the training set. Early stopping criteria is used to control the number of training epochs, which is triggered if there is no further improvement on the validation set during 20 epochs of training. In our experiments, for the WNUT-2017 dataset, 226 training epochs were used.

Additionally, we examined the effect of the employed tasks in the multi-task network on the performance by trying the following three scenarios. First, we measured the original multi-task network, which holds both tasks, Binary and Fine-Grained, and we denote them as *B* and *FG*, respectively. The *B* task has one single-neuron layer with a sigmoid activation function, and it determines whether the input token is an entity or not. The *FG* task has 13 neurons with a softmax activation function and it indicates the category of the input token and whether it is a *Beginning*, an *Inside* or an *Outside* tag using BIO encoding. The second scenario uses a single-task for the network by removing the *B* task and keeping the *FG*. Finally, in the third scenario, we introduced an additional fine-coarse task, which we denote as *FC*, which indicates the category of the input token regardless of being a *Beginning* or an *Inside* tag. Accordingly, the *FC* has seven neurons layer with a softmax function, six of them for the six categories of the W-NUT-2017 dataset, and one for the *Outside* tag.

Regarding the extended version of the W-NUT-2017 dataset, the NUToT dataset, we considered 80% of the samples for training and 20% testing the model, while the other

[12]The evaluation script is available online at the website of the WNUT shared task `https://noisy-text.github.io/2017/files/wnuteval.py`

parameters are kept the same. The experiments were conducted on a PC with an Intel(R) Core(TM) i7 processor with 32 GB of RAM under Windows-10. We used Python3 with Keras framework[13] for the implementation.

4.2.5. Results and Discussion

Comparison with the state-of-the-art

Given the W-NUT-2017 dataset, the goal was to label every token in the BIO scheme (*B*: beginning, *I*: inside, *O*: outside). Next, we report the state-of-the-art performance, as well as various configurations of our proposal to measure the contribution of each task in the multi-task network (*B*: binary, *FC*: fine-coarse, and *FG*: fine-grained) and each input component (*W*: word, *C*: char, *G*: gazetteer, and LDN: local distance neighbor):

- **W-C-G/ B-FG**: it represents the baseline model as it considers the features of the tokens, characters sequences of each token, and a gazetteer. The network has binary and fine-grained tasks only.
- **W-C-G/ B-FC-FG**: it has a similar input to the previous one, but the network has an additional *FC* task.
- **W-C-G/ FG**: the inputs are similar to the first one, but only the *FG* task was used, which makes it a regular network with a single-task.
- **W-C-LDN/ B-FG**: it substitutes the gazetteer input with the LDN feature, while it preserves the multi-task proposed in the baseline model.
- **W-C-LDN/ B-FC-FG**: it has similar input to the previous one but introduces the *FC* task.
- **W-C-LDN/ FG**: it is a single task network, only *FG*, with similar input to the previous one.

We realized that by running the experiments several times but changing only the seed value of the random number generator, the results vary slightly as described in Reimers and Gurevych (2017). Therefore, we re-ran the implementations 10 times and measured the average entity and surface F1 score results.

Table 4.5 presents a comparison between the model of Aguilar et al. (2017), the model of Akbik et al. (2019), and the previously commented proposals on the W-NUT-2017 dataset. Using the LDN feature instead of the gazetteer and without any change to the network tasks, we obtained a performance better than Aguilar et al. model. The average entity and surface F1 scores have increased from 41.84% to 46.11% by 4.27% and from 39.95%[14] to

[13] `https://github.com/fchollet/keras`

[14] Aguilar et al. (2017) reported the maximum value for the entity F1 score as 41.86% and the surface F1 score of 40.24%. However, here we are comparing with the averaged F1 score.

45.74% by 5.79%, respectively. However, the proposal of Akbik et al. (2019) obtained an entity and surface F1 scores of 49.92% and 49.11%, respectively.

Table 4.5: A comparison of the averaged F1 score between Aguilar et al. (2017) model (underlined), Akbik et al. (2019) model, and our proposals on the W-NUT-2017 dataset. The Avg. and the Std. dev. stand for the average and the standard deviation, respectively. The values in bold are the best scores across the proposals.

Model	Network Tasks	Avg. F1 score (entity) %	Std. dev. F1 (entity)	Avg. F1 score (surface) %	Std. dev. F1 (surface)
W-C-G	B-FG (Aguilar et al., 2017)	<u>41.84</u>	<u>0.62</u>	<u>39.95</u>	<u>0.74</u>
	B-FC-FG	41.68	0.97	39.92	0.95
	FG	**42.20**	0.51	**40.39**	0.64
W-C-LDN	B-FG	46.11	0.89	45.74	0.90
	B-FC-FG	46.17	0.85	45.74	0.91
	FG	**46.47**	0.55	**46.02**	0.54
[15]**Akbik et al. (2019)**	FG	**49.92**	-	**49.11**	-

Furthermore, Table 4.5 shows that the usage of a single-task network, with a *FG* task only, is capable of achieving even better performance than the multi-task one from Aguilar et al. (2017). Our proposal **W-C-LDN/ FG** has an entity and a surface F1 scores of 46.47% and 46.02%, respectively. As can be observed, this is also applicable to Aguilar et al. model because, in our experiments, it obtained an entity F1 score of 42.20% and 40.39% for the surface one with a *FG* task only.

Out of the 10 executed runs, Table 4.6 reports the highest result obtained of 47.28% and 46.96% for entity and surface F1 score, respectively. The table compares our proposal with Aguilar et al. and Akbik et al. models, and it shows that Aguilar et al. approach enhanced with LDN feature, i.e., W-C-LDN / FG, outperforms Akbik et al. work in three categories: Product Person, and Group. These results make our approach, W-C-LDN / FG, more suitable that Akbik's proposal for the application of detecting NE on Tor Darknet.

Figure 4.11 reports the growth of the entity F1-score measure over the training and validation sets using the best proposal, i.e., W-C-LDN/ FG. It shows that after 226 training epochs, we yield a training and a validation F1 scores of 66.68% and 49.08%, respectively. At this point, when the training is terminated, we obtained a test F1 score of 46.20%. Next, after training a CRF classifier, the performance boosted to 47.28%.

Extended version of the W-NUT-2017 Dataset (NUToT)

We compared the best proposal observed for the W-NUT-2017, i.e., *W-C-LDN/ FG*, with the models Aguilar et al. and Akbik et al. on the extended version of the W-NUT-2017 dataset (Table 4.7). Since the model of Aguilar el al. requires an external knowledge

[15]Thanks to FLAIR framework (`https://github.com/zalandoresearch/flair`), we reproduced the results on the W-NUT-2017 on the category level.

Table 4.6: The entity F1 score using **W-C-LDN/ FG** proposal, Aguilar et al. (2017) model, and Akbik et al. (2019) model on the W-NUT-2017 dataset respecting each category. The bold values refer to its superiority in terms of the F1 score metric.

Category	Aguilar et al. model F1 (%)		Akbik et al. model F1 (%)		Our Proposal W-C-LDN/ FG F1(%)	
	Entity	**Surface**	**Entity**	**Surface**	**Entity**	**Surface**
Corporation	28.57	23.40	**37.84**	37.62	29.93	28.12
Creative-work	11.63	12.05	**30.62**	27.72	24.36	23.85
Group	25.10	22.55	29.08	27.87	**34.52**	33.03
Location	51.90	49.79	**61.70**	59.26	53.83	55.22
Person	58.95	57.41	64.53	64.63	**65.34**	64.74
Product	15.38	14.10	25.56	23.21	**41.41**	38.31
Total	41.94	39.73	**49.92**	49.11	47.28	46.96

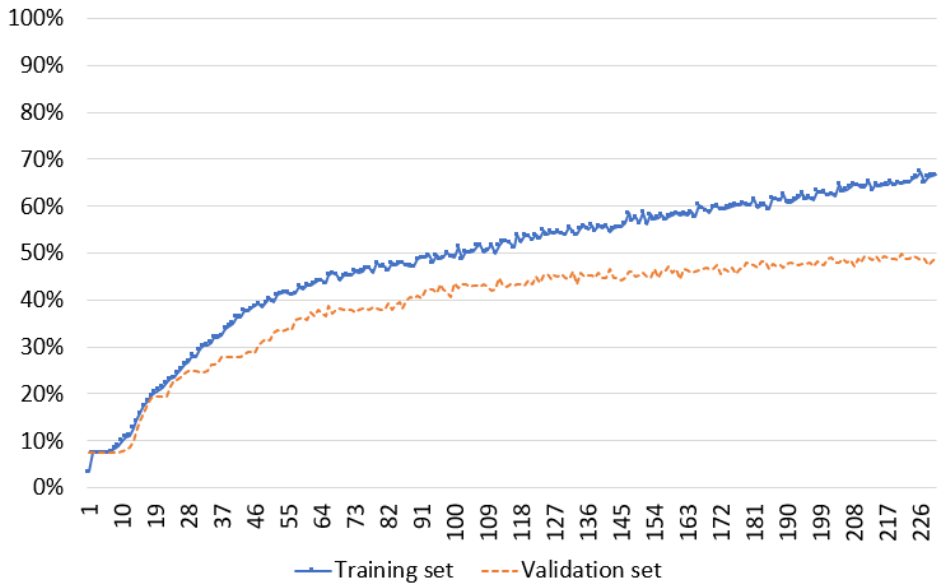


Figure 4.11: Performance in terms of F1 score on the training and validation sets. The X-axis refers to the training epochs, while the Y-axis indicates the F1-score

resource, we followed the same procedure explained in their work to construct the gazetteer.

Table 4.7 shows that our proposal outperforms the benchmarked models with an entity and surface F1 scores of 52.96% and 50.57%, respectively, representing the baseline

results for the NUToT dataset we created. Similarly to the previous result, our proposal obtained state-of-the-art results on the categories Person, Group, and Product on the NUToT dataset.

It is worth mentioning that the category *Products* has the highest increment in terms of entity and surface F1 scores because most of the added samples referred to that category. Another possible reason behind the F1 score increase is related to the nature of product names found in the category of drugs and weapons. The drug names are close in the embedding space. So, when a new drug name is introduced, the LDN can guess its category with high confidence, and the same scenario occurred with the weapons. Indeed, when the tokens of a given category are close semantically, we get the most benefit of LDN. In contrast, the categories *creative-word* and *Corporation* suffer from the lowest F1 score due to the sparsity of their entities tokens. For example, the individual words of a creative-work entity such as *Beauty and the Beast* are possible to fall under several categories, while the words of a product entity like *Sauer P229S* are less likely to be under any other category as they refer to a weapon model. We consider that this advantage is, at the same time, a drawback in the LDN algorithm that we look forward to handling it our future research.

Table 4.7: Comparison between Aguilar et al. (2017) approach **W-C-G/ B-FG**, Akbik et al. (2018) model, and our proposal **W-C-LDN/ FG** on the NUToT dataset. The *Products* entities obtained the highest entity and surface F1 scores of 63.15% and 61.57%, respectively.

Category	**Aguilar et al. model F1 (%)**		**Akbik et al. model F1 (%)**		**Our proposal F1 (%)**	
	Entity	**Surface**	**Entity**	**Surface**	**Entity**	**Surface**
Corporation	18.96	19.46	**29.36**	31.91	21.92	20.94
Creative-work	18.86	21.35	**26.58**	23.14	22.83	24.31
Group	21.14	22.54	23.38	23.36	**26.39**	27.91
Location	35.06	33.10	**60.43**	61.00	54.58	54.87
Person	61.48	61.01	62.05	61.86	**62.15**	62.36
Product	53.25	53.53	53.64	51.95	**63.15**	61.57
Total	44.73	43.29	52.17	50.53	**52.96**	**50.57**

Insights on NER for Tor Marketplaces

According to Table 4.3, the category *Product* is the most popular tag in the Tor domains because nearly all the analyzed onion domains are marketplaces of drugs and weapons. During labeling NUToT dataset, we realized that the majority of the labeled entities - drugs and weapons domains - present their products as a grid of photos, and under

each photo the name of the corresponding product. Another style of product presentation is to show just a list of products, and the customer can click on any of them to open a new page with more detailed information about it (see Figure 4.12). These two exposition techniques let the products entities orphan, with no context or with a short context. In the NUToT dataset, the most frequent sentence length is three to six tokens, e.g., "1G Ice Hash 0.166" sentence has four tokens.

In contrast, the W-NUT-2017 dataset the length ranges between 13 and 17 tokens per sentence. Hence, NER systems that utilize the context of the targeted entities only would struggle, and here is where the LDN feature can fit the best. Nevertheless, if a token does not have a corresponding embedding vector, it will not take advantage of the LDN.

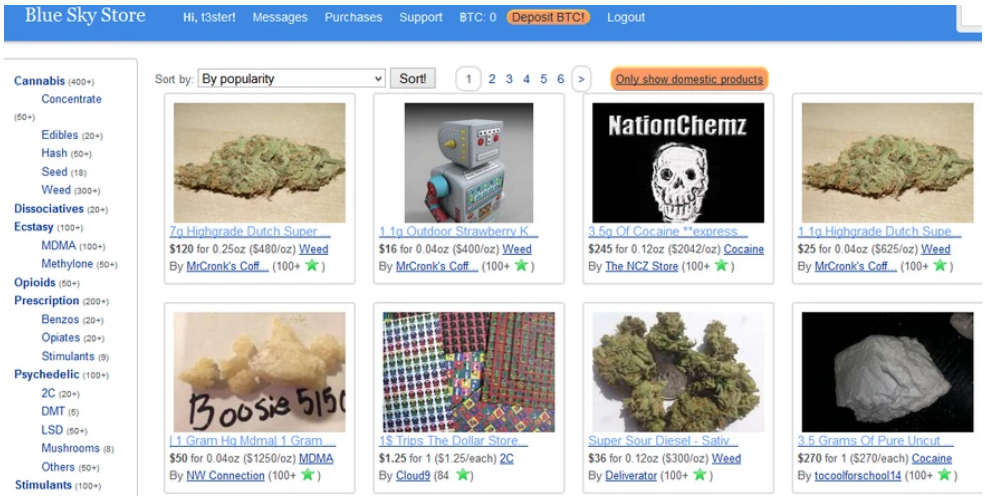
When to use the LDN Feature

The advantage of our work is twofold: first, the LDN feature offers a simple but powerful technique to substitute the gazetteers, which have been used as an external resource of knowledge by several researchers Mishra and Diesner (2016); Aguilar et al. (2017); Ding et al. (2019); Štravs and Zupančič (2019); Dey and Prukayastha (2013). This results in a NER architecture that does not depends on external resources.

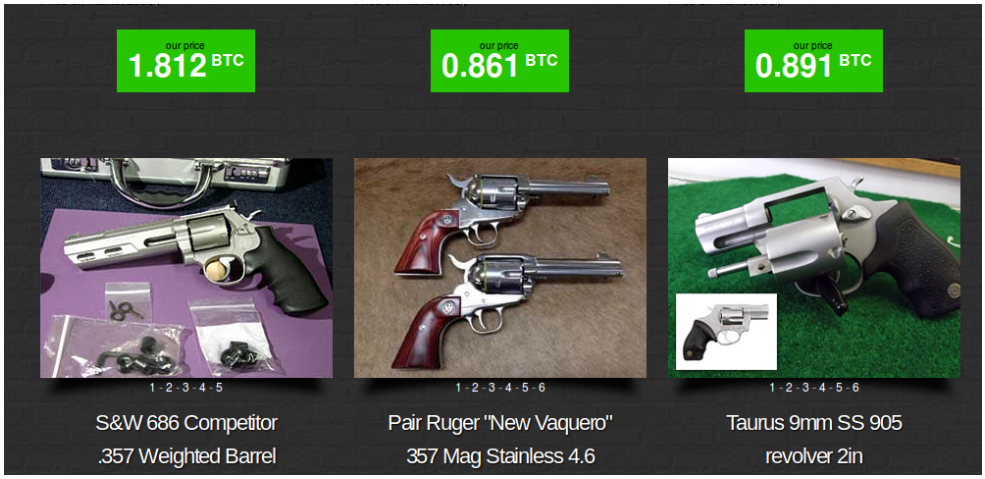
Second, it provides a solution to recognize named entities in the onion domains of the Tor network. As we showed in Table 4.6, our model outperforms the work of Aguilar and competes for the model of Akbik in three categories of the W-NUT-2017: Person, Product, and Group. Furthermore, our proposal surpasses both models in terms of the F1 score metric when tested on the extended version of the W-NUT-2017 dataset (NUToT), as we showed in Table 4.7. Thanks to the design of the LDN feature, it exploits the high cosine similarity between the words embedding vectors whose tokens are semantically similar. Hence, each set of semantically similar tokens will form a small cluster.

The category Product has been the most benefited from this feature, unlike the category Creative-work. We explain this behavior by the small clusters formed by the product entities. For example, drug products like (Cocaine, LDM, MDMA, Hashish) form a cluster, and when a new product from the test set is introduced, like GHB, the LDN feature will recognize its trend as a product, thanks to its neighbors. Likewise, the other types of products, such as car brands, weapons products, and electronic devices.

In contrast, the tokens of the category Creative-work are sparse such as a name of a TV show «What to do in this life». The tokens «What» or «this» are most probably correlated with the category Outside, i.e., not an entity, and in this case, the LDN feature will not be helpful. That is to say, the success of LDN on a particular entity type depends on the distribution of its training tokens. Since our work focus is the marketplaces of the Tor network, which is rich with Product entities, the LDN feature would serve perfectly.



(a) Drugs



(b) Weapons

Figure 4.12: Samples of drugs and weapons markets in the Tor network

4.3. Conclusions

In this chapter, we initially presented a semi-automatic algorithm to detect the emerging products in the Tor Darknet. The proposed algorithm depends on Graph Theory and the k-shell decomposition algorithm. After building a list of products and loading a text corpus (the onion domains), we construct a Products Correlations Graph (PCG). It is an undirected graph where the nodes correspond to the onion domain products, and the edges reflect that two products are offered, simultaneously, by the same onion domain. Hence, a new edge is created between two nodes when they have been offered in the same Tor domain at least once. Then, by running the k-shell algorithm, we decomposed the PCG into levels of importance where the core-shell contains the most popular products. Finally, our emergent detection algorithm is applied to the core-shell to obtain emerging products. Furthermore, from the PCG, we could conclude association rules between the products that are hard to infer when the market transaction logs are not available. We conducted our experiments over a subset of DUTA, only the drug-related categories (both legal and illegal) drug and Black Marketplaces. The results of our experimentation are validated quantitatively by matching them with reports of prestigious international organizations that are concerned with the drugs spread.

Second, we proposed a new feature, Local Distance Neighbor (LDN), which is capable of substituting the usage of an external knowledge resource such as gazetteers. We adopted the model of Aguilar et al. (2017) as a baseline because it obtained the highest F1 score in the 3rd Workshop on Noisy User-generated Text (W-NUT-2017) shared-task. In the experiments carried out, we demonstrated that our proposal outperformed Aguilar's average F1 score and also Akbik's, but in this last case only for the Product, People, and Group categories. During our experimentation, we observed that different combinations of tasks for the multi-task network architecture, such as binary segmentation, fine-coarse, and fine-grained categorical, do not produce a significant impact on the performance. We found that a regular network with a single fine-grained task could outperform a multi-task one.

Furthermore, in this work, we investigated the problem of detecting the NE in the hidden services of the Tor network where suspicious activities can be found. We annotated manually 851 samples extracted from DUTA dataset with 2,970 unique tokens, and we appended them to the W-NUT-2017 dataset to train a model for the Tor network. We found that our proposal using LDN with a fine-grained task outperformed the baseline model. We were capable of detecting the NE in the Tor network with an entity and surface F1 scores of 52.96% and 50.57%, respectively.

The inclusion of the LDN feature has shown promising results with a significant improvement in the F1 score. Future works would be the context representation of the input token while evaluating its LDN vector using Bi-LSTM for the LDN vectors, to evaluate its performance.

Chapter 5

Influence Detection

This chapter focuses on the Influence Detection Unit (IDU) of the Tor monitoring pipeline (see Figure 1.3). In particular, the task of detecting and ranking the most influential onion domains in the Tor network. To tackle this problem, we explore two distinct approaches: Link-based and Content-based ranking.

5.1. Link-based Ranking

The Influence Detection Unit (IDU) is responsible for ranking the onion domains of the Tor network, whereas the top-ranked elements are the most influential ones among the rest. The IDU is fed with a list of onion domains, and it ranks them in a way that reflects their popularity, among others. Recognizing influential Hidden Services (HS) could provide clues to LEAs about the market leaders of each activity. For example, identifying the most prominent drug marketplaces that attract people could be useful to draw insights into the popular products, sellers nicknames, main countries involved, and possible exporting destinations of that market. Hence, the ranking approach that we are presenting might be a supplementary and valuable strategy for leveraging LEAs resources, recommending where to focus their monitoring efforts.

5.1.1. Onion domains dataset

Why to extend DUTA dataset

In our previous work (Al-Nabki et al., 2017a), we introduced DUTA dataset to build a supervised classifier through the Text Classification Unit (TCU). Indeed, despite the lifespan of some DUTA domains is short, as they might be taken down by some LEA or closed by their hosts, the value of DUTA is preserved. Apart from giving insights into Tor in a specific period, it could be used to research several problems, including, but not limited to, the detection of emerging products in the onion domains (Al-Nabki et al., 2017b), image classification (Fidalgo et al., 2017; Matilla et al., 2018), text summarization (Joshi et al., 2018), or recognition of onion domains services (Biswas et al., 2017). Therefore, we decided to extend DUTA, and we collected new onion domains between May and July 2017.

DUTA-10K extension procedure

The Tor metrics website indicated that there are more than 100K HS alive at the time of conducting this research. However, for security reasons, the Tor network structure does not have a public DNS server where all the HS addresses are registered. Instead, it uses a Hidden Service Directory (HSDir), which is a Tor relay that functions as a middle point between an HS, as it publishes its descriptors there, and clients, who communicate with it to learn the address of the HS's introduction points (Biryukov et al., 2013, 2014). However, a Tor relay needs a specific flag to be assigned by Tor authorities to function as HSDir. Instead of asking for that flag, we built a crawler that searched the Web for new onion addresses.

To extend DUTA, we incorporated more onion addresses by searching in different sources. First, we developed a customized crawler that looks for onion addresses in three resources: 1) the Online Notepad Services (ONS) in the Surface Web, 2) the search engines of the Tor network, and 3) DUTA dataset hyperlinks. Then, we detected addresses using a parser that employs regular expression pattern to match the onion ones.

The Surface Web has plenty of addresses that are posted by anonymous people in public notepad websites, such as Pastebin [1]. Fortunately, Pastebin is powered by Google search engine, what allowed us to search for onion addresses by typing keywords like *Onion Addresses*, *hidden services 2018*, *darknet links 2019*, and *.onion links*. We scraped the content of the retrieved pastes and parsed the onion addresses. Additionally, we used Tor network search engines, such as *Ahmia.fi* and *onion.link*, looking for random words like *onion address*, *Tor services*, *Tor markets*, and *Tor products*. Then, we scraped the retrieved onion web pages and parsed the onion addresses. Finally, we used the content of DUTA samples to parse new HS addresses; then, we scraped their content and iteratively repeated this process two more times. Those three strategies returned 124,589 new unique onion addresses, but only 3,536 ones were active at that moment, while the majority were down, i.e., they returned a connection time-out error message. For each active onion domain, we crawled the root page and the first level in-depth for the sub-pages. Next, we concatenated the pages of each domain into a single HTML file.

Therefore, the collected samples represent a real case of the domains in the Tor network, without any bias towards some specific category. We saved each sample of DUTA-10K in a textual format after removing the HTML tags. To this end, we used a simple workaround; we loaded the HTML pages with Lynx[2], a text-based web browser, then we saved the cleaned text into a text file. We extended DUTA by adding the 3,536 newly collected samples and denoted it as DUTA-10K because it holds 10,367 unique onion addresses.

[1] `https://pastebin.com/`
[2] `http://lynx.browser.org/`

DUTA-10K labeling procedure

To ease the labeling process of the new samples, we split this task into two phases. The first one utilizes the previously proposed text classifier to isolate the suspicious activities from the normal ones. Then, we validate the assigned labels manually, one by one. The second phase is manual labeling for the samples that were classified as *Others*. We divided the *Others* samples among several members of our research group, and each one labeled their designated part. We respected the same regulations as the previous version of DUTA: 1) we label a domain based on the user-visible textual content only, 2) a domain must receive one single tag only according to its activity. In case it holds more, it is tagged as marketplace black or white following the suspiciousness of the activity, and 3) if any member hesitates about the tag, we conducted an open discussion among the involved people to judge that onion domain. Finally, to check the consistency of the procedure, Wesam Al-Nabki reviewed the tags assigned to each HS by analyzing random samples of the categorization made by the others.

The DUTA-10K samples are distributed over 25 classes with some small changes in their names compared to the original DUTA. In DUTA-10K, we joined the «Leaked-data» category to «Fraud» because both had a small number of samples, and they are relatively related to the same topic. In the same way, we mixed the category «Wiki» with the category «Hosting/ Directory». Additionally, DUTA-10K presents a new category, *CryptoLocker*, related to domains used to pay a ransom to decrypt a machine infected with Ransomware, like the WannaCry virus (Mitchell, 2017) (Table 5.1). INCIBE (Spanish National Cybersecurity Institute) develops services and solutions to support the daily activities of Spanish LEAs. Due to our close collaboration, we received a list of activities that are considered interesting for Spanish LEAs. Based on that, we tagged the activities related to this list as *Suspicious Activities*, whereas the rest were denoted as *normal Activities*.

In Table 5.1, it can be observed a new activity type, named *Unknown*, which contains HS whose content could not be assessed by the crawler. It comprises three categories of domains. (i) Locked, domains that require human interaction, such as solving a CAPTCHA or entering a log-in credential to access. (ii) *Empty*, domains without text or with graphical content only. Also, based on our previous work (Al-Nabki et al., 2017a), we assigned to this category small HS, with less than five words. (iii) *Down*, when the crawler returns an error while downloading the textual content. For example, many HS require JavaScript activation, which was disabled in the crawler due to security reasons.

Every sample of the dataset contains the HTML code of the web page, the language[3], and the assigned activity by the annotators. Some activities were divided into sub-activities, and, therefore, subcategory labels were assigned to them. For example, the documents forging activity has a main activity named *Counterfeit Personal Identification* with three sub-activities branches named: *Passport*, *ID*, and *Driving License*.

[3]The LangDetect 1.0.7 library was used for the language detection.

Table 5.1: DUTA-10K dataset activities distribution. The letter C. denotes *Counterfeit* activity.

Activity Type	Activity	Sub-Activity	#HS
Suspicious Activities 20%; 2,013 HS	Pornography	Child	105
		Adult	148
	Drugs	-	465
	Violence	Hate	19
		Hitman	28
		Weapons	48
	C. Credit Cards		399
	C. Money		83
	C. Personal Identification	Passport	48
		ID	20
		Driving-License	4
	Hacking	-	205
	Cryptolocker	-	185
	Marketplace	Suspicious	127
	Services	Suspicious	20
	Forum	Suspicious	63
	Fraud	-	43
	Human-Trafficking	-	3
Normal Activities 48%; 5,016 HS	Art & Music	-	15
	Casino & Gambling	-	29
	Services	Normal	341
	Cryptocurrency	-	868
	Forum	Normal	163
	Marketplace	Normal	138
	Library & Books	-	45
	Personal	-	616
	Politics	-	12
	Religion	-	21
	Hosting & Software	File-Sharing	205
		Folders	99
		Search-Engine	92
		Server	1,416
		Software	411
		Directory	133
	Social-Network	Blog	219
		Chat	79
		Email	72
		News	42
Unknown 32%; 3,338 HS	Down	-	864
	Empty	-	1,653
	Locked	-	821
Total sum			10,367

5.1.2. Statistical analysis of DUTA-10K

To have a deeper understanding of the onion domains, we made a statistical analysis of DUTA-10K with respect to: (i) the activities distribution, (ii) the domains content replication, and (iii) the used languages.

Activities distribution

Out of the 10,367 samples of DUTA-10K, we found that the suspicious activities represent 20% of the Tor HS and the normal ones 48%. The third category, i.e., the content which cannot be accessed, adds up the 32% left. During the sampling period of DUTA-10K, we found that the *Drugs* trading is the most popular suspicious activity on the onion domains, representing 23% of the suspicious HS. It is followed by *Credit Cards Counterfeit* and *Hacking*, being 20% and 10%, respectively, of the suspicious HS. Conversely, the *Human-Trafficking* activity counts the lowest presence with only 0.2% (see Figure 5.1).

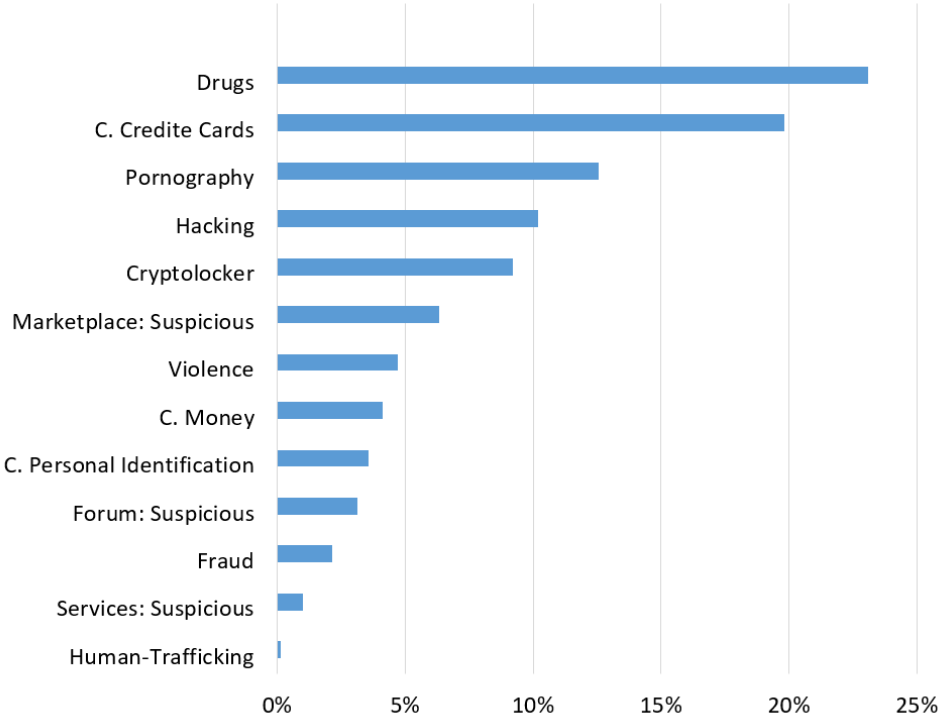


Figure 5.1: The distribution of the suspicious activities in DUTA-10K.

In contrast, for the normal activities (see Figure 5.2), we found that the HS that offer *Hosting Servers* represent 47% of the normal onion domains, followed by *Cryptocurrency*

and *Bitcoin* trading, which accounts a 17%. Next, the category *Personal* represents 12% of DUTA-10K normal activities. The smallest category was the political websites, which occupies only 0.2%.

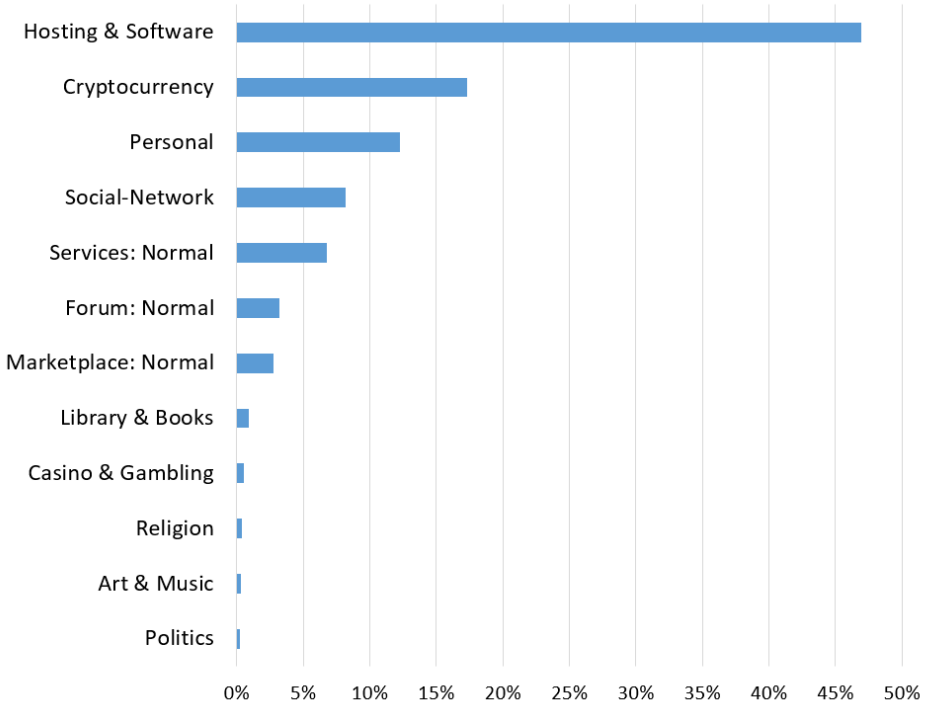


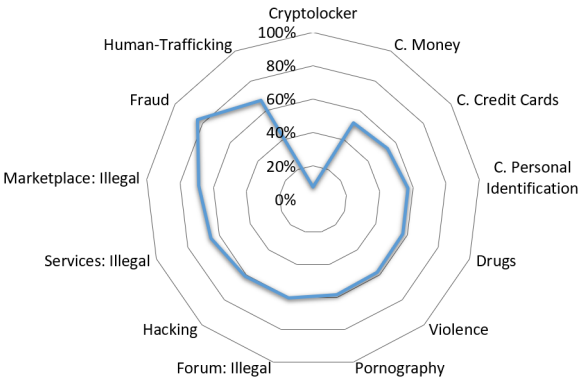
Figure 5.2: The distribution of the normal activities in DUTA-10K.

Domains content replication

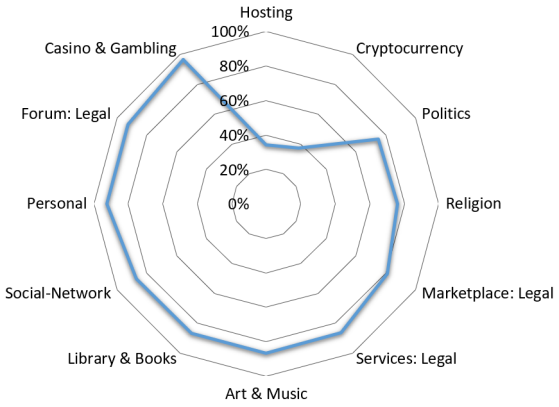
During the DUTA-10K labeling process, we detected that some samples had identical or quasi-identical textual content, but they were hosted under different addresses. The latter ones refer to HS that after preprocessing their text, for example, by removing the PGP signature, the dates, or the price of the products, become exactly identical. We obtained the MD5 hash (Rivest, 1992) for each cleaned onion domain and grouped them into the defined 25 categories. After that, we found only 5,368 unique texts based on their hashes, i.e., 51% of DUTA-10K samples vary between two and 496 copies per domain.

As shown in Figure 5.3a, the majority of the suspicious onion domains are cloned, and they appear several times using various onion addresses. For example, only 60% of the HS, which were labeled as *Drugs* have unique content, and only 10% of the *Cryptolocker* domains are unique. In contrast, the normal activities domains present reverse

behavior (see Figure 5.3b). However, *Hosting* category is an exception because only 40% of its domains are unique. The reason behind the high number of replications is due to a hosting company called «Freedom Hosting»[4] that occupies 35% of those clones. The high number of suspicious HS clones reflects the possible concern of their owners to provide smooth access for the customers in case any LEA takes some of their domains down.



