Applied Intelligence
https://doi.org/10.1007/s10489-023-04671-9

Supervised ranking approach to identify infLuential websites in the darknet

MHD Wesam Al Nabki[1,2] · Eduardo Fidalgo[1,2] · Enrique Alegre[1,2] · Deisy Chaves[1,2]

Accepted: 25 April 2023

Abstract
The anonymity and high security of the Tor network allow it to host a significant amount of criminal activities. Some Tor domains attract more traffic than others, as they offer better products or services to their customers. Detecting the most influential domains in Tor can help detect serious criminal activities. Therefore, in this paper, we present a novel supervised ranking framework for detecting the most influential domains. Our approach represents each domain with 40 features extracted from five sources: text, named entities, HTML markup, network topology, and visual content to train the learning-to-rank (LtR) scheme to sort the domains based on user-defined criteria. We experimented on a subset of 290 manually ranked drug-related websites from Tor and obtained the following results. First, among the explored LtR schemes, the listwise approach outperforms the benchmarked methods with an NDCG of 0.93 for the top-10 ranked domains. Second, we quantitatively proved that our framework surpasses the link-based ranking techniques. Third, we observed that using the user-visible text feature can obtain comparable performance to all the features with a decrease of 0.02 at NDCG@5. The proposed framework might support law enforcement agencies in detecting the most influential domains related to possible suspicious activities.

Keywords Supervised learning · Learning-to-rank · Influence detection · Feature extraction · Darknet · Tor Hidden services

1 Introduction

The onion router (Tor) network, known as one of the most famous darknet networks, gives end users a high level of privacy and anonymity. The Tor project was proposed in the mid-1990 s by US military researchers to secure intelligence communications. However, a few years later, as part of their secret strategy, they made the Tor project available to the public [1]. Currently, onion domains are proliferating rapidly, and the latest statistics stated by the onion metrics website[1] show a significant increase in the number of domains, exceeding 500, 000.

There are many legal uses for the Tor network, such as personal blogs, news domains, and discussion forums [2, 3]. However, due to its level of anonymity, Tor darknet is being exploited by services traders, allowing them to promote their products freely, including but not limited to child sexual abuse (CSA) [4], drug trading [4–9], and counterfeit personal identifications [10–12]. Moreover, the high level of privacy and anonymity provided by the Tor network obstructed the authorities' monitoring tools from controlling the content or even identifying the IP address of the hosts behind any suspicious service. To address this problem, we collaborate with the Spanish National Cybersecurity Institute (INCIBE[2]) to develop tools that can ease the task of monitoring the Tor darknet and detecting existing or new suspicious content. The proposed Tor monitoring framework is summarized in Fig. 1.

Eduardo Fidalgo, Enrique Alegre and Deisy Chaves contributed equally to this work

✉ MHD Wesam Al Nabki
wesam.alnabki@unileon.es

Eduardo Fidalgo
eduardo.fidalgo@unileon.es

Enrique Alegre
enrique.alegre@unileon.es

Deisy Chaves
deisy.chaves@unileon.es

[1] Department of Electrical, Systems and Automation, Universidad de León, León, Spain

[2] Researcher at INCIBE, Spanish National Cybersecurity Institute, León, Spain

[1] https://metrics.torproject.org/hidserv-dir-onions-seen.html

[2] In Spanish, it stands for "Instituto Nacional de Ciberseguridad de España"

Published online: 04 July 2023

The first module of our Tor monitoring tool is an onion domain classifier, which detects and isolates categories of suspicious onion domains. For this task, we used the supervised text classifier already presented in [12], which categorizes hidden services (HS) into eight classes: pornography, cryptocurrency, counterfeit credit cards, drugs, violence, hacking, counterfeit money, and counterfeit personal identification, including driving-licence, identification, and passport.

The second module, which is the focus of this study, addresses the problem of *ranking* the HS that were classified as suspicious. Once they are ranked, a police officer can prioritize the work by focusing on the most influential onion domains. In our previous work [2], we presented ToRank, a ranking algorithm to sort onion domains by analysing the connectivity of their hyperlinks, a linked-based approach. In this work, we propose a content-based approach for ranking, including features extracted from the text, named entities, HTML code, domain position, and visual content, as explained in the following sections.

One of the difficulties we faced was defining the influence of a given onion domain. The literature is rich with definitions of the term *influencers*. In the social network analysis (SNA) field, it denotes highly participating members [13], key members [14], members who encourage others to participate [15], or members who can change the perspective of others using a sentiment analysis algorithm [16]. In the terrorist network analysis field, *influencers* refer to people who have connectivity with the majority of the network members, such as financial managers [17]. Furthermore, in the viral marketing discipline, it stands for opinion leaders who can persuade their audience to purchase or subscribe to a product or a service [18]. In this paper, we borrow this definition of *influencers* to refer to onion domains that can attract customers to visit their websites and potentially buy their products. The attractiveness of the onion domain website is subjective, and it can be determined through its public reputation among buyers, its confidentiality and reliability, or even the service quality it offers [19]. However, ranking onion domains considering these subjective factors is difficult because they depend heavily on customers' opinions and impressions [20].

This work overcomes this difficulty by presenting a supervised ranking approach to sort onion domains based on various features extracted from the content and structure of the onion domains. The ranking function learns how to map between human opinion, i.e., the ground-truth order, and the extracted features from the domains. Therefore, it assigns each domain a score that reflects its influence, whereas the higher the score is, the more significant influence it has.

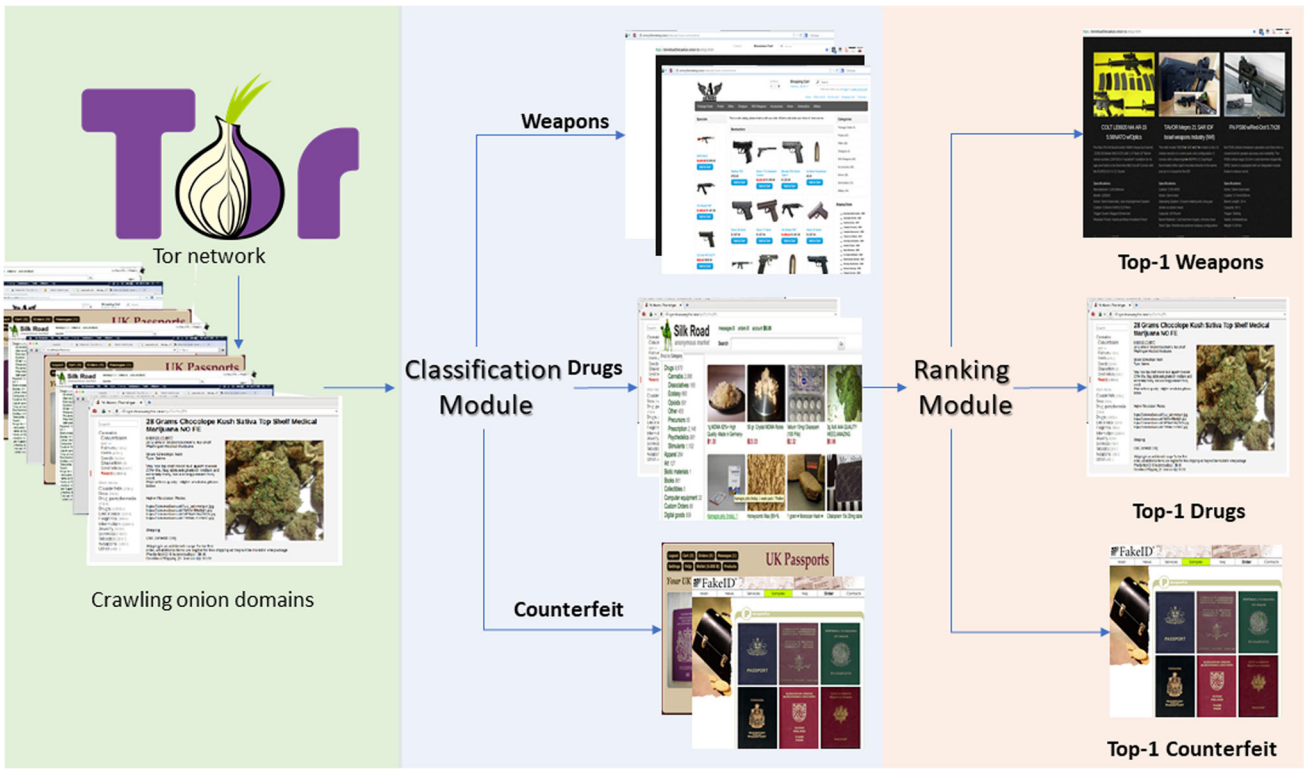

Fig. 1 Overview of the Tor network monitoring tool. After crawling onion domains from the Tor network, the classification module [12] classifies them according to their crime category, and the proposed ranking module sorts the domains per category following their influence level

Thanks to the text classification module [12], the proposed ranking framework works at the activity level of the domains and detects the influential HS in each category of domains. Hence, this paper aims to answer the following question: *What are the most influential onion domains in a determined area of activity?*

Answering this question can improve the capability of LEAs to keep a close eye on suspicious domains that are more influential by concentrating their efforts on monitoring them. Moreover, if an LEA takes a suspicious domain down, the proposed ranking module can recognize it even if it was hosted under a new address if it still hosts the same content. Additionally, when a new domain is released and hosts suspicious content similar to a previously recognized influential domain, our ranking module can capture it before becoming popular among Tor users. Therefore, LEAs can strike suspicious domains preemptively.

