









# CrawlBot: A Domain-Specific Pseudonymous Crawler

Vidyesh Shinde<sup>1</sup> , Shahil Dhotre<sup>1</sup> , Vedant Gavde<sup>1</sup> , Ashwini Dalvi<sup>2</sup> ,  
Faruk Kazi<sup>1</sup> , and S. G. Bhirud<sup>2</sup> 

<sup>1</sup> Department of Electronics Engineering, V.J.T.I. Mumbai, Mumbai, India  
{vvshinde\_b16, smdhotre\_b16, vsgavde\_b16, fskazi}@el.vjti.ac.in

<sup>2</sup> Department of Computer Engineering, V.J.T.I. Mumbai, Mumbai, India  
{aadalvi\_p19, sgbbhirud}@ce.vjti.ac.in

**Abstract.** The Dark Net, a dark side of the internet, became a perfect hosting ground for criminal activities and services, including significant drug market-places, child and women sexual abuse, forged documents, homemade explosives, and likewise. The nature of the Dark Net makes it indifferent to searching through the indexed mechanism. Diverse approaches are suggested and attempted to gain useful insights from the Dark Net. In this paper, a focused crawling framework for uncovering material of child and women abuse that lies in the Surface and the Dark Net is proposed. A focused crawler is domain-specific and trained to selectively retrieve web pages that are related to a given domain from the Internet. One of the novel attributes of the crawler is traversing the Surface Net and the Dark Net in a pseudonymous way. During a single crawl session, it automatically selects hyperlinks, which are most likely to give related web pages and downloads pages that are relevant. The hyperlink selection method uses anchor text and local context of the hyperlink as a feature vector. Numerous web data mining and classification tasks utilize local contexts of hyperlinks. Focused or Topical crawlers have a major dependence on local contexts of hyperlinks. The relevancy of the page and hyperlinks is calculated using natural language processing and a sophisticated Artificial Neural Network (ANN) based classifier. The evaluation of the crawler indicates the high harvest rate and effectiveness of the crawler.

**Keywords:** TOR · Dark web crawling · ANN · Child sexual abuse material · Pseudonymous crawling · Hyperlink selection

## 1 Introduction

The history of the hidden web is almost as old as the history of the surface web. The same architecture and design that made the Surface Web possible, also makes the Dark Net possible. The Surface Web consists of web documents that search engines can find and then provide according to users' needs. As the tip of an iceberg is visible to us, a conventional web spider can see just a tiny fraction of the data that's available on the internet. The remaining part of the information which cannot be found on the regular search engines forms the Deep Net. It is gigantic, unlike the Surface Web. More

than 555 million domains have registered on the web and each contains multiple web pages. Many of these pages are not indexed and thus become part of the Deep Net. This enormous Deep Net has a small part known as the Dark Net. It is purposely hidden and is practically inaccessible via ordinary web browsers. The proper setup of the software and its configuration allows users to enter the Dark Net. It has gained popularity because it offers anonymity and facilitates secret communications. These significant features of the Dark Net attracted many cybercriminals to host illicit content and soon became a scary place to visit on the internet.

Child Sexual Abuse Material (CSAM) is content that portrays the sexual exploitation of children under the age of 18-years-old [1]. As a general term, CSAM belongs to child pornography as well as other forms of online child sexual offense. The usage of the Internet in CSAM crimes has caused substantial concern over the last decade, mainly as it links to the use of the Dark Net or content hosted on anonymized software services such as TOR [2]. Although the true prevalence of CSAM present on the Dark Net is unexplored, recent reports have revealed the popularity of child pornography sites among users of hidden services. A study of TOR conducted at the University of Portsmouth revealed that more than 83% of the Dark Net is created by visits to websites hosting child abuse material [3]. CSAM is different from other Dark Web-based illicit product marketplaces because only a tiny portion of CSAM is commercial. Surveys suggest only about 7.5% of CSAM on the Dark Net is sold for a profit. This is because the majority of CSAM functions as a barter system in which images are collected and traded within peer-to-peer (P2P) networks. The technological constraints required for accessing the elusive world of the Dark Net make crawling schemes employed for it more intricate, unlike the Surface Net.

Many organizations are working on this domain to prevent the online exploitation of children. ‘Thorn’ [4] is a non-profit body established in 2012 which focuses to protect children from sexual offense. One of its flagship products ‘Spotlight’ [5] was developed to enable law enforcement to collaborate beyond jurisdictions or national borders in identifying victims. Another initiative was taken by Thorn in this domain by the development of a tool called ‘Safer’ [6]. It provides companies with proper tools to eradicate CSAM from their platform.