(a) Suspicious Activities



(b) Normal Activities

Figure 5.3: Illustration of the replication of Tor HS. The percentages represent the number of unique domains in the corresponding category. The majority of the suspicious services (upper char) tends to have duplicated copies of their HS, while the majority of the normal ones (bottom chart) have unique content.

[4] `https://en.wikipedia.org/wiki/Freedom_Hosting`

Language analysis

We did not find representative differences between normal and suspicious activities in terms of DUTA-10K language analysis. We detected 38 languages in DUTA-10K domains, but only five of them, which are illustrated in Figure 5.4, have a frequency higher than or equal to 1%. English is the most common language, which is used in 84% of the samples, followed by Russian with 6%. From the point of view of a researcher or a LEA, this finding means that training a model only on an English corpus would be sufficient to cover the majority of the Tor Darknet HS.

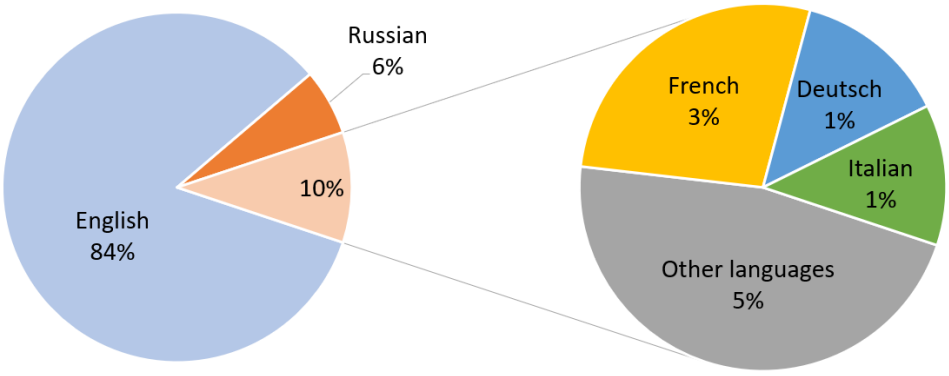


Figure 5.4: The languages used in onion domains of DUTA-10K.

5.1.3. Methodology

The core task of the Influence Detection Unit (IDU) is ranking onion domains in a way that positions the most influential at the top of a list. The expected output of the IDU is a list of ranked onion domains. In the following, we present a novel algorithm, called ToRank, that ranks hidden services in Tor better than the known algorithms used for the Surface Web, such as PageRank (Page et al., 1999), HITS (Kleinberg, 1999), and Katz (Katz, 1953). Using Graph Theory, we model the Tor network by a graph of nodes and edges, and we call it Suspicious Activities Graph (SAG). ToRank algorithm analyzes the interconnections between the nodes to determine the influential domains. Figure 5.5 illustrates a graphical overview of ToRank algorithm.

Onion domains representation

Due to our close collaboration with INCIBE and the interest of Spanish LEAs on certain activities carried out in Tor HS, i.e., suspicious, we focus our effort on ranking only the 13 classes of DUTA-10K labeled as suspicious in Table 5.1. Nevertheless, the experiments

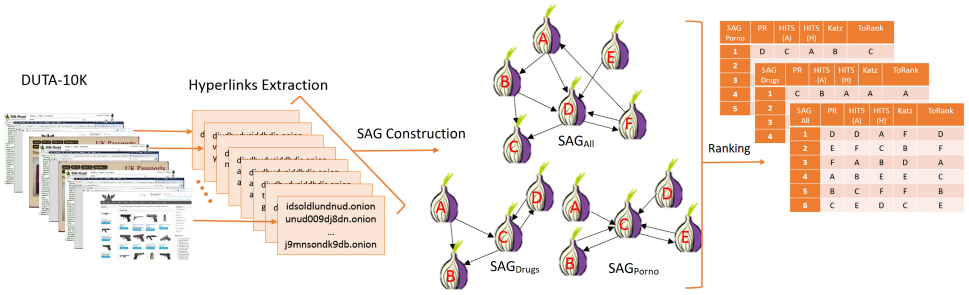


Figure 5.5: An overview of the followed procedure to detect and to rank the influential HS. First, the hyperlinks of the dataset are extracted. Afterward, for every single category of suspicious activities, a Suspicious Activities Graph (SAG) is constructed, like SAG$_{\text{Porno}}$ and SAG$_{\text{Drugs}}$. Additionally, another graph is built for all the activities, named SAG$_{\text{All}}$. Ultimately, the ranking algorithms are applied to rank and to detect the most influential HS.

section is not limited to suspicious activities only, and it addresses the normal ones as well.

Hyperlinks extraction. For each onion domain in DUTA-10K, we extracted only the incoming and outgoing *HTTP* and *HTTPS* hyperlinks. Next, we removed the ones pointing to the Surface Web, to focus our analysis only on onion domains. We also excluded the duplicated links, which have the same source and destination, to avoid having a multi-graph. We found that some of the suspicious activity domains were referenced or were pointing to web pages within Tor. However, either they were not related to the designated suspicious categories or were web pages that did not exist in DUTA-10K. In both cases, we added those nodes to the graph: when a new sample exists in DUTA-10K, we labeled it based on DUTA-10K classification; otherwise, we assigned a different labeled called «new node». In the end, each domain ended up with two lists of hyperlinks, one for the addresses that were referencing it and another list containing the domains inside Tor that this domain was pointing to.

Interesting activity graph implementation. Once the hyperlinks for the suspicious activities were extracted, we modeled the Tor network as a directed graph and denoted it as *Suspicious Activity Graph* (SAG), as shown in Figure 5.6. The $SAG = (N, E)$ model is composed of a set of nodes denoted by N, which are the HS, and a set of edges E, which correspond to their hyperlinks. A new edge is created from an onion domain A to an onion domain B either if A has referenced B or if B has been referenced by A at least once.

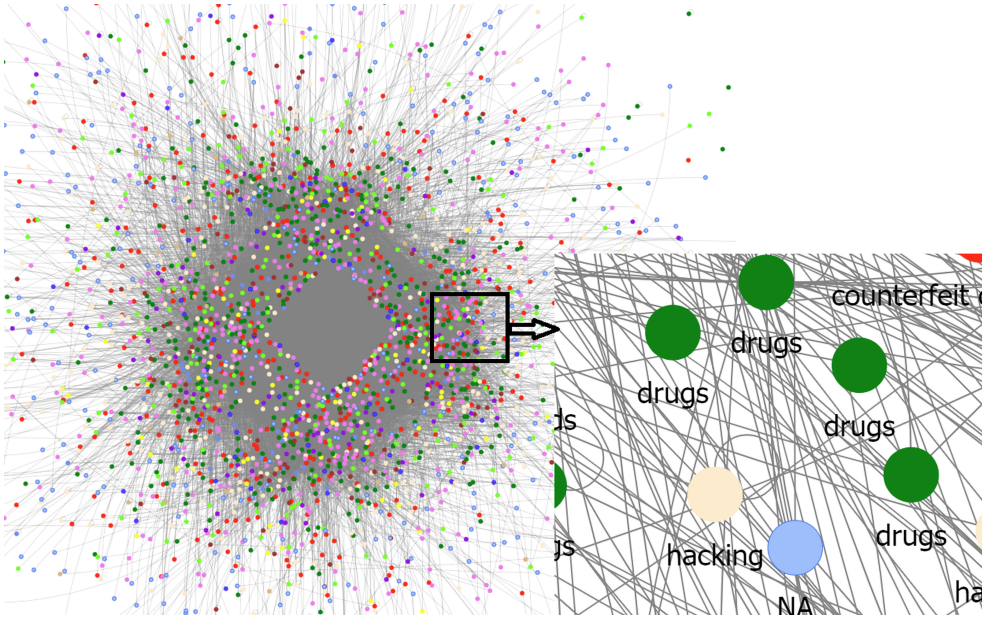


Figure 5.6: Snapshot of the DUTA-10K SAG_{All}, where the dots correspond to HS. Each color represents an activity in DUTA-10K. The gray lines reflect the hyperlinks between the domains. Due to the density of the connections in the center of the graph, it appears as a gray spot.

ToRank algorithm

The objective of the proposed algorithm, ToRank, is to identify the most influential node in a graph by measuring the number of nodes to which traffic can be propagated or from which it can be received. An obvious strategy could sort the domains them by the number of clients' requests they receive. However, the design of the Tor network is dedicated to preventing this behavior (Arma, 2019). The design of ToRank overcomes this limitation by addressing the hyperlinks connectivity rather than measuring the network traffic. The algorithm consists of two phases, a weights initialization and a weights update. The algorithm starts by assigning an initial weight for each node. Given a SAG with N nodes, the initial weight of the node $n \in N$ is computed in the following equation (Equation 5.1.

$$W_n = D_n^i + D_n^o, \tag{5.1}$$

where W_n is the initial weight for node n. The D_n^i and the D_n^o correspond to the in-degree and the out-degree of the node n, respectively.

Next, we accumulate the weight of n's followers, which are the nodes that point to it, and the weight of its followings, the nodes that are referenced by n. Finally, the weights

are calculated again for each node using 5.2. A rank value, TR_n, is assigned according to the weight of the node, such that the higher weight corresponds to the higher rank.

$$TR_n = W_n \log(1 + \alpha W_{fr} + \beta W_{fw}), \tag{5.2}$$

where W_{fr} and W_{fw} are the accumulated weight of followers and the followings of the node n, respectively. The parameters α and β control the contribution of the followers and the followings nodes to the weight of n. More specifically, ToRank formula considers the accumulated weight of the follower and the following nodes. Influenced by the PageRank formula, the neighboring nodes do not have an equal impact; however, both are still valuable factors to identify the influence of the node n. The role of α and β comes to capture this difference in the weight of the followings and followers nodes. When both are set to zero, TR_n would be exactly the degree centrality, which is not a recommended approach - see the description of the degree centrality in Section 5.1.3. In contrast, when both are equal to one, the formula will not reflect the intended purpose of giving higher importance for the following nodes.

During the labeling process, we observed the existence of nodes that function like directories, or wiki pages, that are pointing out to hundreds or even thousands of domains in the network, but they are referenced by very few domains or none. The removal of such nodes would fragment the graph strongly; unfortunately, their detection is worthless as they are not practicing any suspicious action. Hence, in the Tor network, ToRank recommends assigning a high value to α to increase the weight of the follower's nodes but a low value to β to decrease the impact of the following nodes.

ToRank is intended to detect the most critical nodes in the Tor network, but no ground truth tells that. The straightforward approach is to use degree centrality, but its main limitation is that it does not consider the graph structure. We use the degree centrality to initialize the weights of the nodes in a graph. Then, to overcome the shortcoming of the degree centrality, we enhanced the formula of ToRank by incorporating the neighbor's degree. ToRank introduces the usage of the logarithm function to the neighbor's weight, so they do not overshadow the weight of the studied node. Therefore, in ToRank algorithm, the rank value depends on the weight of the studied node and its neighbors, and consequently, the weights of those neighbors depend on their neighbors in cascade. The weight W_n of the node n is calculated accordingly with the weight of its neighbors. The logarithm function is used to respond to deviation towards the nodes which have a very high degree, and the first term of the expression, the number one, is added to avoid the indeterminate form when the value of log argument is zero.

Benchmarked link-based algorithms

PageRank. It was developed by Page et al. (1999), and it is considered as an enhanced version of the in-degree centrality. It assumes that an influential node is likely to receive more links from other influential nodes. The influence of a given node i is calculated

iteratively using the equation (Equation 5.3), and it automatically stops when it converges or reaches a maximum number of iterations.

$$PR(i) = (1-d) + d \sum_{j \in B(i)} \frac{PR(j)}{N_j}, \tag{5.3}$$

where d is the damping factor which indicates the probability of a random surfer to continue or stop navigating the graph nodes, i and j are nodes of a directed graph G, $B(i)$ is the set of nodes that point to i, $PR(i)$ and $PR(j)$ are rank scores of the nodes i and j respectively. N_j indicates the number of outgoing links of the node j. A high-rank value reflects a greater influence of a node over the other nodes.

HITS. The Hyperlink-Induced Topic Search (HITS) algorithm (Kleinberg, 1999) measures the nodes importance recursively by assigning two mutual scores for each node: a hub score hub_i, which grows higher if a node is referencing many high authority scores nodes, and an authority score $auth_i$, which increases if a node is referenced by many high hub scores nodes. To this end, the recursion behavior is defined: good hubs are those nodes that reference many good authorities, which are those referenced by many good hubs. Each node i in a graph G holds two non-negative scores, an authority score $auth_i$ and a hub score hub_i, and they are initialized with arbitrary nonzero values. Next, the scores are updated iteratively until convergence; which reaches a stationary solution. Equation (5.4) shows an update to the hub and another to the authority scores, which captures the intuitive notions behind the HITS algorithm.

$$\begin{aligned} auth_i^{(k)} &= \sum_{j \to i \in E} hub_j^{(k)} \\ hub_i^{(k)} &= \sum_{i \to j \in E} auth_j^{(k)}, \end{aligned} \tag{5.4}$$

where j is a node in G and $i \to j$ indicates a hyperlink from the node i to the node j out of the graph edges set E. k is an iterator index that starts from 1 and increases to ∞, but in practice, this loop is beaked where there is no significant change between consecutive iterates or according to a maximum number of iterations variable.

Katz. Introduced by Katz (1953), it is used to measure the centrality of a node by assigning a score that depends on the first-degree neighbors and the nodes connected with them. In mathematical form, the rank is calculated according to Equation (5.5).

$$k_i = \alpha \sum_j A_{ij} k_j + \beta, \tag{5.5}$$

where k_i and k_j are Katz centrality values for the nodes i and j in a given graph G, A_{ij}

is the adjacency matrix of G that captures the connectivity of the nodes. β corresponds to the initial centrality, and α corresponds to the attenuation factor.

Graph robustness metrics

Fronzetti Colladon and Gloor (2018) carried out a comprehensive study regarding graph robustness and stability metrics. Below, we briefly explore a few of them that we used to evaluate the benchmark algorithms.

Degree centrality. Based on the number of connected nodes that are direct neighbors of a given node, the degree centrality measure uses three different values in directed graphs: in-degree, out-degree, and degree. The in-degree counts only the number of incoming links from a given node, whereas the out-degree counts the number of outgoing ones (Seidman, 1983). The degree of a node is calculated as the sum of the in-degree and the out-degree values. However, this approach ignores the global structure of the graph and focus only on the direct neighbors of the targeted node (Wei et al., 2018). To overcome this limitation, Srinivas and Velusamy (2015) proposed an enhanced version of the degree centrality that incorporates the clustering coefficient.

Graph density. It is defined as the number of existing edges over the number of possible ones. Hence, the more connected is the graph, the higher the density and vice versa. When the graph is fully connected, the density is equal to one, and it is equal to zero when it is free of edges. The density of a directed graph G is computed as shown in Equation (5.6) where E is the number of edges, and N is the number of nodes in the directed graph G.

$$D = \frac{E}{N(N-1)} \tag{5.6}$$

Average shortest path (ASP). It refers to the average length of the shortest paths along with all possible pairs of network nodes (Mao and Zhang, 2017).

Diameter (Dim). It is defined as the longest path among the shortest paths between any two nodes in a given graph (Ye et al., 2010). The removal of central nodes that occupies a core location in the graph would increase the shortest paths, and consequently, the diameter of the graph will increase.

Clustering coefficient (CC). It calculates the number of triangles in a graph. It is calculated by dividing the number of closed triples of nodes by the total number of connected triples in the network (Watts and Strogatz, 1998).

Giant component (GC). It refers to the largest fraction of nodes that are connected, i.e., there exists a path between each pair of nodes in that component. An attack on the graph robustness could be measured by the reduction in the giant component mass (Holme et al., 2002).

5.1.4. Experimental Results

Experimental setting

The experiments were conducted on a PC with an Intel(R) Core(TM) i7 processor with 32 GB of RAM under Windows-10. The domain addresses were extracted from DUTA-10K using a regular expression. The SAG was constructed using the NetworkX library[5] with Python3. For the graph visualization, we used vis.js library[6]. Concerning the ranking algorithms, we compared ToRank with the link-based ranking algorithms presented in Section 5.1.3, namely PageRank, HITS, and Katz. We tuned all the methods by evaluating a range of values for each parameter, as shown in Table 5.2, and we selected the ones, which achieved the highest performance in our experiments. ToRank has two configurable parameters α and β that were set empirically to to 0.9 and 0.2, respectively[7].

Table 5.2: Evaluated values for the parameters of the ranking algorithms. Bolded numbers correspond to the selected configuration that achieved the lowest area under the GDC curve.

Algorithm Name	Parameter	Experienced values
PageRank	alpha	0.5, 0.70, 0.75, 0.80,**0.85**,0.90
	max_iter	10, 100, **1,000**, 10,000
HITS	max_iter	10, 100, **1,000**, 10,000
Katz	alpha	0.01, **0.1**, 0.2, 0.3, 0.4, 0.6, 0.9
	beta	0.1, 0.3, 0.5, 0.7, 0.9, **1.0**
	max_iter	10, 100, **1,000**, 10,000
ToRank	alpha	0.1. 0.2, 0.4, 0.6, 0.8, **0.9**, 1.0
	beta	0.1, **0.2**, 0.4, 0.6, 0.8, 0.9, 1.0

Evaluation measure

Consistently with previous research (Booker, 2012; Fronzetti Colladon and Gloor, 2018; Fronzetti Colladon and Vagaggini, 2017), we employed several standard metrics to judge the structure of the studied graph - see their explanation in Section 5.1.3). Concerning the density criterion, the evaluation procedure starts by peeling away the top-ranked nodes one by one iteratively, and at every cycle, the graph density is evaluated. The iterator stops when the graph is completely disconnected while the density is zero.

[5] `https://networkx.github.io/`

[6] `http://visjs.org/`

[7] we refer the reader to Section 5.1.3 for a deeper explanation of these parameters

We consider that the ranking algorithm that achieves the lowest area under the Graph Density Curve (GDC) corresponds to the algorithm that better measures the influence of a domain inside the Tor network.

Consequently, the GDC is used as a proxy to evaluate the graph robustness (Wang et al., 2014), and hence, the iterative removal of the top-ranked nodes with their edges would result in a reduction in the graph density. Therefore, if the nodes are correctly ranked, the top-ranked nodes would lead to a high reduction in the density because the removal of this node should cause a harmful fragmentation to the graph structure. However, if the removed nodes are not influential, its removal will not break the graph, and hence, it should not be ranked at the top of the list.

Besides looking at the graph density, we consider the reduction in the size of the giant component (GC) and the decrease in the clustering coefficient (CC) as efficient indicators of the produced disruption. However, the one by one node removal strategy is an expensive process due to the time needed to calculate the GC and the CC of the graph at every iteration. Alternatively, we analyze only the removal of the top-(1^{st}, 5^{th} and 10^{th}) percentile of the ranked nodes, and hence, the GC and CC metrics are evaluated three times only.

A higher decrease in the giant component size, the graph density, and the clustering coefficient reflect more disruption to the graph robustness (Chang, 2017; Iyer et al., 2013) and consequently a better ranking. Similarly, the diameter and the average shortest path measures can be used to test the robustness at multi-levels of top-ranked nodes removal[8] (Cohen and Havlin, 2010). An increase in the graph diameter and the average shortest path indicate a better ranking, due to removing the top-ranked nodes from the graph. Hence, the higher the ASP and the Dim and the lower the CC and the GC are, the better the ranking algorithm.

Analysis of the suspicious activities in Tor

We created two types of Suspicious Activities Graph (SAG): first, a SAG for all the suspicious activities, and second, a SAG for every single suspicious activity in DUTA-10K. We denoted them by SAG_{All} and SAG_X, respectively, where X corresponds to the activity name, for example, SAG_{Drugs} refers to the drugs HS. Table 5.3 shows the specifications of SAG_{All} and SAG_X graphs.

Figure 5.7 shows five different Graph Density Curves (GDC) of SAG_{All} for the four ranking algorithms[9]. ToRank outperforms the other methods because it achieves the lowest area under the GDC, with a value of 1.31, presenting as well a homogeneous decrease in the density curve. In contrast, PageRank suffers from a sudden increase in its curve, then a sharp decrease, which yields the highest GDC of 2.07 (Table 5.4). This phenomenon is discussed in Section 5.1.5. In Figure 5.7 it can be observed how the density reaches zero

[8] We could not manage a one-by-one node removal due to metrics complexity; hence we did it over the top-1, 5, and 10 percentile of the ranked nodes only

[9] HITS algorithm produces two curves, one for the authorities, and another for the hubs

Table 5.3: Details of the created Suspicious Activities Graphs. The *#Activity nodes* column refers to the number of nodes related to the studied activity, the *SAG #nodes* and the *SAG #edges* columns represent the number of nodes and the number of edges in the corresponding SAG.

Activity name	**#Activity domains**	**SAG #nodes**	**SAG #edges**	**Average degree**
All	2,013	2,908	14,511	4.99
C. Credit Cards	399	583	2,622	4.49
Forum: Suspicious	63	436	1527	3.50
Violence	95	240	795	3.31
C. Money	83	202	796	3.94
C. Personal Identification	72	180	763	4.23
Marketplace: suspicious	127	389	1670	4.29
Drugs	465	743	4,130	5.55
Hacking	205	402	1,381	3.43
CryptoLocker	185	198	611	3.08
Services: suspicious	20	46	76	1.65
Pornography	253	686	2,765	4.03
Fraud	43	145	386	2.66
human-trafficking	3	15	16	1.06

but the domain count is 13. Those nodes are normal HS that were referenced by suspicious onion domains such as wiki pages or Tor directories pages.

Table 5.4: Comparison between the four ranking algorithms over the SAGs in terms of GDC. The bold numbers correspond to the lowest GDC value.

Activity name	**PageRank**	**HITS$_{\text{Auth}}$**	**HITS$_{\text{Hub}}$**	**Katz**	**ToRank**
All	2.07	1.63	1.96	1.43	**1.31**
C. Credit Cards	2.00	1.65	2.13	1.51	**1.48**
C. Money	0.64	0.56	0.72	**0.52**	**0.52**
C. Personal Identification	0.73	0.63	0.95	**0.60**	**0.60**
CryptoLocker	2.41	2.53	3.48	2.32	**2.29**
Drugs	2.24	1.74	2.19	1.58	**1.49**
Forum: Suspicious	0.48	0.36	0.35	0.35	**0.26**
Fraud	0.68	0.62	0.57	0.54	**0.42**
Hacking	0.90	0.72	0.80	0.68	**0.63**
Marketplace: Suspicious	1.25	0.94	1.10	0.87	**0.80**
Pornography	0.76	0.52	0.67	0.50	**0.42**
Services: Suspicious	0.44	0.43	0.52	0.34	**0.33**
Violence	0.69	0.60	0.66	0.52	**0.51**

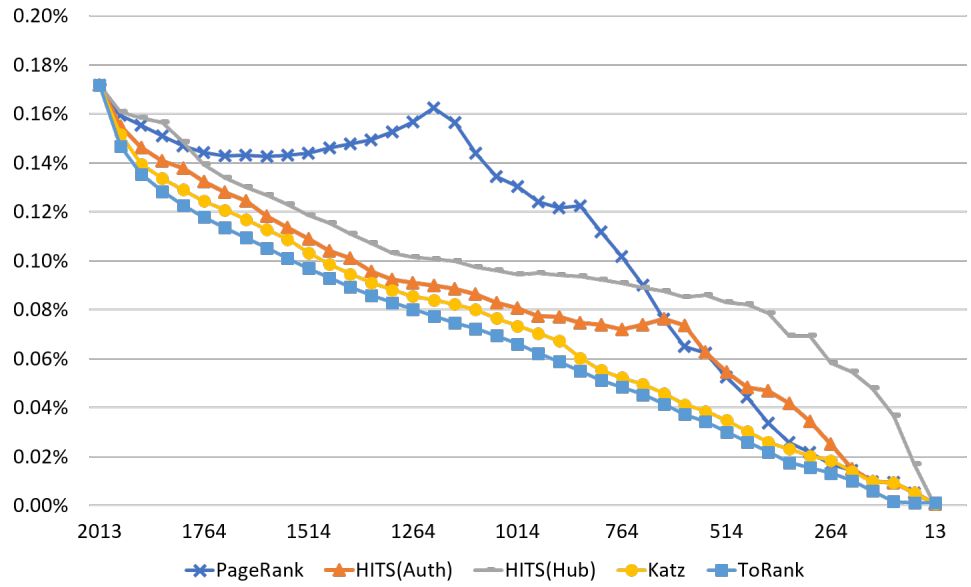


Figure 5.7: Density Analysis for the SAG_{All} of DUTA-10K. The X-axis refers to the current number of nodes, while the Y-axis the corresponding graph density. It can be seen that ToRank outperforms the other algorithm as it achieves the lowest GDC.

Figure 5.8 compares the Graph Density Curve (GDC) of the four ranking algorithms on SAG_{All} and only 12 single activities graphs. Human-Trafficking was not evaluated, because it contains three onion domains, two with identical content, yielding two unique domains without any hyperlink between them. The lowest GDC area obtained by ToRank proves its superiority to the other algorithms. However, despite the excellence of ToRank over Katz, the latter is approaching ToRank in all of the suspicious categories except the counterfeit of personal identification and the counterfeit of money where they have the same GDC value (see Table 5.4). This is because ToRank and Katz use the in-degree, but the advantage of ToRank is based on the use of the out-degree of the domains and its weighting.

In addition to the density analysis, Table 5.5 explores four graph structure metrics: it compares the full network structure before applying the top-ranked node removal, with the three levels of nodes reduction (1, 5, and 10 percentile of the full network). From the table, we can see that ToRank achieved the sharpest reduction in the GC with a high decrease in the CC. Also, with ToRank, the Average Shortest Path (ASP) increased to 8.9, which reflects a higher disruption to the network structure by removing the core nodes first. This observation is reflected in the graph dimension as it increased from nine to 22 after removing the top-10% of the top-ranked nodes. Those observations prove that ToRank outperforms the other ranking algorithms.

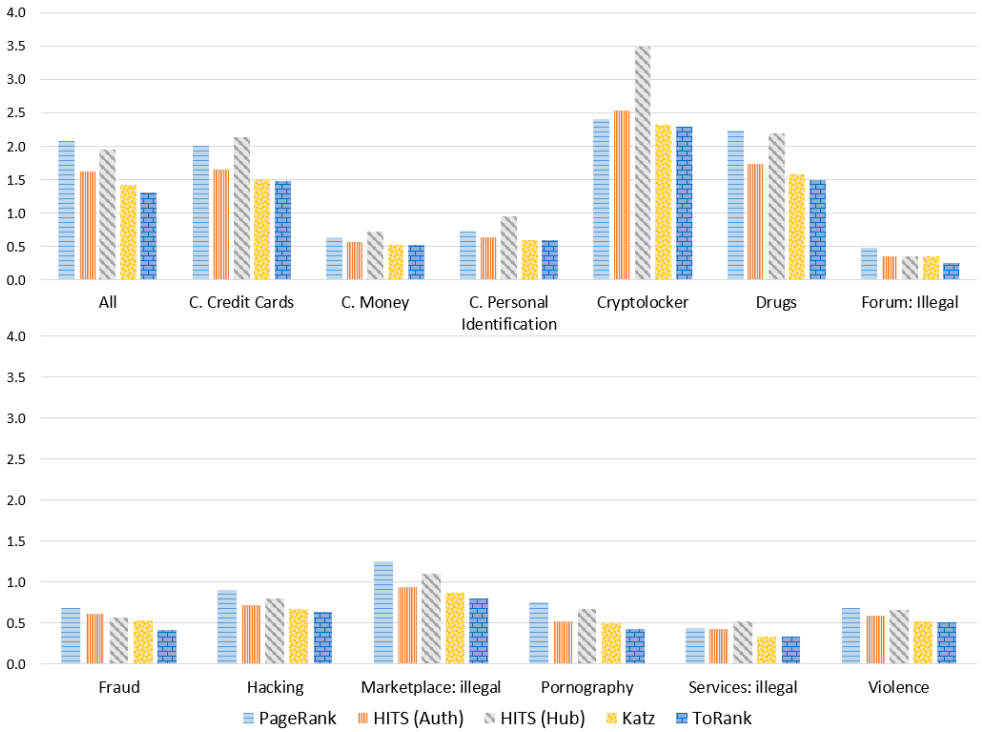


Figure 5.8: Comparison of the GDC value on four ranking algorithms with respect to the suspicious activities of DUTA-10K. The vertical axis represents the GDC value for the corresponding ranking algorithm, whereas the horizontal axis shows the 12 individual activities SAG_X, plus all the interesting activities graph SAG_{All}.

In Table 5.6, we explore the top-10 onion domains nominated by ToRank algorithm as influential HS within SAG_{All}.

Analysis of the Normal Activities in Tor

We carried out the same analysis for the normal activities of DUTA-10K (see Figure 5.9). We created a Normal Activities Graph NAG. Then, we ranked the nodes according to the four ranking algorithms and evaluated the performance using the GDC measure. The NAG has 22,965 nodes where 5,016 of them are from the normal activities. The left ones are HS that are connected to them, and the edges count to 85,699 with an average node degree of 3.73. Figure 5.9 shows that ToRank has the lowest area under the GDC with value of 0.02 where $HITS_{Hub}$, Katz, $HITS_{Auth}$, and PageRank achieved 0.03, 0.17, 0.22 and 0.26 respectively. The good performance of ToRank and $HITS_{Hub}$ is due to the early detection onion domains that are directories, which have a high number of connections with other HS in Tor. Due to their impact on the GDC curve, the conclusion is that the

Table 5.5: Impact of top-ranked nodes removal on four graph metrics with respect to three graph datasets: suspicious activities, normal activities, and 9/11 Hijackers Network. Clustering Coefficient (CC), Average Shortest Path (ASP), Giant Component (GC) and Diameter (Dim), and FN refers to the full network before nodes removal. The bolded values refer to the best performance whereas the object is to decrease the CC and the GC and to increase the ASP and the Dim variables.

		Suspicious Activities Network				**Normal Activities Network**				**9/11 Hijackers Network**			
Metrics	**Algorithms**	**FN**	**Top-1%**	**Top-5%**	**Top-10%**	**FN**	**Top-1%**	**Top-5%**	**Top-10%**	**FN**	**Top-1%**	**Top-5%**	**Top-10%**
	ToRank		**0.042**	**0.029**	**0.023**		**0.009**	**0.004**	**0.003**		0.465	**0.334**	**0.303**
	PR		0.048	0.041	0.042		0.299	0.285	0.267		**0.452**	**0.334**	0.342
	HITS (Hub)		0.055	0.043	0.023		0.013	0.004	0.004		0.471	0.458	0.449
CC	HITS (Auth)	0.056	0.055	0.051	0.039	0.3577	0.357	0.341	0.252	0.476	0.465	0.446	0.441
	Katz		0.051	0.034	0.027		0.355	0.370	0.391		0.465	0.446	0.339
	ToRank		**1,539**	**998**	**406**		**1,156**	**42**	**32**		**59**	55	34
	PR		2,691	2,498	2,351		15,690	14,760	13,620		**59**	55	28
	HITS (Hub)		1,880	1,048	452		2,964	619	129		**59**	57	54
GC	HITS (Auth)	2,748	2,705	2,484	2,272	22,572	22,281	21,316	14,118	60	**59**	**31**	**26**
	Katz		2,686	2,399	2,166		21,249	20,481	19,086		**59**	**31**	**26**
	ToRank		**5.234**	**7.684**	**8.902**		**7.340**	**6.066**	**5.371**		**4.291**	**4.698**	3.695
	PR		3.174	3.193	3.255		2.665	2.698	2.725		3.653	**4.698**	3.183
	HITS (Hub)		4.658	5.956	6.441		7.014	1.997	1.985		3.635	3.677	**4.253**
ASP	HITS (Auth)	3.151	3.169	3.225	3.281	2.690	2.755	2.800	2.934	3.606	**4.291**	3.179	3.129
	Katz		3.180	3.229	3.281		2.593	2.550	2.399		**4.291**	3.179	3.129
	ToRank		**15**	**19**	**22**		18	**14**	**12**		**9**	**11**	**9**
	PR		9	9	9		8	8	8		7	**11**	7
	HITS (Hub)		12	17	18		**20**	2	2		7	7	**9**
Dim	HITS (Auth)	9	9	10	10	8	8	8	8	7	**9**	7	7
	Katz		10	10	10		8	7	6		**9**	7	7

Table 5.6: The top-10 HS ranked using ToRank algorithm

Rank	**HS Address**	**Activity Category**	**HS Title**	**Short Description**
1	matangareonmy6bg.onion	Drugs	-	Online market for suspicious drugs
2	y3nau3mnibjbpmh4.onion	Pornography	Tor Links 2.0	A Tor directory for pornography content
3	hansamkt2rr6nfg3.onion	Marketplace suspicious	HANSA Market	A famous marketplace for suspicious product
4	vfvfq64rtrefmdtd.onion	Drugs	-	Russian market for suspicious drugs
5	silkkitiehdg5mug.onion	Drugs	Silkkitie	Valhalla Market (known by its Finnish name, Silkkitie)
6	shops3jckh3dexzy.onion	Drugs	-	Online market for suspicious drugs
7	abbujjh5vqtq77wg.onion	C. personal identification	Onion Identity Services	HS for producing fake passports
8	gxmrzk2s56oxzb3e.onion	Pornography	German Pi-X-Board	Multi-languages forum for child-pornography
9	boysopidonajtogl.onion	Pornography	Central Park	Guides and links to porno websites
10	newpdsuslmzqazvr.onion	Drugs	Peoples Drug Store	Online market for suspicious drugs

onion directories are one of the main ways to redirect Tor users to the activities domains.

Table 5.5 shows that ToRank outperformed other algorithms, even for the normal activities graph. This superiority is observed in the reduction of the CC and the GC as well as an increase in the ASP. However, in this case, the PageRank achieved better diameter after removing the top-1% of the nodes only.

Analysis of the 9/11 Hijackers Network

With the intention of evaluating ToRank's generalization capability, we looked for other similar datasets or at least other problems where the main purpose was to rank collected data, like the 9/11 Hijackers Network. This is a famous dataset containing in-

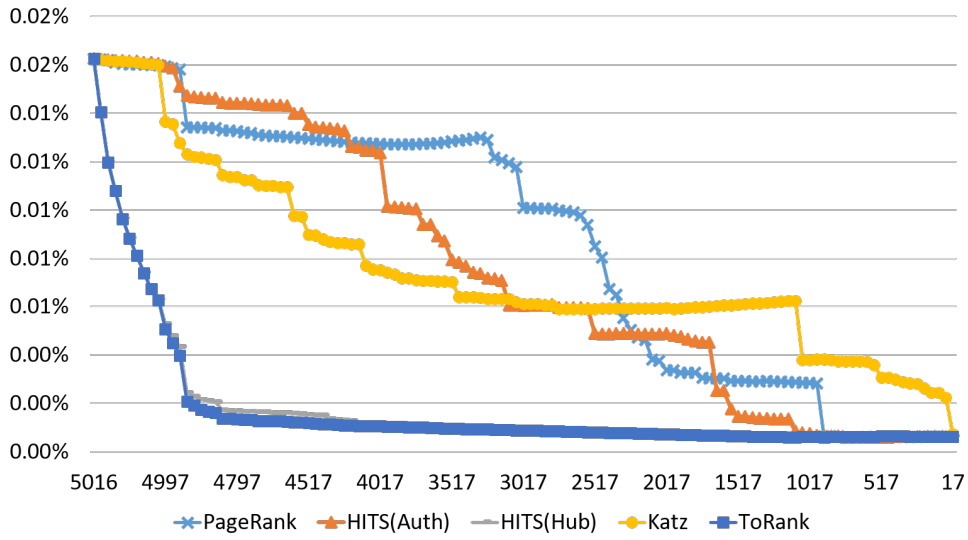


Figure 5.9: Graph density analysis for the NAG. The X-axis refers to the current number of nodes, while the Y-axis the corresponding graph density.

formation about the terrorists involved in the 9/11 bombing of the World Trade Center, in 2001 (Krebs, 2002). The objective is to detect the most influential people who contributed to that attack, whereas the most influential node, the key-player, is the one whose removal would produce a significant break in the connectivity between the other network members. To apply the link-based ranking algorithms, we represented the 9/11 Hijackers Network by a directed graph of nodes, which refers to the hijackers, and edges, to describe a relation between any two people. Each undirected edge was replaced with two directed ones.