A straightforward strategy for detecting influential onion domains is to sort them by the number of client requests, i.e., analysing the network traffic. However, the design of the Tor network is oriented to preventing this behaviour [21]. Chaabane et al. [22] conducted a deep analysis of Tor network traffic by establishing six *exit nodes* distributed worldwide with the default exit policy. Nonetheless, this approach cannot assess the traffic of onion domains that are not reachable through these exit nodes. Furthermore, it can be risky because the Tor network users can reach any onion domain, regardless of its legality, through the IP addresses of the machines dedicated to that purpose. Biryukov et al. [23] attempted to exploit the concept of *entry guard nodes* [24] to deanonymize clients of a Tor hidden service. However, this proposal will not be feasible as soon as the vulnerability is fixed.

Another strategy reported in the literature to detect influential onion domains is using a link-based ranking algorithm such as ToRank [2], PageRank [25], hyperlink-induced topic search (HITS) [26], or Katz [27]. We explored link-based ranking algorithms in our previous work [2] and concluded that the main drawback of this approach lies in its dependency on hyperlink connectivity between onion domains [28]. Hence, if an influential but isolated domain exists in the network, this technique cannot recognize it as an essential item.

This paper presents an alternative approach for detecting influential onion domains by extracting features from domain content to train a learning-to-rank (LtR) algorithm [29–31]. In particular, given a list of HSs, our model ranks onion domains based on two key steps: content feature extraction and onion domain ranking. First, we represent each onion domain by forty element feature vectors extracted from five different resources: 1) the textual content of the domain, 2) the textual named entities (NEs) in the user-visible text such as product names and organization names, 3) the HTML markup code by taking advantage of specific HTML tags, 4) the visual content such as the images exposed in the domain, and finally, 5) the position of the targeted onion domain in the Tor network topology. Second, the extracted features are cleaned and normalized to train a ranking function using the LtR approach to rank the domains and to propose the top-k domains as the most influential.

The ranking problem addressed in this work is close to the information retrieval (IR) field but with a significant difference. Both retrieve a ranked list of elements similar to how search engines work. For example, the Google search engine considers more than 200 factors to generate a ranked list of websites concerning a query [32]. However, in the context of our problem, we do not have a search term to order the results accordingly. Instead, our objective is to rank the domains based on a virtual query: *What are the most attractive onion domains in a determined area of activities?* Therefore, this model adopts IR to solve the problem of ranking and detecting the most influential onion domain in the Tor network without having an available search term.

Nevertheless, the proposed framework is not restricted to ranking the onion domains of the Tor network. It can be generalized and adapted to different areas with slight modifications in the feature vector, such as document ranking, web pages of the surface web, or users in a social network, among others.

The main contributions of this work are as follows:

- We propose a novel framework to *rank* the onion domains and *detect* the most influential domains. Our strategy exploits five groups of features extracted from the Tor network via a hidden service modelling unit (HSMU). We used the extracted features to train the supervised learning-to-rank unit (SLRU). Our approach outperforms link-based ranking techniques, such as ToRank, PageRank, HITS, and Katz, when tested on samples of onion domains related to drug marketing (Fig. 2).
- We propose 40 features extracted from five resources: 1) user-visible text, 2) textual NEs, 3) the HTML markup code, 4) the visual content, and 5) features drawn from the Tor network topology. In particular, we address the effects of representing an onion domain by several variations in features on the ranking framework. We identify the most efficient combination of features compared to their cost of extraction in terms of the prediction time and the resources needed to build the feature extraction model.
- We evaluated our approach on a manually ranked dataset of 290 domains extracted from the Tor network and dedicated to trading illegal drugs. Each onion domain was judged by three members and received its influence score based on the majority voting strategy.

The rest of the paper is organized as follows. Section 2 summarizes the related work. Next, in Sect. 3, we present a procedure followed to build the dataset. Section 4 introduces

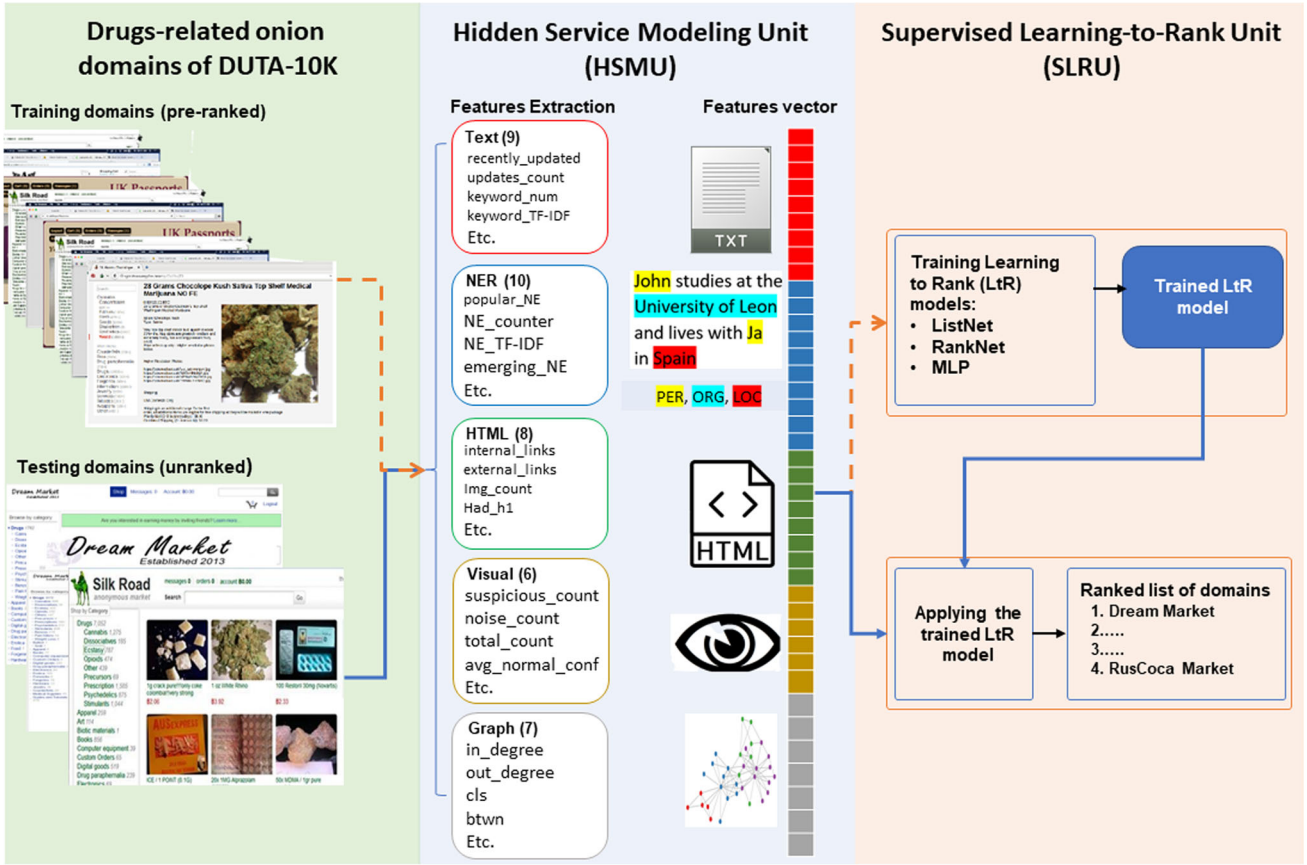

Fig. 2 A general view of the proposed framework for ranking and detecting the influential onion domains in the Tor network. The dashed orange arrows indicate the training pipeline of the system, while the solid blue arrows indicate the testing/production phase

the proposed ranking framework, including its main components. Section 5 describes the experimental settings and the configuration of the framework units. Section 6 addresses a case study to test the effectiveness of the proposed framework in a real-case scenario. Finally, Sect. 7 presents the main conclusions of this work and introduces other approaches that we are planning to explore in the future.

2 Related work

Several researchers have analysed suspicious activities on the darknet, including illicit drug markets [33–35], terrorist activities [36, 37], arms smuggling, violence, and cybercrime [6, 12, 38]. However, a few have focused on detecting the most influential domains.