The increasing number of content being uploaded on the web and technical complexities has made the manual investigation a very tedious task. Still, some websites like ‘The Hidden wiki’ and darknet search engines like Candle, Torch, etc. provide entry points to the Dark Net. The entry points gathered from these pages often lead to inactive websites and make the investigation futile. This paper highlights an effective tool that can be used by law enforcement and researchers to get the insight of the illicit content on the web. The proposed crawler can maneuver not only surface web but also the Dark Net to discover web resources on child and women abuse content. The crawler flawlessly follows related hyperlinks by adapting unique identities every time visiting a website. The next section talks about various works related to our crawler. Further, Sect. 2 discusses proposed architecture followed by pseudonymous and non-blocking crawling strategy, construction of feature vectors for the page classifier and link classifier. Section 4 displays the outcomes of the experiment for the CSAM domain and finally, Sect. 5 talks about the conclusion and future work.

## 2 Related Work

Several crawlers have developed to obtain the information from the Surface Net and the Hidden Web. Preliminary research has proposed various techniques and tools to obtain data from the visible web and the Hidden Net [7, 8]. Many studies describe the entry points of the Hidden Net [9, 10]. Generally, an exhaustive crawler and the focused crawler are employed in research areas. An Exhaustive or a Generic [11] crawler is the one that extracts information from the sites irrespective of its relevance to the domain and thus delays retrieval of domain-related information. Focused Crawler or Domain-specific crawler crawls topic or domain-related web pages and extracts information from it. ACHE crawler [12] can automatically search for information on a particular domain, producing a large number of relevant results from the web. We will now discuss a few crawlers related to our work.

Basic Exhaustive crawler proposed by Pant et al. [11] extracts all the information from the web without checking the relevancy of the data to the domain of interest. The basic flow of this type of crawler is to maintain the list of unvisited URLs called frontier. Initially, they consist of seed URLs provided by the users or another program. After this, the crawling loop starts by popping out one URL at a time from the frontier, visits that URL and extract all the information and hyperlinks present on the page and mark that page as visited so that the crawler won't encounter the same page again while crawling. This marking method helps for fast crawling process. Hyperlinks extracted from the page are added into the frontier, implemented as a FIFO queue because the crawler is working on the breadth-first algorithm. Now, if the crawler is ready to crawl another page, the next URL is popped out and the crawling process mentioned above continues till the termination. When the crawler reaches the depth (a certain number of pages have been crawled) or situation signals a dead-end (Frontier is empty-The crawler has no new page to fetch) the whole crawling process gets terminated.

ACHE crawler [12] is a domain-specific crawler developed at New York University. The darknet sites are generally not accessible by generic crawlers. The servers are located in the TOR network and require specific protocols for being accessed. ACHE uses external HTTP proxies such as Privoxy to crawl such sites. These proxies allow the crawler to route the request through the TOR network. It uses machine learning-based or regular expression based classifier to judge relevant and irrelevant web pages. It relies upon the title, body, and URL of the webpage as features for the classifier. The ACHE crawler employs the hard-focus and the soft-focus crawling strategy. In the hard-focus strategy, the crawler will discard all hyperlinks from irrelevant webpages. In the soft-focus strategy, it will discard hyperlinks only if the score calculated by the link classifier is lesser than the configured threshold value. It also uses a link classifier to determine the order of crawling. The link classifier assigns a score to the link based on its depth and discards links with depth higher than the predetermined threshold. ACHE uses a seed finder tool to collect a large set of seed URLs related to the chosen domain. The tool accepts a query from a user and redirects it to a search engine to obtain relevant seed URLs. It stores raw information and metadata of webpages as documents in the elasticsearch index.

The E-FFC (Enhanced Form-Focused Crawler) is a Deep Web Database (WDB) focused crawler proposed by Li et al. [13]. This crawler tries to tackle problems such as

mining high-quality information from massive WDBs by automatically discovering and recognizing domain-specific entry points. It aims to overcome some common crawler deficiencies such as low stability of harvest and coverage rate. The crawler implements a two-step page classifier and link scoring to locate its target webpages by the breadth search strategy. E-FFC in its page classifier uses a page similarity threshold to check if a page belongs to a certain domain. The page similarity threshold is computed by extracting domain-specific pages from an available data set and comparing their domain feature vectors. The link classifier implemented in E-FFC considers a link scoring strategy with two features, immediate benefit link and delayed benefit link. Due to correlation among domains, the aforementioned methods also grab WDBs forms from other domains. To eliminate this, a generic (domain-independent) searchable form classifier and a domain-specific form classifier are introduced in a sequence in the E-FFC to filter domain-specific searchable forms. Further, to increase the rate of crawling, the E-FFC has adopted two types of link queues, single-level site link queue, and multi-level in-site link priority queues. As crawling stopping criteria, the E-FFC quits from a website after finding 4 forms in a crawling website or if the depth of visited pages away from its root page exceeds 5 levels.