There are plenty of research papers regarding the analysis of the network structure (Husslage et al., 2015; Memon and Larsen, 2006), but fewer works about ranking them. Therefore, we considered the rank proposed by Choudhary and Singh (2016) as the ground truth to compare with our work. Kendall's tau coefficient (Kendall, 2008),was used to assess the correlation between two ranked lists. Its value ranges between $+1$ and -1, such that the closer the value to $+1$ or -1, the stronger the relationship, while the closer the value to zero, the weaker the relationship. Table 5.7 shows the top-10[10] Hijackers nominated by each ranking algorithm and a comparison with the ground truth rank with respect to Kendall's tau measure.

Additionally, we tested the GDC evaluation procedure used in this work to compare the ranking algorithms (see Figure 5.10), and we found that, again, ToRank outperformed the other algorithms with a GDC of 1.13, while the other ranking algorithms were as fol-

[10]We selected the top-10 only because the ground truth enumerates only the top-10 Hijackers.

Table 5.7: The top-10 Hijackers in 9/11 dataset

Ground truth	**PageRank**	**$HITS_{Auth}$**	**$HITS_{Hub}$**	**Katz**	**ToRank**
M.Atta	M. Al-Shehhi	M.Atta	R.al-Shibh	M.Atta	M.Atta
E.Khemail	M. Atta	M.Al-Shehhi	S.Bahaji	M.Al-Shehhi	E.Khemail
Z.Moussaoui	E. Khemail	Z.Jarrah	Z.Essabar	Z.Jarrah	M.Al-Shehhi
H.Hanjour	Z. Jarrah	R.al-Shibh	A.Budiman	E.Khemail	D.Benghal
N.Alhazmi	A. Al-Omari	S.Bahaji	M.Motassadeq	Z.Moussaoui	Z.Moussaoui
M.Al-Shehhi	W. Alshehri	Z.Essabar	L.Raissi	D.Benghal	R.al-Shibh
S.Suqami	H. Alghamdi	M.Motassadeq	M.Darkazanli	A.Qatada	T.Maaroufi
D.Benghal	D. Benghal	Z.Moussaoui	Z.Moussaoui	T.Maaroufi	Z.Jarrah
R.al-Shibh	H. Hanjour	A.Qatada	M.al-Hisawi	H.Hanjour	A.Qatada
Z.Jarrah	N. Alhazmi	D.Benghal	A.Al-Omari	R.al-Shibh	N.Alhazmi
Kendall's tau	0.47	0.43	0.05	0.63	**0.70**

lows: 1.48 Katz, 1.70 PageRank, 2.55 $HITS_{Auth}$, and 2.98 $HITS_{Hub}$.

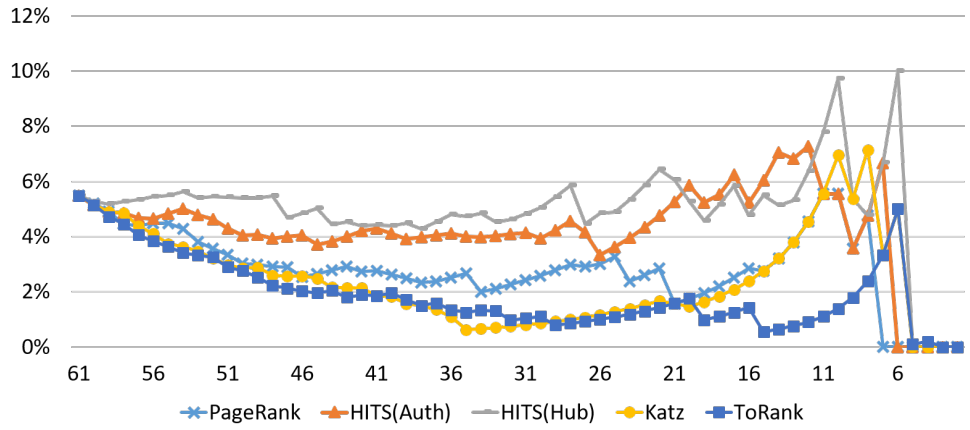


Figure 5.10: Graph Density Analysis for the 9/11 Hijackers dataset

Table 5.5 shows that ToRank is slightly better than the other algorithms. However, the results of the algorithms are significantly close due to the small number of nodes in this graph.

5.1.5. Discussion

We designed the proposed algorithm, ToRank, to detect the nodes that contribute the most to both propagating and receiving traffic to and from the other nodes. By setting the initial weight of each node as its degree, ToRank establishes an initial measure of each node's influence. Later on, the starting value is adjusted using the weights of its adjacent nodes. Therefore, the weight of every node depends on the weight of its direct and its in-

direct neighbors, and it considers the degree of the followings and follower's nodes. Also, what distinguishes ToRank from the benchmarked algorithms is that it is not iterative like PageRank or HITS, so it does not have a convergence behavior, and it considers the degree of the followings and follower's nodes.

It was surprising to find out that, probably, the most popular ranking algorithm for the surface web, PageRank, performs poorly for this problem according to the graph density analysis metric. We found that the iterative removal of the top-ranked nodes using PageRank yielded the bigger area under the density curve, especially after removing between 10% and 20% of the top-ranked nodes. The reason for that is that PageRank assigns high ranks to low connectivity nodes, and during the iterative evaluation, those nodes were dropped first, resulting in a decrease in the number of nodes without a significant impact on the graph connectivity. Hence, the number of edges is kept high while the count of nodes decreases, which leads to an increment in the density curve and a high GDC value. Additionally, the GDC evaluation procedure measures the efficiency of a ranking algorithm in breaking down the connectivity of a given graph, whereas the PageRank design was not for that end. Therefore, we conclude that despite the popularity of PageRank, it is not a suitable ranking algorithm for this task.

However, notwithstanding the success of ToRank algorithm in detecting and ranking the influential nodes in Tor, our method fails in assessing onion domains that are isolated with no incoming or outgoing hyperlinks. Luckily, this issue barely has an impact on our problem because in SAG_{All} there are only 94 onion domains that do not have any hyperlinks to or from HS within Tor, which counts only 3% out of the total.

ToRank Computational Complexity

ToRank has two phases: the initialization phase, which iterates over all the nodes to assign an initial weight to them, and a ranking phase that calculates a rank value thanks to ToRank equation. Hence, ToRank complexity is proportional to the number of nodes N, and the time complexity would be $O(2N)$. Table 5.8 compares the needed time to rank relatively large and small graphs with their corresponding processing time.

ToRank with Big Graphs

Besides the explored graph structures, we ran the ranking algorithms over three relatively large graphs, as it is depicted in Table 5.8. In particular, we experimented with Google web graph [11] (Leskovec et al., 2009), Stanford web graph[12] (Leskovec et al., 2009), and Note Dame web graph[13] (Albert et al., 1999). We explored only the reduction in the giant component metric as we could not manage to the other metrics due to their high time complexity using NetworkX library. Also, we compared the processing time of ToRank

[11] `https://snap.stanford.edu/data/web-Google.html`
[12] `https://snap.stanford.edu/data/web-Stanford.html`
[13] `https://snap.stanford.edu/data/web-NotreDame.html`

Table 5.8: A comparison for the processing time, in terms of seconds, of multiple ranking algorithms. The first column to the left refers to the graph name and its number of nodes (N) and edges (E). Using NetworkX library, we were not able to rank relatively big graphs, so we replaced it with (-) sing.

Graph Structure	PageRank	HITS	Katz	ToRank
Google web graph Source: (Leskovec et al., 2009) (N=875713, E=5,105,039)	284.99	128.95	-	**32.42**
Note Dame web graph Source: (Albert et al., 1999) (N=325,729, E=1,497,134)	90.40	74.87	-	**11.14**
Stanford web graph Source: (Leskovec et al., 2009) (N=281,903, E=2,312,497)	141.52	22.90	-	**11.72**
Suspicious Activities (N=2,908, E=14,511)	1.47	0.42	0.78	**0.14**
Normal Activities (N=22,965, E=85,699)	5.98	36.33	99.85	**0.80**
9/11 Hijackers Network (N=60, E=194)	**0.03**	0.07	0.62	0.05

with the other ranking algorithm with respect to the commented graphs. Table 5.9 shows that both ToRank and PageRank algorithms are competing for the first position. ToRank outperformed PageRank in Note Dame web graph but failed in Google Web graph. However, considering the GC reduction time, shown in Table 5.8, ToRank is superior to PageRank for large graphs.

Table 5.9: Comparison of the reduction in the Giant Component (GC) of multiple large graphs. The lower GC is, the better ranking.

Graph Structure	Full network giant component	Removal	PR	HITS(Auth)	HITS(Hub)	Katz	ToRank
Google web graph	855,802	Top-1%	**771,062**	818,528	845,134	-	772,966
		Top-5%	**626,437**	714,767	801,869	-	665,315
		Top-10%	**502,855**	603,827	742,537	-	525,710
Note Dame web graph	325,729	Top-1%	254,729	315,000	322,468	-	**228,957**
		Top-5%	207,088	285,246	280,401	-	**125,635**
		Top-10%	180,889	225,480	157,022	-	**50,600**
Stanford web graph	255,265	Top-1%	204,408	232,634	252,396	-	**200,723**
		Top-5%	**127,013**	22,1680	241,120	-	129,735
		Top-10%	**77,765**	183,794	223,617	-	104,960

5.2. Content-based Ranking

The main advantage of the link-based approaches is that it does not require a training dataset, as it targets the connectivity of the hyperlinks between the onion domains. However, it would fail in evaluating isolated domains, which do not have connections to the rest of the community. Furthermore, it ignores a fundamental factor while evaluating the importance of a node, which is the content of the domain itself. Therefore, as we will show in the experiments section, the link-based approach is surpassed by a supervised ranking approach. The content-based ranking overcomes the aforementioned limitation of the link-based technique. Typically, it extracts features from the content of the domains, and use them for training a supervised ranking model. The main benefit of using a supervised content-based ranking approach is its capability of adapting predefined ranking criteria. Concretely, we employ a Learning to Rank (LtR) algorithm to capture characteristics of a given ranked list and to map the learned rank into a new unsorted list of items.

5.2.1. Methodology

Our ranking model has two components: 1) *Hidden Service Modeling Unit (HSMU)*, which analyzes and extracts features from a given onion domain, and 2) the *Supervised Learning to Rank Unit (SLRU)* that learns a function to order a collection of domains according to the pattern captured from previously sorted samples, a training set (see Figure 5.11).

Hidden Service Modeling Unit

Given a domain $d_i \in D$, where D is a set of onion domains or hidden services (HS), the HSMU analyzes d_i to extract features belonging to five different categories: text, named entities, HTML, visual content, and network topology. Then, the HSMU encodes those features into numerical values that represent the HSs analyzed.

Text Features. The set of text features involves four types of descriptors constructed from the text in d_i, which is visible to the user.

- **Date and Time.** A binary feature indicates whether d_i has been updated recently or not. A domain might be updated by its owner or after receiving reviews from customers concerning the offered service. We parsed the date and time patterns, and we compared them with a configurable threshold, computing this binary feature. This threshold is a particular date-time point, whereas the actions beyond it are considered obsolete.

  Moreover, by counting these updates, we measured the number of recent changes that d_i has received in the past. We refer to these two features as *recently_updated* and *updates_counts*, respectively.

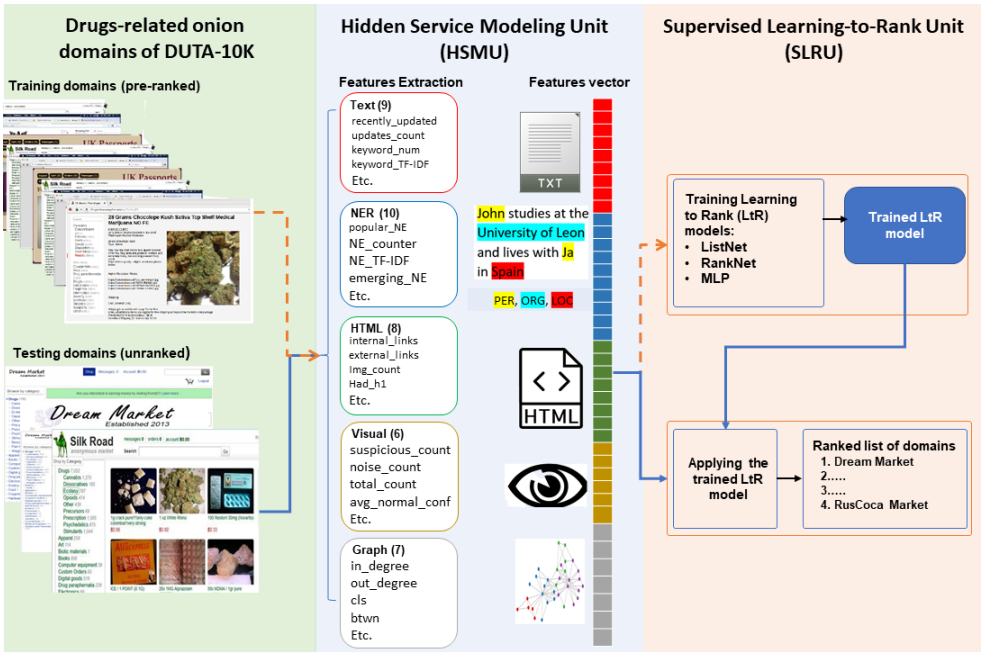


Figure 5.11: A general view for the proposed model for ranking and detecting the influential onion domains in the Tor network (better viewed in color). The dashed orange arrows indicate the training pipeline of the system, while the solid blue arrows indicate the testing/production phase.

- **HS Name.** An onion address consists of 16 characters randomly generated. The prefix of an HS can be customized with tools like Shallot[14], allowing that the onion domain address includes words attractive to the customers. For example, a drug marketplace could add the words *Cocaine* or *LSD* to its HS's URL. Nevertheless, the customization process is extensively time-consuming; for example, customizing the first seven characters requires one day of machine-time while customizing 10 characters takes 40 years of processing.

  To explore this characteristic, we split the concatenated words using a probabilistic approach based on English Wikipedia unigram frequencies thanks to Wordninja tool[15]. We obtained two features: (i) the number of human-readable words and (ii) the number of their letters. We named them as *address_words_count* and *address_letters_count*, respectively.

- **Clones Rate.** The number of onion domains that host the same content under different addresses. After reviewing the Darknet Usage Addresses Text (DUTA-10K)

[14] https://github.com/katmagic/Shallot
[15] https://github.com/keredson/wordninja

dataset[16], we realized that some onion domains have almost the same textual content but hosted under different addresses. This feature, i.e., *HS duplication*, might reflect concerns of the domains' owners about their services of being taken down by authorities. The *clones_rate* feature of d_i reflects the frequency of the MD5 (Rivest, 1992) hash code of its text.

- **Term Frequency-Inverse Document Frequency (TF-IDF) Vectorizer.** We used this text vectorization technique to extract and to weight domain-dependent keywords (Onan et al., 2016). It allowed us to draw the following four features: 1) *keyword_num*: the number of words that are in common between the TF-IDF feature vector and the domain's words - we consider these words as the *keywords* of the domain, 2) *keyword_TF-IDF_Acc*: the accumulated TF-IDF weights of the keywords, 3) *keyword_avg_weight*: the average weight of the keywords, and 4) *keyword_to_total*: the number of the domain's words dived by the number of its keywords.

Named Entities Features. A textual Named Entity (NE) refers to a real proper name of an object, including, but not limited to, persons, organizations, or locations. To extract those named entities, our previous work (Al-Nabki et al., 2019b) adopted a Named Entity Recognition (NER) model proposed by Aguilar et al. (2017) to the Tor Darknet to recognize six categories of entities: persons (PER), locations (LOC), organizations (ORG), products (PRD), creative-work (CRTV), corporation (COR), and groups (GRP). We map the extracted NEs into the following five features:

- **NE Number.** It counts the total number of entities in d_i regardless of the category; we refer to this feature by *NE_counter*.
- **NE Popularity.** An entity is popular if its appearance frequency value is above or equal to a specific threshold that we set to five, as is explained in Section 5.2.2. For every category identified by the NER model, we use a binary representation to capture the existence of popular entities in the domain (1) or (0) otherwise. We refer to this feature as $popular\_NE_X$ where X is the corresponding NER category.
- **NE TF-IDF.** It accumulates the TF-IDF weight of all the detected NE in d_i. This feature is denoted by *NE_TF-IDF*.
- **TF-IDF Popular NE.** It accumulated TF-IDF weight of the popular NE, and it is named *popular_NE_TF-IDF*.
- **Emerging NE.** The frequency of the emerging product entities in d_i. We used our previous work (Al-Nabki et al., 2017b) based on K-shell algorithm (Carmi et al., 2007) and graph theory to detect these entities in the Tor Darknet. We denote this feature by *emerging_NE*.

[16] http://gvis.unileon.es/dataset/duta-darknet-usage-text-addresses/

HTML Markup Features. Using regular expressions, we parse the HTML markup code of d_i to build the following eight features:

- **Internal Hyperlinks.** It refers to the number of unique hyperlinks that share the same domain name as d_i. We denote it by *internal_links*.
- **External Hyperlinks.** It indicates the number of pages referenced by d_i on the Tor network or in the Surface Web. We refer to this feature by *external_links*.
- **Image Tag Count.** It denotes the number of images referenced in d_i. It is calculated by counting the $< img >$ HTML tag in the HTML code of d_i. We denote it by *img_count*.
- **Login and Password.** A binary feature to indicate whether the domain needs login and password credentials or not. We use a regular expression pattern to parse such inputs. This feature is called *needs_credential*.
- **Domain Title.** A binary feature that checks whether the $< title >$ HTML tag has a textual value or not. We call it *has_title*.
- **Domain Header.** A binary feature that analyzes if the $< H1 >$ HTML tag has a header or not. We named it *has_H1*.
- **Title and Header TF-IDF**. An accumulation of the TF-IDF weight for the d_i title and header text. It is denoted by *TF-IDF_title_H1*.
- **TF-IDF Image Alternatives.** It indicates the TF-IDF weight accumulation of the alternative text. Some websites use an optional property called $< alt >$ inside the image tag $< img >$ to hold a textual description for the image. This text becomes visible to the end-user to substitute the image when it is not loaded properly. It is denoted as *TF-IDF_alt*.

Visual Content Features. It could be an attractive factor for drawing the attention of end-users, especially in the Tor HSs when the customer seeks to have a real image of the product before buying it. A suspicious services trader might incorporate real images of products to create an impression of credibility to a potential customer. However, the visual contents that are interesting for LEAs could be mixed up with other noisy contents, such as banners and images of logos.

To isolate the interesting images, we built a supervised image classifier that categorizes the visual content into nine categories, where eight of them are suspicious, and one is not. The definition of these categories is based on our previous works (Al-Nabki et al., 2017a, 2019c), where we created DUTA dataset and its extended version, DUTA-10K. The image classification model was built using the Transfer Learning (TL) technique (Lu et al., 2015) on the top of a pre-trained Inception-ResNet V2 model (Szegedy et al., 2017). The visual content feature vector has six dimensions distributed in the following manner:

- **Image Count.** It represents the total number of images in d_i, both suspicious and non-suspicious images regardless of their category. Suspicious stands for images that could contain illicit content. We denote these features by *total_count*, *suspicious_count* and *noise_count*, respectively.

- **Average Classification Confidence.** It represents the averaged confidence score of multiple images per category. These features are named *avg_suspicious_conf* and *avg_normal_conf*, respectively.

- **Majority Class.** A binary flag to indicate whether the majority of the images published in d_i are suspicious or not. This flag is denoted by *suspicious_majority*.

Network Structure Features. Following our previous work (Al-Nabki et al., 2019c), we modeled the Tor Darknet by a directed graph of nodes and edges. The nodes refer to onion domains, and the hyperlinks between domains are captured by the edges. This representation allowed us to built the following seven features:

- **In-degree.** It refers to the number of onion domains pointing to the domain d_i. It is called *in_degree*.

- **Out-degree.** It indicates the number of hidden services referenced by d_i, and it is named *out_degree*.

- **Centrality Measures.** For each domain d_i in the Tor network graph, we evaluated three node's centrality measures: *closeness*, *betweenness*, and *eigenvector* (Freeman, 1978). In particular, the closeness measures how short the shortest paths are from d_i to all domains in the network, and it is named *cls*. The betweenness measures the extent to which d_i lies on paths between other domains, and it is named *btwn*. Finally, the eigenvector reflects the importance of d_i for the connectivity of the graph and it is denoted *eigvec*.

- **ToRank Value.** ToRank is a link-based ranking algorithm to order the items of a given network following their centrality (Al-Nabki et al., 2019c). We applied ToRank to the Tor network to rank the onion domains, and we used the assigned rank as a feature of the node. Moreover, we used a binary flag to indicate whether d_i is in the top-X domains of ToRank or not. We refer to those features as *ToRank_rank* and *ToRank_top_X*, respectively.

After computing the forty features described (Table 5.10), we concatenate them to form a feature vector. However, given the variety of the scales of the features, we normalize them by removing the mean and scaling to unit variance.

Table 5.10: Summarization of the HSMU feature vector

Feat. Class	**Feat. Count**	**Feat. Source**	**Feat. Name**
Textual	9	Date and Time	- recently_updated - updates_count
		HS Name	- address_words_count - address_letters_count
		Clones Rate	- clones_rate
		TF-IDF Vectorizer	- keyword_num - keyword_TF-IDF - keyword_avg_weight - keyword_to_total
Named Entities	10	NE Popularity	- popular_NE (x)
		Total NE Number	- NE_counter
		TF-IDF NE	- NE_TF-IDF
		TF-IDF Popular NE	- popular_NE_TF-IDF
		Emerging NE	- emerging_NE
HTML Markup	8	Internal Hyperlinks	- internal_links
		External Hyperlinks	- external_links
		Image Tag Count	- img_count
		Login and Password	- needs_credential
		Domain Title	- has-title
		Domain Header	- has_H1
		TF-IDF Title and Header	- TF-IDF_title_H1
		TF-IDF Image Alternatives	- TF-IDF_alt
Visual Content	6	Images Count	- suspicious_count - noise_count - total_count
		Average Classification Count	- avg_suspicious_conf - avg_normal_conf
		Majority Class	- suspicious_majority
Network Structure	7	In-degree	- in_degree
		Out-degree	- out_degree
		Centrality Measures	- cls (closeness) - btwn (betweenness) - eigvec (eigenvector)
		ToRank Value	- ToRank_rank - ToRank_top_X
Total Features	40		

Supervised Learning to Rank Unit

To learn an optimal order of the onion domains through their descriptors, we adopt a Learning to Rank (LtR) approach adapted from the Information Retrieval (IR) field. In a

traditional IR problem, a training sample is a vector of three components: the relevance judgment, which can be binary (Wang et al., 2017c) or with multiple levels of relevance (Li, 2011a), the query ID, and the feature vector that describes the ranked instance.

However, our ranking system needs to answer a unique question: *What are the most attractive onion domains in a determined area of activities?*, and in the context of our problem, the relevance judgment of an item is a numerical score that is calculated and assigned per item. These scores are set manually by a group of annotators that we considered as ground truth for the ranking, whereas the annotation procedure is described in Section 5.2.3. Hence, the input vector for the SLRU is limited to the features of d_i and its relevant judgment r_i score in R, where R refers to the ground truth rank. The vector of each sample d_i can be modeled as $V = \{r_i, d_{i,1}, d_{i,2}, ..., d_{i,n}\}$, $n \in N$, where $d_{i,n}$ is the n_{th} feature of the domain d_i and N is the total number of the ranking features, i.e., $N = 40$.

Our LtR schema aims to learn a function f that projects a feature vector into a rank value $(d_{i,1}, d_{i,2}, ..., d_{i,n}) \xrightarrow{f} r_i$. Therefore, the goal of an LtR scheme is to obtain the optimal ranking function f that ranks D in a way similar to R. Typically, there are three popular architectures to design a LtR system: pointwise, pairwise, and listwise. In the following, a detailed description of each architecture is listed.

Pointwise. The loss function of the pointwise approach considers only a single instance of onion domains at a time (Friedman, 2001). It is a supervised classifier/regressor that predicts a relevance score for each query domain independently. The ranking is achieved by sorting the onion domains according to yield scores. For this LtR schema, we explore the Multi-layer Perceptron (MLP) regressor (Ciaramita et al., 2008).

Pairwise. It transforms the ranking task into a pairwise classification task. In particular, the loss function takes a pair of items at a time and tries to optimize their relative positions by minimizing the number of inversions comparing to the ground truth (Burges et al., 2005). In this work, we use the RankNet algorithm (Burges et al., 2005), which is one of the most popular pairwise LtR schemes.

Listwise. This approach extends the pairwise schema by looking at the entire list of samples at once (Xia et al., 2008). One of the most well-known listwise schemes is ListNet algorithms (Cao et al., 2007). Given two ranked lists, the human-labeled scores and the predicted ones, the loss function minimizes the cross-entropy error between their permutation probability distributions.

5.2.2. Experimental Settings

In this work, we presented a content-based ranking approach to sort the onion domains and to identify the most influential ones. While designing a ranking algorithm, we expressed each onion domain by 40 features that are extracted from five different sources,

that are: text, named entity, HTML, visual content, and network topology. Also, we explored three LtR architectures: pointwise, pairwise, and listwise. The conducted experiments in this section try to answer the following research questions:

- What is the most suitable LtR schema for ranking the onion domains in the Tor network and for detecting the influential ones?
- What are the advantages of each ranking approaches, the content-based and the link-based?
- And what is the best combination of features for the LtR model performance?

In the following, we discuss these questions, describing in detail the analytical approach that we conducted, and finally, we present our findings.

Evaluation Measure

The two most popular metrics for ranking Information Retrieval systems are Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain ($NDCG$) (Järvelin and Kekäläinen, 2000; Manning et al., 2010). The main difference between the two is that the MAP assumes a binary relevance of an item according to a given query, while $NDCG$ allows relevance scores in the form of real numbers. Two characteristics suggest the use of $NDCG$ to evaluate the problem addressed in this work. Thanks to the first component of the Tor network monitoring pipeline - the classification components-, all the addressed onion domains are already relevant to the query, and second, the ground truth and the predicted rank produced by any of the previously commented LtR schemes are real numbers, not binary ones. To obtain the $NDCG@K$ of a given query, we calculate the $DCG@K$ flowing Equation 5.7.

$$DCG@K = G_1 + \sum_{i=2}^{K} \frac{G_i}{log_2(i)}, \tag{5.7}$$

where G_1 is the gain score at the first position in the obtained ranked list, G_i is the gain score of the item i in that list, and K refers to the first K items to calculate the DCG. To obtain a normalized version of $DCG@K$ is necessary to divide it by $IDCG@K$, which is the ideal $DCG@K$ sorted by the gain scores in descending order (Equation 5.8).

$$NDCG@K = \frac{DCG@K}{IDCG@K} \tag{5.8}$$

Modules Configurations

Hardware Configurations. Our experiments were conducted on a 2.8 GHz CPU (Intel i7) PC running Windows 10 OS with 16G of RAM. We implemented the ranking models using Python3.

HSMU Configurations. In the TF-IDF text vectorizer, we set the feature vector length to 10,000 with a minimum frequency of three, following our previous work (Al-Nabki et al., 2017a). We used a NER model trained on WNUT-2017 dataset[17]. To set the popularity threshold of the *popular_NE$_X$* feature, we examined four values (3,5,10,15), and we assigned it to five, experimentally. Additionally, we set the threshold of the *recently_updated* feature to three months earlier to the dataset scraping date. To extract features from the HTML code, we used BeautifulSoup library[18]. To construct the Tor network graph, we used NetworkX[19] library.

In the image classifier, we used the Transfer Learning (TL) technique with the pre-trained Inception-ResNet V2 model released by Google[20]. The model was trained and tested with a dataset of 9,000 and 2,700 samples, respectively, equally distributed over nine categories. The motivation behind selecting these categories is to have the same classes as in our text classifier (Al-Nabki et al., 2017a). The dataset was collected from Google Images using a chrome plugin called *Bulk Image Downloader*. Table 5.11 shows the image classifier performance per category.

Table 5.11: The obtained F1 score by the image classifier over the testing set. The letter C. denotes *Counterfeit* activity.

Category Name	**F1 Score (%)**
C. Credit Cards	92.45
C. Money	96.78
C. Personal Identification	95.16
Cryptocurrency	94.60
Drugs	91.60
Pornography	98.53
Violence	93.80
Hacking	97.63
Others	86.78

SLRU Configurations. We used the dataset described in Section 5.2.3 to train and test the three LtR models introduced in Section 5.2.1. Due to the small number of samples in the drug domain, 290, we conducted 5-fold cross-validation following recommendations from previous works (Cao et al., 2007). On each iteration, three folds are used for training the ranking model, one fold for validation, and one fold for testing. For the three LtR models, the number of iterations is controlled by an early stopping framework, which is triggered when there is no remarkable changes in the validation set at $NDCG@10$ (Lai et al., 2013).

[17] https://noisy-text.github.io/2017/emerging-rare-entities.html
[18] https://pypi.org/project/beautifulsoup4/
[19] https://networkx.github.io/
[20] http://download.tensorflow.org/models/inception_resnet_v2_2016_08_30.tar.gz

The three LtR schemes commented in Section 5.2.1 shares the same network structure but with different loss function. The neural network has two layers, with 128 and 32 neurons, respectively. For non-linearity, a Rectifier Linear Unit (ReLU) activation function is used (Xu et al., 2015). To avoid overfitting, the ReLU layer is followed by a dropout layer with a value of 0.5 (Hinton, 2002).

5.2.3. Results and Discussion: Drugs Case Study

Dataset Construction

Darknet Usage Text Addresses 10K (DUTA-10K) is a publicly available dataset proposed by Al-Nabki et al. (2019c) that holds 10,367 onion domains downloaded from the Tor network and distributed in 25 categories. In this case study, we address the ranking of the onion domains that were classified as *Drugs* in DUTA-10K. This category contains drugs-topics activities, including manufacture, cultivation, and marketing of drugs, in addition to drug forums and discussion groups. Out of 465 drug domains in DUTA-10K, we selected only English language domains what makes a total of 290 domains. This ranking approach could be adapted to any category of DUTA-10K, but we selected the drugs-related domains due to its popularity in our dataset, DUTA-10K (Al-Nabki et al., 2019c). We also want to stress that our approach is not only limited to web domains. It could be extended to further fields, such as document ranking or influencers detection in social networks, when a previously ranked list for training is available.

To build a ranked dataset, which served as ground truth, 13 persons, including the author, Wesam Al-Nabki, ranked manually the 290 drugs-related domains . For keeping the consistency in the ranking criteria among the annotators, we created a unified questionnaire of 23 subjective binary questions (Table 5.12) that were answered by the annotators for each domain. The ground truth was built in a pointwise manner assigning an annotator a value to each domain, coming from answering to every question with a one or zero, that corresponds to *Yes* or *No*, respectively.

This process was repeated three times, assigning to each annotator a new batch of approximately 23 domains every time, different in each iteration. Thus, each onion domain was judged three times by three different annotators, and as a result, each domain was represented by three binary vectors of answers. Following the majority voting approach, we unified these answers' vectors of every hidden service into a single vector of 23 dimensions that correspond with the number of questions. Finally, for a given domain, the vector of answers was aggregated into a single real number, a gain value, by adding up its elements, corresponding the obtained sum with the ground truth rank of that domain. The higher the gain value, the higher the rank an onion domain obtains.

Table 5.12: Binary questionnaire used to build a ground truth rank for the drugs onion domains.

Questions	
Has a satisfactory FAQ?	Has a communication channel?
Has a professional design?	Has real images for the products?
Has a subjective title?	Sells between 2 to 10 products?
Provides safe shipping?	Domain name has a meaning?
Offers reward or discount?	Products majority are illegal?
Sell more than 10 products?	Still accessible in TOR network?
Shipping worldwide service?	Sells at least one popular product?
Reputation content?	Requires login/ registration?
Accepts only Cryptocurrency?	Recently updated?
Can customers add a review/ feedback?	Do you feel that this domain is trustable?
Need text spotting for the products' images	Are you satisfied with the products description?
Has more than 10 sub-pages?	

Learning to Ranking Schema Selection

In Section 5.2.1, we evaluated three well-known LtR schemes, namely, pointwise, pairwise, and listwise, and for each one, we explored a supervised ranking algorithm: MLP, RankNet, and ListNet, respectively. We wanted to know what is the most suitable LtR schema for the task of ranking the onion domains in the Tor network and detecting the influential ones. Figure 5.12 compares the three LtR algorithms using the $NDCG@k$ metric for 10 different values of $K = \{1,3,5,7,9,15,25,35,45,55\}$. The reader can notice that the values of k are not equally sampled, that we selected five values between zero and 10, while the other five values are greater than 10. This distribution is because having the head of a ranked list sorted correctly is more important than its tail (Cao et al., 2006; Wang et al., 2017c). The superiority of the listwise approach is evidence of its suitability among the other methods (see Figure 5.12). The same figure shows that the $NDCG@1$ of the ListNet is equal to one, which means that during the five folds of cross-validation, the algorithm ranked correctly the first domains tested, exactly as the ground truth. At $NDCG@4$, the curve starts dropping; however, the lowest $NDCG$ value was 0.88 for K equals to 25. Also, as can be seen in Figure 5.12, the pointwise approach, which is the MLP in our case, obtained the worst performance, which agrees with the conclusion of other researchers (Li, 2011a).

In addition to comparing the performance in terms of the $NDCG@K$, we registered the duration of the time required to train and to test each LtR model. More precisely, we compare them starting from the moment the model receives a list of domains encoded by HSMU (Section 5.2.1) until it produces their rank. On average, for the five-folds, the ListNet model took 8.30 seconds for training and 0.08 seconds for testing. The RankNet

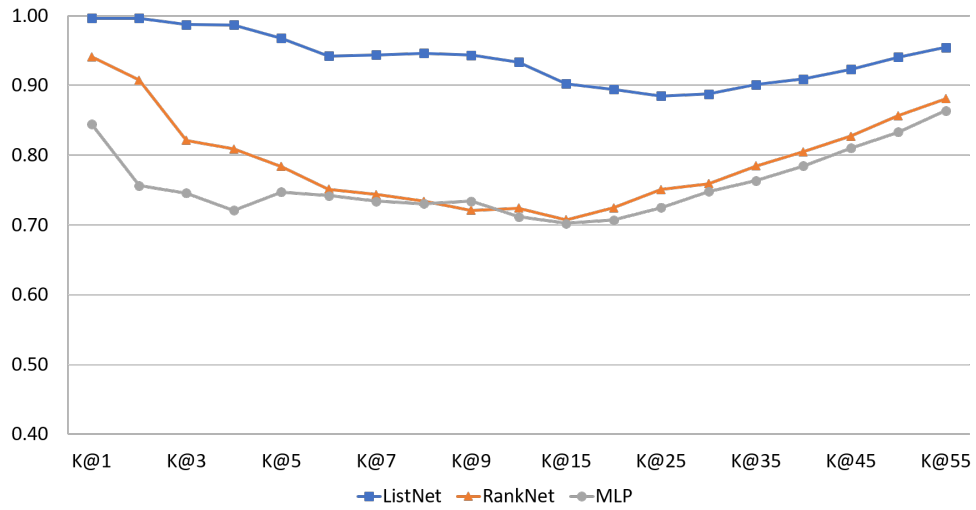


Figure 5.12: Comparison between three LtR algorithms against multiple values of $NDCG@K$. The X-axis refers to K value and the Y-axis one indicates the $NDCG$ scores of the algorithms, obtained at each value of K.

took 7.35 seconds for training and 0.007 seconds for testing. Finally, the fastest one was the MLP model, which took 3.34 seconds for training and 0.0009 seconds for testing. This comparison shows that the ListNet model is the slowest one due to the complexity of its loss function comparing to the ones of the RankNet and the MLP algorithms.