Some have used social network analysis (SNA) techniques to mine networks. Chen et al. [39] conducted a comprehensive exploration of terrorist organizations to examine the robustness of their networks against attacks. They simulated the attacks by removing the items with the highest in-degree or betweenness scores [40]. Al-Nabki et al. [2] proposed an algorithm called ToRank to rank and detect the most influential domains in the Tor network. ToRank represents the Tor network by a directed graph of nodes and edges; the most influential nodes are those whose removal would reduce the nodes' connectivity. However, link-based approaches fail to evaluate isolated nodes that do not connect to the rest of the community.

Choi et al. [41] built hand-crafted features to identify key cyberbullies in social networks. They collected features from various network centrality measures, including degree centrality, betweenness centrality, closeness centrality, and PageRank, to analyse the connectivity of community members. Additionally, they used the Losada ratio, a ratio of positive-to-negative text sentiment, and a cyberbullying index, a ratio of insulting words that appear in the text. Similarly, [42] addressed the Twitter social network to identify key actors using the same network centrality measures along with sentiment analysis.

Anwar et al. [43] presented a hybrid algorithm to detect the influential leaders of radical groups in darknet forums. Their proposal is based on mining the content of the user's profiles and their historical posts to extract textual features representing their radicalness. Then, they incorporated the obtained features in a customized link-based ranking algorithm based

on PageRank [25] to build a ranked list of radically influential users.

A different perspective was carried out by Biryukov et al. [23], who exploited the *entry guard node* concept [24] to deanonymize clients of an onion domain in the Tor network. The popularity of onion domains in the Tor network is estimated by measuring its incoming traffic; nevertheless, this approach will not be feasible when the vulnerability is fixed.

The LtR framework has been used widely in the IR domain [44–47]. Li et al. [48] proposed an algorithm to help software developers deal with unfamiliar application programming interfaces (APIs) by offering software documentation recommendations and training an LtR model with 22 features extracted from four resources. Agichtein et al. [49] employed the RankNet algorithm to leverage search engine results by incorporating features from user behaviour. Wang et al. [50] presented an LtR-based framework to rank input parameter values of online forms. They used 6 categories of features extracted from user contexts and patterns of user inputs. Moreover, LtR was used for mining social networks [51–53] or to detect and rank critical events in Twitter social networks [54].

3 Dataset construction

Darknet Usage Text Addresses 10K (DUTA-10K) is a publicly available dataset proposed by Al-Nabki et al. [2] that contains 10, 367 onion domains from the Tor network distributed into 25 categories. In this paper, we consider the domains of the category *Drugs* as a case study to rank its domains using the proposed ranking framework. This category contains drug manufacturing, cultivation, and marketing topics, as well as drug forums and discussion groups. Out of 465 drug domains in DUTA-10K, we selected only English language domains, which totalled 290 domains. This ranking approach could be adapted to any collection of web domains, but we selected the drug-related domains owing to their high popularity in the Tor network. In addition, our approach is more comprehensive than HS ranking. It can be extended to document ranking or influence detection in social networks.

To annotate the dataset, thirteen people, including the authors, manually ranked the 290 drug-related domains. To secure consistent ranking criteria among the annotators, we created a unified questionnaire of 23 subjective binary questions (Table 1) that the annotators answered for each domain. The ground-truth is built in a pointwise manner, assigning an annotator a value to each domain, coming from answering every question with a 1 or 0, corresponding to *Yes* or *No*, respectively.

We repeated the process three times, assigning each annotator a new batch of approximately 23 domains every time. 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 domain into a single vector of 23 dimensions that corresponded with the number of questions. Finally, we summed the answers of each domain to obtain a score value for each domain, representing a ground-truth rank while training. In this context, a higher score means a more significant influence.

4 Proposed ranking framework

This work presents a ranking framework for automatically ranking hidden services (HSs), i.e., the Tor network websites, according to user-defined criteria captured from a training set (Fig. 2). Our design has two components: 1) the *hidden service modelling unit (HSMU)* for extracting features from a given website domain in the Tor network and 2) a *supervised learning-to-rank unit (SLRU)* that trains a supervised ranking model.

4.1 Hidden service modelling unit

Given a hidden service domain $d_i \in D$ collected from the Tor network D, which is represented in the HSMU by a feature vector extracted from sources: 1) the text, 2) the NEs, 3) the HTML code, 4) the visual content, and 5) the topology of D and the position of d_i in D.

4.1.1 Text features

Given the text of d_i, we extract nine features from the following four sources.

Date and Time 1 binary feature to indicate whether d_i has been updated recently. If the most recent date is close to today's date, it is marked as "updated" or "obsolete" otherwise. Additionally, we count date patterns within a date window to measure the number of recent changes in d_i. We refer to these two features as *recently_updated* and *update_counts*, respectively.

Website URL 1 URL address of an onion domain[3] consists of 16 characters generated using a 1024-bit RSA key pair, and the public key is hashed using the SHA-1 algorithm. Then, the first 80 bytes of the hash are encoded using a Base32 encoder, and the suffix ".onion" is added. Therefore, most generated onion domain URLs do not involve readable or meaningful words and can be seen as a random sequence of

[3] This paper uses onion domains of version 2 since the experimented dataset is on the same version https://support.torproject.org/onionservices/v2-deprecation/

Table 1 Binary questionnaire used to build a ground-truth rank for the drug onion domains

Qestions	
- 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?	- Does the domain name has a meaning?
- Offers reward or discount?	- Does the majority of the products are illegal?
- Sell more than 10 products?	- Still accessible in the 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 product's description?
- Has more than 10 subpages?	

16 characters. However, there are open-source tools capable of generating customized addresses, such as Shallot.[4] These tools allow the onion domain address to include attractive, catchy words, such as *cocaine* or *LSD*, for a hidden service selling illegal drugs. The main challenge here is the exponential time required to customize domain names; for example, customizing seven characters takes one day of machine time, while customizing 10 characters requires 40 years of processing. We used a probabilistic model based on English Wikipedia unigram frequencies to extract the URL features. The model splits concatenated letters into potential words, thanks to the Wordninja tool.[5] For the URL words, we obtain two features: (i) the number of human-readable words identified using the Nostril tool [55] and (ii) the number of their letters. We name these features *URL_word_count* and *URL_letter_count*, respectively.

Clone rate 1 *refers to the number of HS that host the same content under different addresses. In our previous work [2], we recognized that some onion domains have identical or semi-identical text hosted under different URLs, particularly those with suspicious content. To detect duplication, we calculate the MD5 hash [56] after preprocessing it by removing numbers, special characters, date and time formats, and the PGP signature. The clone_rate of d_i reflects the frequency of its MD5 hash code.*

Term frequency-inverse document frequency (TF-IDF) vectorizer an algorithm comprised of two components, the term frequency (TF) and the inverse document frequency (IDF). The TF counts the number of times a word is used in a domain, while the IDF finds how important a word is in the list of onion domains. It is calculated by dividing the number of onion domains by the number of domains that contain that word. Finally, the TF-IDF is computed as (Eq. 1).

$$w_{(i,d)} = TF_{(i,d)} \times log_2 \frac{N}{DF_i}, \tag{1}$$

where $w_{(i,d)}$ is the weight of word i in domain d, N is the size of domain set D, $TF_{(i,d)}$ is the term frequency of word i in d, and DF_i is the document frequency of word i in D.

Typically, it is good practice to filter out infrequent words by adjusting the max features parameter of the TF-IDF algorithm.[6] Hence, only a specific number of features are considered. Following our previous work [12], we set the $max\_features$ parameter to $10,000$ sorted by the TF-IDF weight. The TF-IDF algorithm represents the text of each onion domain by a feature vector of $10,000$ dimensions. In addition, it returns a dictionary (TF-IDF_dict) of length $max\_features$ that holds the keywords and their weights. Applying the TF-IDF algorithm to a dataset of drugs HS, we obtained the following top-10 words (cannabis, cocaine, quantity, kush, gram, crystal, heroin, psychedelic, drug, and strain). We consider the common words between the TF-IDF_dict and domain d_i as the domain keywords.

[4] https://github.com/katmagic/Shallot

[5] https://github.com/keredson/wordninja

[6] This behaviour is controlled using the $max\_features$ parameter in the Scikit-Learn library.

Consequently, we define the following four features: 1) *keyword_num*: the number of keywords identified in d_i, 2) *keyword_TF-IDF_Acc*: the accumulated TF-IDF keyword weights, 3) *keyword_avg_weight*: the average keyword weight, and 4) *keyword_to_total*: the number of the domain's keywords divided by the number of its words.

4.1.2 Named entities features

A named entity (NE) refers to a real proper name of an object, including but not limited to persons, organizations, or locations. In the Tor network, most entities come from sparse text without context, such as the product entity names mentioned under the product image in a marketplace. Therefore, it is vital to use a named entity recognition model that does not depend heavily on the context. Hence, we used our previous work [57], which was designed especially for this case, rather than contextualized-based models such as the bidirectional encoder representations from transformers (BERT) [58]. The named entity recognition (NER) model recognizes six categories of named entities: persons (PER), locations (LOC), organizations (ORG), products (PRD), creative work (CRTV), corporations (COR), and groups (GRP). We map the extracted NEs into the following five features:

NE number 1 counts the total number of entities in d_i regardless of the category; we name this feature *NE_counter*.