Link-based versus Content-based Ranking

Having two distinct ranking strategies raises a question: *which is the most suitable ranking approach? Content-based or link-based?*. To answer it, we explore four link-based algorithms, namely, ToRank (Al-Nabki et al., 2019c), PageRank (Page et al., 1999), Hyperlink-Induced Topic Search (HITS) (Kleinberg, 1999), and Katz (Katz, 1953). In particular, we compare the best LtR model of our approach, i.e., ListNet, which is considered as a supervised ranking algorithm, with the four link-based algorithms that are unsupervised. We represent the Tor network by a directed graph that consists of nodes and directed edges, where nodes represent the onion domains, and the hyperlinks between them are captured using the directed edges.

Comparison configuration. Unlike our approach, the link-based algorithms do not require training data. To obtain a fair comparison, we carried out 5-folds cross-validation with the same random seed for both ranking approaches. Then, we constructed a directed graph out of the testing nodes and applied the link-based algorithms; finally, both ap-

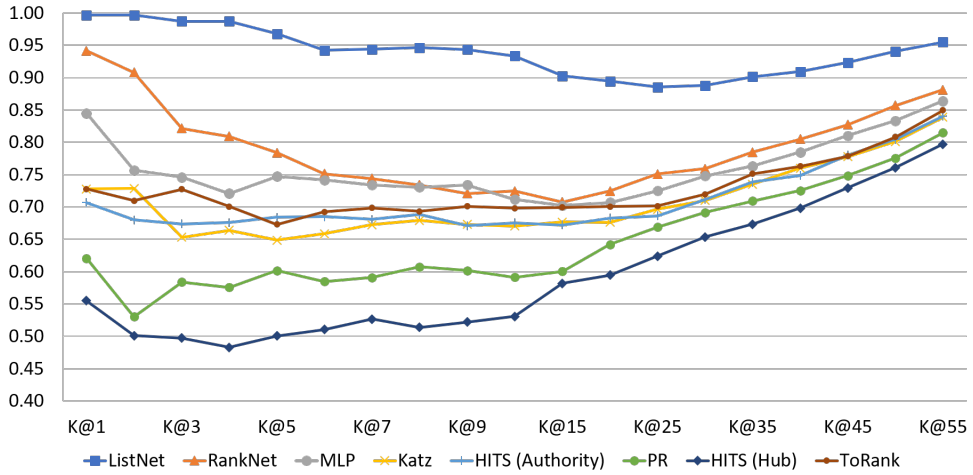


Figure 5.13: A comparison between the content-based versus link-based ranking algorithms with respect to multiple values of K. The horizontal axis refers to K value and the vertical one indicates the $NDCG$ scores the algorithms obtained at each value of K.

proaches were evaluated using the same test set. For the link-based algorithms, we evaluated several configuration parameters and selected the ones that obtained the highest $NDCG$ (Table 5.13).

Table 5.13: The examined parameters for the link-based ranking algorithms. Bold numbers correspond to the selected configuration that achieved the highest $NDCG$ value.

Algorithm Name	Parameter	Evaluated values
PageRank	alpha	0.5, 0.70, 0.75.0.80,**0.85**,0.90
	max_iter	10, 100, **1000**, 10000
ToRank	alpha	0.50, 0.70, 0.80, **0.90**, 1.00
	beta	0.1, **0.2**, 0.3, 0.4, 0.5, 0.6
HITS	max_iter	10, 100, **1000**, 10000
Katz	alpha	0.01, **0.1**, 0.2, 0.3, 0.4, 0.6, 0.9
	beta	0.1, 0.3, 0.5, 0.7, 0.9, **1.0**
	max_iter	10, 100, **1000**, 10000

Figure 5.13 shows that ListNet highly surpasses all the link-based ranking algorithms. We observe that the weakest LtR approach, i.e., MLP, which obtained a $NDCG@10$ of 0.71, outperforms the best link-based ranking algorithm, ToRank, which scored $NDCG@10$ of 0.69. This result emphasizes the importance of considering the content of domains rather than their hyperlinks connectivity only.

Feature Selection

In the previous sections, we concluded that ListNet outperforms the RankList and MLP content-based, and the four link-based algorithms when the proposed forty features represent an onion domain. However, the cost of these features varies. Some of them, such as the visual content, requires building a dedicated image classification model, while other features could be extracted merely using a regular expression. The cost is reflected in the time necessary to obtain the features and to build the ranking model, in addition to the inference time. On average, per domain, the prediction of the image classification model was the most expensive and took 109 seconds, followed by the NER model with 22 seconds, and then it was the text features that required 12 seconds. Finally, the HTML and the graph features were the fastest ones to be extracted, spending 3 and 2 seconds, respectively.

In the following, we want to answer the question: *what is the feature or combination of features that produce the best performance of the LtR model?* To answer this question, we conducted feature analysis for the best LtR model, the ListNet one, using features and combining them from the five different resources. To discover the best combination of features for the ranking system performance, we trained and tested five ListNet models for each source of features and compared their performance.

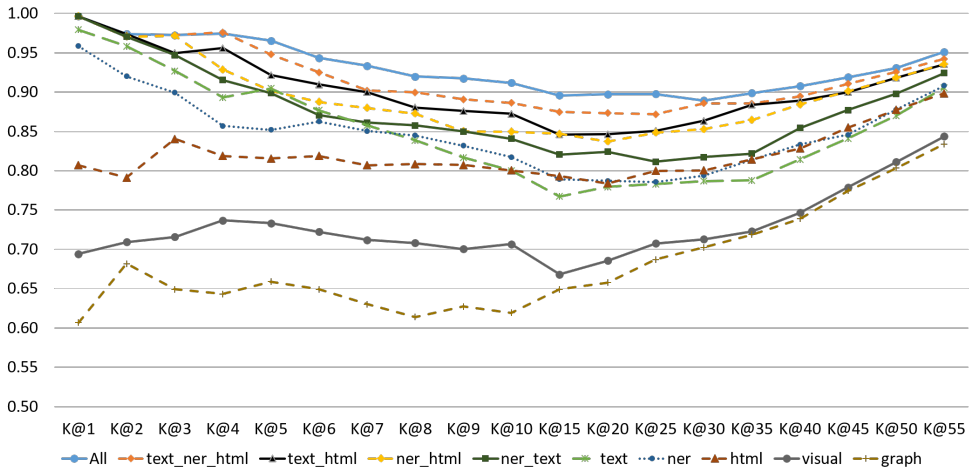


Figure 5.14: The effect of using different types of features along with their combinations on the ListNet ranking model. The vertical axis refers to the *NDCG* value, while the horizontal axis denotes the value of *K*. Each curve refers to a source of features: textual (*text*), featured produced by a Named Entity Recognition (*NER*), HTML markup features (*HTML*), visual features (*visual*), graph features (*graph*), and all the features fused, denoted as (*All*).

Furthermore, we analyzed only features coming from a single source, without combining them (see Figure 5.14). The features that were extracted only from text, denoted

by *text* achieved the highest $NDCG@5$ of 0.90. In the second position came the features extracted from the named entities, which obtained a $NDCG@5$ of 0.85. After that, using only features extracted from HTML, the ListNet model obtained a $NDCG@5$ of 0.81. In contrast, the graph features obtained the lowest $NDCG@5$ of 0.65, which indicates their weakness for ranking the onion domains, unlike the features that were extracted from the text, which have shown a significant and positive impact on the $NDCG$ metric. Hence, we conclude that the features extracted from the user-visible text are more representative comparing to the ones coming from the visual characteristics of the domain or the graph ones.

After the previous analysis, we decided to investigate the effect of combining features from different sources, to measure the impact of those combinations on the model performance. In Figure 5.14 it can be seen that the performance increased when the top-3 individual features, i.e. *text*, *NER*, and *HTML* were fused. Also, those three features combined could obtain a $NDCG$ close to the one yield by combining all the features (*All*). Consequently, we can remove the graph and the visual features and keep the ranking performance relatively high and very close to the case when all the features are used.

5.3. Conclusions

This chapter tackled two ranking techniques to detect the most influential onion domains in the Tor network: link-based and content-based. Regarding the link-based, we presented and made publicly available a dataset, DUTA-10K, with 10,367 Hidden Services (HS), manually labeled into 25 categories. Against the widespread belief that most of Tor's content is related to criminal activities, the statistical analysis on DUTA-10K showed that only around 20% of the tested onion domains are associated with suspicious activities, while 48% are related to normal ones. We also verified that the language of 84% of the crawled onion domains is English, which corroborates the idea that a text-based model to classify Tor content, trained only on this language, will cover the majority of the onion domains in Tor Darknet. Additionally, we found out that the domains related to suspicious activities tend to have multiple clones under different addresses, which can be even used as an additional feature for identifying them.

One of the main contributions of this chapter is ToRank, a new algorithm proposed to rank Tor HS, based on their links. In order to facilitate the process of monitoring the HS, ToRank was designed to identify and to rank the most influential onion domains. We employed graph theory to model the Tor network, where nodes correspond to the HS, and edges refer to the hyperlinks between them. ToRank was evaluated quantitatively by peeling away the top-ranked nodes iteratively and checking if the density of the graph decreases in every cycle. Its performance was compared against three popular link-based ranking algorithms, finding that the area under the Graph Density Curve - lower is better - was of 1.31 for ToRank, 1.41 Katz, 1.63 $\text{HITS}_{\text{Auth}}$, 1.96 HITS_{Hub}, and 2.07 PageRank.

Concerning the Content-based, we compared three supervised ranking strategies using Learning to Rank (LtR) to detect the most influential onion domains in the Tor Darknet using five sources of features. The proposed framework consists of two components: the Hidden Service Modeling Unit (HSMU) and the Supervised Learning to Rank Unit (SLRU). The HSMU represents an onion domain by 40 features extracted from five different resources: the domain user-visible text, the HTML markup of the web page, the named entities in the domain text, the visual content, and the Tor network structure. The SLRU learns how to rank the domains using an LtR approach. To train the LtR model, we built a manually sorted dataset of 290 drugs-related onion domains.

We tested the effectiveness of our framework on a manually ranked dataset of onion domains related to drug trading. We explored and evaluated three common LtR algorithms: MLP, RankNet, and ListNet, considering that the method which obtained the highest Normalized Discount Cumulative Gain ($NDCG$) is the best one. ListNet algorithm obtained the best results, with a $NDCG@10$ of 0.95. Moreover, we contrasted our framework with four link-based ranking algorithms, PageRank, HITS, Katz, and ToRank .We observed that the worst LtR algorithm, MLP, outperformed the best link-based one, ToRank, with a $NDCG@10$ of 0.71 and 0.69, respectively.

Finally, we analyzed how the different kinds of features impact the ranking perform-

ance in ListNet. We found that using only the features extracted from the user-visible textual content - text, named entities, and HTML - the model achieves a $NDCG@4$ of 0.97, the same result than the model with all the features. However, at $NDCG@10$, the performance drops slightly to 0.88 comparing to 0.95 using the five different sources of features. Considering both the cost to obtain the features and to create the models and its score, we recommend the text-NER-HTML model because its cost is low, and its score is almost the same as the more complex approach that uses all the features.

In the future, we plan to explore additional methods from the listwise approach such as RankBoost Freund et al. (2003), LambdaMart Burges (2010), and neural networks (Prakash and Sarkar, 2018).

Chapter 6

Conclusions and Outlook

6.1. Work Summary

The growing number of unlawful activities in the Darknet and Online Notepad Services (ONS) facilitates the accessibility of end-users to online products and services that could be related to criminal activities. The Onion Router (Tor) network has a considerable amount of domains with outlawed content, including, but not limited to, Child Sexual Abuse (CSA), drug smuggling, or weapons trading. These kinds of materials should be removed or blocked to achieve a safe environment for users. Hence, Law Enforcement Agencies (LEAs) strive to fight against these crimes by all available means, including Artificial Intelligence techniques and algorithms. As far as we know, most of the current monitoring strategies applied by LEAs are based on manual approaches or simple keyword filtering systems, which are neither efficient nor adequately effective. In this thesis, we propose new methods to monitor the Darknet of the Tor network using supervised machine learning, deep learning, and graph theory. Particularly, this thesis dissertation presents a comprehensive framework of three units: 1) a Text Classification Unit (TCU), to categorize suspicious activities in Tor Darknet and Pastebin; 2) a Text Mining Unit (TMU), to extract useful features and information from these domains; and 3) an Influence Detection Unit (IDU), to sort the onion domains and to detect the most influential ones, per activity, based on their content and their internal hyperlinks connectivity. The algorithms and techniques presented in this work would help LEAs in detecting suspicious content automatically.

The remainder of this chapter presents a summary of the contributions and possible areas that may expand to our research.

6.2. Summary of Contributions

In this dissertation, we present a framework of three units to detect suspicious activities and identify the most influential ones using different supervised machine learning techniques. The applications of our work are not limited to the Tor Darknet and Pastebin ONS. They can be further extended to other areas facing suspicious content, such as social media networks and the Surface Web. In this section, we sum up the obtained conclusions

to illustrate how this work contributes to monitoring suspicious activities in Tor Darknet and Pastebin ONS. A summary of the contributions are listed below:

- *We built a new labeled dataset in the domain of suspicious activities of the Tor Darknet.* In Chapter 3, we presented the first publicly available dataset, called *Darknet Usage Text Addresses* (DUTA), which was extracted from the Tor Darknet, and it holds 6,831 unique onion domains distributed on 26 activities monitored on Tor network during the sampling period.

- *We trained a supervised classifier with eight categories of suspicious activities in the Tor network.* In Chapter 3, we identified the key stages that influence the performance of a supervised text classifier. Also, we explored three supervised classifiers: Logistic Regression (LR), Support Vector Machine (SVM), and Naive Bayes (NB), and two text representation techniques: Bag of Words (Bow) and Term Frequency-Inverse Document Frequency (TF-IDF). Experimental results on DUTA showed that the combination of the TF-IDF vectorizer and the LR classifier outperforms the benchmarked models with an average F1 score of 93.7%.

- *We designed a supervised classifier using Active Learning (AL) to recognize suspicious activities in Pastebin ONS.* The presented classifier in Chapter 3 consists of three cascading classifiers: 1) a binary classifier to screen out code snippet instances; 2) a second binary classifier to distinguish the readable from the none-readable text; 3) and finally a multi-class and a binary classifier to detect suspicious activities in Pastebin. We found that a subset of 25,000 readable pastes can represent up to 86.40% of Pastebin sampled entries. On this subset, we adopted the two approaches of AL: exploitation-based and exploration-based, to select the most informative training samples. Using 180 pastes selected by the exploration-based AL approach, we obtained an average class recall of 92.84%. Furthermore, by initializing the exploitation-based AL with 20% of the labeled sampled, we obtained a binary classifier and another multi-class classifier with an average class recall of 95.24% and 80.33%, respectively. Our results strengthen the superiority of AL to random sampling in terms of saving effort and labor time for building a supervised classifier.

- *We introduced a semi-automatic algorithm to detect emerging products in the Tor Darknet using graph theory and the k-shell decomposition algorithm.* Chapter 4 described the use of graph theory to build a Products Correlations Graph (PCG), whereas the nodes correspond to the products, and the edges reflect a simultaneous offering of two products. Using the k-shell algorithm, we decomposed the PCG into shells, and our algorithm analyzed these shells to recognize the most popular products along with a list of the emerging ones. Furthermore, our algorithm derived association rules between the products that are hard to infer when the market

transaction logs are not available. Experimental results on the DUTA dataset indicated that the *Ecstasy*, which is a famous member of the MDMA drug family, was the most emerging drug during the crawling period of DUTA, and it was followed by *Ketamine* and the *Dimethyltryptamine* (DMT).

- *In the Named Entity Recognition field, we presented Local Distance Neighbor (LDN), a novel feature to substitute the usage of an external knowledge resource, such as a gazetteer.* In Chapter 4, we presented the LDN feature, and we proved how it could effectively replace gazetteers, which had been used widely in the literature. Also, we demonstrated that by incorporating the LDN feature, our proposal outperformed the state-of-the-art method in three classes of entities: Product, People, and Group in W-NUT-2017 dataset.

- *In the same research line, we incorporated the LDN feature to build a neural network to detect six categories of Named Entities in the onion domains of the Tor network.* In Chapter 4, we enhanced the W-NUT-2017 dataset with 851 manually annotated samples extracted from onion domains of the DUTA dataset with 2,970 unique tokens. We encapsulated the new samples into a new dataset called *Noisy User-generated Text on Tor* (NUToT), and we made it publicly available. Using NUToT, we trained a supervised NER model to recognize six Named Entities (NE) in onion domains. Experimentally, we found that our proposal using LDN with a fine-grained task outperformed the baseline model. We were capable of detecting the NE in the Tor network with an entity and surface F1 scores of 52.96% and 50.57%, respectively.

- *We presented DUTA-10K, an extension of the DUTA dataset, with* 10,367 *onion domains manually labeled into* 25 *categories. The dataset has been attached with a comprehensive analysis in terms of the activities distribution, the used languages, and the domain clones.* Against the widespread belief that most of Tor domains' contents are related to criminal activities, in Chapter 5, we showed that only 20% of the tested samples were correlated with suspicious activities, while the left 80% were either normal activities (48%) or inaccessible and were classified as unknown (32%). Furthermore, we verified that 84% of the hidden services of Tor are in English, which encourages using text-based models trained only on English to cover the majority of the onion pages. Additionally, we found that 51% of DUTA-10K samples vary between two and 496 copies per domain.

- *We presented ToRank, a new link-based ranking algorithm to sort Tor web pages, detecting the most influential ones.* The algorithm in Chapter 5 employs graph theory to model the Tor network, where nodes correspond to onion domains, and edges refer to the hyperlinks between them. ToRank obtained the lowest area under the Graph Density Curve (GDC) of 1.31, on the Suspicious Activities Graph (SAG_{All}), when compared against three popular link-based ranking algorithms:

Katz, $HITS_{Auth}$, $HITS_{Hub}$ and PageRank, with GDC of 1.41 , 1.63 , 1.96, and 2.07 respectively. In addition to the suspicious activities graph, ToRank outperformed the aforementioned algorithms on the normal activities graph and the 9/11 hijackers network.

- *We presented a content-based ranking framework, based on Learning to Rank (LtR), to detect first and rank later the most influential onion domains in the Tor Darknet.* The framework presented in Chapter 5 consists of two components: 1) the Hidden Services Modeling Unit (HSMU), to model an onion domain by 40 features that were extracted from five different resources: the domain user-visible text, the HTML markup of the web page, the named entities in the domain text, the visual content, and the Tor network structure; and the 2) Supervised Learning to Rank Unit (SLRU) to learn how to rank the domains using the LtR approach. As a case study, we selected 290 drug-related domains from the DUTA-10K dataset. To rank these domains, 13 people, including the author, collaborated in answering 23 subjective binary questions. The obtained rank served as ground truth for the ranking algorithm. We examined three common LtR algorithms: MLP, RankNet, and ListNet, whereas the best one is the one that obtains the highest $NDCG$. We found that the ListNet algorithm outperforms the rest of the ranking algorithms with a $NDCG@10$ of 0.95. Moreover, we compared our content-based ranking approach with four link-based ranking algorithms. We found that the MLP algorithm, which was the worst LtR algorithm with a $NDCG@10$ of 0.71, is better than the best link-based one, ToRank, which obtained a $NDCG@10$ of 0.69.

- *We analyzed the features which impact the best LtR algorithm, ListNet.* In Chapter 5, we found that using only the features extracted from the user-visible textual content, including the text, the named entities, and HTML markup code, the ranking model could achieve a $NDCG@4$ of 0.97 as if all the features were considered. However, at $NDCG@10$, the performance drops slightly to 0.88 comparing to 0.95 using the five features sources. Hence, we concluded that for ranking onion domains, it is enough to use only the user-visible textual content rather than including the visual and the graph features.

6.3. Open Problems and Future Work

Next, we present some research lines arisen during this work, which could be interesting to be addressed in the future.

- *Explore different protocols in Tor network.* In this thesis, we addressed the port 80 in the Tor network only to classify the eight categories of suspicious activities. Yet, there remain other significant ports, like 443, where several unlawful are taking

place (Biryukov et al., 2014). Furthermore, it is essential to tackle more networks like FreeNet[1], which focus primarily on Child Sexual Abuse (CSA) activity, and I2P[2].

- *Incorporate further features, apart from the user-visible, to leverage the text classifier.* The text classifier presented in this thesis judges an onion domain using its text but ignoring the information of the HTML markup code and the images. This extension is particularly important when an HS is presenting its services as visual content only, without text.

- *Extend the current text classifier to handle multi-label training samples.* A significant number of marketplaces in the Tor network are involved in more than one type of suspicious activity. The actual Text Classification Unit (TCU) excludes these samples from the training set, resulting in losing crucial porting of the training data.

- *Examine various text representation techniques.* In particular, regular text embedding techniques, like Word2Vec Mikolov et al. (2013), GloVe Pennington et al. (2014), and fastText Bojanowski et al. (2017), and contextualized character-level word embedding like Flair Akbik et al. (2018), as well as transformer-based models, such as BERT(Devlin et al., 2019), RoBERTa (Liu et al., 2019), and XLNet (Yang et al., 2019).

- *Improving the LDN feature by considering the context of the input token.* Currently, the LDN of an input token is estimated separately, without considering its context, i.e., the LDN vectors of the right and the left neighboring tokens using Bi-LSTM units.

- *Linking the recognized Named Entities to construct a knowledge graph for the Tor network.* The recognized NEs in onion domains are valuable assets to draw insights about the presented products, companies, people, and locations. However, linking these entities would enrich our knowledge about the Tor network (Derczynski et al., 2015; Hachey et al., 2011).

- *Extending ToRank algorithm to consider further features extracted from the visual content presented in the domains.* Currently, ToRank is a link-based ranking algorithm, and it surpasses other well-known algorithms, even in large graphs. We are planning to extract visual information by categorizing the images in Tor HS (Fidalgo et al., 2017) and generating textual descriptions for them, using image captioning (You et al., 2016; Tan and Chan, 2019). Our idea is to combine those features with ToRank to improve the ranking. Furthermore, we are also considering to introduce in our analysis the hyperlinks related to the surface Web, which might help to understand and to determine which the influential domains are.

[1] `https://freenetproject.org/`
[2] `https://geti2p.net`

- *Exploring more supervised ranking methods, such as RankBoost Freund et al. (2003), LambdaMart Burges (2010), and neural networks (Prakash and Sarkar, 2018).* Moreover, we are looking forward to exploring StarSpace Wu et al. (2018) that attempted to learn objects representations into a common embedding space that could be used to entities ranking and recommendation systems.

Capítulo 7

Conclusiones y perspectiva

7.1. Resumen del trabajo

El creciente número de actividades ilegales en la Red oscura (Darknet) y Servicios de bloc de notas en línea (ONS, del inglés Online Notepad Services) facilita la accesibilidad de los usuarios finales a productos y servicios en línea que podrían estar relacionados con actividades delictivas. La red Tor (del inglés, The Onion Router) tiene una cantidad considerable de dominios con contenido ilegal, que incluye, entre otros, Abuso Sexual Infantil (CSA, del inglés Child Sexual Abuse), contrabando de drogas o comercio de armas. Este tipo de materiales deben eliminarse o bloquearse para lograr un entorno seguro para los usuarios de Tor. Por lo tanto, las agencias de aplicación de la ley (LEAs, del inglés Law Enforcement Agencies) se esfuerzan por luchar contra estos delitos por todos los medios disponibles, incluidas las técnicas y algoritmos de inteligencia artificial. Hasta donde sabemos, la mayoría de las estrategias de monitoreo actuales aplicadas por las LEAs se basan en enfoques manuales o sistemas simples de filtrado de palabras clave, que no son ni eficientes ni suficientemente efectivos. En esta tesis, proponemos nuevos métodos para monitorear la red Tor Darknet utilizando aprendizaje automático supervisado, aprendizaje profundo y teoría de grafos. En particular, esta disertación presenta una propuesta integral formada por tres unidades: 1) Una solución para la Clasificación de Texto (TCU, del inglés Text Classification Unit), para clasificar las actividades sospechosas en la red Tor y Pastebin; 2) Una propuesta basada en minería de texto (TMU, del inglés Text Mining Unit), para extraer características e información útiles de estos dominios; y 3) La Detección de dominions influyentes (IDU, del inglés Influence Detection Unit), para clasificar los dominios de cebolla y detectar los más influyentes, por actividad, en función de su contenido y de la conectividad de sus hipervínculos internos. Los algoritmos y técnicas presentados en este trabajo son de utilidad para que las LEAs puedan detectar automáticamente contenido sospechoso.

El resto de este capítulo presenta un resumen de las contribuciones y posibles líneas futuras de investigación.

7.2. Conclusiones generales

En esta disertación, presentamos un conjunto de tres unidades que se pueden utilizar para detectar actividades sospechosas e identificar las más influyentes en la red Tor, utilizando diferentes técnicas de aprendizaje automático supervisado. Las aplicaciones de nuestro trabajo no se limitan a la red Tor y al servicio Pastebin. Pueden extenderse también a otras áreas donde también aparece contenido sospechoso, como son las redes sociales y la web superficial. En esta sección, resumimos las conclusiones obtenidas para ilustrar cómo este trabajo contribuye a monitorear actividades sospechosas en la red Tor y en Pastebin. Ls contribuciones realizadas se resumen a continuación.

- *Creamos un nuevo conjunto de datos etiquetados conteniendo dominios de la Red Oscura Tor relacionados con actividades posiblemente criminales.* En el capitulo 0.3, presentamos el primer conjunto de datos disponible públicamente, llamado «**D**arknet **U**sage **T**ext **A**ddresses» (DUTA), que se extrajo de Tor Darknet, y contiene 6,831 dominios de cebolla únicos distribuidos en 26 actividades monitoreadas en la red Tor durante el período de muestreo.

- *Entrenamos un clasificador supervisado con ocho categorías de actividades sospechosas en la red Tor.* En el Capítulo 0.3, identificamos las etapas clave que influyen en el rendimiento de un clasificador de texto supervisado. Además, exploramos tres clasificadores supervisados: Regresión logística (LR, de inglés Logistic Regression), Máquina de vectores de soporte (SVM, de inglés Support Vector Machine) y Naive Bayes (NB), y dos técnicas de representación de texto: Bolsa de palabras (BoW, de inglés Bag of Words) y Frecuencia de término - frecuencia inversa de documento (TF- IDF, de inglés Term Frequency-Inverse Document Frequency). Los resultados experimentales en el conjunto de datos DUTA mostraron que la combinación del vectorizador TF-IDF y el clasificador LR supera a los otros modelos obteniendo un valor F1 de 93,7 %.

- *Diseñamos un clasificador supervisado utilizando aprendizaje activo (AL, del inglés Active Learning) para reconocer actividades sospechosas en Pastebin ONS.* El clasificador presentado en el Capítulo 0.3 consta de tres clasificadores en cascada: 1) un clasificador binario para descartar pastes conteniendo fragmentos de código; 2) un segundo clasificador binario para distinguir el texto legible del texto no legible; y 3) un doble clasificador, tanto multiclase como binario, para detectar actividades sospechosas en Pastebin. Descubrimos que con 25,000 pastes legibles podemos representar hasta el 86,40 % de las entradas de la muestra de Pastebin con la que trabajamos. En este subconjunto, adoptamos los dos enfoques convencionales de AL: basado en la explotación y en la exploración, para seleccionar las muestras más informativas y entrenar de este modo un clasificador. Usando 180 pastes seleccionadas mediante el enfoque AL basado en la exploración, obtuvimos un recall

promedio de clase de 92,84 %. Además, al inicializar un AL basado en la explotación con el 20 % de elementos de la muestra, obtuvimos un clasificador binario y otro clasificador multiclase con un recall promedio de 95,24 % y 80,33 %, respectivamente. Nuestros resultados demuestran la superioridad de AL frente al muestreo aleatorio, en cuanto al ahorro tanto en esfuerzo como en tiempo para construir un clasificador supervisado.

- *Introdujimos un algoritmo semiautomático para detectar productos emergentes en Tor Darknet utilizando la teoría de gráficos y el algoritmo de descomposición de k-shell.* El capítulo 0.4 describe el uso de la teoría de grafos para construir un Gráfico de Correlaciones de Productos (PCG, del inglés Product Correlation Graph), mientras que los nodos corresponden a los productos y las aristas reflejan que dos productos se ofrecen simultáneamente en ese dominio. Usando el algoritmo k-shell, descompusimos el PCG en distintas regiones similares a conchas (shells), y nuestro algoritmo analizó estos shells para reconocer los productos más populares conjuntamente con una lista de los emergentes. Además, nuestro algoritmo deriva reglas de asociación entre los productos, que son difíciles de inferir cuando los registros de transacciones del mercado no están disponibles, como es el caso. Los resultados experimentales en el conjunto de datos DUTA indicaron que el *Ecstasy*, que es un miembro famoso de la familia de drogas MDMA, es la droga más emergente durante el período de rastreo de DUTA, seguido por *Ketamine* y el *Dimethyltryptamine* (DMT).

- *En el campo de Reconocimiento de entidades con nombre, presentamos una característica novedosa que denominamos Vecino de Distancia Local (LDN, del inglés Local Distance Neighbor), con la intención de sustituir el uso de un recurso de conocimiento externo, como es el caso de los gazetteers (diccionarios de términos).* En el Capítulo 0.4, presentamos la función LDN y demostramos que permite reemplazar a los diccionarios de términos (gazetteers), que se han utilizado ampliamente en la literatura. Además, demostramos que al incorporar la función LDN, nuestra propuesta supera el estado del arte en tres clases de entidades: Producto, Gente y Grupo en el conjunto de datos W-NUT-2017.

- *En la misma línea de investigación, incorporamos la función LDN para construir una solución de red neuronal para detectar seis categorías de entidades nombradas en los dominios de cebolla de la red Tor.* En el Capítulo 0.4, mejoramos el conjunto de datos W-NUT-2017 con 851 muestras anotadas manualmente, extraídas de dominios de cebolla del conjunto de datos DUTA con tokens únicos de 2,970. Encapsulamos las nuevas muestras en un nuevo conjunto de datos llamado «Texto ruidoso generado por el usuario en Tor» (NUToT, del inglés Noisy User-generated Text on Tor), y lo pusimos a disposición del público. Usando NUToT, entrenamos un modelo NER supervisado para reconocer seis Entidades Nombradas (NE, del in-

glés Named Entity) en dominios de cebolla. Experimentalmente, demostramos que nuestra propuesta usando LDN supera al modelo de referencia. Pudimos detectar NE en la red Tor con valores F1 de entidad y superficie de 52,96 % y 50,57 %, respectivamente.

- *Presentamos DUTA-10K, una extensión del conjunto de datos DUTA, con* 10,367 *dominios de cebolla etiquetados manualmente en* 25 *categorías diferentes.* Contra la creencia generalizada de que la mayoría de los contenidos de los dominios Tor están relacionados con actividades delictivas, en el Capítulo 0.5, mostramos que solo se puede afirmar que el 20 % de las muestras analizadas están correlacionadas con actividades sospechosas, siendo el restante 80 % o bien actividades normales (48 %) o bien no fue posible acceder a sus contenidos (32 %), por lo que se clasificaron estas últimas como desconocidas.

- *Presentamos ToRank, un nuevo algoritmo de clasificación basado en enlaces para ordenar las páginas web de Tor, detectando las más influyentes.* En el Capítulo 0.5 empleamos la teoría de grafos para modelar la red Tor, donde los nodos corresponden a dominios de cebolla, y las aristas representan los hipervínculos entre ellos. ToRank obtuvo el área más baja bajo la Curva de Densidad del Gráfico (GDC, del inglés Graph Density Curves), 1,31, en el Gráfico de actividades sospechosas (SAG_{All}), en comparación con tres algoritmos populares de clasificación basados en enlaces: Katz, $HITS_{Auth}$, $HITS_{Hub}$ y PageRank, con un GDC de 1,41, 1,63, 1,96 y 2,07 respectivamente.

- *Presentamos una propuesta adicional de clasificación basada en el contenido de los dominios, utilizando el paradigma de Aprender a Clasificar (LtR, del inglés Learning to Rank), para detectar primero y clasificar más tarde los dominios de cebolla más influyentes en la Darknet de Tor.* El algoritmo presentado en el Capítulo 0.5 consta de dos componentes: 1) la Unidad de modelado de servicios ocultos (HSMU, del inglés Hidden Services Modeling Unit), mediante la cual modelamos un dominio de cebolla mediante 40 características extraídas de cinco tipos de recursos diferentes: el texto visible para el usuario, el contenido HTML de la página web, las entidades nombradas en el texto de dicho dominio, el contenido visual y la estructura de la red Tor; y 2) Unidad supervisada para Aprender-a-rankear (SLRU, del inglés Supervised Learning to Rank Unit) para aprender a establecer el orden de prioridad de los dominios utilizando el enfoque LtR. Como caso de estudio, seleccionamos 290 dominios relacionados con drogas que se encuentran en el conjunto de datos DUTA-10K. Examinamos tres algoritmos comunes de LtR: Perceptrón multicapa (MLP, del inglés Multilayer Perceptron), RankNet y ListNet, considerando el mejor aquel que obtuviera el mayor $NDCG$. Descubrimos que el algoritmo ListNet supera al resto de los algoritmos de clasificación con un $NDCG@10$ de 0,95. Además, comparamos nuestro enfoque de rankeado basado en contenido con cuatro algoritmos de

rankeado basados en enlaces. Descubrimos que el algoritmo MLP, que era el peor algoritmo LtR con un $NDCG@10$ de 0,71, es mejor que el mejor basado en enlaces, ToRank, que obtuvo un $NDCG@10$ de 0,69.

7.3. Trabajos futuros

En esta sección, resumimos las principales líneas de trabajo que permanecen abiertas para cada una de las aplicaciones estudiadas.

- *Explorar diferentes protocolos en la red Tor*. En esta tesis, usamos el puerto 80 en la red Tor solo para clasificar las ocho categorías de actividades sospechosas. Sin embargo, quedan otros puertos importantes, como el 443, que suele utilizarse comunmente para realizar actividades ilegales (Biryukov et al., 2014). Además, se podría abordar el análisis de más redes, como FreeNet, que se enfoca principalmente en la actividad de Abuso Sexual Infantil (CSA, del inglés Child Sexual Abuse), e I2P [1].

- *Incorporar características adicionales, además de las visibles para el usuario, para aprovechar el clasificador de texto.* El clasificador de texto presentado en esta tesis juzga un dominio de cebolla (HS, del inglés Hidden Service) usando su texto pero ignorando la información del código de marcado HTML y las imágenes. Esta extensión es particularmente importante cuando un HS presenta sus servicios solo como contenido visual, sin texto.

- *Extender el clasificador de texto actual para manejar muestras de entrenamiento con etiquetas múltiples.* Un número significativo de mercados en la red Tor están involucrados en más de un tipo de actividad sospechosa. La unidad de clasificación de texto (TCU, del inglés Text Classification Unit) excluye estas muestras del conjunto de entrenamiento, lo que resulta en la pérdida de datos de entrenamiento muy valiosos.

- *Examinar técnicas adicionales de representación de texto.* En particular, técnicas regulares de inclusión de texto, como Word2Vec Mikolov et al. (2013), GloVe Pennington et al. (2014) y fastText Bojanowski et al. (2017), y la inserción de palabras contextualizadas a nivel de caracteres como Flair Akbik et al. (2018), así como modelos basados en transformadores, como BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) y XLNet (Yang et al., 2019).

- *Mejorar la función LDN considerando el contexto del token de entrada.* Actualmente, el LDN de un token de entrada se estima por separado, sin tener en cuenta su contexto, es decir, los vectores LDN de los tokens vecinos derecho e izquierdo utilizando unidades Bi-LSTM .

[1] https://geti2p.net

- *Detectar asociaciones entre las Entidades nombradas reconocidas para construir un gráfico de conocimiento para la red Tor*. Las NE reconocidas en dominios de cebolla son activos valiosos para obtener información sobre los productos, empresas, personas y ubicaciones presentados. Por ello, crear asociaciones entre estas entidades enriquecería nuestro conocimiento sobre la red Tor (Derczynski et al., 2015; Hachey et al., 2011).

- *Extender el algoritmo ToRank para considerar otras características extraídas del contenido visual presentado en los dominios*. Actualmente, ToRank es un algoritmo de clasificación basado en enlaces, y supera a otros algoritmos conocidos, incluso en gráficos grandes. Estamos planeando extraer información visual clasificando las imágenes (Fidalgo et al., 2017) en Tor HS y generando descripciones textuales para ellas, usando el subtitulado de imágenes (You et al., 2016). Nuestra idea es combinar esas características con ToRank para mejorar el ranking. Además, también estamos considerando introducir en nuestro análisis los hipervínculos relacionados con la Web superficial, lo que podría ayudar a comprender y determinar cuáles son los dominios influyentes.

- *Explorar otros métodos de clasificación supervisada, como RankBoost Freund et al. (2003), LambdaMart Burges (2010) y redes neuronales (Prakash and Sarkar, 2018)*. Además, estamos ansiosos por explorar StarSpace Wu et al. (2018) que intenta aprender representaciones de objetos en un espacio de incrustación común que podría usarse para clasificar las entidades y los sistemas de recomendación.

Bibliography

Abe, H. and Tsumoto, S. (2010). Trend detection from large text data. In *2010 IEEE International Conference on Systems, Man and Cybernetics*, pages 310–315. IEEE.

Agichtein, E., Brill, E., and Dumais, S. (2006). Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–26. ACM.

Aguilar, G., Maharjan, S., Monroy, A. P. L., and Solorio, T. (2017). A multi-task approach for named entity recognition in social media data. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 148–153.

Aizawa, A. (2003). An information-theoretic perspective of tf–idf measures. *Information Processing and Management*, 39(1):45–65.

Akbik, A., Bergmann, T., and Vollgraf, R. (2019). Pooled contextualized embeddings for named entity recognition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 724–728.

Akbik, A., Blythe, D., and Vollgraf, R. (2018). Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649.

Al-Nabki, M. W., Fidalgo, E., Alegre, E., and Chaves, D. (2019a). Content-based features to rank influential hidden services of the tor darknet. *arXiv preprint arXiv:1910.02332*.

Al-Nabki, M. W., Fidalgo, E., Alegre, E., and de Paz, I. (2017a). Classifying illegal activities on tor network based on web textual contents. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, volume 1, pages 35–43.

Al-Nabki, M. W., Fidalgo, E., Alegre, E., and Fernández-Robles, L. (2019b). Improving named entity recognition in noisy user-generated text with local distance neighbor features. *Neurocomputing*, XXX:XXX–XXX. Accepted but not published yet.

Al-Nabki, M. W., Fidalgo, E., Alegre, E., and Fernández-Robles, L. (2019c). Torank: Identifying the most influential suspicious domains in the tor network. *Expert Systems with Applications*, 123:212–226.

Al-Nabki, M. W., Fidalgo, E., Alegre, E., and González-Castro, V. (2017b). Detecting emerging products in tor network based on k-shell graph decomposition. In *III Jornadas Nacionales de Investigación en Ciberseguridad (JNIC)*, volume 1, pages 24–30.

Al-Nabki, M. W., Fidalgo, E., and Velasco Mata, J. (2019d). Darkner: A platform for named entity recognition in tor darknet. In *Jornadas Nacionales de Investigación en Ciberseguridad (JNIC2019)*, volume 1, pages 279–280.

Albert, R., Jeong, H., and Barabási, A.-L. (1999). Internet: Diameter of the world-wide web. *nature*, 401(6749):130.

ALI, C. (2019). Circl ail - analysis information leak framework - training materials. `https://www.circl.lu/services/ail-training-materials/`. Accessed : 2019-05-29.

Alvarez, V. M. (2019). The pattern matching swiss knife for malware researchers 2019. `https://virustotal.github.io/yara/`. Accessed: 2019-05-29.

Anger, I. and Kittl, C. (2011). Measuring influence on twitter. In *Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies*, page 31. ACM.

Angluin, D. (1988). Queries and concept learning. *Machine Learning*, 2(4):319–342.

Anwar, T. and Abulaish, M. (2015). Ranking radically influential web forum users. *IEEE Transactions on Information Forensics and Security*, 10(6):1289–1298.

Aramaki, E., Miura, Y., Tonoike, M., Ohkuma, T., Mashuichi, H., and Ohe, K. (2009). Text2table: Medical text summarization system based on named entity recognition and modality identification. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*, pages 185–192. Association for Computational Linguistics.

Arma (2019). one cell is enough to break tor's anonymity | tor blog 2019. `https://blog.torproject.org/one-cell-enough-break-tors-anonymity`. Accessed: 2019-06-17.

Arrigo, L. and Goosdeel, A. (2016). European drug report trends and developments. `http://www.emcdda.europa.eu/system/files/publications/2637/TDAT16001ENN.pdf`. Accessed 28 February 2017.

Backstrom, L. and Kleinberg, J. (2014). Romantic partnerships and the dispersion of social ties: a network analysis of relationship status on facebook. In *Proceedings of the 17th ACM conference on Computer supported cooperative work and social computing*, pages 831–841. ACM.

Balcan, M.-F., Broder, A., and Zhang, T. (2007). Margin based active learning. In *International Conference on Computational Learning Theory*, pages 35–50. Springer.

Barbosa, L., Freire, J., and Silva, A. (2007). Organizing hidden-web databases by clustering visible web documents. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 326–335. IEEE.

Barratt, M. J. and Aldridge, J. (2016). Everything you always wanted to know about drug cryptomarkets*(* but were afraid to ask). *International Journal of Drug Policy*, 35:1–6.

Beluch, W. H., Genewein, T., Nürnberger, A., and Köhler, J. M. (2018). The power of ensembles for active learning in image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9368–9377.

Bergman, M. K. (2001). White paper: the deep web: surfacing hidden value. *Journal of electronic publishing*, 7(1).

Berzinji, A., Kaati, L., and Rezine, A. (2012). Detecting key players in terrorist networks. In *Intelligence and Security Informatics Conference (EISIC), 2012 European*, pages 297–302. IEEE.

Bidoki, A. M. Z., Ghodsnia, P., Yazdani, N., and Oroumchian, F. (2010a). A3crank: An adaptive ranking method based on connectivity, content and click-through data. *Information Processing and Management*, 46(2):159 – 169.

Bidoki, A. M. Z., Ghodsnia, P., Yazdani, N., and Oroumchian, F. (2010b). A3crank: An adaptive ranking method based on connectivity, content and click-through data. *Information processing and management*, 46(2):159–169.

Bidoki, A. M. Z. and Yazdani, N. (2008). Distancerank: An intelligent ranking algorithm for web pages. *Information Processing and Management*, 44(2):877–892.

Biryukov, A., Pustogarov, I., Thill, F., and Weinmann, R.-P. (2014). Content and popularity analysis of tor hidden services. In *2014 IEEE 34th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 188–193. IEEE.

Biryukov, A., Pustogarov, I., and Weinmann, R.-P. (2013). Trawling for tor hidden services: Detection, measurement, deanonymization. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 80–94. IEEE.

Biswas, R., Fidalgo, E., and Alegre, E. (2017). Recognition of service domains on tor dark net using perceptual hashing and image classification techniques. *8th International Conference on Imaging for Crime Detection and Prevention, ICDP-2017*, 14:15.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Booker, L. B. (2012). The effects of observation errors on the attack vulnerability of complex networks. Technical report, MITRE CORP MCLEAN VA.

Borodin, A., Roberts, G. O., Rosenthal, J. S., and Tsaparas, P. (2005). Link analysis ranking: algorithms, theory, and experiments. *ACM Transactions on Internet Technology (TOIT)*, 5(1):231–297.

Brian, M. (2019). Pastebin: How a popular code-sharing site became a hacker hangout. https://thenextweb.com/socialmedia/2011/06/05/pastebin-how-a-popular-code-sharing-site-became-the-ultimate-hacker-hangout/. Accessed: 2019-08-27.

Brinker, K. (2003). Incorporating diversity in active learning with support vector machines. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 59–66.

Broséus, J., Rhumorbarbe, D., Mireault, C., Ouellette, V., Crispino, F., and Décary-Hétu, D. (2016). Studying illicit drug trafficking on darknet markets: structure and organisation from a canadian perspective. *Forensic science international*, 264:7–14.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. (2005). Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM.

Burges, C. J. (2010). From ranknet to lambdarank to lambdamart: An overview. *Microsoft Research Technical Report*, 11(23-581):81–99.

Buxton, J. and Bingham, T. (2015). The rise and challenge of dark net drug markets. *Policy brief*, 7:1–24.

Cao, Y., Xu, J., Liu, T.-Y., Li, H., Huang, Y., and Hon, H.-W. (2006). Adapting ranking svm to document retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186–193. ACM.

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., and Li, H. (2007). Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136. ACM.

Carmi, S., Havlin, S., Kirkpatrick, S., Shavitt, Y., and Shir, E. (2007). A model of internet topology using k-shell decomposition. *Proceedings of the National Academy of Sciences*, 104(27):11150–11154.

Chaabane, A., Manils, P., and Kaafar, M. A. (2010). Digging into anonymous traffic: A deep analysis of the tor anonymizing network. In *2010 Fourth International Conference on Network and System Security*, pages 167–174. IEEE.

Chang, V. (2017). A cybernetics social cloud. *Journal of Systems and Software*, 124:195–211.

Chaurasia, N. and Tiwari, A. (2013). Efficient algorithm for destabilization of terrorist networks. *IJ Information Technology and Computer Science*, 12:21–30.

Chen, H. (2011). *Dark web: Exploring and data mining the dark side of the web*. Springer Science and Business Media.

Chen, H., Chung, W., Qin, J., Reid, E., Sageman, M., and Weimann, G. (2008). Uncovering the dark web: A case study of jihad on the web. *Journal of the American Society for Information Science and Technology*, 59(8):1347–1359.

Chen, H., Mckeever, S., and Delany, S. J. (2017). Harnessing the power of text mining for the detection of abusive content in social media. In *Advances in Computational Intelligence Systems*, pages 187–205. Springer.

Chen, H., McKeever, S., and Delany, S. J. (2018). A comparison of classical versus deep learning techniques for abusive content detection on social media sites. In Staab, S., Koltsova, O., and Ignatov, D. I., editors, *Social Informatics*, pages 117–133, Cham. Springer International Publishing.

Chen, Y., Lasko, T. A., Mei, Q., Denny, J. C., and Xu, H. (2015). A study of active learning methods for named entity recognition in clinical text. *Journal of biomedical informatics*, 58:11–18.

Chen, Y., Perozzi, B., Al-Rfou, R., and Skiena, S. (2013). The expressive power of word embeddings. In *ICML 2013 Workshop on Deep Learning for Audio, Speech, and Language Processing*, page 9, Atlanta, GA, USA.

Cherman, E. A., Papanikolaou, Y., Tsoumakas, G., and Monard, M. C. (2019). Multi-label active learning: key issues and a novel query strategy. *Evolving Systems*, 10(1):63–78.

Choudhary, P. and Singh, U. (2016). Ranking terrorist nodes of 9/11 network using analytical hierarchy process with social network analysis. In *International Symposium on the Analytic Hierarchy Process (ISAHP 2016)*, pages 1–10.

Ciancaglini, V., Balduzzi, M., Goncharov, M., and McArdle, R. (2013). Deepweb and cybercrime. *Trend Micro Report*, 9.

Ciancaglini, V., Balduzzi, M., McArdle, R., and Rösler, M. (2016). Below the surface: Exploring the deep web. *Trend Micro Incorporated. As of*, 12.

Ciaramita, M., Murdock, V., and Plachouras, V. (2008). Online learning from click data for sponsored search. In *Proceedings of the 17th international conference on World Wide Web*, pages 227–236. ACM.

Cockbuen, A. and Mckenzie, B. (2001). What do web users do? an empirical analysis of web use. *International Journal of Human-Computer Studies*, 54(6):903–922.

Cohen, R. and Havlin, S. (2010). *Complex networks: structure, robustness and function*. Cambridge university press.

Cossu, J.-V., Dugué, N., and Labatut, V. (2015). Detecting real-world influence through twitter. In *Network Intelligence Conference (ENIC), 2015 Second European*, pages 83–90. IEEE.

Dalal, M. K. and Zaveri, M. A. (2011). Automatic text classification: a technical review. *International Journal of Computer Applications*, 28(2):37–40.

Dalins, J., Wilson, C., and Carman, M. (2018). Criminal motivation on the dark web: A categorisation model for law enforcement. *Digital Investigation*, 24:62 – 71.

Das, P. and Das, A. K. (2018). Crime pattern analysis by identifying named entities and relation among entities. In Bhattacharyya, S., Chaki, N., Konar, D., Chakraborty, U. K., and Singh, C. T., editors, *Advanced Computational and Communication Paradigms*, pages 75–84, Singapore. Springer Singapore.

Dasgupta, S. and Hsu, D. (2008). Hierarchical sampling for active learning. In *Proceedings of the 25th international conference on Machine learning*, pages 208–215. ACM.

Davidson, G. S., Hendrickson, B., Johnson, D. K., Meyers, C. E., and Wylie, B. N. (1998). Knowledge mining with vxinsight: Discovery through interaction. *Journal of Intelligent Information Systems*, 11(3):259–285.

Deepwebsiteslinks (2019). 25 best deep web drugs store. = https://www.deepwebsiteslinks.com/best-deep-web-drugs-store/. Accessed: 2018-05-29.

Derczynski, L., Maynard, D., Rizzo, G., van Erp, M., Gorrell, G., Troncy, R., Petrak, J., and Bontcheva, K. (2015). Analysis of named entity recognition and linking for tweets. *Information Processing and Management*, 51(2):32 – 49.

Derczynski, L., Nichols, E., van Erp, M., and Limsopatham, N. (2017). Results of the WNUT2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 140–147.

Derhami, V., Khodadadian, E., Ghasemzadeh, M., and Bidoki, A. M. Z. (2013). Applying reinforcement learning for web pages ranking algorithms. *Applied Soft Computing*, 13(4):1686–1692.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dey, A. and Prukayastha, B. S. (2013). Named entity recognition using gazetteer method and n-gram technique for an inflectional language: A hybrid approach. *International Journal of Computer Applications*, 84(9).

Ding, R., Xie, P., Zhang, X., Lu, W., Li, L., and Si, L. (2019). A neural multi-digraph model for chinese ner with gazetteers. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 1462–1467.

Dolliver, D. S. (2015). Evaluating drug trafficking on the tor network: Silk road 2, the sequel. *International Journal of Drug Policy*, 26(11):1113–1123.

Dong, F., Yuan, S., Ou, H., and Liu, L. (2018). New cyber threat discovery from darknet marketplaces. In *2018 IEEE Conference on Big Data and Analytics (ICBDA)*, pages 62–67. IEEE.

Donlevy, C. (2005). A human response to ambiguity: A psycholinguistic analysis.

Duan, Y., Jiang, L., Qin, T., Zhou, M., and Shum, H.-Y. (2010). An empirical study on learning to rank of tweets. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 295–303. Association for Computational Linguistics.

Duggan, B. (2016). Uganda elections: Government shuts down social media - cnn @misc. `https://edition.cnn.com/2016/02/18/world/uganda-election-social-media-shutdown/`.

Duijn, P. A., Kashirin, V., and Sloot, P. M. (2014). The relative ineffectiveness of criminal network disruption. *Scientific reports*, 4:4238.

Dumais, S. and Chen, H. (2000). Hierarchical classification of web content. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 256–263. ACM.

Elahi, T., Bauer, K., AlSabah, M., Dingledine, R., and Goldberg, I. (2012). Changing of the guards: A framework for understanding and improving entry guard selection in tor. In *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*, pages 43–54. ACM.

Eliacik, A. B. and Erdogan, N. (2018). Influential user weighted sentiment analysis on topic based microblogging community. *Expert Systems with Applications*, 92:403–418.

Fidalgo, E., Alegre, E., Fernández-Robles, L., and González-Castro, V. (2019). Classifying suspicious content in tor darknet through semantic attention keypoint filtering. *Digital Investigation*.

Fidalgo, E., Alegre, E., González-Castro, V., and Fernández-Robles, L. (2017). Illegal activity categorisation in darknet based on image classification using creic method. In *International Joint Conference SOCO'17-CISIS'17-ICEUTE'17 León, Spain, September 6–8, 2017, Proceeding*, pages 600–609. Springer.

Foley, S., Karlsen, J., and Putniņš, T. J. (2018). Sex, drugs, and bitcoin: How much illegal activity is financed through cryptocurrencies? *SSRN Electronic Journal*.

Freeman, L. C. (1978). Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239.

Freund, Y., Iyer, R., Schapire, R. E., and Singer, Y. (2003). An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Ann. Statist.*, 29(5):1189–1232.

Fronzetti Colladon, A. and Gloor, P. A. (2018). Measuring the impact of spammers on e-mail and twitter networks. *International Journal of Information Management*.

Fronzetti Colladon, A. and Remondi, E. (2017). Using social network analysis to prevent money laundering. *Expert Systems with Applications*, 67:49–58.

Fronzetti Colladon, A. and Vagaggini, F. (2017). Robustness and stability of enterprise intranet social networks: The impact of moderators. *Information Processing and Management*, 53(6):1287–1298.

Gallagher, S. and UTC (2016). Whole lotta onions: Number of tor hidden sites spikes-along with paranoia. `https://bit.ly/2MGTkrU`.

Gangwar, A., Fidalgo, E., Alegre, E., and González-Castro, V. (2017). Pornography and child sexual abuse detection in image and video: A comparative evaluation. In *8th International Conference on Imaging for Crime Detection and Prevention (ICDP 2017)*. Institution of Engineering and Technology.

Glance, N., Hurst, M., and Tomokiyo, T. (2004). Blogpulse: Automated trend discovery for weblogs. In *WWW 2004 workshop on the weblogging ecosystem: Aggregation, analysis and dynamics*, volume 2004. New York.

Globaldrugsurvey.com (2016). The global drug survey 2016 findings | global drug survey. `https://www.globaldrugsurvey.com/past-findings/the-global-drug-survey-2016-findings/`. Accessed 28 February 2017.

Godin, F., Vandersmissen, B., De Neve, W., and Van de Walle, R. (2015). Multimedia Lab @ ACL WNUT NER shared task: Named entity recognition for Twitter Microposts using distributed word representations. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 146–153.

Gohari, F. S. and Mohammadi, S. (2014). A comprehensive framework for identifying viral marketing's influencers in twitter. *International SAMANM Journal of Marketing and Management*, 2(1):27–43.

Goudjil, M., Koudil, M., Bedda, M., and Ghoggali, N. (2018). A novel active learning method using svm for text classification. *International Journal of Automation and Computing*, 15(3):290–298.

Graczyk, M. and Kinningham, K. (2015). Automatic product categorization for anonymous marketplaces.

Greenberg, N., Bansal, T., Verga, P., and McCallum, A. (2018). Marginal likelihood training of bilstm-crf for biomedical named entity recognition from disjoint label sets. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2824–2829.

Gupta, P., Jindal, R., and Sharma, A. (2018). Community trolling: an active learning approach for topic based community detection in big data. *Journal of Grid Computing*, 16(4):553–567.

Hachey, B., Radford, W., and Curran, J. R. (2011). Graph-based named entity linking with wikipedia. In *International conference on web information systems engineering*, pages 213–226. Springer.

Haldenwang, N., Ihler, K., Kniephoff, J., and Vornberger, O. (2018). A comparative study of uncertainty based active learning strategies for general purpose twitter sentiment analysis with deep neural networks. In Rehm, G. and Declerck, T., editors, *Language Technologies for the Challenges of the Digital Age*, pages 208–215, Cham. Springer International Publishing.

Han, J., Sun, A., Cong, G., Zhao, W. X., Ji, Z., and Phan, M. C. (2017). Linking fine-grained locations in user comments. *IEEE Transactions on Knowledge and Data Engineering*, 30(1):59–72.

Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3):146–162.

Hasan, O., Brunie, L., Bertino, E., and Shang, N. (2013). A decentralized privacy preserving reputation protocol for the malicious adversarial model. *IEEE Transactions on Information Forensics and Security*, 8(6):949–962.

Hasegawa, T., Sekine, S., and Grishman, R. (2004). Discovering relations among named entities from large corpora. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, pages 415–422, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hauck, T. (2014). *scikit-learn Cookbook*. Packt Publishing Ltd.

He, S., He, Y., and Li, M. (2019). Classification of illegal activities on the dark web. In *Proceedings of the 2019 2Nd International Conference on Information Science and Systems*, ICISS 2019, pages 73–78, New York, NY, USA. ACM.

Henni, K., Mezghani, N., and Gouin-Vallerand, C. (2018). Unsupervised graph-based feature selection via subspace and pagerank centrality. *Expert Systems with Applications*, 114:46–53.

Herath, H. (2017). Web information extraction system to sense information leakage. Master's thesis, University of Moratuwa, Sri Lanka.

Hidalgo, J. M. G., García, F. C., and Sanz, E. P. (2005). Named entity recognition for web content filtering. In *International Conference on Application of Natural Language to Information Systems*, pages 286–297. Springer.

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800.

Holme, P., Kim, B. J., Yoon, C. N., and Han, S. K. (2002). Attack vulnerability of complex networks. *Physical review E*, 65(5):056109.

Holmgren, Å. J. (2007). A framework for vulnerability assessment of electric power systems. In *Critical Infrastructure*, pages 31–55. Springer.

Hosmer Jr, D. W. and Lemeshow, S. (2004). *Applied logistic regression*. John Wiley and Sons.

Hu, R., Delany, S. J., and Mac Namee, B. (2010a). Egal: Exploration guided active learning for tcbr. In *International Conference on Case-Based Reasoning*, pages 156–170. Springer.

Hu, R., Mac Namee, B., and Delany, S. J. (2010b). Off to a good start: Using clustering to select the initial training set in active learning. In *Twenty-Third International FLAIRS Conference*.

Hu, R., Mac Namee, B., and Delany, S. J. (2016a). Active learning for text classification with reusability. *Expert Systems with Applications*, 45:438–449.

Hu, Y., Wang, S., Ren, Y., and Choo, K.-K. R. (2018). User influence analysis for github developer social networks. *Expert Systems with Applications*, 108:108–118.

Hu, Y., Zhang, J., Bai, X., Yu, S., and Yang, Z. (2016b). Influence analysis of github repositories. *SpringerPlus*, 5(1):1268.

Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional lstm-crf models for sequence tagging. *CoRR*, abs/1508.01991.

Husslage, B., Borm, P., Burg, T., Hamers, H., and Lindelauf, R. (2015). Ranking terrorists in networks: A sensitivity analysis of al qaeda's 9/11 attack. *Social Networks*, 42:1–7.

Intelliagg (2015). Deep light shining a light on the dark web. *Magazine*.

Iyer, S., Killingback, T., Sundaram, B., and Wang, Z. (2013). Attack robustness and centrality of complex networks. *PloS one*, 8(4):e59613.

Jansen, R., Tschorsch, F., Johnson, A., and Scheuermann, B. (2014). The sniper attack: Anonymously deanonymizing and disabling the tor network. Technical report, OFFICE OF NAVAL RESEARCH ARLINGTON VA.

Järvelin, K. and Kekäläinen, J. (2000). Ir evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 41–48. ACM.

Ji, S., Li, W., Gong, N. Z., Mittal, P., and Beyah, R. (2016). Seed-based de-anonymizability quantification of social networks. *IEEE Transactions on Information Forensics and Security*, 11(7):1398–1411.

Jiang, H., Nie, L., Sun, Z., Ren, Z., Kong, W., Zhang, T., and Luo, X. (2016). Rosf: Leveraging information retrieval and supervised learning for recommending code snippets. *IEEE Transactions on Services Computing*, pages 34–46.

John, B. (2019). Pastebin, the text sharing website, updates with an emphasis on code – techcrunch. `https://techcrunch.com/2015/12/16/pastebin-the-text-sharing-website-updates-with-an-emphasis-on-code/`.

Joshi, A., Fidalgo, E., Alegre, E., and Al-Nabki, M. W. (2018). Extractive text summarization in dark web: A preliminary study. *International Conference of Applications of Intelligent Systems*, 0.

Joshi, A. J., Porikli, F., and Papanikolopoulos, N. (2009). Multi-class active learning for image classification. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2372–2379. IEEE.

Kadari, R., Zhang, Y., Zhang, W., and Liu, T. (2018). Ccg supertagging via bidirectional lstm-crf neural architecture. *Neurocomputing*, 283:31 – 37.

Kan, M.-Y. (2004). Web page classification without the web page. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers and posters*, pages 262–263. ACM.

Kan, M.-Y. and Thi, H. O. N. (2005). Fast webpage classification using url features. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 325–326. ACM.

Kashiwazaki, H. (2018). Personal information leak in a university, and its cleanup. In *Proceedings of the 2018 ACM on SIGUCCS Annual Conference*, SIGUCCS '18, pages 43–50, New York, NY, USA. ACM.

Katz, L. (1953). A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43.

Kaur, P. (2014). Web content classification: A survey. *International Journal of Computer Trends and Technology (IJCTT)*, 10(2):97–01.

Kelleher, J. D., Mac Namee, B., and D'arcy, A. (2015). *Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies*. MIT Press.

Kendall, M. G. (2008). *Kendall Rank Correlation Coefficient*, pages 278–281. Springer New York, New York, NY.

Kenter, T., Borisov, A., and de Rijke, M. (2016). Siamese cbow: Optimizing word embeddings for sentence representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 941–951. Association for Computational Linguistics.

Khodabakhsh, M., Kahani, M., Bagheri, E., and Noorian, Z. (2018). Detecting life events from twitter based on temporal semantic features. *Knowledge-Based Systems*, 148:1–16.

Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632.

Kontostathis, A., Galitsky, L. M., Pottenger, W. M., Roy, S., and Phelps, D. J. (2004). A survey of emerging trend detection in textual data mining. In *Survey of text mining*, pages 185–224. Springer.

Krebs, V. E. (2002). Mapping networks of terrorist cells. *Connections*, 24(3):43–52.

Kwon, A., AlSabah, M., Lazar, D., Dacier, M., and Devadas, S. (2015). Circuit fingerprinting attacks: Passive deanonymization of tor hidden services. In *24th USENIX Security Symposium (USENIX Security 15)*.

Kwon, S., Ko, Y., and Seo, J. (2019). Effective vector representation for the korean named-entity recognition. *Pattern Recognition Letters*, 117:52 – 57.

Lai, H., Pan, Y., Liu, C., Lin, L., and Wu, J. (2013). Sparse learning-to-rank via an efficient primal-dual algorithm. *IEEE Transactions on Computers*, 62(6):1221–1233.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270. Association for Computational Linguistics.

Latapy, M., Magnien, C., and Fournier, R. (2013). Quantifying paedophile activity in a large p2p system. *Information Processing and Management*, 49(1):248–263.

Le, N. T., Mallek, F., and Sadat, F. (2016). UQAM-NTL: Named entity recognition in Twitter messages. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 197–202.

Leskovec, J., Lang, K. J., Dasgupta, A., and Mahoney, M. W. (2009). Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123.

Levene, M. (2011). *An introduction to search engines and web navigation*. John Wiley and Sons.

Li, H. (2011a). Learning to rank for information retrieval and natural language processing. *Synthesis Lectures on Human Language Technologies*, 4(1):1–113.

Li, H. (2011b). A short introduction to learning to rank. *IEICE TRANSACTIONS on Information and Systems*, 94(10):1854–1862.

Li, J., Xing, Z., and Kabir, A. (2018). Leveraging official content and social context to recommend software documentation. *IEEE Transactions on Services Computing*, 1(1):1–14.

Li, M., Luo, L., Miao, L., Xue, Y., Zhao, Z., and Wang, Z. (2016a). Friendrank: A personalized approach for tweets ranking in social networks. In *Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*, pages 896–900. IEEE.

Li, R., Lei, K. H., Khadiwala, R., and Chang, K. C.-C. (2012). Tedas: A twitter-based event detection and analysis system. In *Data engineering (icde), 2012 ieee 28th international conference on*, pages 1273–1276. IEEE.

Li, Y., Tripathi, A., and Srinivasan, A. (2016b). Challenges in short text classification: The case of online auction disclosure. In *MCIS*, page 18.

Limsopatham, N. and Collier, N. (2016). Bidirectional lstm for named entity recognition in twitter messages. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 145–152. The COLING 2016 Organizing Committee.

Lin, B. Y., Xu, F., Luo, Z., and Zhu, K. (2017). Multi-channel BiLSTM-CRF model for emerging named entity recognition in social media. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 160–165.

Ling, Z., Luo, J., Wu, K., Yu, W., and Fu, X. (2015). Torward: Discovery, blocking, and traceback of malicious traffic over tor. *IEEE Transactions on Information Forensics and Security*, 10(12):2515–2530.

Liu, T.-Y. et al. (2009). Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Loy, C. C., Hospedales, T. M., Xiang, T., and Gong, S. (2012). Stream-based joint exploration-exploitation active learning. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1560–1567. IEEE.

Lu, J., Behbood, V., Hao, P., Zuo, H., Xue, S., and Zhang, G. (2015). Transfer learning using computational intelligence: a survey. *Knowledge-Based Systems*, 80:14–23.

Lughofer, E. (2012). Hybrid active learning for reducing the annotation effort of operators in classification systems. *Pattern Recognition*, 45(2):884–896.

Luo, J., Zhou, W., and Du, Y. (2018). An active learning based on uncertainty and density method for positive and unlabeled data. In Vaidya, J. and Li, J., editors, *Algorithms and Architectures for Parallel Processing*, pages 229–241, Cham. Springer International Publishing.

Ma, X. and Hovy, E. (2016). End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074. Association for Computational Linguistics.

MacAvaney, S., Yates, A., Hui, K., and Frieder, O. (2019). Content-based weak supervision for ad-hoc re-ranking. In *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, pages 993–996, New York, NY, USA. ACM.

Manning, C., Raghavan, P., and Schütze, H. (2010). Introduction to information retrieval. *Natural Language Engineering*, 16(1):100–103.

Mao, G. and Zhang, N. (2017). Fast approximation of average shortest path length of directed ba networks. *Physica A: Statistical Mechanics and its Applications*, 466:243–248.

Matic, S., Fattori, A., Bruschi, D., and Cavallaro, L. (2012). Peering into the muddy waters of pastebin. *ERCIM News: Special Theme Cybercrime and Privacy Issues*, pages 16–18.

Matic, S., Kotzias, P., and Caballero, J. (2015). Caronte: Detecting location leaks for deanonymizing tor hidden services. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1455–1466. ACM.

Matilla, D., González Castro, V., Fernández Robles, L., Fidalgo, E., and Al-Nabki, M. W. (2018). Color sift descriptors to categorize illegal activities in images of onion domains. *Actas de las XXXIX Jornadas de Automática, Badajoz, 5-7 de Septiembre de 2018*.

McCallum, A., Nigam, K., et al. (1998). A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer.

Memon, N. and Larsen, H. L. (2006). Structural analysis and destabilizing terrorist networks. In *DMIN*, pages 296–302. Citeseer.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Minelli, M., Chambers, M., and Dhiraj, A. (2012). *Big data, big analytics: emerging business intelligence and analytic trends for today's businesses*. John Wiley and Sons.

Miorandi, D. and De Pellegrini, F. (2010). K-shell decomposition for dynamic complex networks. In *8th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, pages 488–496. IEEE.

Mirante, D. and Cappos, J. (2013). Understanding password database compromises. Technical report, Dept. of Computer Science and Engineering Polytechnic Inst. of NYU, Tech. Rep. TR-CSE-2013-02.

Mishra, S. and Diesner, J. (2016). Semi-supervised named entity recognition in noisy-text. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 203–212.

Mitchell, J. (2017). Want to cry? *ITNOW*, 59(3):12–13.

Mohamad, M. and Selamat, A. (2015). An evaluation on the efficiency of hybrid feature selection in spam email classification. In *2015 International Conference on Computer, Communications, and Control Technology (I4CT)*, pages 227–231. IEEE.

Moore, D. and Rid, T. (2016). Cryptopolitik and the darknet. *Survival*, 58(1):7–38.

Morris, C. and Hirst, G. (2012). Identifying sexual predators by svm classification with lexical and behavioral features. In *CLEF (Online Working Notes/Labs/Workshop)*, volume 12, page 29.

Nafziger, B. (2017). Data mining in the dark: Darknet intelligence automation. *SANA Institute*, 1:1–43.

Nakamoto, S. et al. (2008). Bitcoin: A peer-to-peer electronic cash system. *NA*.

News, B. (2017). The growing popularity, and potency, of ecstasy and mdma - bbc news. `http://www.bbc.com/news/uk-37156380`. Accessed 28 February 2017.

Ng, A. Y. and Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848.

Noor, U., Rashid, Z., and Rauf, A. (2011). A survey of automatic deep web classification techniques. *International Journal of Computer Applications*, 19(6):43–50.

Norbutas, L. (2018). Offline constraints in online drug marketplaces: An exploratory analysis of a cryptomarket trade network. *International Journal of Drug Policy*, 56:92–100.

Nouh, M. and Nurse, J. R. (2015). Identifying key-players in online activist groups on the facebook social network. In *Data Mining Workshop (ICDMW), 2015 IEEE International Conference on*, pages 969–978. IEEE.

Nunes, E., Diab, A., Gunn, A., Marin, E., Mishra, V., Paliath, V., Robertson, J., Shakarian, J., Thart, A., and Shakarian, P. (2016). Darknet and deepnet mining for proactive cybersecurity threat intelligence. In *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, pages 7–12.

Ohana, B., Delany, S. J., and Tierney, B. (2012). A case-based approach to cross domain sentiment classification. In *International Conference on Case-Based Reasoning*, pages 284–296. Springer.

Onan, A. (2016). Classifier and feature set ensembles for web page classification. *Journal of Information Science*, 42(2):150–165.

Onan, A., Korukoğlu, S., and Bulut, H. (2016). Ensemble of keyword extraction methods and classifiers in text classification. *Expert Systems with Applications*, 57:232–247.