NE popularity 1 an entity is popular if its frequency is above or equal to a threshold that we set to five, as explained in Sect. 5.2.2. For every category identified by the NER model, we use a binary representation to capture the existence of popular entities in domain (1), or (0) otherwise. We refer to this feature as $popular\_NE_X$, where X is the corresponding NER category.

NE TF-IDF 1 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 1 accumulates the TF-IDF weight of the popular NE, and it is named *popular_NE_TF-IDF*.

Emerging NE 1 the frequency of the emerging product entities in d_i. We used our previous work [5] based on the K-Shell algorithm [59] and graph theory to detect emerging entities in HS. We denote this feature by *emerging_NE*.

4.1.3 HTML markup features

Among the available HTML parsing techniques, we used a regular expression pattern to detect hyperlinks because we realized that some onion domain pages reference other domains by mentioning their addresses within the text flow without the $< HREF >$ HTML tag. Hence, libraries such as Beautiful Soup[7] cannot detect them. For the rest of the HTML markup code of d_i, we used the Beautiful Soup library to extract the following features:

Internal hyperlinks 1 counts the number of unique hyperlinks that share the same domain name as d_i. We denote it by *internal_links*.

External hyperlinks 1 refers to the number of pages referenced by d_i on the Tor network or Surface Web. We refer to this feature by *external_links*.

Image tag count 1 corresponds to 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 1 a binary feature to indicate whether the domain needs login and password credentials. We used a regular expression pattern to parse such inputs. This feature is called *needs_credential*.

Domain Title 1 a binary feature to check whether the $< title >$ HTML tag has a textual value. We called it *has_title*.

Domain header 1 a binary feature that checks if the $< H1 >$ HTML tag has a header, and we named it *has_H1*.

Title and header TF-IDF 1 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 1 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 in case it is not loaded properly. This feature refers to the TF-IDF weight accumulation of the alternative text and is denoted as *TF-IDF_alt*.

4.1.4 Visual content features

The visual content can be more attractive than the text to draw the customer's attention. A suspicious services trader might incorporate authentic product images to create an impression of credibility to customers. However, the interesting images for LEAs can be confused with other noisy images, such as banners and logo images. To isolate the interesting images, we built a supervised image classifier that categorizes the visual content into nine categories, where eight are suspicious and one is others. The definition of these categories is based on our previous works [2, 12]. For the image classifier, we fine-tune the Inception-ResNet V2 model [60]. The following features represent the visual content:

[7] https://www.crummy.com/software/BeautifulSoup/

Image count 1 corresponds to the total number of images in d_i, both suspicious and nonsuspicious, regardless of their category. Suspicious stands for images that can contain illicit content. We denote these features by *total_count*, *suspicious_count* and *noise_count*, respectively.

Average classification confidence 1 represents the averaged confidence score of multiple images per category. These features are named *avg_suspicious_conf* and *avg_normal_conf*, respectively.

Majority class 1 a binary flag to indicate whether the majority of the images published in d_i are suspicious. This flag is denoted by *suspicious_majority*.

4.1.5 Network structure features

We modelled the Tor network as a directed graph of nodes and edges. The nodes refer to onion domains, and the edges capture the hyperlinks between domains. This representation allowed us to build the following features:

In-degree 1 the number of onion domains pointing to domain d_i. It is called the *in-degree*.

Out-degree 1 the number of HS referenced by d_i, and it is named *out-degree*.

Centrality measures 1 for each domain d_i in the Tor network graph, we evaluated three node centrality measures: closeness, betweenness, and eigenvector [61, 62]. The closeness metric computes the length of the shortest paths from d_i to the network domains. The betweenness measures the extent to which d_i lies on paths between other domains. Finally, the eigenvector centrality reflects the importance of d_i based on the centrality of its neighbours. Formally, given a graph $G = (V, E)$ with a set of V nodes and E edges, the closeness centrality is calculated as the inverse of the sum of the shortest path distances between a domain d_i and the remaining $|V| - 1$ domains in G, and it is defined in Eq. 2. as:

$$cls(d_i) = \frac{|V| - 1}{\sum_{v=1}^{|V|-1} dis(d_i, d_v)}, \tag{2}$$

where $cls(d_i)$ is the closeness of d_i and $dis(d_i, d_v)$ is the shortest path distance between domains d_i and d_j.

The betweenness of domain d_i is the sum of the fraction of all-pairs shortest paths that pass through d_i; it is given by Eq. 3.

$$btwn(d_i) = \sum_{d_j, d_k \in V and (d_i \sigma d_j, d_i \neq d_k)} \frac{\sigma(d_j, d_k \bar{d_i})}{\sigma(d_j, d_k)}, \tag{3}$$

where $btwn(d_i)$ is the betweenness of d_i, $\sigma(d_j, d_k \bar{d_i})$ and corresponds to the number of shortest paths between domains d_j and d_k that pass through node d_i, and $\sigma(d_j, d_k)$ is the number of shortest paths between domains d_j and d_k.

The eigenvector centrality score of domain d_i, denoted by $eigvec(d_i)$, is proportional to the sum of the eigenvector scores of all connected domains. Therefore, the relative score of domain d_i is defined by Eq. 4.

$$eigvec(d_i) = \frac{1}{\lambda} \sum_{d_j \in G, d_j \neq d_i} a_{(d_i, d_j)} eigvec(d_j), \tag{4}$$

It can be rewritten as $Ax = \lambda x$, where λ is an eigenvalue and $a_{(d_i, d_j)}$ is the adjacency matrix of graph G. If there are hyperlinks between domains d_i and d_j, $a_{(d_i, d_j)} = 1$; otherwise, $a_{(d_i, d_j)} = 0$. Matrix A has multiple eigenvalues, but the components of A are all nonnegative. According to the Perron-Frobenius theorem [63], there is only a unique eigenvalue that satisfies a positive eigenvector of x. The eigenvector centrality calculation is as follows: all the node centralities are initialized to one and multiplied by A. The resulting vectors are normalized, and the process is repeated until convergence [64].

ToRank value 1 ToRank is a link-based ranking algorithm to order the items of a given network following their centrality [2]. We applied ToRank to the Tor network to rank the onion domains and used the assigned rank as a node feature. Moreover, we used a binary flag to indicate whether d_i is in the top-X domains of ToRank. We refer to those features as *ToRank_rank* and *ToRank_top-X*, respectively.

After computing the features described (Table 2), 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.

4.2 Supervised learning-to-rank unit

We adopt the LtR approach widely used in the information retrieval (IR) field. In a traditional IR problem, a training sample has three components: the query ID, a ranked list of answers to the query and their relevance score, which can be either binary [50] or multiple levels of relevance [65]. However, looking at our ranking problem, there are two significant differences. First, we do not have queries; we have a single abstract question: *What are the most attractive onion domains in a determined area of activities?* Second, the relevant, i.e., practising the same activity, thanks to the classification component demonstrated in the Tor monitoring pipeline (see Fig. 1). Simultaneously, the relevance score cannot be multilevel because each domain has received a numerical score calculated and assigned manually by human annotators, as described in Sect. 3. These scores represent the ground-truth while training LtR. Therefore, a training sample d_i has a feature vector and a score r_i in R, where R refers

Table 2 Summary of the HSMU feature vector

Feat. Class	Feat. Count	Feat. Source	Feat. Name
Textual	9	Date and Time	- recently_updated - update_count
		HS Name	- URL_word_count - URL_letter_count
		Clone Rate	- clone_rate
		TF-IDF Vectorizer	- keyword_num - keyword_TF-IDF_Acc - keyword_avg_weight - keyword_to_total
Named Entities	10	NE Popularity	- popular_NE (x^a)
		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		

[a]This feature is repeated for the six named entity types, as explained in Sect. 4.1.2

to the ground-truth set. The feature vector of each sample d_i can be modelled 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 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 similar way to R, i.e., $D \xrightarrow{f} R$. The learning loss function depends on the LtR architecture and is explained in the following three subsections.

4.2.1 Pointwise

The loss function of the pointwise approach considers only a single instance of onion domains at a time [66]. It is a supervised classifier/regressor that independently predicts a relevance score for each query domain. The ranking is achieved by sorting the onion domains according to yield scores. For this LtR schema, we explore the multilayer perceptron (MLP) regressor [67]. This approach estimates the loss function based on a single item, i.e., onion domain, as shown in Eq. 5.

$$L(f; D, R) = \sum_{i=1}^{|R|} (f(d_i) - r_i)^2 \tag{5}$$

4.2.2 Pairwise

Pairwise transforms the ranking task into a pairwise classification task. In particular, the loss function takes a pair of items at a time and attempts to optimize their relative positions by minimizing the number of inversions compared to the ground-truth [68]. We use the RankNet algorithm [68], which is one of the most popular pairwise LtR schemes. The loss function of RankNet is given by Eq. 6, as:

$$L(f; D, R) = \sum_{i=1}^{|R|-1} \sum_{j=1}^{|R|} \theta(f(d_i) - f(d_j)), \tag{6}$$

where θ is logistic function $\theta(z) = log(1 + \exp^{-z})$.

4.2.3 Listwise

This approach extends the pairwise schema by looking at the entire list of samples at once [69]. One of the most well-known listwise schemes is the ListNet algorithms [70]. Given two ranked lists, the human-labelled scores and the predicted scores, the loss function minimizes the cross-entropy error between their permutation probability distributions. The ListNet loss function is defined for all onion domains in R by Eq. 7, as:

$$L(f; d_i, r_i) = -\sum_{j=1}^{|R|} P_{d_i}(j) log P_{f(r_i)}(j) \tag{7}$$

where $P_s(j)$ is a Plackett-Luce probability model [71] of j according to s, which is given by Eq. 8, as:

$$P_s(j) = \prod_{j=1}^{|R|} \frac{exp(s_{j\,j})}{\sum_{k=j}^{|R|} exp(s_{j\,k})} \tag{8}$$

5 Experimental settings

To evaluate the proposed ranking framework, we tailored the experiments to answer three research questions:

- What is the most suitable LtR schema for ranking the onion domains in the Tor network and detecting the influential domains?
- When is each ranking approach used: the content-based and the link-based??
- What is the best combination of features for the LtR model performance?

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

5.1 Evaluation measure

The two most popular metrics for ranking an information retrieval system are mean average precision (MAP) and normalized discounted cumulative gain ($NDCG$) [72, 73]. The main difference between the two is that the MAP assumes a binary relevance of an item according to a given query, i.e., an item can be either relevant or nonrelevant. Additionally, $NDCG$ allows the use of a numerical relevance score. Therefore, the $NDCG$ is better suited for two reasons. First, thanks to the onion domain classification component (see Fig. 1), all domains are relevant, i.e., all have the same category, drug-related domains, in this case. Second, the ground-truth and the predicted rank score are numerical scores produced by the LtR schemes.

To obtain the $NDCG@K$, we calculate the $DCG@K$ following formula (Eq. 9).

$$DCG@K = G_1 + \sum_{i=2}^{K} \frac{G_i}{log_2(i)} \tag{9}$$

Table 3 The image classification performance using the F1 score over a test set of nine classes

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

where G_1 is the gain score at the first position in the obtained ranked list, G_i is the gain score of item i in that list, and K refers to the first K items to calculate the DCG. To obtain a normalized version of $DCG@K$, it is necessary to divide it by $IDCG@K$, which is the ideal $DCG@K$ sorted by the gain scores in descending order (Eq. 10).

$$NDCG@K = \frac{DCG@K}{IDCG@K} \tag{10}$$

5.2 Module configuration

5.2.1 Hardware configurations

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

5.2.2 HSMU configurations

We set the feature vector length of the TF-IDF text vectorizer to 10, 000 with a minimum frequency of 3, following our previous work [12]. We used an NER model trained on the WNUT-2017 dataset.[8] To set the popularity threshold of the *popular_NE_X* feature, we examined four values (3, 5, 10, 15), and we assigned it to 5 experimentally. Additionally, we set the threshold of the *recently_updated* feature to three months earlier than the dataset scraping date. To extract features from the HTML code, we used the BeautifulSoup library.[9] To construct the Tor network graph, we used the NetworkX[10] library.

For the image classifier, we fine-tuned the Inception-ResNet V2 model [60] on a dataset of 11, 700, split as 9, 000 for training and 2, 700 for testing, and equally distributed over nine categories, as shown in Table 3. We collected the images from Google Images using a chrome plugin called *Bulk Image Downloader*.

5.2.3 SLRU configurations

We used the dataset described in Sect. 3 to train and test the three LtR models. Due to the small number of samples in the drug domain, only 290 onion domains, we conducted a 5-fold cross-validation following recommendations from previous works [70]. On each iteration, three folds were used for training the ranking model, one for validation and one for testing. For the three LtR models, the number of iterations is controlled by early stopping criteria, which is triggered when there is no change in the validation set at $NDCG@10$ [74].

The three LtR schemes commented on in Sect. 4.2 share the same network structure but differ in their loss functions. The neural network has two layers, with 128 and 32 neurons. For nonlinearity, a rectifier linear unit (ReLU) activation function is used [75], and a ReLU layer is followed by a dropout layer with a value of 0.5 [76] to avoid overfitting.

6 Results and discussion: drug case study

6.1 Learning to ranking schema selection

In Sect. 4.2, we explored 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 the most suitable LtR schema for ranking the onion domains in the Tor network and detecting the influential ones. Figure 3 compares the three LtR algorithms using the $NDCG@k$ metric for 10 different values of $K = \{1, 3, 5, 7, 9, 15, 25, 35, 45, 55\}$, whereas 55 refers to the complete test set. The values of k are not equally sampled, and we select five values between zero and ten, while the other five values are greater than ten. This distribution is chosen because a correct rank on the head of a ranked list is more important than its tail [50, 77]. The superiority of the listwise approach is evidence of its suitability among the other methods (Fig. 3). The same figure shows that the $NDCG@1$ of ListNet is equal to one, which means that during the five folds of cross-validation, the algorithm ranked the first domains in the test set correctly, exactly as the ground-truth. It obtained $NDCG@5$ and $NDCG@10$ values 0.97 and 0.93, respectively. However, the lowest value was at $NDCG@25$ of 0.88. Additionally, as shown in Fig. 3, the pointwise approach, which is the MLP in our case, obtained the worst performance, which agrees with the conclusion of other researchers [65].

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

[9] https://pypi.org/project/beautifulsoup4/

[10] https://networkx.github.io/

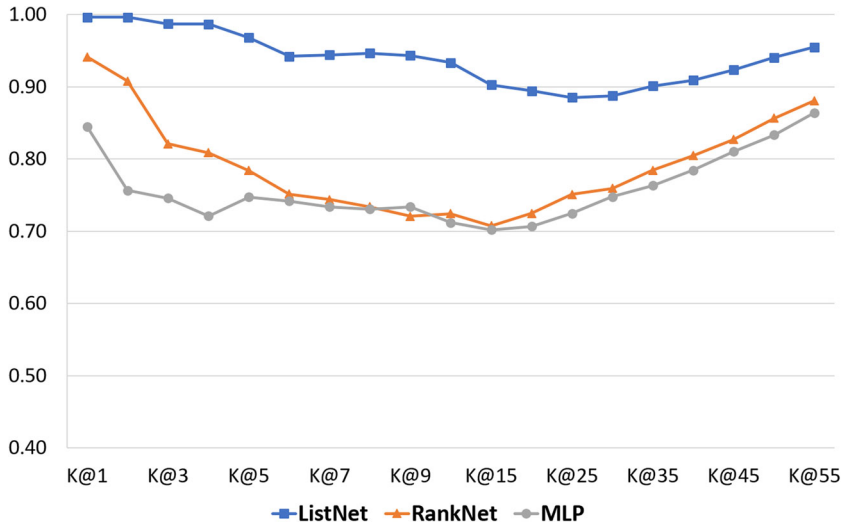

Fig. 3 A comparison between three LtR algorithms against multiple values of averaged $NDCG@K$ over the five fold cross-validation. The horizontal axis refers to the K value, and the vertical axis indicates the $NDCG$ scores of the algorithms obtained at each value of K

The superiority of the ListNet scheme comes from its ability to map a list of scores to a probability distribution, whereas the loss is calculated using the cross entropy between the predicted probability distribution and a target probability distribution. Therefore, we can say that ListNet considers the complete list of ranked items, while pointwise and pairwise ignore this structure.

Table 4 presents the top-10 drug domains nominated by each ranking algorithm.

In addition to comparing the performance using $NDCG@K$, we register the total time required to train and test each LtR model. More precisely, we compare these times from when the model receives a list of domains encoded by the HSMU (Sect. 4.1) until it produces the rank. On average, for the five folds, the ListNet model took 8.30 seconds for training and 0.08 seconds for testing. The RankNet took 7.35 seconds for training and 0.007 seconds for testing. Finally, the MLP model was the fastest, requiring 3.34 seconds for training and 0.0009 for testing. This comparison shows that the ListNet model is the slowest due to the complexity of its loss function compared to the RankNet and MLP algorithms.