O'Neill, J., Delany, S. J., and Namee, B. M. (2016). Activist: A new framework for dataset labelling. In *Proceedings of the 24th Irish Conference on Artificial Intelligence and Cognitive Science, AICS 2016, Dublin, Ireland, September 20-21, 2016.*, pages 140–148.

Owoputi, O., O'Connor, B., Dyer, C., Gimpel, K., Schneider, N., and Smith, N. A. (2013). Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–390. Association for Computational Linguistics.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Pang, B., Lee, L., et al. (2008). Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135.

Pannu, M., Kay, I., and Harris, D. (2019). Using dark web crawler to uncover suspicious and malicious websites. In Ahram, T. Z. and Nicholson, D., editors, *Advances in Human Factors in Cybersecurity*, pages 108–115, Cham. Springer International Publishing.

Park, J., Mun, H., and Lee, Y. (2018). Improving tor hidden service crawler performance. In *2018 IEEE Conference on Dependable and Secure Computing (DSC)*, pages 1–8. IEEE.

Pastebin.com (2018). pastebin.com - frequently asked questions 2018. `https://pastebin.com/faq#1`. Accessed: 2019-05-29.

Peersman, C., Schulze, C., Rashid, A., Brennan, M., and Fischer, C. (2014). icop: Automatically identifying new child abuse media in p2p networks. In *Security and Privacy Workshops (SPW), 2014 IEEE*, pages 124–131. IEEE.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Phan, M. C. and Sun, A. (2019). Collective named entity recognition in user comments via parameterized label propagation. *Journal of the Association for Information Science and Technology*.

Porter, A. L. and Detampel, M. J. (1995). Technology opportunities analysis. *Technological Forecasting and Social Change*, 49(3):237–255.

Prakash, C. and Sarkar, A. (2018). Ranking with deep neural networks. In *2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT)*, pages 1–4. IEEE.

Raeder, T. and Chawla, N. V. (2011). Market basket analysis with networks. *Social network analysis and mining*, 1(2):97–113.

Reimers, N. and Gurevych, I. (2017). Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348. Association for Computational Linguistics.

Reyes, O., Morell, C., and Ventura, S. (2018). Effective active learning strategy for multi-label learning. *Neurocomputing*, 273:494–508.

Riesco, A., Fidalgo, E., Al-Nabki, M. W., Jáñez-Martino, F., and Alegre, E. (2019). Classifying pastebin content through the generation of pastecc labeled dataset. In Pérez García, H., Sánchez González, L., Castejón Limas, M., Quintián Pardo, H., and Corchado Rodríguez, E., editors, *Hybrid Artificial Intelligent Systems*, pages 456–467, Cham. Springer International Publishing.

Ritter, A., Clark, S., Etzioni, O., et al. (2011a). Named entity recognition in tweets: an experimental study. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1524–1534. Association for Computational Linguistics.

Ritter, A., Clark, S., Mausam, and Etzioni, O. (2011b). Named entity recognition in tweets: An experimental study. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Rivest, R. (1992). The md5 message-digest algorithm. *Internet Engineering Task Force*.

Robertson, S. and Zaragoza, H. (2009). The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Rodrigues, M., Gama, J., and Ferreira, C. A. (2012). Identifying relationships in transactional data. In *Ibero-American Conference on Artificial Intelligence*, pages 81–90. Springer.

Ron, D. and Shamir, A. (2014). How did dread pirate roberts acquire and protect his bitcoin wealth? In *International Conference on Financial Cryptography and Data Security*, pages 3–15. Springer.

Rubens, N., Elahi, M., Sugiyama, M., and Kaplan, D. (2015). Active learning in recommender systems. In *Recommender systems handbook*, pages 809–846. Springer.

Rudesill, D. S., Caverlee, J., and Sui, D. (2015). The deep web and the darknet: A look inside the internet's massive black box. *Woodrow Wilson International Center for Scholars, STIP*, 3.

Ruhnau, B. (2000). Eigenvector-centrality—a node-centrality? *Social networks*, 22(4):357–365.

Sabbah, T., Selamat, A., Selamat, M. H., Ibrahim, R., and Fujita, H. (2016). Hybridized term-weighting method for dark web classification. *Neurocomputing*, 173:1908 – 1926.

Sachan, D., Zaheer, M., and Salakhutdinov, R. (2018). Investigating the working of text classifiers. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2120–2131, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Salton, G. (1971). *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Sanchez-Rola, I., Balzarotti, D., and Santos, I. (2017). The onions have eyes: A comprehensive structure and privacy analysis of tor hidden services. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1251–1260. International World Wide Web Conferences Steering Committee.

Schubert, E., Weiler, M., and Kriegel, H.-P. (2014). Signitrend: scalable detection of emerging topics in textual streams by hashed significance thresholds. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 871–880. ACM.

Scott, J. (1988). Social network analysis. *Sociology*, 22(1):109–127.

Seidman, S. B. (1983). Network structure and minimum degree. *Social networks*, 5(3):269–287.

Settles, B. (2009). Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Settles, B. and Craven, M. (2008). An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1070–1079. Association for Computational Linguistics.

Shaori Al-Ash, H. and Wibowo, W. C. (2018). Fake news identification characteristics using named entity recognition and phrase detection. In *2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE)*, pages 12–17.

Silvestre Castillo, R. (2019). A software to extract criminal networks from unstructured text in spanish; the case of peruvian criminal networks. In Aiello, L. M., Cherifi, C., Cherifi, H., Lambiotte, R., Lió, P., and Rocha, L. M., editors, *Complex Networks and Their Applications VII*, pages 3–15, Cham. Springer International Publishing.

Somda, Y. (2019). How does guesslang guess? — guesslang 0.9.4 documentation. `https://guesslang.readthedocs.io/en/latest/how.html`. Accessed: 2019-09-03.

Squire, M. and Smith, A. K. (2015). The diffusion of pastebin tools to enhance communication in floss mailing lists. In *IFIP International Conference on Open Source Systems*, pages 45–57. Springer.

Srinivas, A. and Velusamy, R. L. (2015). Identification of influential nodes from social networks based on enhanced degree centrality measure. In *Advance Computing Conference (IACC), 2015 IEEE International*, pages 1179–1184. IEEE.

Štravs, M. and Zupančič, J. (2019). Named entity recognition using gazetteer of hierarchical entities. In *Advances and Trends in Artificial Intelligence. From Theory to Practice*, pages 768–776. Springer International Publishing.

Suchacka, G. and Chodak, G. (2017). Using association rules to assess purchase probability in online stores. *Information Systems and e-Business Management*, 15(3):751–780.

Sun, A., Lim, E.-P., and Ng, W.-K. (2002). Web classification using support vector machine. In *Proceedings of the 4th international workshop on Web information and data management*, pages 96–99. ACM.

Suykens, J. A. and Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300.

Swan, R. and Allan, J. (2000). Automatic generation of overview timelines. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 49–56. ACM.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 4278–4284.

Taha, K. and Yoo, P. D. (2016). Siimco: A forensic investigation tool for identifying the influential members of a criminal organization. *IEEE Transactions on Information Forensics and Security*, 11(4):811–822.

Taha, K. and Yoo, P. D. (2017). Using the spanning tree of a criminal network for identifying its leaders. *IEEE Transactions on Information Forensics and Security*, 12(2):445–453.

Takaaki, S. and Atsuo, I. (2019). Dark web content analysis and visualization. In *Proceedings of the ACM International Workshop on Security and Privacy Analytics*, IWSPA '19, pages 53–59, New York, NY, USA. ACM.

Tan, Y. H. and Chan, C. S. (2019). Phrase-based image caption generator with hierarchical lstm network. *Neurocomputing*, 333:86–100.

Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Tseng, H., Chang, P., Andrew, G., Jurafsky, D., and Manning, C. (2005). A conditional random field word segmenter for sighan bakeoff 2005. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 168–171.

Tucker, C. and Kim, H. (2011). Predicting emerging product design trend by mining publicly available customer review data. In *DS 68-6: Proceedings of the 18th International Conference on Engineering Design (ICED 11), Impacting Society through Engineering Design, Vol. 6: Design Information and Knowledge, Lyngby/Copenhagen, Denmark, 15.-19.08. 2011.*

Usbeck, R., Ngonga Ngomo, A.-C., Röder, M., Gerber, D., Coelho, S. A., Auer, S., and Both, A. (2014). Agdistis - graph-based disambiguation of named entities using linked data. In Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., and Goble, C., editors, *The Semantic Web – ISWC 2014*, pages 457–471, Cham. Springer International Publishing.

Voorhees, E. M. (1985). The effectiveness and efficiency of agglomerative hierarchic clustering in document retrieval. Technical report, Cornell University.

Wang, D., Irani, D., and Pu, C. (2011). A social-spam detection framework. In *Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference*, pages 46–54. ACM.

Wang, K., Zhang, D., Li, Y., Zhang, R., and Lin, L. (2016). Cost-effective active learning for deep image classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(12):2591–2600.

Wang, M., Min, F., Zhang, Z.-H., and Wu, Y.-X. (2017a). Active learning through density clustering. *Expert Systems with Applications*, 85:305 – 317.

Wang, R., Wang, X.-Z., Kwong, S., and Xu, C. (2017b). Incorporating diversity and informativeness in multiple-instance active learning. *IEEE transactions on fuzzy systems*, 25(6):1460–1475.

Wang, S., Zou, Y., Ng, J., and Ng, T. (2017c). Context-aware service input ranking by learning from historical information. *IEEE Transactions on Services Computing*, 1(01):1–1.

Wang, Y., Nelissen, N., Adamczuk, K., De Weer, A.-S., Vandenbulcke, M., Sunaert, S., Vandenberghe, R., and Dupont, P. (2014). Reproducibility and robustness of graph measures of the associative-semantic network. *PloS one*, 9(12):e115215.

Wang, Z., Du, B., Zhang, L., Zhang, L., and Jia, X. (2017d). A novel semisupervised active-learning algorithm for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(6):3071–3083.

Wasserman, S. and Faust, K. (1994). *Social network analysis: Methods and applications*. Cambridge university press.

Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of 'small-world'networks. *nature*, 393(6684):440.

Wei, H., Pan, Z., Hu, G., Zhang, L., Yang, H., Li, X., and Zhou, X. (2018). Identifying influential nodes based on network representation learning in complex networks. *PloS one*, 13(7):e0200091.

Wei, K., Iyer, R., and Bilmes, J. (2015). Submodularity in data subset selection and active learning. In *International Conference on Machine Learning*, pages 1954–1963.

Weimann, G. (2016). Terrorist migration to the dark web. *Perspectives on Terrorism*, 10(3):40–44.

Whitelaw, C., Kehlenbeck, A., Petrovic, N., and Ungar, L. (2008). Web-scale named entity recognition. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 123–132. ACM.

Wu, L., Fisch, A., Chopra, S., Adams, K., Bordes, A., and Weston, J. (2018). Starspace: Embed all the things! In *AAAI Conference on Artificial Intelligence*, pages 5569–5577.

Xia, F., Liu, T.-Y., Wang, J., Zhang, W., and Li, H. (2008). Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199. ACM.

Xian, X., Zhao, P., Fang, W., Xin, J., and Cui, Z. (2009). Automatic classification of deep web databases with simple query interface. In *Industrial Mechatronics and Automation, 2009. ICIMA 2009. International Conference on*, pages 85–88. IEEE.

Xu, B., Wang, N., Chen, T., and Li, M. (2015). Empirical evaluation of rectified activations in convolutional network. In *Deep Learning Workshop, ICML 15*.

Xu, H., Zhang, C., Hao, X., and Hu, Y. (2007). A machine learning approach classification of deep web sources. In *Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)*, volume 4, pages 561–565.

Xu, X., Zhou, C., and Wang, Z. (2009). Credit scoring algorithm based on link analysis ranking with support vector machine. *Expert Systems with Applications*, 36(2):2625–2632.

Xue, Y. and Hauskrecht, M. (2019). Active learning of multi-class classification models from ordered class sets. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*.

Yang, B., Sun, J.-T., Wang, T., and Chen, Z. (2009). Effective multi-label active learning for text classification. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 917–926. ACM.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237.

Yang, Z., Salakhutdinov, R., and Cohen, W. W. (2017). Transfer learning for sequence tagging with hierarchical recurrent networks. In *5th International Conference on Learning Representations ICLR 2017*, pages 258–268.

Ye, Q., Wu, B., and Wang, B. (2010). Distance distribution and average shortest path length estimation in real-world networks. In *International Conference on Advanced Data Mining and Applications*, pages 322–333. Springer.

You, Q., Jin, H., Wang, Z., Fang, C., and Luo, J. (2016). Image captioning with semantic attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4651–4659.

Yu, K., Zhu, S., Xu, W., and Gong, Y. (2008). trnon-greedy active learning for text categorization using convex ansductive experimental design. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 635–642. ACM.

Zeiler, M. D., Ranzato, M., Monga, R., Mao, M., Yang, K., Le, Q. V., Nguyen, P., Senior, A., Vanhoucke, V., Dean, J., et al. (2013). On rectified linear units for speech processing. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3517–3521. IEEE.

Zhang, C. and Zhang, S. (2002). *Association rule mining: models and algorithms*. Springer-Verlag.

Zhang, Y., Bao, Y., Zhao, S., Chen, J., and Tang, J. (2015). Identifying node importance by combining betweenness centrality and katz centrality. In *Cloud Computing and Big Data (CCBD), 2015 International Conference on*, pages 354–357. IEEE.

Zheng, S., Hao, Y., Lu, D., Bao, H., Xu, J., Hao, H., and Xu, B. (2017). Joint entity and relation extraction based on a hybrid neural network. *Neurocomputing*, 257:59 – 66. Machine Learning and Signal Processing for Big Multimedia Analysis.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. (2016). Learning deep features for discriminative localization. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 2921–2929. IEEE.

Zhu, J., Wang, H., and Hovy, E. (2008a). Multi-criteria-based strategy to stop active learning for data annotation. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 1129–1136. Association for Computational Linguistics.

Zhu, J., Wang, H., Yao, T., and Tsou, B. K. (2008b). Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 1137–1144. Association for Computational Linguistics.

Annex B

SUMMARY OF THE THESIS IN SPANISH

RESUMEN DE LA TESIS EN CASTELLANO

Encumplimiento del punto 7 de la normativa complementaria del Real Decreto 778/1998, de 30 de Abril y de las normas para la aplicación del mismo, aprobadas por acuerdo de la Junta de Gobierno de fecha 10 de mayo de 1999, se adjunta un resumen en castellano de cada uno de los capítulos de esta tesis doctoral para que pueda admitirse a trámite.

1 Introducción

1.1 Motivación

Una representación famosa de Internet es un iceberg en medio del océano. La parte visible es la *Web Superficial*, y se refiere a la parte de la Web indexada por los buscadores, como Google o Yahoo. En cambio, la parte oculta del iceberg se conoce como la *Web Profunda* (del inglés, Deep Web) (Bergman, 2001; Noor et al., 2011; Al-Nabki et al., 2017a). En lo más profundo de la *Web Profunda*, hay una parte de la Web al que solo se puede acceder a través de navegadores específicos o proxies, es y la llamada *Web Oscura*, (del inglés Dark Web) (Moore and Rid, 2016; Al-Nabki et al., 2017a) (ver Figura 1).

La Web Oscura está formada por varias redes oscuras (del inglés, Darknet), siendo Tor (Tor del inglés, The Onion Router) [2] unas de las más populares. Los sitios web o dominios de Tor se denominan *servicios ocultos* (HS del inglés, Hidden Services). Existen sitios web con métricas extraídas de Tor[3] donde se pueden ver que el número de dominios de Tor ha llegado a alcanzar picos de $120K$ servicios ocultos (ver Figura 2).

[2] www.torproject.org
[3] https://metrics.torproject.org/

Figura 1: Representación iceberg de Internet

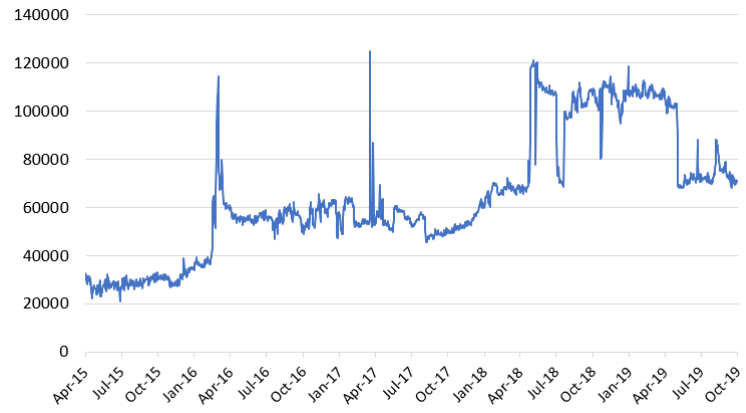


Figura 2: Número de dominios de Tor vivos entre abril de 2015 y octubre de 2019. El eje horizontal representa los años, mientras que el eje vertical corresponde al número de direcciones Tor únicas.

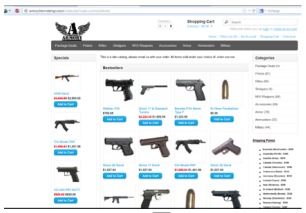
La privacidad y el anonimato que ofrece Tor promueven su posible uso fraudulento. Estas actividades incluyen, entre otras, la venta de drogas ilegales, comercio de armas y abuso sexual infantil (Ling et al., 2015; Ciancaglini et al., 2013; Moore and Rid, 2016; Fidalgo et al., 2017; Gangwar et al., 2017; Norbutas, 2018; Foley et al., 2018; Fidalgo et al., 2019).

INCIBE, el Instituto Nacional Español de Ciberseguridad trabaja con Fuerzas y Cuerpos de Seguridad del Estado (FFCCSE) - ó en inglés Law Enforcement Agencies - LEA - proporcionándoles herramientas y servicios automatizados para luchar contra el cibercrimen. Una de estas herramientas permite a las LEA españolas la supervisión automática de Tor, evitando que los agentes exploren manualmente miles de dominios diariamente. Dicha herramienta analiza el contenido de los dominios ocultos, extrayendo información significativa que ayudaría a los agentes en la lucha contra los delitos informáticos.

Gracias a nuestra colaboración con INCIBE, desarrollamos soluciones basadas en Inteligencia Artificial para estas herramientas y servicios orientados a la Ciberseguridad. En esta tesis, hemos creado varias soluciones para INCIBE basados en el procesamiento del lenguaje natural (NLP, del inglés Natural Language Processing). Nuestra investigación y soluciones para INCIBE se pueden dividir en tres componentes o unidades principales (ver Figura 3), que describiremos con un poco más de detalle en las siguientes subsecciones:

1. *Unidad de Clasificación de Texto (TCU, del inglés Text Classification Unit).*
2. *Unidad de Minería de Texto (TMU, del inglés Text Mining Unit).*
3. *Unidad de Detección de Influencia (IDU, del inglés Influence Detection Unit).*

Aunque los resultados de este trabajo están siendo utilizados para supervisar la red



Dirección de dominio	Armory34omvkkgr.onion
Categoría	Ilegal
Tipo de actividad	Armas
Entidades detectadas	**Productos:** Walther P22, Glock 17 ... **Ubicaciones:** Estados Unidos, España ... **Persona:** Jack Morgan **Organizaciones:** ARMORIA **Grupos:** ninguno **Trabajo creativo:** ninguno
Productos emergentes	Glock 32 Gen4
Posición Mundial	23
Rango de armas	2

Figura 3: Componentes propuestos para la supervisión automática de la red oscura Tor

oscura de la red Tor, podrían generalizarse para textos extraídos de otras fuentes, como redes sociales o foros.

1.1.1 Clasificación de Texto

Dada el volumen de datos contenido en la red Tor, no es posible realizar una supervisión de todos los dominios que son creados a diario, dado que implicaría una alta inversión en tiempo y recursos humanos. Las FFCCSE necesitan herramientas y servicios que les permitan supervisar la red Tor de una manera efectiva, extrayendo información relevante que les permita identificar dominios que albergan contenido *legal* o *ilegal*.

Existen estudios que ofrecen diferentes perspectivas y clasificaciones de las actividades de la red oscura Tor (Intelliagg, 2015; Moore and Rid, 2016; Ling et al., 2015). Pero prácticamente todos coinciden en que es posible encontrar dominios en Tor que alojan *actividades ilegales*, las cuales son objeto de nuestro trabajo, y *actividades legales* (ver Figura 4).

El objetivo de TCU es *clasificar* los dominios de la red oscura Tor en diferentes categorías (Al-Nabki et al., 2017a).

(a) Venta de drogas ilegales (b) Mercado de armas ilegales

(c) Dominio personal (d) Dominio de noticias

Figura 4: Capturas de pantalla de cuatro servicios ocultos de Tor. (a) y (b) muestran un dominio de venta de drogas y armas, respectivamente - actividades ilegales. (c) y (d) presentan dos dominios con contenido legal.

1.1.2 Minería de Texto

La relevancia de los diferentes productos ofertados en la red Tor varía siguiendo varios factores relacionados con las necesidades y la demanda del mercado. Estas tendencias se podrían analizar a través de transacciones de compra (Minelli et al., 2012; Suchacka and Chodak, 2017), sin embargo, éstas no son accesibles en los mercados de la red oscura (Buxton and Bingham, 2015).

Para solucionar ese problema, la TMU extrae un conjunto de características que ayudarían a evaluar la importancia o relevancia de un dominio en la red Tor. Esta relevancia se ha medido en esta tesis a través de dos enfoques: la detección de productos emergentes (EPD) (Al-Nabki et al., 2017b) (Ver Figura 5) y el Reconocimiento de Entidades Nombradas (NER).

La técnicas de NER identifican entidades nombradas (NE, del inglés Entity Names), que son palabras desconocidas, como un tipo específico de entidad, como ubicaciones, personas o nombres de productos. Durante la revisión de la literatura, se observó que las técnicas de NER no se habían utilizado para analizar los dominios ocultos de la red Tor. La investigación desarrollada en esta tesis ha cubierto este vacío en la literatura (Al-Nabki et al., 2019d).

Además, en esta tesis se presenta una nueva característica para mejorar algoritmos de NER existentes, que permiten reconocer seis tipos diferentes de NE: nombres de personas, marcas de productos, organizaciones, grupos, trabajos creativos y ubicaciones (Al-

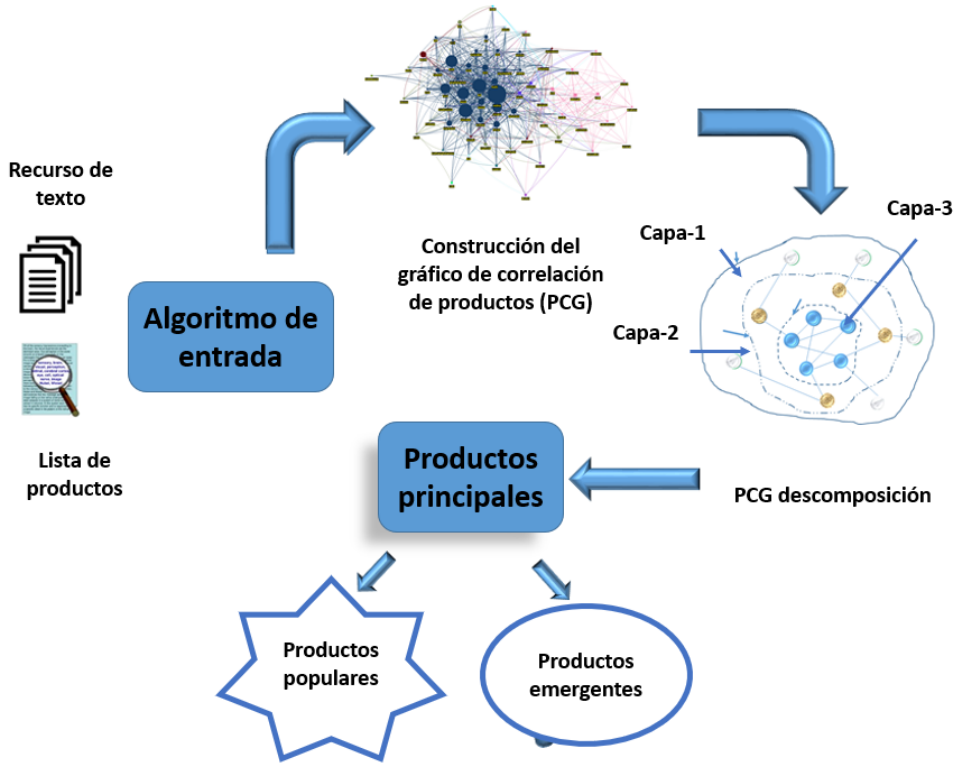


Figura 5: Etapas del algoritmo de detección de productos emergentes.

Nabki et al., 2019b) (ver Figura 6).

1.1.3 Detección de influencia

Una estrategia para detectar sitios influyentes es la supervisión del tráfico de red. Sin embargo, en la red Tor no es posible, debido a su diseño (Arma, 2019).

Dada una lista de dominios de la red Tor, y sus características extraídas previamente, la UDI asigna un valor de ranking a cada dominio. De este modo, los dominios con un valor más alto del ranking serían los dominios más influyentes dentro de la red oscura Tor.

En este trabajo, por un lado, utilizamos una aproximación basada en hiperenlaces para analizar la conectividad de hipervínculos entre los nodos (Al-Nabki et al., 2019c). Por otra parte, utilizamos una aproximación basada en el contenido, en la que se utiliza el contenido de los dominios para entrenar una función de clasificación usando el esquema *Aprender a Clasificar* (LtR, del inglés Learning to Rank) (Li, 2011b; Al-Nabki et al., 2019a).

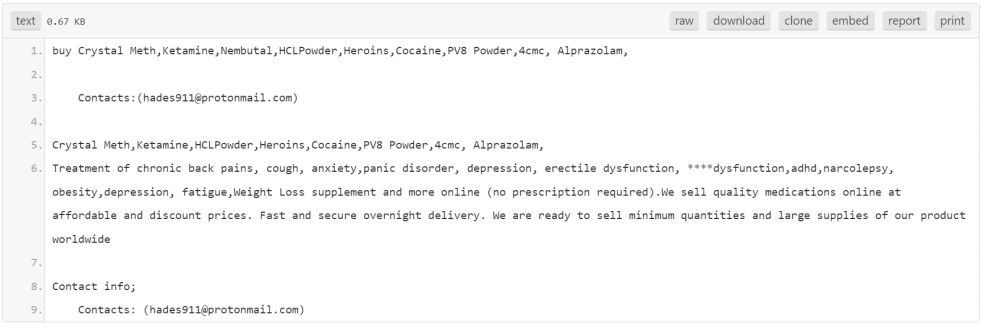
Figura 6: Extracción de entidades nombradas de un dominio de Tor que ofrece medicamentos. Los rectángulos se refieren a las entidades reconocidas. Los colores, amarillo, azul y rojo, indican entidades de tipos: organización, productos/marca y ubicación/dirección, respectivamente. El producto subrayado con una línea roja indica un producto emergente.

1.2 Clasificación de servicios de bloc de notas en línea: Pastebin

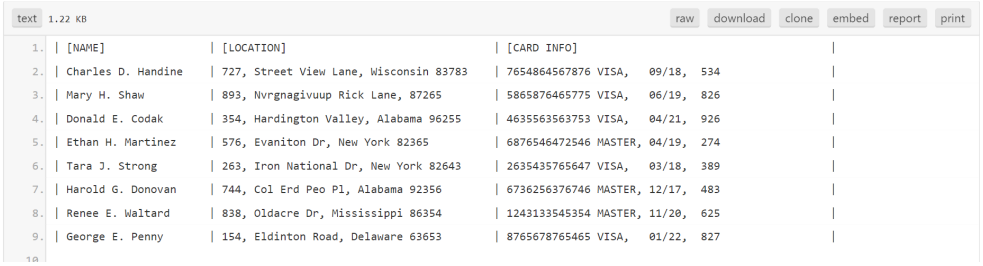
En este trabajo también se ha trabajado en una pequeña porción de la Web Superficial, más específicamente en los Bloc de notas en línea (ONS, del inglés Online Notepad Services). Un ONS permite acceder a notas de texto en la nube, a las cuales se puede acceder a través de un Localizador Uniforme de Recursos (URL, del inglés Uniform Resource Locator) único. Pastebin [4] (Squire and Smith, 2015), uno de los ONS más populares, aloja en su mayoría código fuente y otro contenido legítimo. Pero también hay usuarios que lo utilizan con fines maliciosos, como filtrar información confidencial, ataques de piratería o incluso propaganda de actividades ilegales realizadas en Darknet (Matic et al., 2012; Mirante and Cappos, 2013; Herath, 2017; Kashiwazaki, 2018; Brian, 2019; Riesco et al., 2019) (Ver Figura 7). En esta tesis, diseñamos un sistema de clasificación de texto para detectar estas actividades utilizando Aprendizaje Activo (AL, del inglés Active Learning) (Settles,

[4] https://pastebin.com

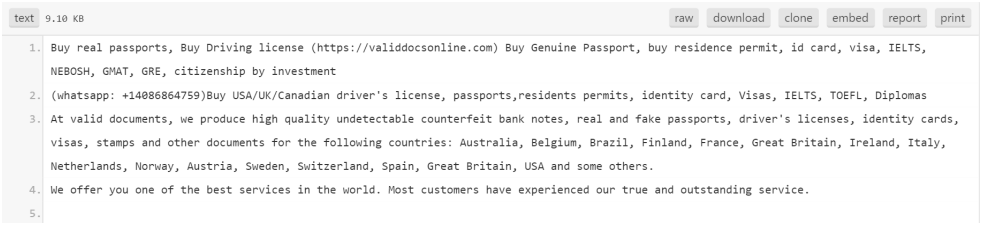
2009).



(a) Drogas ilegales



(b) Tarjetas de crédito falsificadas



(c) Identificación personal falsificada

Figura 7: Tres pastas del servicio de bloc de notas en línea Pastebin.

1.3 Organización del Resto del Documento

En el Capítulo 0.2 se revisa el estado del arte de las técnicas de NLP relacionadas con la clasificación de texto, minería de texto y ordenación. En el Capítulo 0.3 se aborda el problema de la clasificación de dominios ocultos en la red Tor y la detección de las actividades sospechosas que realizan. El Capítulo0.4 presenta un algoritmo semiautomático para detectar productos emergentes en los mercados online de la red oscura Tor, además de las contribuciones en el campo del reconocimiento de entidades nombradas. Posteriormente, en el Capítulo 0.5, se presentan los resultados obtenidos tras explorar méto-

dos de ordenación de dominios basados en los enlaces y en el contenido. Finalmente, el Capítulo 0.6 expone las conclusiones de esta tesis doctoral y las líneas futuras de trabajo.

2 Revisión del Estado de la Técnica

2.1 Clasificación de Texto

La clasificación de texto es una tarea fundamental en el procesamiento del lenguaje natural (NLP, del inglés Natural Language Processing), y su objetivo es la clasificación de un fragmento de texto en una de una lista predefinida de clases (Sachan et al., 2018).

La variedad de actividades y el creciente uso de la red Tor para actividades de carácter ilegal en determinados países ha atraído recientemente la atención de los investigadores (Graczyk and Kinningham, 2015; Moore and Rid, 2016; Al-Nabki et al., 2017a; Park et al., 2018; Dalins et al., 2018; Dong et al., 2018; Takaaki and Atsuo, 2019; Al-Nabki et al., 2019c). Con respecto a la Web Profunda, Xu et al. (2007) presentó un clasificador de texto supervisado y utilizó Ganancia de Información (IG del inglés, Information Gain) para extraer características del texto. Para ponderar estas características, utilizaron una versión personalizada de Frecuencia de término - Frecuencia de documento inversa (TF-IDF, del inglés Term Frequency - Inverse Document Frequency), llamada Frecuencia de término de web profunda. Barbosa et al. (2007) abordó el problema a través del aprendizaje no supervisado, combinando las técnicas TF-IDF con un clasificador K-means.

Noor et al. (2011) exploró técnicas convencionales utilizadas para extraer contenido de la Web profunda, como el «Query Probing», que se usa comúnmente para algoritmos de aprendizaje supervisados, y «Visible Form Features» (Xian et al., 2009). Dalins et al. (2018) presentó un nuevo modelo, llamado Tor-use Motivation Model, que permite clasificar los servicios ocultos de la red Tor con mayor granularidad. Al-Nabki et al. (2017a) propusieron DUTA (DUTA del inglés, Darknet Usage Text Addresses), el conjunto de datos textuales más amplio y públicamente disponible sobre la red Tor, conteniendo 6,831 dominios etiquetados en 25 categorías diferentes. Sobre este dataset, evaluaron dos codificadores de texto, junto a tres clasificadores, viendo que los mejores resultados se obtenían con la combinación de TF-IDF y un clasificador de regresión logística. Al-Nabki et al. (2019c) extendieron DUTA hasta las 10,367 muestras, en una nueva versión del dataset denominada DUTA-10K. Más recientemente, He et al. (2019) propuso un clasificador supervisado para las actividades ilegales en la red Darknet de Tor utilizando la técnica TF-IDF y el clasificador Naive Bayes (NB).

Otra alternativa para solucionar el problema de la clasificación de texto es el uso del Aprendizaje Activo (AL, del inglés Active Learning). AL se ha utilizado en muchos campos, como la clasificación de imágenes (Joshi et al., 2009; Wang et al., 2016, 2017d), recuperación y clasificación de información (Rubens et al., 2015; O'Neill et al., 2016), reconocimiento de entidades nombradas (Chen et al., 2015) y clasificación de texto (Yang et al., 2009; Goudjil et al., 2018; Reyes et al., 2018). AL permite reducir el coste de asignación de categorías a datos no etiquetados , ya que selecciona solo las muestras más informativas para que un experto las anote.

2.2 Minería de Texto

En la literatura se investiga el problema de la detección de productos emergentes (EPD, del inglés Emerging Product Detection) en un corpus de texto dado. El proceso EPD habitualmente sigue dos estrategias (Kontostathis et al., 2004): semi-automática y automática. Por un lado, en las estrategias semi-automáticas se reciben dos entradas: una lista predefinida de términos y un corpus de texto. A partir de estas entradas, un algoritmo de EPD procesa el corpus - o conjunto de datos - para identificar los términos emergentes (Davidson et al., 1998; Swan and Allan, 2000; Tucker and Kim, 2011; Schubert et al., 2014). Por otro lado, las estrategias automáticas reciben solo un corpus de texto como entrada y el algoritmo de EPD encuentra automáticamente los términos emergentes (Glance et al., 2004; Abe and Tsumoto, 2010).

En los mercados de la red Tor, los registros de transacciones de compras no están disponibles, lo que impide el uso de cualquiera de las técnicas anteriores. Nuestro algoritmo de EPD emplea la teoría de grafos para representar los productos de los mercados de Tor mediante un grafo no dirigido de nodos y bordes. Los nodos se refieren a los productos del mercado, y los bordes expresan la presencia de dos productos dentro del mismo mercado (Al-Nabki et al., 2017b). Usando el algoritmo K-shell (Carmi et al., 2007), descomponemos el grafo en capas (en inglés shells) de acuerdo con la conectividad entre los productos en el grafo. Los productos presentes en la capa más profunda (en inglés core-shell), son identificados por el algoritmo como productos emergentes.

Con respecto al reconocimiento de entidades nombradas en el texto, Aguilar et al. (2017) propuso un modelo basado en redes neuronales. En dicho modelo, existe una característica que se extrae de un recurso de datos externo, un diccionario geográfico, lo que le permitió obtener unas puntuaciones de entidad y superficie F1 de 41,86 % y 40,24 %, respectivamente, en el conjunto de datos W-NUT-2017 (Derczynski et al., 2017).