6.2 Link-based versus content-based ranking

Having two distinct ranking strategies raises a question: *What is the most suitable ranking approach, content-based or link-based?* To answer this question, we explore four link-based algorithms: ToRank [2], PageRank [25], hyperlink-induced topic search (HITS) [26], and Katz [27]. Ranking the onion domains of the Tor network using a link-based approach requires a directed graph representation. The graph nodes represent onion domains, and the directed edges capture the hyperlinks between domains. We compare these four link-based algorithms against the best LtR model, i.e., ListNet, which depends on the 40 features described in Sect. 4.1.

6.2.1 Comparison configuration

Unlike our supervised ranking approach [2], the link-based approach does not require training data; it can be seen as an unsupervised ranking. In contrast, LtR uses a portion for training and another for testing. Therefore, to perform a fair comparison between these two approaches, we use a five fold cross-validation. We split the dataset into five parts, and each time, one-fold is held out for testing, and the remaining four folds are used to train LtR. Hence, both approaches are tested on the same test set. Finally, we report the average $NDCG$ of both. We evaluated several configuration parameters for the link-based algorithms and selected the parameters that obtained the highest $NDCG$ (Table 5).

Figure 4 shows that ListNet surpasses all the link-based ranking algorithms. We observe that the weakest LtR approach, i.e., MLP, which obtained an $NDCG@10$ 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 hyperlink connectivity only. Nonetheless, the link-based approach, such as ToRank, is still valid, with an $NDCG@10$ of 0.69 without labelling cost.

6.3 Feature selection

In the previous sections, we concluded that ListNet outperformed the benchmarked techniques when a feature vector of forty dimensions represented each hidden service. However, the computational cost of these features varies. Some of them, such as the visual content, require 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 build the rank-

Table 4 An example of the top 10 ranking algorithm outputs sorted from the highest to the lowest influence. The rank is estimated only based on the output of the first fold of the cross-validation. ListNet has the highest NDCG@10

Order	Ground-truth	ListNet	RankNet	MLP
1	coinrx6j4gqspquq.onion	torpharmzxholobn.onion	iv2w26wwal6tnnpl.onion	tdupp6lmgnpex5ss.onion
2	torpharmzxholobn.onion	coinrx6j4gqspquq.onion	tdupp6lmgnpex5ss.onion	gpostalfauulvzhs.onion
3	gpostalfauulvzhs.onion	tdupp6lmgnpex5ss.onion	newpdsuslmzqazvr.onion	newpdsuslmzqazvr.onion
4	xdsa5xcrrrxxxolc.onion	gpostalfauulvzhs.onion	kbvbh4kdddiha2ht.onion	smoke77v445xp3oc.onion
5	tdupp6lmgnpex5ss.onion	xdsa5xcrrrxxxolc.onion	smoke77v445xp3oc.onion	newpdioehu3fhxph.onion
6	eeyovrly7charuku.onion	rso4h34eooxjlg75.onion	smokerhv5hlklzh2.onion	nlgro7qqgwi2jjnv.onion
7	eupillu4np223oxe.onion	eupillu4np223oxe.onion	rso4h34eooxjlg75.onion	smokerhv5hlklzh2.onion
8	pharmasuzik56e4l.onion	smokerhv5hlklzh2.onion	newpdioehu3fhxph.onion	rso4h34eooxjlg75.onion
9	artsmankindxgcv5.onion	eeyovrly7charuku.onion	nlgro7qqgwi2jjnv.onion	kbvbh4kdddiha2ht.onion
10	limaconzruthefg4.onion	artsmanvesyngaz5.onion	drugszun7tvsgsaa.onion	pms5n4czsmblkcjl.onion
NDCG@10		**0.90**	0.62	0.65

ing model and the inference time. On average, per domain, the prediction of the image classification model was the most expensive. It took 109 seconds, followed by the NER model with 22 seconds and the text features that required 12. Finally, the HTML and graph features were the fastest to be extracted, requiring 3 and 2 seconds, respectively.

Furthermore, we used asymptotic notation to generalize the processing time of features and compare their time complexities. In particular, textual features have a time complexity of $O(nLlog(nL))$, i.e., the complexity of computing the TF-IDF feature vector, where n is the total number of text sequences and L is the average length of these sequences [78]. In contrast, the HTML feature has a time complexity of $O(n)$. Regarding the network structure features, ToRank has a complexity of $O(2n)$, and the remaining features have a time complexity of $O(n)$. Last, the complexity of the neural network-based models, such as the image classifier Inception-ResNet V2 or the NER model, depends on the structure of the neural network, i.e., the number of convolution layers and kernels [79]. Because of this, visual content features are the most time-consuming.

To answer the question: *What feature or combination of features produces the best LtR model performance?* We compare a ListNet ranker trained on different collections of features, as shown in Fig. 5.

We found that the features extracted only from text, denoted by *text*, achieved the highest $NDCG@5$ of 0.90. The features extracted from the NEs came in the second position, which obtained an $NDCG@5$ of 0.85. After that, using only features extracted from HTML, the ListNet model obtained an $NDCG@5$ of 0.81. In contrast, the graph features obtained the lowest $NDCG@5$ of 0.65, which indicates their weakness in ranking onion domains, unlike the features extracted from the text, which showed a significant and positive impact on the $NDCG$ metric. Hence, the features extracted from the user-visible text are more representative than those from the visual content or the graph structure.

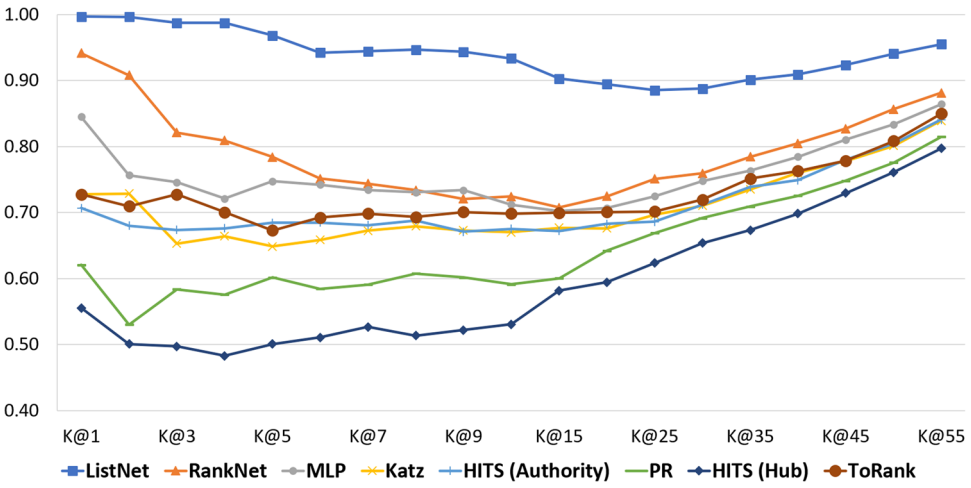

Fig. 4 A comparison between the content-based versus link-based ranking algorithms concerning multiple values of K. The horizontal axis refers to the K value, and the vertical axis indicates the $NDCG$ scores of the algorithms obtained at each value of K

Table 5 The evaluated parameter for the link-based ranking algorithms. Bold values correspond to the selected configuration with the highest $NDCG$

Algorithm name	Parameter	Evaluated values
PageRank	alpha	0.5, 0.70, 0.75.0.80,**0.85**,0.90
	max_iter	10, 100, **1000**, 10,000
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**, 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, **1000**, 10,000

Furthermore, we examined the impact of aggregating the user-visible text features. Figure 5 shows an increase in the NDCG when the *text*, *NER*, and *HTML* features were combined. They scored an $NDCG@5$ of 0.95 compared to 0.97 when all the features were used. Hence, the graph and the visual features can be ignored with a 0.02 decrease in the $NDCG$. However, at $NDCG@10$, user-visible text features scored 0.88 and 0.93 for all the features, which means a decrease of 0.05. This result emphasizes the ability of the proposed ranking framework to rank onion domains regardless of whether they were isolated in the network or carried visual content. Therefore, further exploration of textual features, mainly textual semantic representation, such as BERT [80], for onion domains will significantly boost the ranking results.

6.4 Limitations of the content-based ranking

The content-based ranking approach has some limitations. As it falls under the supervised learning umbrella, it requires preranked data, which can be labour extensive. Moreover, building a training set requires answering subjective questions based on the annotators' opinions, such as "Do you feel that this domain is trustable?" If the answers are not normalized to a standard, more noise will be introduced in the dataset. Furthermore, when a domain blocks the crawler from exploring the content by requesting login credentials, a content-based ranker will not be able to analyse the content and produce the expected output.

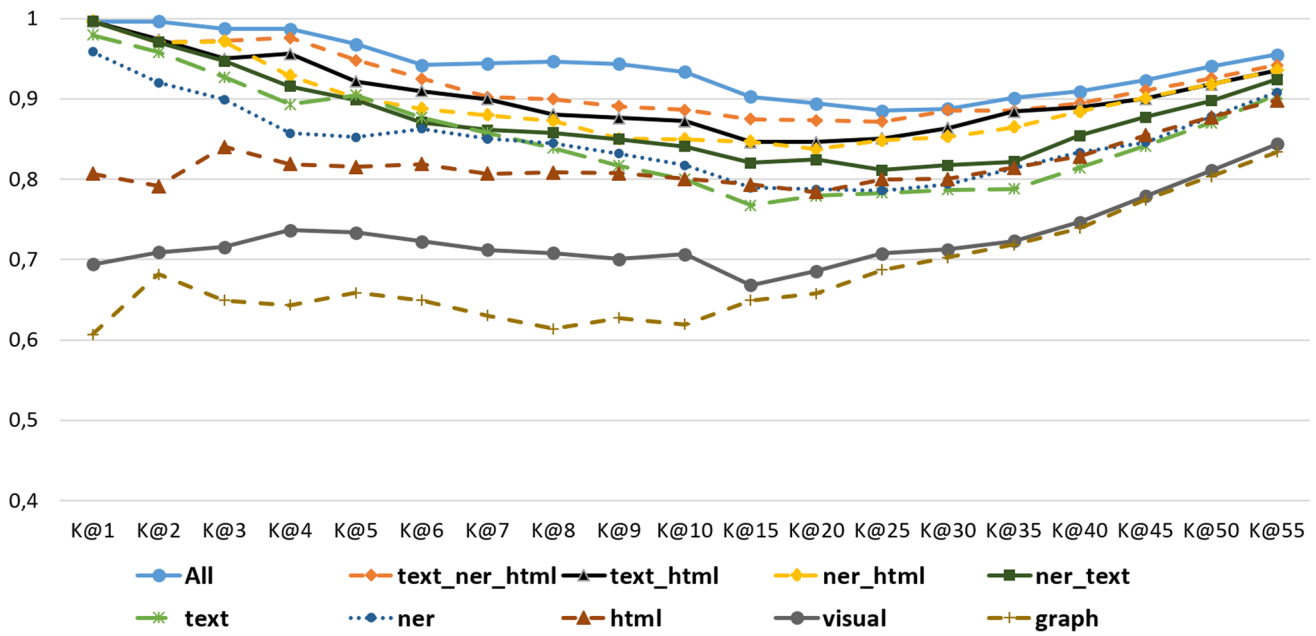

Fig. 5 The effect of using different types of features along with their combinations on the ListNet ranking model. The vertical axis refers to the $NDCG$ value, while the horizontal axis denotes the value of K. Each curve refers to a source of features: textual (*text*), featured produced by a named entity recognition (*NER*), HTML markup features (*HTML*), visual features (*visual*), graph features (*graph*), and all the features fused, denoted as (*All*)

7 Conclusions and future work

The Tor network hosts suspicious activities that LEAs might be interested in monitoring. Ranking the onion domains according to their influence inside the Tor network will help LEAs prioritize the domains to leverage the monitoring process.

In this paper, we benchmarked three supervised learning-to-rank (LtR) algorithms, MLP, RankNet, and ListNet, to detect and rank the most influential onion domains. The proposed framework consists of two components: 1) a hidden service modelling unit (HSMU), which represents an onion domain by 40 features extracted from the domain user-visible text, the HTML markup of the web page, the NEs in the domain text, the visual content, and the Tor network structure; and 2) a supervised learning-to-rank unit (SLRU), which builds a ranking model.

We tested the effectiveness of our framework on a manually ranked dataset of 290 onion domains related to drug trading. We found that the ListNet algorithm outperforms with an $NDCG@10$ of 0.93.

Furthermore, we analysed the impact of the feature collections on ranker performance. We found that using only the user-visible textual features extracted from the text, NEs, and HTML markup code, the model achieves 0.95 at $NDCG@5$, and it decreased to 0.88 at $NDCG@10$, in comparison to 0.97 and 0.93, respectively, when all the features were used. Hence, using only features from the user-visible text allows the model to perform comparably with less complexity.

In the future, we plan to boost the explored features with a contextualized language model, such as BERT, to extract semantic features from the onion domain text [80].

Acknowledgements This work was supported by the framework agreement between the University of León and INCIBE (Spanish National Cybersecurity Institute) under Addendum 01 and 22.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Declarations

Conflict of interest The authors declare that they have no conflicts of interest.

References

1. The Tor Project I (2022) Who uses Tor? Accessed: 17 April 2022. https://www.torproject.org/about/torusers.html.en
2. Al-Nabki MW, Fidalgo E, Alegre E, Fernández-Robles L (2019) Torank: Identifying the most influential suspicious domains in the tor network. Expert Syst Appl 123:212–226. https://doi.org/10.1016/j.eswa.2019.01.029
3. Choshen L, Eldad D, Hershcovich D, Sulem E, Abend O (2019) The language of legal and illegal activity on the darknet. In: Proc. of ACL
4. Foley S, Karlsen JR, Putniņš TJ (2019) Sex, drugs, and bitcoin: How much illegal activity is financed through cryptocurrencies? The Review of Financial Studies 32(5):1798–1853
5. Al Nabki MW, Fidalgo E, Alegre E, González-Castro V (2017) Detecting emerging products in tor network based on k-shell graph decomposition. III Jornadas Nacionales de Investigación en Ciberseguridad (JNIC) 1(1):24–30
6. Ciancaglini V, Balduzzi M, Goncharov M, McArdle R (2013) Deepweb and cybercrime. Trend Micro Report 9:1–22
7. 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
8. Anjum A, Kaur C, Kondapalli S, Hussain M, Begum A, Hassen S, Boush D, Benjeed A, Abdulraheem D (2021) A mysterious and darkside of the darknet: A qualitative study. Webology 18(4)
9. Wang V, Gee J, Button M (2022) In: Gill M (ed.) Crime on the Darknet The Case of Brand Abuse pp. 447–467. Springer, Cham
10. Ling Z, Luo J, Wu K, Yu W, Fu X (2015) Torward: Discovery, blocking, and traceback of malicious traffic over tor. IEEE Trans Inf Forensics Secur 10(12):2515–2530
11. Biswas R, Fidalgo E, Alegre E (2017) Recognition of service domains on tor dark net using perceptual hashing and image classification techniques. In: 8th International Conference on Imaging for Crime Detection and Prevention (ICDP 2017) pp. 7–12. https://doi.org/10.1049/ic.2017.0041
12. Al Nabki MW, Fidalgo E, Alegre E, 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: Volume 1, Long Papers vol. 1 pp. 35–43
13. Probst G, Borzillo S (2008) Why communities of practice succeed and why they fail. Eur Manag J 26(5):335–347
14. Ríos SA, Aguilera F, Nuñez-Gonzalez JD, Graña M (2019) Semantically enhanced network analysis for influencer identification in online social networks. Neurocomputing 326–327:71–81
15. Bourhis A, Dubé L, Jacob R et al (2005) The success of virtual communities of practice: The leadership factor. The Electronic Journal of Knowledge Management 3(1):23–34
16. Eliacik AB, Erdogan N (2018) Influential user weighted sentiment analysis on topic based microblogging community. Expert Syst Appl 92:403–418
17. Berzinji A, Kaati L, Rezine A (2012) Detecting key players in terrorist networks. In: Intelligence and Security Informatics Conference (EISIC), 2012 European pp. 297–302. IEEE
18. Gohari FS, 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
19. Nurmi J, Kaskela T, Perälä J, Oksanen A (2017) Seller's reputation and capacity on the illicit drug markets: 11-month study on the