Varios investigadores han utilizado diccionarios geográficos para capturar características adicionales del texto de entrada (Mishra and Diesner, 2016; Aguilar et al., 2017; Štravs and Zupančič, 2019; Dey and Prukayastha, 2013; Ding et al., 2019). Sin embargo, su uso se podría considerar una limitación debido a las dificultades para construirlo y mantenerlo actualizado para hacer frente a nuevos términos y entidades. Recientemente, Akbik et al. (2019) presentó una técnica de incrustaciones contextualizadas agrupadas, que logra resultados del estado del arte, con puntuaciones F1 en el conjunto de datos W-NUT-2017 de 49,59 %.

2.3 Detección de Influencia

Un algoritmo de ordenación de sitios web puede estar basado en los hiperenlaces entre los diferentes dominios, en su contenido o puede ser híbrido, es decir, basado en ambas aproximaciones.

Xu et al. (2009) propuso un algoritmo que combina un clasificador basado en enlaces con uno de máquinas de vectores de soporte, para determinar la elegibilidad de los solicitantes para una tarjeta de crédito o préstamos a bancos. Fronzetti Colladon and Remondi (2017) exploró el uso de métricas de las redes sociales, como la centralidad interna, externa y de proximidad para combatir los delitos de lavado de dinero. Taha and Yoo (2017) presentó un algoritmo que utiliza el Árbol de Expansión Mínima (MST, del inglés Minimum Spanning Tree) para construir una red de delincuentes. A cada nodo se le asignó una puntuación proporcional al número de nodos cuya existencia depende de la existencia del nodo objetivo. Hu et al. (2018) propuso UserRank, una evolución de PageRank orientada a la red de desarrolladores de GitHub.

A pesar de la efectividad del enfoque basado en hiperenlaces, ésta fallaría al evaluar nodos aislados o sin conexiones con el resto. En estos casos, un enfoque basado en el contenido podría superar este problema y mejorar las técnicas basadas en enlaces. Anwar and Abulaish (2015) presentó un algoritmo para detectar los líderes influyentes de grupos radicales en los foros de Darknet. Extrajeron el contenido de los perfiles de usuario, junto con sus publicaciones, para extraer características textuales que midieran su radicalidad. A continuación, incorporaron estas características a PageRank (Page et al., 1999), un algoritmo de ordenación basado en enlaces, creando una lista ordenada de usuarios influyentes según su radicalidad. MacAvaney et al. (2019) propuso un algoritmo de ordenación basado en el contenido mediante la extracción de características del contenido utilizando el algoritmo BM25 (Robertson and Zaragoza, 2009).

El enfoque propuesto en este trabajo es diferente a lo revisando en la literatura. Durante la realización de este doctorado, no se encontraron trabajos que hubieran abordado la ordenación de dominios ocultos de Tor utilizando su contenido. En esta tesis, se ha utilizado un algoritmo de Aprender a Ordenar (LtR, del inglés Learning to Rank) para aprender automáticamente la ordenación de dominios según unos criterios de ordenación previamente establecidos.

3 Clasificación de Texto

En esta sección, el objetivo de esta tesis es la clasificación automática de textos, en una serie de categorías predefinidas. Inicialmente se describirá la clasificación en la red oscura Tor, y a continuación de describirá la clasificación del Bloc de Notas Online (ONS) Pastebin utilizando Aprendizaje Activo.

3.1 Clasificación de Servicios Ocultos en la red oscura Tor

En la red Tor, diseñamos un clasificador supervisado para detectar y clasificar actividades sospechosas en sus dominios ocultos (HS, del inglés Hidden Services).

3.1.1 Conjunto de Datos DUTA

Durante la realización de esta tesis se observó en la literatura que no existía ningún conjunto de datos público que contuviera texto extraído de los dominios ocultos (HS) de Tor (Biryukov et al., 2014; Moore and Rid, 2016; Intelliagg, 2015). Por ese motivo, se creó DUTA (del inglés, Darknet Usage Text Addresses). Primero, se creó un rastreador que buscaba en un listado de enlyaces, y accedía al contenido de HS, utilizando fuentes como onion.city[5] y ahmia.fi[6]. Inicialmente se obtuvieron $250K$ enlaces de dominios, pero solo 6,831 estaban en línea durante el rastreo: el resto estaban inactivos o no respondían. A continuación, se accedió al contenido de los dominios en línea y se unieron las páginas HTML de cada HS en un solo archivo HTML. El resultado fue DUTA, el primer dataset de texto público basado en la red Tor, conteniendo 26 clases que se enumeran en la Tabla 1

3.1.2 Metodología

Un clasificador de texto consta de tres etapas: pre-procesamiento de texto, extracción, codificación de características y clasificación. En este trabajo se han combinado dos técnicas de codificación de texto con tres clasificadores supervisados, obteniendo seis propuestas de clasificadores de texto diferentes.

Durante el pre-procesamiento, se eliminan las etiquetas HTML, caracteres especiales y palabras vacías (del inglés, stop words). Finalmente, se mapean todos los correos electrónicos y URL en las siguientes etiquetas <*EML*> y <*URL*>, respectivamente.

Para la extracción de características se utilizó la Bolsa de palabras (BOW, del inglés Bag of Words) (Harris, 1954) y la Frecuencia de Término - Frecuencia Inversa del Documento (TF-IDF, del inglés Term Frequency - Inverse Document Frequency) (Aizawa, 2003).

En cuanto a los clasificadores, se utilizaron el Clasificador Ingenuo de Bayes (NB, del inglés Naïve Bayes) (McCallum et al., 1998), Máquinas de Vectores de Soporte (SVM, del

[5] www.onion.city

[6] www.ahmia. fi

[7] Esta clase incluye 57 muestras únicas con 857 muestras adicionales extraídas de un solo foro.

Clase principal	Subclase	Countar	Clase principal	Contar
Violencia	Odio	4	Arte/ Música	8
	Sicarios	11	Casino/Juego	26
	Armas	47	Servicios	285
Contrabando Personal Identificación	Carnet Conducir	4	Criptomonedas	586
	ID	7	Caídas	608
	Pasaporte	37	Vacías	1,649
Hosting y Software	Intercambio de ficheros	111	Foros	104
	Carpetas	63	Hacking	90
	Servidores	95	Datos filtrados	12
	Software	121	Bloqueadas	435
	Directorios	142	Personal	405
Drogas	Ilegal	230	Política	8
	Legal	9	Religión	6
Tiendas	Ilegales	63	Fraude	4
	Legales	67	Librería. Libros	27
Pornografía	CSA-Material	914 ([7])	Contrabando Dinero	55
	Pornografía-en general	83	Contrabando Tarjetas Crédito	240
Redes Sociales	Blogs	71	Tráfico de personas	2
	Chat	47		
	Email	56		
	Noticias	32		
Total				**6,831**

Tabla 1: Categorías del conjunto de datos DUTA

inglés Support Vector Machine) (Suykens and Vandewalle, 1999), y Regresión Logística (LR, del inglés Logistic Regression) (Hosmer Jr and Lemeshow, 2004).

3.1.3 Evaluación Empírica y Resultados

Seleccionamos un subconjunto de ocho categorías de DUTA, como se muestra en la Tabla 2, tratando de cubrir las actividades ilegales más representativas. Se dispone, finalmente, de 5,635 muestras distribuidas en nueve clases, es decir, las ocho clases relacionadas con actividades ilegales más una categoría adicional denominada *Otros*. Los detalles sobre cada categoría se pueden ver en la Tabla 3.2. Después de la etapa de *preprocesamiento*, quedaron 5,002 dominios ocultos, que se dividieron en dos conjunto de entrenamiento y pruebas, con 3,501 y 1,501 muestras respectivamente, representando un 70/30.

En la Tabla 3, se puede comprobar como la combinación de TF-IDF y LR obtiene los mejores resultados, con una puntuación F1 y precisión de 93,7 % y 96,6 %, respectivamente. Por el contrario, la combinación de TF-IDF con NB obtuvo el peor resultado, con una

Categorías	# Muestras
Pornografía	963
Criptomonedas	578
Contrabando Tarjetas Crédito	209
Drogas	169
Violencia	60
Hacking	57
Contrabando Moneda	46
Contrabando Identificaciones personales (Carné de conducir, ID, Pasaportes)	40
Otros	3,513

Tabla 2: Subconjunto de ocho categorías de DUTA, más una novena denominada *Otros*, para el entrenamiento del clasificador supervisado.

puntuación F1 y precisión de 46,0 % y 86,3 %, respectivamente.

Métodos/ Métricas		Promedio (macro %)	Promedio (micro %)	Promedio (ponderación %)	Validación Cruzada %
BOW LR	P	95.2	96.5	96.5	95.8 +/- 0.010
	R	88.9	96.5	96.5	
	F1	91.6	96.5	94.6	
TF-IDF LR	P	98.2	97.4	97.5	**96.6 +/- 0.010**
	R	90.2	97.4	97.4	
	F1	**93.7**	**97.4**	**97.4**	
BOW SVM	P	87.7	94.1	94.2	93.2 +/- 0.013
	R	87.5	94.1	94.1	
	F1	87.4	94.1	94.1	
TF-IDF SVM	P	98.3	97.1	97.2	96.0 +/- 0.009
	R	88.2	97.1	97.1	
	F1	92.4	97.1	97.0	
BOW NB	P	86.5	94.1	94.3	92.4 +/- 0.009
	R	97.0	94.1	94.1	
	F1	81.2	94.1	94.0	
TF-IDF NB	P	53.0	88.5	88.5	86.3 +/- 0.012
	R	42.5	88.5	88.5	
	F1	46.0	88.5	86.0	

Tabla 3: Comparación de resultados de los seis clasificadores de texto propuesto, reportando el valor medio de la validación cruzada de 10 iteraciones (10-fold CV, del inglés Cross Validation), utilizando la precisión (P), sensibilidad (R) y puntuación F1 para micro, macro y ponderado .

La Figura 8 muestra la puntuación F1 de cada combinación de codificadores y clasificadores por categoría. Se puede observar que las las clases que suelen contener palabras clave distintas, como las clases *Pornografía* y *Criptomoneda*, obtienen una puntuación F1

relativamente más alta en todos clasificadores de texto evaluados.

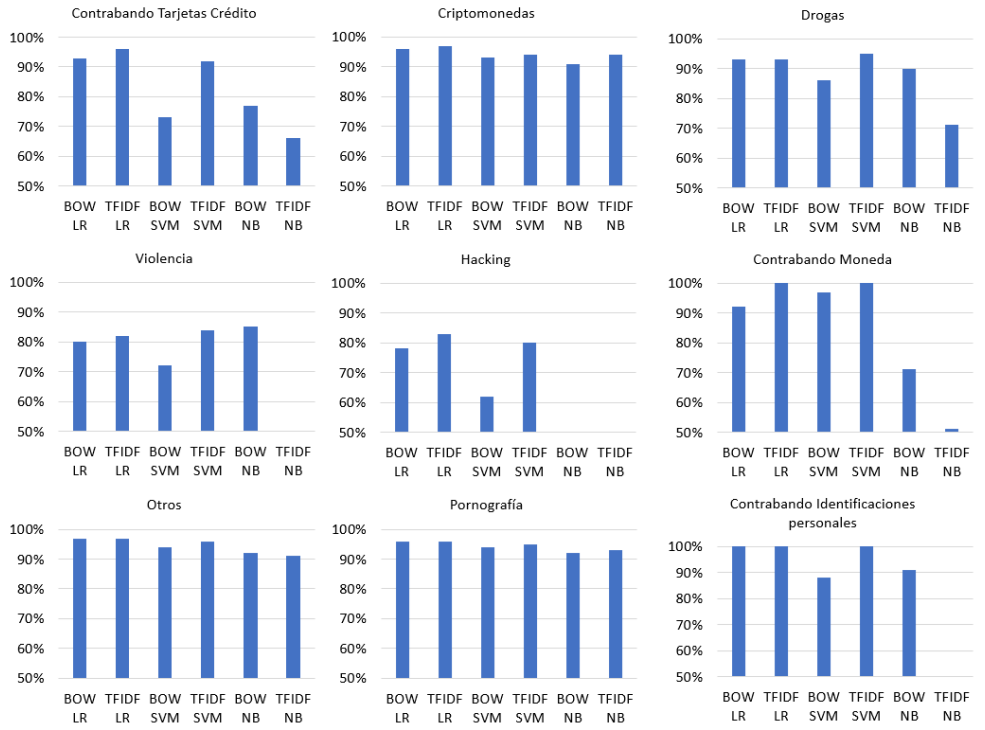


Figura 8: Comparación de puntuación F1 por clase de los clasificadores de texto propuestos. Cuando no se muestra una barra, significa que su valor es cero.

3.2 Aprendizaje Activo

Un servicio de Bloc de Notas en línea (ONS, del inglés Online Notepad Services) permite a los usuarios publicar texto no estructurado de forma anónima. Pastebin es uno de los ONS más populares, y representa una fuente inagotable de texto generado por los usuarios.

Debido al anonimato que proporcionan estos servicios, hay ciertos usuarios que lo utilizan para publicar contenido inapropiado, como material de abuso sexual a menores (CSEM, del inglés Child Sexual Exploitation Material) o enlaces a mercados de venta de drogas. Por ese motivo, sería de interés proporcionar un sistema de clasificación que permitiera, por ejemplo, a Fuerzas y Cuerpos de Seguridad del Estado (FFCCSE) la monitorización automática de ONS y combatir posibles delitos informáticos.

Sin embargo, uno de los inconvenientes a la hora de trabajar con sistemas supervisa-

dos es la falta de conjuntos de datos etiquetados. El aprendizaje activo (AL, del inglés Active Learning) es una técnica que facilita la creación de un conjunto de datos etiquetados a partir de una extensa colección de muestras no etiquetadas, lo que reduce el esfuerzo de etiquetado manual. Solo las muestras nominadas por el algoritmo AL se etiquetarían manualmente y se usarían para entrenar (Settles, 2009).

3.2.1 Metodología

En Pastebin, existen varios tipos de texto, desde fragmentos de código o archivos codificados en binario hasta registros, historias o noticias. Debido a esta diversidad, se abordó el objetivo de la clasificación de texto en Pastebin en en tres niveles, utilizando para cada nivel las técnicas más adecuadas para codificar el texto en función de su tipología. Se proponen tres clasificadores de texto diferentes: de fragmentos de código (CSC, del inglés Code Snippt Classifier), de texto legible / no legible (RNC, del inglés Readable/ Non-Readable Classifier) y de actividades sospechosas (SAC, del inglés Suspicious Activities Classifiers).

3.2.2 Evaluación Empírica y Resultados

El clasificador CSC está basado en la librería guesslang[8]. A través de CSC, se evaluaron 2,360,000 muestras - o pastes - de Pastebin, resultando el 59 % como fragmentos de código. El resto, 1,640,000 pastes, se clasificó como texto no considerado como código fuente.

Para identificar cuáles son las posibles categorías ilegales dentro de Pastebin, se utilizó como punto de partida el dataset etiquetado PasteCC_17K (Riesco et al., 2019). En aquellas categorías con un número bajo de muestras, como «Falsificación de documentos de identidad», se utilizó el motor de búsqueda de Pastebin para adquirir más muestras. Además, dicho motor se utilizó para conseguir más muestras de pastes con ciertas palabras clave, como «pasaportes falsos», «EEUU tarjeta de residencia» y «licencia de conducir original». La tabla 4 resume el número de categorías utilizadas, así como el número de muestras por categoría.

Para entrenar el clasificador legible / no legible (RNC), se utilizó el enfoque del Aprendizaje Activo (AL) basado en la exploración, usando la estrategia de selección de Muestreo de Margen (MS, del inglés, Margin Sampling) (Balcan et al., 2007).

Para entrenar el clasificador legible / no legible (RNC), exploramos tres estrategias de AL para la selección de muestras del grupo: AL basado en exploración, basado en explotación e híbrido.

La figura 9 muestra que usando 180 muestras seleccionadas usando el AL basado en exploración, se obtiene un clasificador con sensibilidad de clase promedio (del inglés, Average Class Recall) de 92,84 %.

[8] https://github.com/yoeo/guesslang

Tabla 4: Especificaciones del conjunto de prueba del *Clasificador de actividades sospechosas (SAC)*. La letra «C» se refiere a la actividad de falsificación (C, del inglés Actividades de Contrabando).

Categoría	**# Muestras**
Otros	1,696
C. Tarjetas Crédito	54
Hacking	75
Filtrado de datos	103
C. Identificación personal	96
Drogas	72
Pornografía	89
Total	2,185

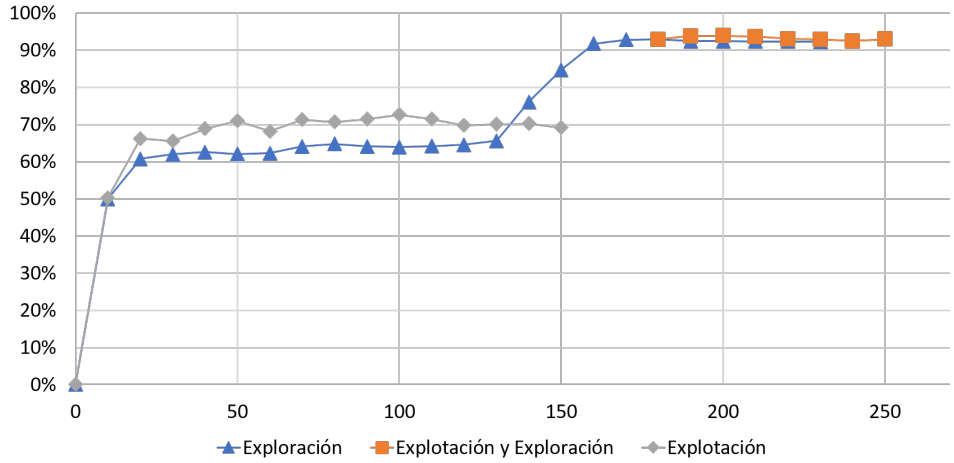


Figura 9: Comparación de los tres enfoques AL para construir un clasificador legible / no legible (RNC). El eje X se refiere al número de muestras etiquetadas, mientras que el eje Y presenta la sensibilidad promedio de clase del RNC.

Para entrenar el clasificador de actividades sospechosas (SAC), utilizamos el aprendizaje activo basado en la explotación. En particular, SAC está compuesto por dos clasificadores: un primer clasificador binario para indicar si una muestra es sospechosa o no, y un clasificador multiclase con seis categorías.

Para el clasificador binario, la Figura 10 muestra que al entrenar al clasificador con 437 muestras - 20 % del conjunto de datos - se obtiene una sensibilidad promedio de clase del 94,21 %. A partir de la décima iteración, cuando se habían etiquetado 540 muestras, el clasificador incrementó su sensibilidad al 95,24 %, activándose el criterio de detención del algoritmo.

Para el clasificador multiclase, la Figura 11 muestra que después de 16 iteraciones y 160 muestras etiquetadas, el rendimiento alcanza el 80,33 %. Tras obtener 530 muestras

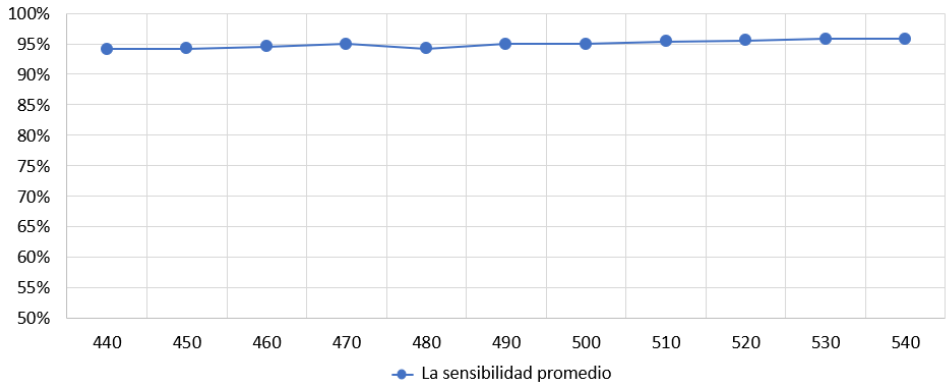


Figura 10: AL basado en la exploración para la porción texto legible(RT, del inglés Readable Text) de Pastebin. El eje X se refiere al número de muestras etiquetadas, mientras que el eje Y muestra la sensibilidad promedio de la clase.

etiquetadas, la sensibilidad deja de aumentar, y su valor oscila en torno al 80 %.

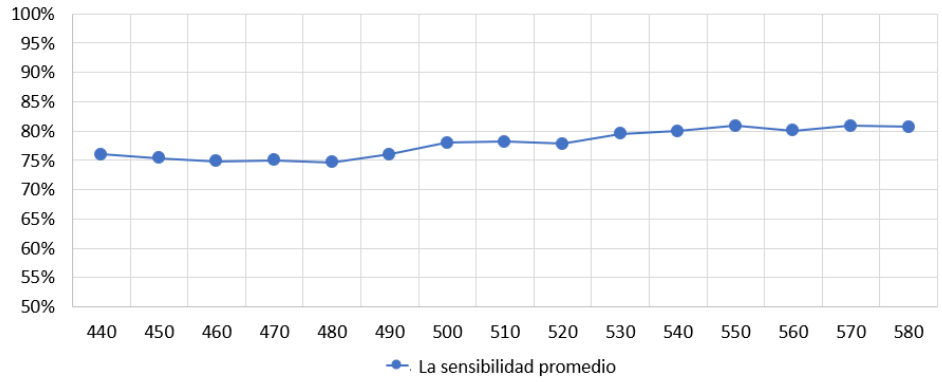


Figura 11: AL basado en la explotación para la porción RT de Pastebin. El eje X se refiere al número de muestras etiquetadas, mientras que el eje Y muestra la recuperación promedio de la clase.

3.2.3 Conclusiones

En este capítulo se presentan dos aproximaciones para la realización de clasificaciones de texto automáticas: la primera a través de datos previamente etiquetados, y la segunda a través de técnicas de Aprendizaje Activo para el etiquetado de muestras representativas.

En la primera aproximación, se presenta el dataset DUTA, un nuevo conjunto de datos que contiene 6,831 muestras etiquetadas manualmente en 26 categorías. Sobre un

subconjunto de 9 categorías de DUTA, se evaluaron seis arquitecturas de clasificación de texto, resultantes de la combinación de dos codificadores de texto - TF-IDF y BOW - con tres clasificadores - SVM, LR y NB. La combinación de TF-IDF con LR obtiene el mejor resultado, con una precisión del 96,6 % y una puntuación macro F1 de 93,7 %, en la tarea de clasificar el contenido ilegal de los dominios de Tor.

En segundo lugar, diseñamos y presentamos una arquitectura de clasificación supervisada, utilizando Aprendizaje Activo, para identificar actividades sospechosas en Pastebin. Descubrimos que usando 25,000 de pastes legibles es posible representar hasta 86 % de las entradas obtenidas de Pastebin.

Sobre este subconjunto de 25,000 muestras, se han adoptado dos enfoques de Aprendizaje Activo para seleccionar las muestras más informativas para el entrenamiento: basados en la explotación y en la exploración. Seleccionando solo 180 pastes con el enfoque de exploración, podemos obtener un clasificador con una sensibilidad de clase promedio (del inglés, Average Class Recall) de 92,84 %. Además, al inicializar un algoritmo AL basado en la explotación con 20 % de las muestras etiquetadas, es posible obtener un clasificador binario y otro clasificador multi-categoría con una sensibilidad promedio de 95,24 % y 80,33 %, respectivamente.

4 Minería de texto

En este capítulo se proponen dos técnicas de minería de texto. En primer lugar, un algoritmo para detectar productos emergentes en los dominios ocultos de mercado online de la red Tor. En segundo lugar, una arquitectura de red entrenada y mejorada para reconocer entidades nombradas en textos generado por usuarios.

4.1 Detección de Productos Emergentes en la red oscura Tor

La aplicación de técnicas de minería de datos para comprobar registros de compras en mercados online puede resolver el problema de detección de productos emergentes de manera eficiente (Raeder and Chawla, 2011). Sin embargo, cuando estos registros son inaccesibles, como es el caso de los mercados de la red Tor, se deben buscar soluciones alternativas.

4.2 Metodología

En esta sección se presenta un algoritmo semiautomático para detectar productos emergentes en los mercados de la red Tor.

Primero, el algoritmo se inicia con un corpus - o conjunto de datos - y una lista de productos predefinida, en este caso, un listado de drogas. Después, se modela el problema de la detección de productos emergentes como un grafo no dirigido, llamado grafo de correlación de productos (PCG, del inglés Products Correlation Graph). Los nodos representan productos, y tienen asociados tres parámetros: la frecuencia del producto en el conjunto de datos o corpus, el grado de centralidad en el PCG, y el soporte del producto. Los enlaces representan la oferta mutua de dos productos dentro del mismo mercado online.

Después de construir el PCG, se aplica el algoritmo k-shell para descomponer el grafo en capas. Finalmente, algoritmo de detección de productos emergentes (EPD, del inglés Emerging Product Detection) identifica los productos de la capa más profunda como productos populares o emergentes.

4.2.1 Experimentos y Resultados

En esta tesis, se ha trabajado con el caso de uso de la venta de drogas ilegales. El corpus o conjunto de datos utilizado es DUTA (Al-Nabki et al., 2017a), y se considera un listado de drogas conocidas como listado de productos.

Se extrajeron 395 nombres de drogas - que en el grafo quedan representados por nodos - y 38,347 relaciones de oferta mutua de entre ellas. El algoritmo k-shell descompuso los PCG en 12 capas, donde la capa más interna contenía 27 productos. La tabla 5 muestra los productos de la capa más profunda, siendo estos productos candidatos a ser productos emergentes.

Tabla 5: Los productos farmacéuticos emergentes en la capa más profunda para un subconjunto de datos del corpus DUTA

Productos Capa Profunda	**Candidatos a Emergentes**	**Productos emergentes**
Marijuana, DMT, Edibles, Ecstasy, XTC, Cocaine, MDMA, Speed, Heroin, Weed, Mxe, 2C-B,„Haze, Mescaline, GHB, MDA, Benzos, Cannabis, LSD, Mushrooms, GBL, Meth, Hashish, Amphetamine, Ketamine, Psychedelics, Kush	Ecstasy, Ketamine, DMT, MDA, Marijuana, Hashish, Mescaline, Psychedelics, Amphetamine, GHB, Mushrooms, 2C-B, Edibles, GBL, Mxe, Benzos, XTC	Ecstasy, Ketamine, DMT, MDA, Marijuana, Hashish, Mescaline, GHB, XTC

La figura 12 muestra una lista de los 20 productos más populares en el período de estudio, siendo los tres primeros MDMA, LSD y el Cannabis.

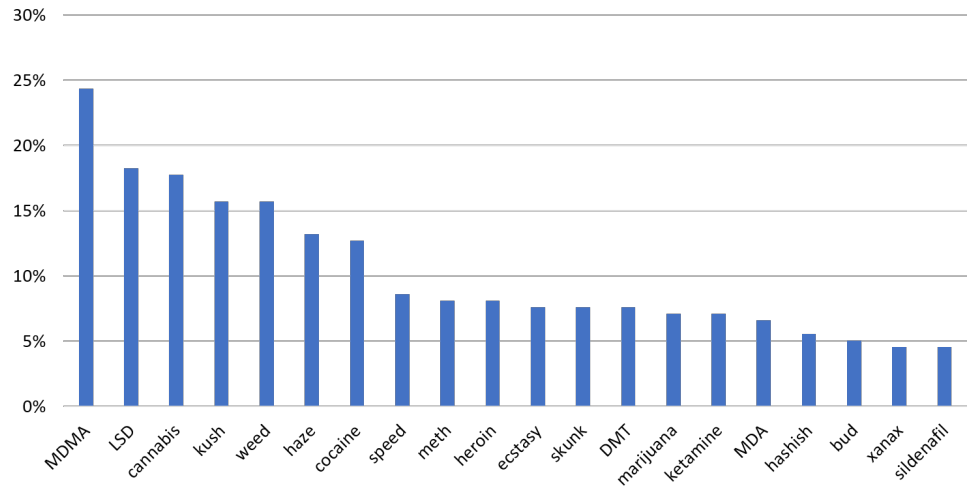


Figura 12: Gráfico con los 20 productos - drogas - más populares en la red Tor en el período de muestreo. El eje Y se refiere al porcentaje de dominios del corpus que contienen los productos del eje X.

A través del estudio, se pudo observar que parejas productos son los que más frecuentemente son ofertados conjuntamente. La figura 13 muestra que entre las 20 parejas más ofertadas, el MDMA y el LSD son los productos más habitualmente ofertados dentro del mismo mercado online.

Además del análisis cuantitativo, se presenta una visualización del grafo de correlaciones de productos (PCG), ya que podría ayudar a comprender las relaciones subyacentes

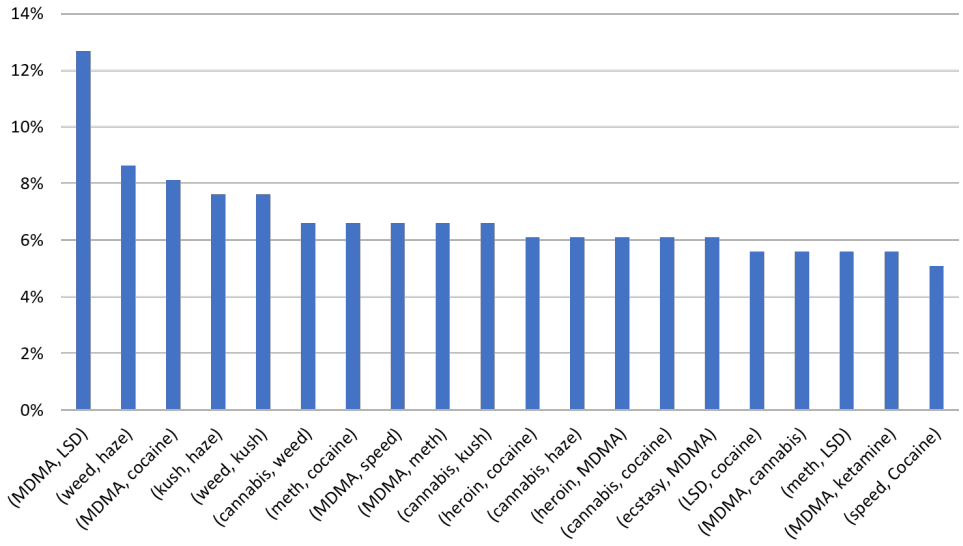


Figura 13: Los 20 pares de productos ofertados más frecuentemente en el conjunto de datos DUTA. El eje X muestra los pares identificados, mientras que el eje Y se refiere al porcentaje de dominios donde se ofertan.

entre los productos en el grafo (Figura 14).

4.3 Reconocimiento de Entidades Nombradas en la red oscura Tor

En esta sección, se presenta una arquitectura de red neuronal de extremo a extremo para reconocer entidades nombradas en texto ruidoso generado por el usuario.

El modelo propuesto está inspirado en el modelo de Aguilar et al. (2017) pero con dos diferencias principales. Primero, la arquitectura propuesta no depende de ningún recurso de conocimiento externo, como un diccionario geográfico, que es costoso de construir y depende del dominio. En segundo lugar, la arquitectura presenta una característica nueva, que denominamos *Local Distance Neighbor* (LDN), y que sustituye el uso de cualquier recurso de conocimiento externo.

4.3.1 Metodología

La característica propuesta, LDN, está inspirada en la forma en que un humano intenta descubrir el significado de un término desconocido o de una palabra ambigua dentro de un texto. Esto se puede hacer a través de un diccionario, o tratar de buscar palabras que sean semánticamente similares al término ambiguo.

LDN intenta emular este comportamiento considerando que cada palabra de un texto es ambigua, es decir, no tiene una etiqueta, y esta etiqueta cambia según otras palabras

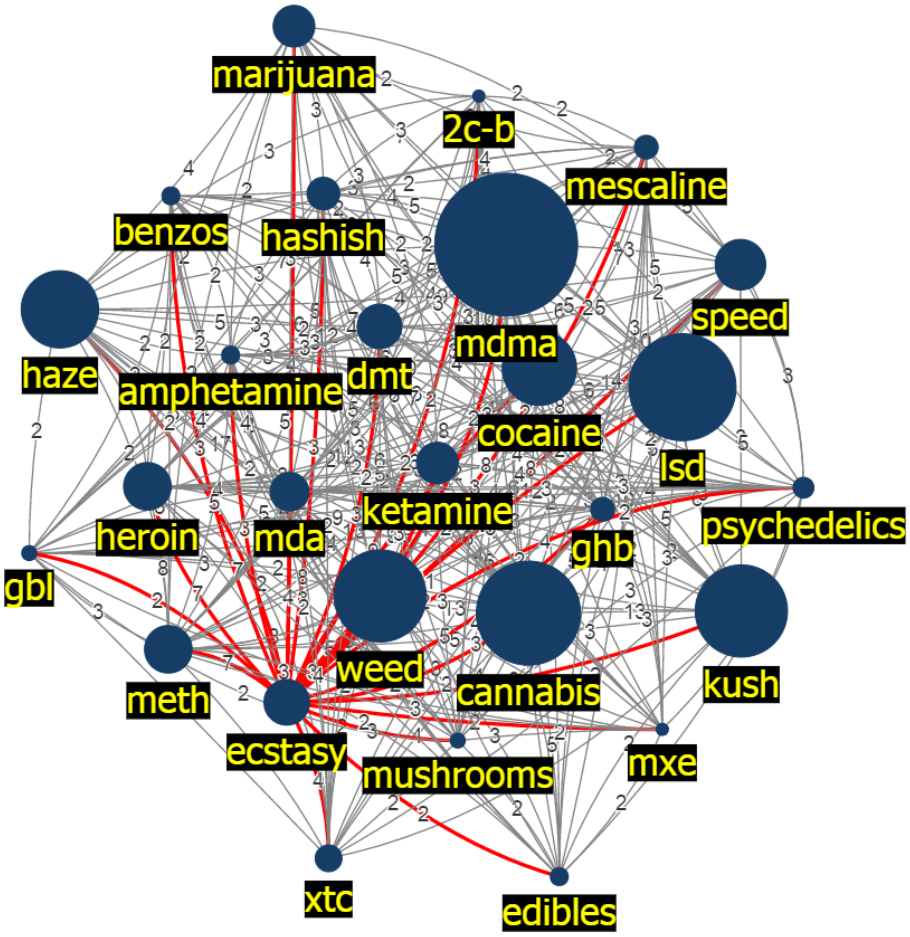


Figura 14: Capa más profunda de PCG. Contiene 27 tipos de drogas o productos farmacéuticos y sobre cada enlace se muestra su peso. Los bordes en rojo corresponden a los enlaces de Éxtasis, el producto más emergente de DUTA.

que son semánticamente similares para ayudar a identificarlo.

En la práctica, el texto de entrada es representado como texto incrustado en un espacio incrustado. Para tratar de resolver la ambigüedad de una palabra desconocida, se compara su representación incrustada con otras representaciones de otras palabras conocidas. Basándose en esa similitud, se obtiene un listado de X palabras similares a la palabra desconocida o palabra de consulta (Chen et al., 2013), de modo que a través de las etiquetas de las palabras de ese listado LDN puede inferir el significado de la palabra desconocida.

El algoritmo LDN consta de dos fases: (i) inicialización, que se activa solo una vez para buscar los vectores de incrustación de las palabras de entrenamiento, y (ii) acumulación, fase que se llama para cada palabra desconocida, con el objetivo de asignarla una suposición inicial. Al entrenar el sistema NER, ambas fases se llaman secuencialmente, pero para predicciones o pruebas, solo se requiere la fase de acumulación. La figura 15 muestra una descripción general del algoritmo y presenta un ejemplo donde se obtiene el vector LDN para la palabra - o token - de consulta *Córdoba*.