finnish version of the silk road. Drug Alcohol Depend 178:201–207. https://doi.org/10.1016/j.drugalcdep.2017.05.018
20. Hardy RA, Norgaard JR (2016) Reputation in the internet black market: an empirical and theoretical analysis of the deep web. J Inst Econ 12(3):515–539
21. Accessed: 17 May 2022 (2019). https://blog.torproject.org/one-cell-enough-break-tors-anonymity
22. Chaabane A, Manils P, Kaafar MA (2010) Digging into anonymous traffic: A deep analysis of the tor anonymizing network. In: 2010 Fourth International Conference on Network and System Security pp. 167–174. IEEE
23. Biryukov A, Pustogarov I, Thill F, Weinmann RP (2014) Content and popularity analysis of tor hidden services. In: Distributed Computing Systems Workshops (ICDCSW), 2014 IEEE 34th International Conference On pp. 188–193. IEEE
24. Elahi T, Bauer K, AlSabah M, Dingledine R, 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 pp. 43–54. ACM
25. Page L, Brin S, Motwani R, Winograd T (November 1999) The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab . http://ilpubs.stanford.edu:8090/422/
26. Kleinberg JM (1999) Authoritative sources in a hyperlinked environment. Journal of the ACM (JACM) 46(5):604–632
27. Katz L (1953) A new status index derived from sociometric analysis. Psychometrika 18(1):39–43
28. Bernaschi M, Celestini A, Guarino S, Lombardi F (2017) Exploring and analyzing the tor hidden services graph. ACM Transactions on the Web (TWEB) 11(4):24
29. Li H (2011) A short introduction to learning to rank. IEICE Trans Inf Syst 94(10):1854–1862
30. Yao J, Liu F, Geng Y (2019) Query-specific optimal convolutional neural ranker. Neural Comput Appl 31(7):3107–3116
31. Yu W, Li S, Tang X, Wang K (2019) An efficient top-k ranking method for service selection based on ε-admopso algorithm. Neural Comput Appl 31(1):77–92
32. Dean B (2018) Google's 200 Ranking Factors: The Complete List. Accessed: 17 April 2022. https://backlinko.com/google-ranking-factors
33. Broséus J, Rhumorbarbe D, Mireault C, Ouellette V, Crispino F, Décary-Hétu D (2016) Studying illicit drug trafficking on darknet markets: structure and organisation from a canadian perspective. Forensic Sci Int 264:7–14
34. Barratt MJ, 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
35. Dolliver DS (2015) Evaluating drug trafficking on the tor network: Silk road 2, the sequel. International Journal of Drug Policy 26(11):1113–1123
36. Weimann G (2016) Terrorist migration to the dark web. Perspectives on Terrorism 10(3):40–44
37. Chen H, Chung W, Qin J, Reid E, Sageman M, Weimann G (2008) Uncovering the dark web: A case study of jihad on the web. J Am Soc Inform Sci Technol 59(8):1347–1359
38. Nunes E, Diab A, Gunn A, Marin E, Mishra V, Paliath V, Robertson J, Shakarian J, Thart A, Shakarian P (2016) Darknet and deepnet mining for proactive cybersecurity threat intelligence. In: IEEE International Conference on Intelligence and Security Informatics: Cybersecurity and Big Data, ISI 2016, pp. 7–12. Institute of Electrical and Electronics Engineers Inc., United States. https://doi.org/10.1109/ISI.2016.7745435
39. Chen H (2011) Dark Web: Exploring and Data Mining the Dark Side of the Web. Springer
40. Wasserman S, Faust K (1994) Social Network Analysis: Methods and Applications. Cambridge University press
41. Choi Y-J, Jeon B-J, Kim H-W (2021) Identification of key cyberbullies: A text mining and social network analysis approach. Telematics Inform 56:101504
42. Oroh AJ, Bandung Y, Zagi LM (2021) Detection of the key actor of issues spreading based on social network analysis in twitter social media. In: 2021 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob), pp. 206–212. IEEE
43. Anwar T, Abulaish M (2015) Ranking radically influential web forum users. IEEE Trans Inf Forensics Secur 10(6):1289–1298
44. Liu TY, et al (2009) Learning to rank for information retrieval. Foundations and Trends® in Information Retrieval 3(3), 225–331
45. Jiang H, Nie L, Sun Z, Ren Z, Kong W, Zhang T, Luo X (2016) Rosf: Leveraging information retrieval and supervised learning for recommending code snippets. IEEE Transactions on Services Computing 34–46. https://doi.org/10.1109/TSC.2016.2592909
46. Macdonald C, Santos RLT, Ounis I (2013) The whens and hows of learning to rank for web search. Inf Retrieval 16(5):584–628. https://doi.org/10.1007/s10791-012-9209-9
47. Ghanbari E, Shakery A (2022) A learning to rank framework based on cross-lingual loss function for cross-lingual information retrieval. Appl Intell 52(3):3156–3174
48. Li J, Xing Z, Kabir A (2018) Leveraging official content and social context to recommend software documentation. IEEE Trans Serv Comput 1:1–1
49. Agichtein E, Brill E, 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 pp. 19–26. ACM
50. Wang S, Zou Y, Ng J, Ng T (2017) Context-aware service input ranking by learning from historical information. IEEE Trans Serv Comput 01:1–1. https://doi.org/10.1109/TSC.2017.2777487
51. Duan Y, Jiang L, Qin T, Zhou M, Shum HY (2010) An empirical study on learning to rank of tweets. In: Proceedings of the 23rd International Conference on Computational Linguistics pp. 295–303. Association for Computational Linguistics
52. Li M, Luo L, Miao L, Xue Y, Zhao Z, Wang Z (2016) 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 pp. 896–900. IEEE
53. Liu C, Cao T, Zhou L (2022) Learning to rank complex network node based on the self-supervised graph convolution model. Knowl-Based Syst 251:109220. https://doi.org/10.1016/j.knosys.2022.109220
54. Li R, Lei KH, Khadiwala R, Chang KCC (2012) Tedas: A twitter-based event detection and analysis system. In: Data Engineering (icde), 2012 Ieee 28th International Conference On pp. 1273–1276. IEEE
55. Hucka M (2018) Nostril: A nonsense string evaluator written in python. Journal of Open Source Software 3(25), 596. https://doi.org/10.21105/joss.00596
56. Rivest R (1992) The md5 message-digest algorithm. Internet Engineering Task Force
57. Al-Nabki MW, Fidalgo E, Alegre E, Fernández-Robles L (2020) Improving named entity recognition in noisy user-generated text with local distance neighbor feature. Neurocomputing 382:1–11
58. Ghaddar A, Langlais P, Rashid A, Rezagholizadeh M (2021) Context-aware adversarial training for name regularity bias in named entity recognition. Transactions of the Association for Computational Linguistics 9:586–604
59. Carmi S, Havlin S, Kirkpatrick S, Shavitt Y, Shir E (2007) A model of internet topology using k-shell decomposition. Proc Natl Acad Sci 104(27):11150–11154

60. Szegedy C, Ioffe S, Vanhoucke V, Alemi AA (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. pp. 4278–4284
61. Freeman LC (1978) Centrality in social networks conceptual clarification. Social networks 1(3):215–239
62. Newman MEJ (2005) A measure of betweenness centrality based on random walks. Social Networks 27(1):39–54. https://doi.org/10.1016/j.socnet.2004.11.009
63. Ruhnau B (2000) Eigenvector-centrality - a node-centrality? Social Networks 22(4):357–365. https://doi.org/10.1016/S0378-8733(00)00031-9
64. Gómez S (2019) Centrality in networks: Finding the most important nodes. In: Business and Consumer Analytics: New Ideas, pp. 401–433. Springer
65. Li H (2011) Learning to rank for information retrieval and natural language processing. Synthesis Lectures on Human Language Technologies 4(1):1–113
66. Friedman JH (2001) Greedy function approximation: A gradient boosting machine. Ann. Statist. 29(5):1189–1232. https://doi.org/10.1214/aos/1013203451
67. Ciaramita M, Murdock V, Plachouras V (2008) Online learning from click data for sponsored search. In: Proceedings of the 17th International Conference on World Wide Web pp. 227–236. ACM
68. Burges C, Shaked T, Renshaw E, Lazier A, Deeds M, Hamilton N, Hullender G (2005) Learning to rank using gradient descent. In: Proceedings of the 22nd International Conference on Machine Learning pp. 89–96. ACM
69. Xia F, Liu TY, Wang J, Zhang W, Li H (2008) Listwise approach to learning to rank: theory and algorithm. In: Proceedings of the 25th International Conference on Machine Learning pp. 1192–1199. ACM
70. Cao Z, Qin T, Liu TY, Tsai MF, Li H (2007) Learning to rank: from pairwise approach to listwise approach. In: Proceedings of the 24th International Conference on Machine Learning pp. 129–136. ACM
71. Ceberio J, Mendiburu A, Lozano JA (2013) The plackett-luce ranking model on permutation-based optimization problems. In: 2013 IEEE Congress on Evolutionary Computation, pp. 494–501. IEEE
72. Järvelin K, 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 pp. 41–48. ACM
73. Manning C, Raghavan P, Schütze H (2010) Introduction to information retrieval. Nat Lang Eng 16(1):100–103
74. Lai H, Pan Y, Liu C, Lin L, Wu J (2013) Sparse learning-to-rank via an efficient primal-dual algorithm. IEEE Trans Comput 62(6):1221–1233
75. Xu B, Wang N, Chen T, Li M (2015) Empirical evaluation of rectified activations in convolutional network. In: Deep Learning Workshop, ICML 15
76. Hinton GE (2002) Training products of experts by minimizing contrastive divergence. Neural Comput 14(8):1771–1800
77. Cao Y, Xu J, Liu TY, Li H, Huang Y, Hon HW (2006) Adapting ranking svm to document retrieval. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval pp. 186–193. ACM
78. Cong Y, Chan Y-B, Ragan MA (2016) A novel alignment-free method for detection of lateral genetic transfer based on tf-idf. Sci Rep 6(1):1–13
79. Jiang K, Zhang J, Wu H, Wang A, Iwahori Y (2020) A novel digital modulation recognition algorithm based on deep convolutional neural network. Appl Sci 10(3):1166
80. Kenton, JDMWC, Toutanova LK (2019) Bert: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of naacL-HLT pp. 4171–4186