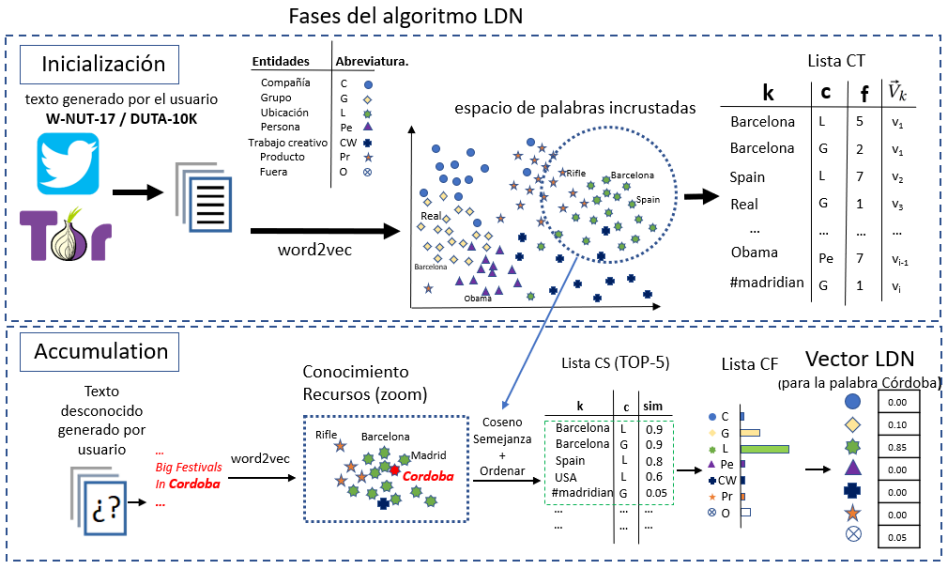


Figura 15: Procedimiento para construir el vector de características Vecinos de distancia local (LDN) dado un espacio de integración del conjunto de entrenamiento y un token de consulta, por ejemplo, *Córdoba* . Primero, en la *Fase de inicialización*, se crea la lista *Category-Token* (CT), donde cada token de entrenamiento tiene una categoría (c), una frecuencia (f) y una inserción vector ($\vec{V}_k$). A continuación, en la *Fase de acumulación*, se calcula la distancia del coseno entre el token de consulta y cada token en el espacio de inserción de entrenamiento. Luego, se clasifican en orden descendente de acuerdo con la puntuación de similitud con el token de consulta y se seleccionan los vecinos superiores X. Además, se calcula la lista *Category-Frequencies* (CF) para los tokens top-X. Finalmente, la información recopilada se alimenta a las fórmulas LDN para evaluar la tendencia del token de consulta. En este ejemplo, se concluye que el token *Córdoba* tiene una probabilidad de 0,85 % para ser etiquetado como Ubicación, 0,10 % como Grupo y 0,05 % como Palabra externa.

4.3.2 Experimentos y resultados

Los experimentos se llevaron a cabo en el conjunto de datos W-NUT-2017 (Derczynski et al., 2017) (Tabla 6)

Tabla 6: Especificaciones del dataset W-NUT-2017

Categoría	#Entrenamiento	#Desarrollo	#Pruebas
Compañías	140	32	60
Grupos	231	38	141
Localizaciones	434	68	125
Personas	546	399	376
Trabajo Creativo	127	100	136
Productos	126	109	117
#Entidades	1604	746	955
#Muestras	3394	1009	1287

En la tabla 7 se puede observar como el mejor resultado es de 47,28 % y 46,96 % para las puntuación de entidad y superficie F1, respectivamente. La tabla compara la propuesta de este trabajo con la de Aguilar y colaboradores, así como la de Akbik y colaboradores. El enfoque mejorado con la función LDN, supera al modelo de Akbik y colaboradores en tres categorías: personas, productos y grupos. Estos resultados hacen que nuestro enfoque sea más adecuado que la propuesta de Akbik y colaboradores para su aplicación de detectar entidades nombradas en la red oscura Tor.

Tabla 7: La puntuación F1 de entidad usando la propuesta **WC-LDN / FG**, Aguilar et al. (2017), y el modelo Akbik et al. (2019) en el conjunto de datos W-NUT-2017 respetando cada categoría. Los valores en negrita se refieren a su superioridad en términos de la métrica de puntuación F1.

Categoría	Aguilar y col. modelo F1 (%)		Akbik y col. modelo F1 (%)		Propuesta W-C-LDN/ FG F1(%)	
	Entidad	**Superficie**	**Entidad**	**Superficie**	**Entidad**	**Superficie**
Compañía	28.57	23.40	**37.84**	37.62	29.93	28.12
Trabajo creativo	11.63	12.05	**30.62**	27.72	24.36	23.85
Grupo	25.10	22.55	29.08	27.87	**34.52**	33.03
Localización	51.90	49.79	**61.70**	59.26	53.83	55.22
Personas	58.95	57.41	64.53	64.63	**65.34**	64.74
Productos	15.38	14.10	25.56	23.21	**41.41**	38.31
Total	41.94	39.73	**49.92**	49.11	47.28	46.96

4.3.3 Versión Extendida del Conjunto de Datos W-NUT-2017

Comparamos nuestro modelo con los modelos Aguilar y colaboradores y Akbik y colaboradores en una versión extendida del conjunto de datos W-NUT-2017 que se creó utilizando datos de productos de DUTA (Tabla 8). Este nuevo dataset extendido se denominó

NUToT (del inglés Noisy User-generated Text on Tor), y contiene 851 muestras adicionales, anotadas manualmente, con 2,970 nuevas entidades. Reentrenando la arquitectura de Aguilar y colaboradores con NUToT, se obtienen los resultados mostrados por la tabla 8, con puntuaciones de entidad y puntuación F1 de superficie de 52,96% y 50,57%, respectivamente. De manera similar al resultado anterior, nuestra propuesta obtuvo resultados del estado del arte en las categorías Personas, Grupos y Productos.

Tabla 8: Comparación entre el modelo Aguilar et al. (2017), el modelo Akbik et al. (2018) y la propuesta de este trabajo.

Categoría	**Aguilar y col. modelo F1 (%)**		**Akbik y col. modelo F1 (%)**		**Propuesta F1 (%)**	
	Entidad	**Superficie**	**Entidad**	**Superficie**	**Entidad**	**Superficie**
Compañía	18.96	19.46	**29.36**	31.91	21.92	20.94
Trabajo creativo	18.86	21.35	**26.58**	23.14	22.83	24.31
Grupos	21.14	22.54	23.38	23.36	**26.39**	27.91
Localizaciones	35.06	33.10	**60.43**	61.00	54.58	54.87
Personas	61.48	61.01	62.05	61.86	**62.15**	62.36
Productos	53.25	53.53	53.64	51.95	**63.15**	61.57
Total	44.73	43.29	52.17	50.53	**52.96**	**50.57**

4.3.4 Conclusiones

En este capítulo, se ha presentado un algoritmo semiautomático para detectar los productos emergentes en la red oscura Tor. El algoritmo, basado en teoría de grafos, representa los productos mediante nodos, y su ofrecimiento simultáneo en el mismo mercado online mediante enlaces, y lo analiza por capas con el algoritmo k-shell. En la capa más profunda, el algoritmo identifica los productos más famosos - MDMA, LSD y Cannabis - y los los emergentes - éxtasis y ketamina. Dichos hallazgos se han validado cuantitativamente contra informes de organizaciones internacionales sobre la temática del consumo de drogas.

Además, en este capítulo de tesis, se ha presentado un nuevo vector de características denominado Local Distance Neighbor (LDN). Este vector está basado en la búsqueda de relaciones semánticas entre una palabra de consulta desconocida y un corpus de palabras conocidas dentro de un espacio incrustado. LDN es capaz de sustituir el uso de diccionarios externos en el ámbito del Reconocimiento de Entidades Nombradas (NER).

Añadiendo LDN al modelo de Aguilar y colaboradores, se han conseguido obtener resultados del estado del arte en tres tipos de entidades nombradas en el conjunto de datos W-NUT-2017: productos, personas y grupos. Además, se ha investigado el reconocimiento de entidades nombradas en los servicios ocultos de la red Tor. Para ello, se propuso (NUToT, del inglés Noisy User-generated Text on Tor), una extensión de W-NUT-2017

con 851 muestras adicionales, anotadas manualmente, con 2,970 nuevas entidades. Se ha utilizado NUToT para reentrenar la arquitectura de Aguilar y colaboradores, observando que incorporando LDN obtiene resultados del estado del arte en la detección de entidades nombradas en Tor, con unas puntuaciones de entidad y F1 de superficie de 52,96 % y 50,57 %, respectivamente.

5 Detección de Influencia

Este capítulo se enfoca en la Unidad de Detección de Influencia (IDU, del inglés Influence Detection Unit) - ver Figura 3. El objetivo es la detección y ordenación de los dominios ocultos más influyentes en la red Tor. Este problema se ha abordado a través de dos enfoques: ordenación basada en enlaces y en contenido.

5.1 Ordenación basada en enlaces

En esta sección se presenta un algoritmo de ordenación de dominios ocultos de Tor basado en hiperenlaces para ordenar los dominios ocultos de la red Tor. Inicialmente se describe DUTA-10K, la extensión del conjunto de datos DUTA sobre el que se va a realizar la organización. A continuación, se describe ToRank y se presentan los resultados obtenidos en DUTA-10K al compararse con otros algoritmos de ordenación de la web superficial.

5.1.1 DUTA-10K: conjunto de datos de dominios ocultos

En un trabajo anterior (Al-Nabki et al., 2017a), se presentó el conjunto de datos DUTA para construir un clasificador supervisado para la Unidad de clasificación de texto (TCU, del inglés Text Classification Unit). Para aumentar el período de muestreo y su utilidad para múltiples trabajos en el ámbito del Procesamiento del Lenguaje Natural (NLP, del inglés, Natural Language Processing), se decidió extender con la preparación de una nueva versión DUTA-10K (Ver Tabla 9).

5.1.2 Metodología

El objetivo de la IDU es la ordenación de los dominios ocultos de Tor, de modo que los más influyentes ocupen las primeras posiciones del ranking. Con este objetivo, en este trabajo se ha presentado ToRank, un algoritmo basado en hiperenlaces que ordena de un modo más eficaz - en términos de disrupción del grafo creado - que los algoritmos utilizados en la Web Superficial: PageRank (Page et al., 1999), HITS (Kleinberg, 1999) y Katz (Katz, 1953).

Usando la teoría de grafos, se ha modelado la red Tor como un grafo de actividades sospechosas (SAG, del inglés Suspicious Activities Graph). Los nodos hacen referencia a los dominios de Tor, y los enlaces entre nodos representan los hipervínculos entre dominios ocultos. Se genera un enlace entre un dominio oculto *A* y otro *B*, siempre que A haga referencia a B al menos una vez, o viceversa. (Ver Figura 16).

ToRank analiza las interconexiones entre los nodos para determinar cuáles son los dominios más influyentes. La figura 17 ilustra una descripción gráfica del algoritmo ToRank.

Tabla 9: Categorías, sub-categorías y número de muestras de DUTA-10K

Tipo de actividad	Actividad	Sub-actividad	#HS
Actividades Sospechosas 20%; 2,013 HS	Pornografía	Infantil	105
		Adulta	148
	Drogas	-	465
	Violencia	Odio	19
		Sicarios	28
		Armas	48
	C. Tarjetas de Crédito		399
	C. Moneda		83
	C. Identificación Personal	Pasaportes	48
		ID	20
		Carné de conducir	4
	Hacking	-	205
	Cryptolocker	-	185
	Mercados	Sospechosos	127
	Servicios	Sospechosos	20
	Foros	Sospechosos	63
	Fraude	-	43
	Tráfico Humano	-	3
Normal Actividades 48%; 5,016 HS	Arte & Música	-	15
	Casino & Juego	-	29
	Servicios	Normal	341
	Criptomoneda	-	868
	Foros	Normal	163
	Mercados	Normal	138
	Librerías & Libros	-	45
	Personal	-	616
	Política	-	12
	Religión	-	21
	Almacenamiento & Software	File-Sharing	205
		Carpetas	99
		Motores de búsqueda	92
		Servidores	1,416
		Software	411
		Directorios	133
	Redes sociales	Blogs	219
		Chats	79
		Email	72
		Noticias	42
Desconocidos 32%; 3,338 HS	Caídos	-	864
	Vacíos	-	1,653
	Bloqueados	-	821
Total			10,367

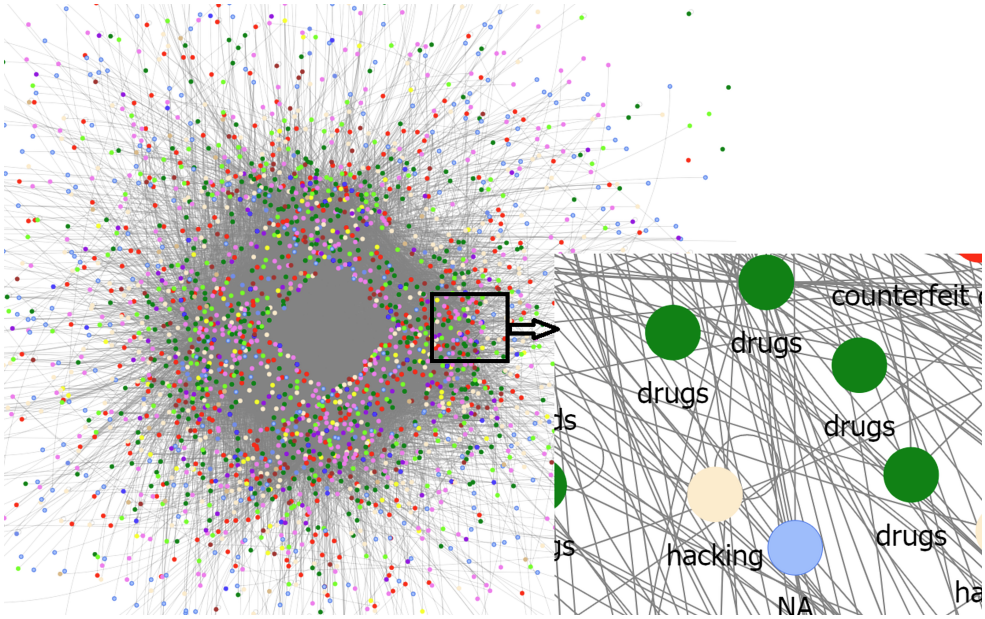


Figura 16: Captura de SAG_{All} para DUTA-10K, donde cada nodo es un servicio oculto, y cada color una actividad o categoría en DUTA-10K. Las líneas grises reflejan los hipervínculos entre los dominios. Debido a la densidad de las conexiones, el centro del grafo aparece como una "macha"gris.

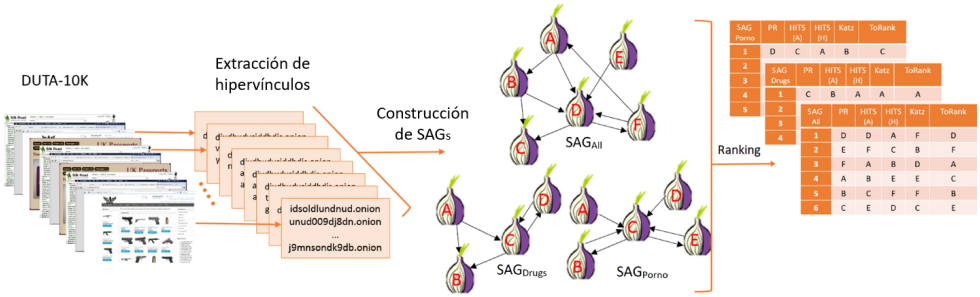


Figura 17: Procedimiento de detección y ordenación de dominios ocultos. Primero, se extraen los hipervínculos del conjunto de datos. Luego, para cada categoría de actividades sospechosas, se construye un grafo de Actividades Sospechosas (SAG), como por ejemplo SAG_{Porno} y SAG_{Drogas}. Además, se crea otro grafo para todas las actividades, denominado SAG_{Todos}. En última instancia, los algoritmos de clasificación se aplican para ordenar y detectar el dominio oculto más influyente.

Algoritmo ToRank. El diseño de la red Tor impide la ordenación de dominios en base a su tráfico de red (Arma, 2019). ToRank supera esta limitación, dado que utiliza la conectividad entre dominios a través de sus hipervínculos en lugar de medir el tráfico.

El algoritmo consta de dos fases: inicialización y actualización de pesos. El algoritmo

comienza asignando un peso inicial a cada nodo. A continuación, comienza un proceso iterativo por el que se acumula el peso de los nodos seguidores de un nodo n y el peso de sus seguidores, los nodos a los que hace referencia n. Finalmente, los pesos se calculan nuevamente para cada nodo. Se asigna un valor de rango, de acuerdo con el peso del nodo, de modo que el peso más alto corresponde al rango más alto y, consecuentemente, el dominio más influyente. Se recuerda que la influencia se mide como la capacidad de disrupción provocada al grafo completo si ese dominio se eliminara.

5.1.3 Resultados Experimentales

En este trabajo, se ha comparado ToRank con tres algoritmos de ordenación frecuentemente utilizados en la Web Superficial: PageRank (Page et al., 1999), HITS (Kleinberg, 1999) y Katz (Katz, 1953).

Siguiendo literatura previa (Booker, 2012; Fronzetti Colladon and Gloor, 2018; Fronzetti Colladon and Vagaggini, 2017), se ha utilizado el criterio de densidad para evaluar el rendimiento del algoritmo (Ver Figuras 18. Partiendo del grafo SAG, el criterio de densidad elimina, uno a uno e iterativamente, los nodos cuya posición es la más alta en el ranking, y en cada ciclo, se evalúa la densidad del gráfico. El iterador se detiene cuando el grafo está completamente desconectado y la densidad es cero. Se considera que el algoritmo de ordenación que alcanza el área más baja bajo la Curva de densidad gráfica (GDC, del inglés Graph Density Curve) corresponde al algoritmo que mide mejor la influencia de un dominio dentro de la red Tor.

La figura 18 muestra cinco diferentes curvas de densidad de gráficos (GDC) de SAG_{All} para los cuatro algoritmos de clasificación [9]. ToRank supera a los otros métodos porque alcanza el área más baja bajo el GDC, con un valor de 1,31, presentando también una disminución homogénea en la curva de densidad. Por el contrario, PageRank sufre un aumento repentino en su curva, luego una fuerte disminución, que produce el mayor GDC de 2,07.

Fue sorprendente que el algoritmo de clasificación más popular para la web de superficie, PageRank, funcione mal para este problema de acuerdo con la métrica de análisis de densidad gráfica. Descubrimos que PageRank produjo el área más grande bajo la curva de densidad, especialmente después de eliminar entre 10 % y 20 % de los nodos mejor clasificados. La razón de esto es que PageRank asigna altos rangos a los nodos de baja conectividad, y durante la evaluación iterativa, esos nodos se eliminaron primero, lo que resulta en una disminución en el número de nodos sin un impacto significativo en la conectividad del gráfico. Por lo tanto, el número de aristas se mantiene alto mientras que el recuento de nodos disminuye, lo que conduce a un incremento en la curva de densidad y un alto valor de GDC.

Sin embargo, a pesar del éxito del algoritmo ToRank en la detección y clasificación de los nodos influyentes en Tor, nuestro método falla al evaluar los dominios de cebolla que

[9] el algoritmo HITS produce dos curvas, una para las autoridades y otra para los centros

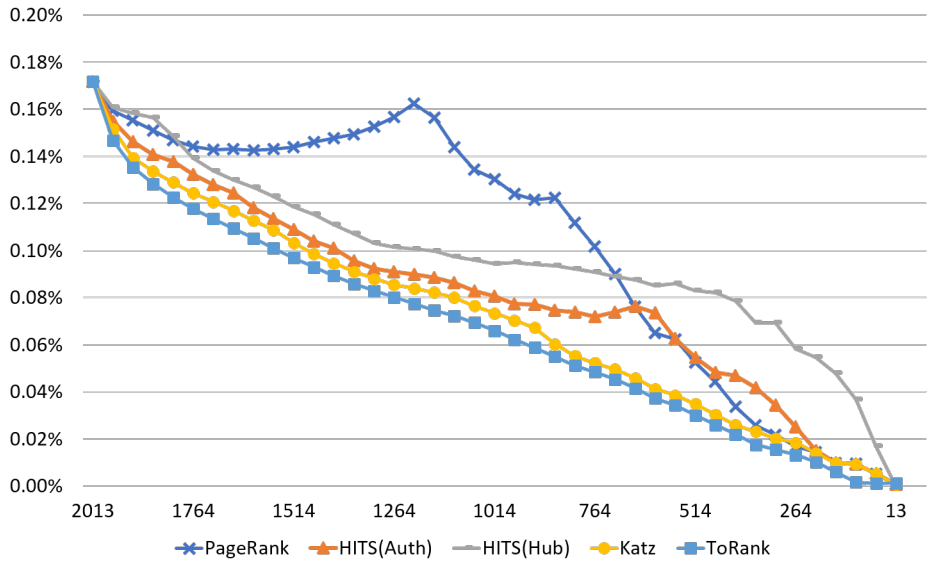


Figura 18: Análisis de densidad para SAG_{All} de DUTA-10K. El eje X se refiere al número actual de nodos, mientras que el eje Y la densidad gráfica correspondiente. Se puede ver que ToRank supera al otro algoritmo ya que alcanza el GDC más bajo.

están aislados sin hipervínculos entrantes o salientes. Afortunadamente, este problema apenas tiene un impacto en nuestro problema porque en SAG_{All} solo hay dominios de cebolla 94 que no tienen hipervínculos hacia o desde HS dentro de Tor, lo que cuenta solo 3 % del total .

5.2 Ordenación basada en contenido

En esta sección se presenta un método de ordenación basado en contenido, que extrae características del contenido de los dominios y las utiliza para entrenar un modelo de clasificación supervisado. El principal beneficio de utilizar un enfoque de clasificación supervisada basada en contenido es su capacidad de adaptar criterios de clasificación predefinidos. Concretamente, empleamos un algoritmo de Aprendizaje para Ordenar (LtR, del inglés Learning to Rank) con el objetivo de capturar las características de una lista clasificada dada y para mapear el rango aprendido en una nueva lista de elementos sin clasificar.

5.2.1 Metodología

El modelo de clasificación tiene dos componentes: 1) *Unidad de modelado de servicios ocultos (HSMU, del inglés Hidden Service Modeling Unit)*, que analiza y extrae característi-

cas de un dominio oculto, y 2) la *Unidad supervisada de aprendizaje para clasificar (SLRU, del inglés Supervised Learning to Rank Unit)* que aprende una función para ordenar una colección de dominios de acuerdo con un patrón capturado de muestras clasificadas previamente, es decir, un conjunto de entrenamiento (ver Figura 19).

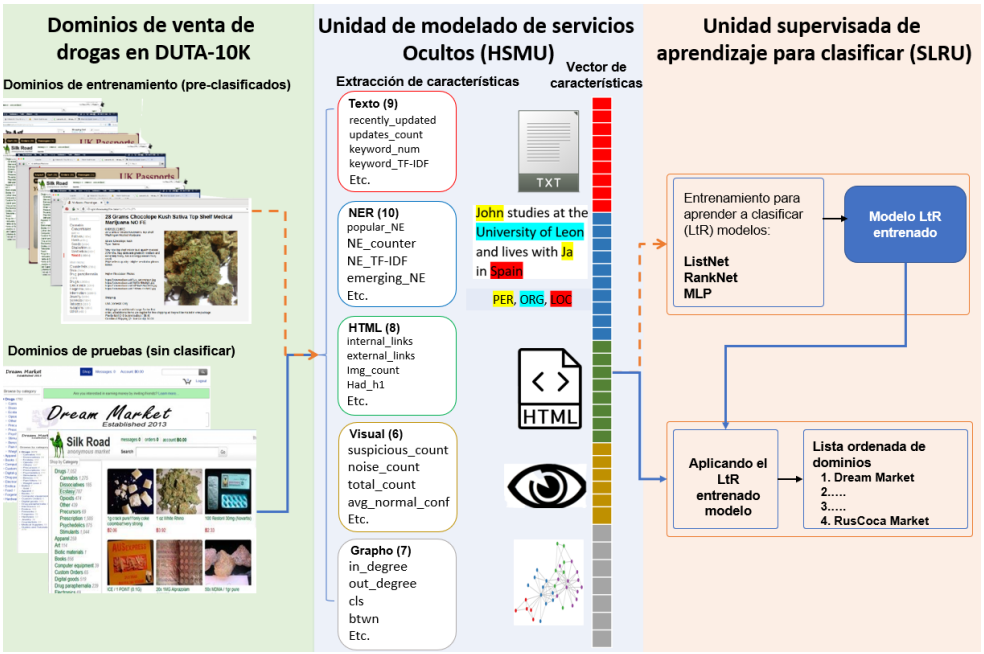


Figura 19: Se recomienda el visionado en color. Vista general del modelo propuesto para ordenar y detectar los dominios ocultos de Tor influyentes en base al contenido de los mismos. Las flechas anaranjadas punteadas indican la fase de entrenamiento del sistema, mientras que las flechas azules continuas indican la fase de pruebas o evaluación.

Dado un dominio $d_i \in D$, donde D es un conjunto de dominios de cebolla o servicios ocultos (HS, del inglés Hidden Services), la HSMU analiza d_i para extraer características que pertenecen a cinco categorías diferentes: texto, entidades con nombre (NER, del inglés Named Entity Recognition), HTML, contenido visual y grapho del red. Luego, la HSMU codifica esas características en valores numéricos que representan las HS analizadas. La tabla 10 resume las características que se extraen de los dominios ocultos a través de HSMU.

Para aprender un orden óptimo de los dominios ocultos a través de sus descriptores, se ha adoptado un enfoque de Aprendizaje para Ordenar (LtR, del inglés Learning to Rank) adaptado del campo Recuperación de información (IR, del inglés Information Retrieval). En la literatura de LtR, existen tres estrategias principales para entrenar un sistema de ordenar: *Pointwise*, *Pairwise* y *Listwise*. En este documento de tesis, se han explorado las tres estrategias con el objetivo de determinar cual es la más adecuada para la

Tabla 10: Resumen de las características extraídas por HSMU

Característica clase	Característica contar	Característica fuente	Característica nombre
Texto	9	Fecha y hora	- recently_updated - updates_count
		HS nombre	- address_words_count - address_letters_count
		Tasa de clones	- clones_rate
		TF-IDF vectorizador	- keyword_num - keyword_TF-IDF - keyword_avg_weight - keyword_to_total
entidades con nombre (NER)	10	NE popular	- popular_NE (x)
		Número total de NE	- NE_counter
		TF-IDF NE	- NE_TF-IDF
		TF-IDF popular NE	- popular_NE_TF-IDF
		Emergente NE	- emerging_NE
HTML Margen	8	Hipervínculos internos	- internal_links
		Hipervínculos externos	- external_links
		Recuento de etiquetas de imagen	- img_count
		Login y Contraseña	- needs_credential
		Título de dominio	- has-title
		Encabezado de dominio	- has_H1
		TF-IDF Título y encabezado	- TF-IDF_title_H1
		TF-IDF Imagen Alternativas	- TF-IDF_alt
Contenido visual	6	Cuenta de imágenes	- suspicious_count - noise_count - total_count
		Clasificación promedio	- avg_suspicious_conf
		Count	- avg_normal_conf
		Clase mayoritaria	- suspicious_majority
Estructura de red (grapho)	7	En grado	- in_degree
		Fuera de grado	- out_degree
		Medidas de centralidad	- cls (cercanía) - btwn (intermediación) - eigvec (eigenvector)
		ToRank valor	- ToRank_rank - ToRank_top_X
Características totales	40		

ordenación de dominios ocultos en Tor. Para cada estrategia, adoptamos un algoritmo: perceptrón multicapa (MLP, del inglés Multilayer Perceptron), RankNet y ListNet, respectivamente.

5.2.2 Experimentación y resultados

Por lo general, la ganancia acumulada normalizada con descuento (NDCG, del inglés Normalized Discounted Cumulative Gain) (Järvelin and Kekäläinen, 2000; Manning et al.,

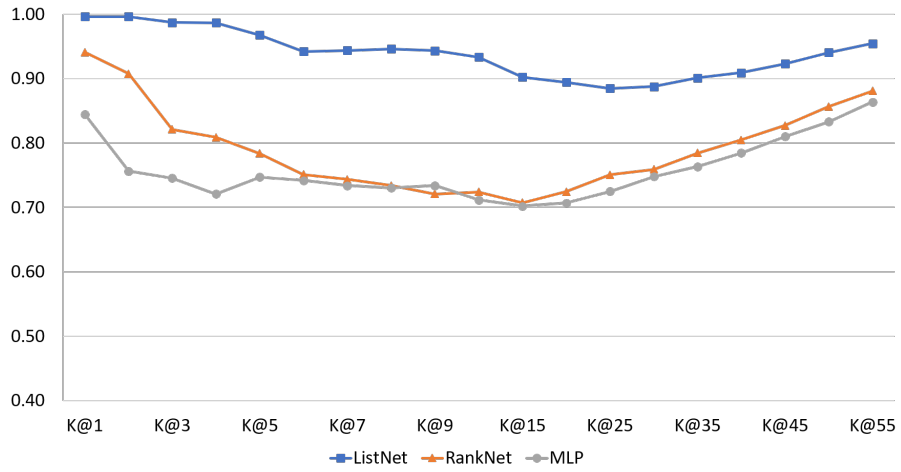


Figura 20: Comparación entre estrategias LtR *Pointwise*, *Pairwise* y *Listwise* contra valores múltiples de $NDCG@K$. El eje X se refiere al valor K y el eje Y indica la puntuación $NDCG$ de los algoritmos, obtenidos en cada valor de K.

2010) se utiliza para evaluar problemas de ordenación. Por lo general, usamos $NDCG@K$ donde K se refiere a los elementos con la orden K más alta.

Para evaluar el método propuesto, se aborda el caso de uso de los dominios ocultos que trafican con drogas. Para ello, se seleccionaron 290 dominios del dataset DUTA-10K (Al-Nabki et al., 2019c) con la categoría *Drogas*. A continuación, para construir un conjunto de datos de referencia, 13 personas respondieron un cuestionario de preguntas SI/NO, a través de las cuales se realizó una ordenación manual de los mismos. Este conjunto de dominios etiquetado se utilizó como conjunto de entrenamiento para el algoritmo LtR.

Se compararon las tres estrategias LtR usando la métrica NDCG, como se muestra en la Figura 20.

La figura 20 muestra que $NDCG@1$ de ListNet es igual a uno, lo que significa que el algoritmo clasificó correctamente los primeros dominios evaluados, siguiendo el patrón del conjunto de datos de referencia. A partir de $NDCG@4$, la curva comienza a decaer; sin embargo, el valor más bajo de $NDCG$ fue de $0,88$ con K igual a 25.

Se comparó la mejor estrategia LtR, i.e. ListNet, contra algoritmos basados en hiperenlaces, como se muestra en la Figura 21. ListNet supera ampliamente todos los algoritmos de clasificación basados en enlaces. Incluso a través de su enfoque más débil, MLP, con un $NDCG@10$ de 0,71, supera al mejor algoritmo de clasificación basado en enlaces, ToRank, que obtiene un $NDCG@10$ de 0,69. Este resultado enfatiza la importancia de considerar el contenido de los dominios en lugar de solo su conectividad de hipervínculos.

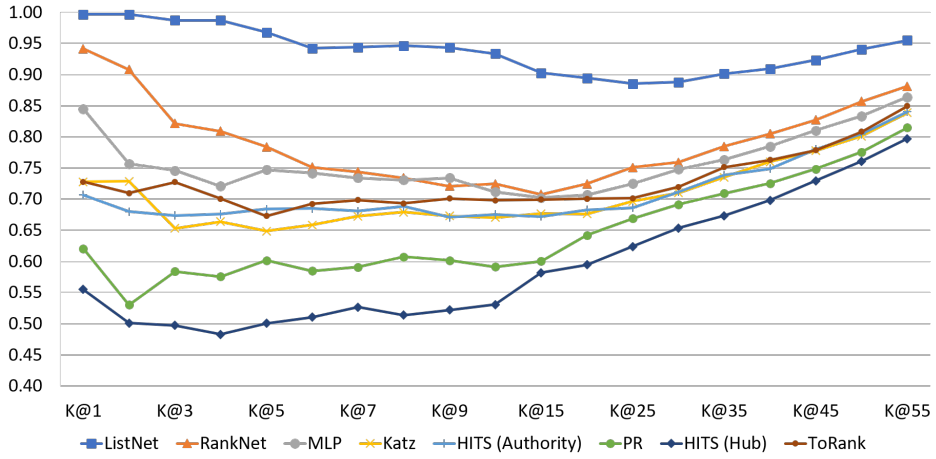


Figura 21: Comparación entre los algoritmos de ordenación basados en contenido y basados en hiperenlaces con respecto a valores múltiples de K. El eje horizontal se refiere al valor de K y el vertical indica las puntuaciones de $NDCG$ que los algoritmos obtuvieron en cada valor de K.

5.2.3 Conclusiones

En este capítulo de tesis se han presentado dos algoritmos de ordenación para detectar los dominios ocultos más influyentes en la red Tor: basados en enlaces y basados en contenido. Todos los experimentos de esta sección se han realizado en DUTA-10K, una versión extendida de DUTA, que contiene 10,367 muestras etiquetadas en 25 categorías.

Basado en enlaces, se presenta ToRank, un algoritmo de ordenación que obtiene mejores resultados que PageRank, HITS y Katz cuando se aplica a la ordenación de dominios ocultos en Tor. ToRank se evaluó cuantitativamente utilizando la Curva de densidad gráfica (GDC, del inglés Graph Density Curve) - más bajo es mejor - obteniendo 1,31, en comparación con Katz, $\text{HITS}_{\text{Auth}}$, HITS_{Hub} y PageRank que obtuvieron 1,41, 1,63, 1,96 y 2,07 respectivamente.

Con respecto al método basado en contenido, se ha utilizado el esquema de Aprendiendo a ordenar (LtR, del inglés Learning to Rank) para detectar los dominios ocultos más influyentes en la red oscura Tor. El algoritmo propuesto consta de una Unidad de Modelado de Servicios Ocultos (HSMU , del inglés Hidden Service Modeling Unit) y una Unidad Supervisada de Aprendizaje para Clasificar (SLRU, del inglés Supervised Learning to Rank Unit). El HSMU representa un dominio oculto con 40 características extraídas de cinco recursos diferentes: el texto visible del usuario del dominio, el marcado HTML de la página web, las entidades nombradas en el texto del dominio, el contenido visual y la estructura de la red Tor. La SLRU aprende a clasificar los dominios utilizando un enfoque LtR. Para entrenar el modelo LtR, se creó un conjunto de datos ordenados manualmente de 290 dominios ocultos relacionados con las drogas. Utilizando la métrica Ganancia

acumulativa de descuento normalizado (*NDCG*, del inglés Normalized Discount Cumulative Gain) se compararon tres esquemas de LtR: Perceptrón multicapa, RankNet y ListNet. La experimentación demostró que ListNet obtiene los mejores resultados, con un *NDCG*@10 de 0,95.

6 Conclusiones de la Tesis y Trabajo Futuro

En cumplimiento del punto 3º del artículo 19 del Reglamento de las enseñanzas oficiales de doctorado y del título de doctor de la Universidad de León, aprobado en Consejo de Gobierno el 25/9/2012, las conclusiones y líneas de trabajo futuro de esta tesis, han sido presentadas en el capítulo 7.