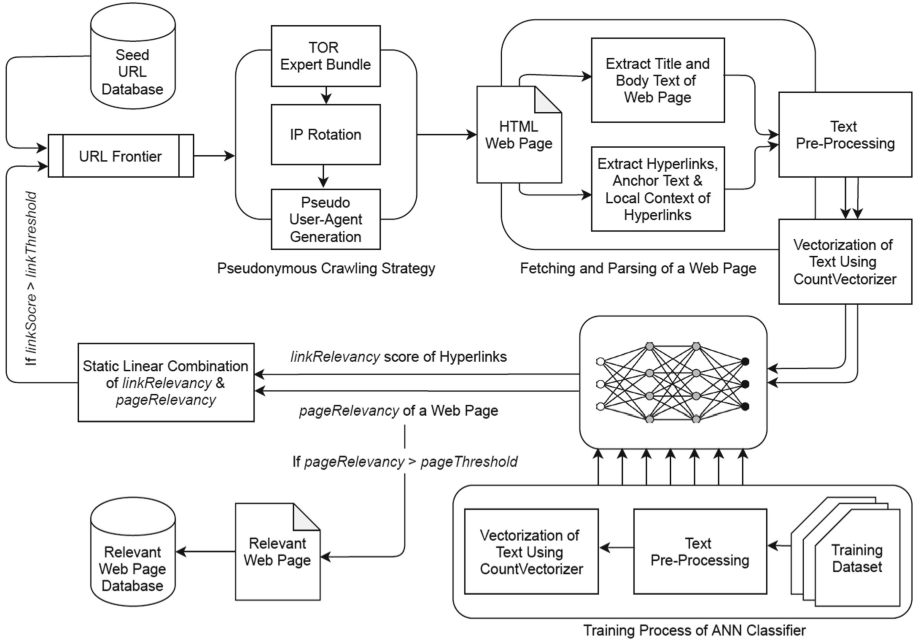
The SmartCrawler proposed by Zhao et al. [7] is a two-stage domain-specific crawler framework that helps to locate hidden-web resources. The first stage is Site locating which uses a backlinks searching technique and site prioritization technique to find relevant links. The backlinks searching technique uses generic search engines to find the main pages of sites. These relevant links act as input to the next stage. The next stage is In-site exploring which utilizes ‘balance link prioritizing’ and ‘form classifier’. This technique helps to harvest hidden web sources (such as searchable and non-searchable forms [13]) by covering a wide portion of the web as much as possible. It uses an adaptive learning technology that leverages data collected during past crawling session. This crawler has an effective harvesting rate for deep web interfaces or hidden web resources but is limited to the site hosted on the surface web.

Iliou et al. [8] proposed a focused crawling framework for extracting information on any given topic from the sites hosted on the Surface Net or the Dark Net (TOR, I2P, Freenet) [9]. They proposed 11 different hyperlink selection methods to find the most relevant sites on a given topic. The crawler uses a classifier-based approach to follow hyperlinks. They used Support Vector Machine (SVM) with RBF kernel classifier to calculate scores of hyperlinks. These methods are based on the dynamic linear combination of hyperlink and web page (parent and destination page) classification. Their results show high precision, recall, Surface Net hits, and Dark Net hits.

Different from the work mentioned above, our crawler CrawlBot is a domain-specific pseudonymous crawler for extracting relevant information and provides the insight of the surface web and Dark Net (TOR). Our crawler hides its identity by using randomly generated pseudo-user agents and IP rotation after every single crawl. It uses the link and the page classifier to collect relevant sources. This framework is bound to the CSAM domain. However, it can be easily reconfigured for any other domain.

### 3 Proposed Methodology

To crawl as many as relevant web pages and to collect data from them more efficiently, we have implemented a crawler framework that can cover the surface web as well as the Dark web. The detailed framework of the proposed crawler is illustrated in Fig. 1. The crawler automatically connects to the Surface Net or the Dark Net based on the link encountered. With the help of a pseudonymous crawling strategy, the crawler navigates the web uninterruptedly. It exploits the trained ANN model which forms the backbone of two main classifiers 1) link classifier, 2) page classifier. The page classifier is used in association with the link classifier to compute a relevancy score for a specific link. The crawler is implemented using Python 3.7 and Requests 2.2 library.



**Fig. 1.** The detailed framework of CrawlBot

First, seed URLs corresponding to the topic of interest are scraped from the search engine like Candle, Torch, etc. and inserted into the URL frontier. A URL is picked up from the frontier and before visiting the corresponding web page the crawler acquires a unique identity by changing user-agent and IP address. Then the web page is fetched and parsed to extract the title, body text, and hyperlinks. The extracted data is used by the page classifier to determine the relevancy score of the downloaded web page. If the score exceeds the threshold value then the corresponding web page is stored in the database. Further, the link classifier assigns relevancy score to each hyperlink considering anchor text and local context of hyperlink. Hyperlinks with a score greater than the predefined threshold are inserted in the frontier. While appending URLs to the frontier hyperlink

with the highest score is appended first which ultimately increases the effectiveness of the crawler. In the coming part of this paper, a thorough explanation of each component of the proposed crawler is given.

### 3.1 Pseudonymous Crawling Strategy

Web crawlers are designed to retrieve data at a much faster pace and can dig deeper into the website than human beings. Multiple requests within a short period cause the server to overwhelm and reduce user experience. Thus, site administrators often employ anti-crawling mechanisms by finding crawling patterns such as a large number of requests from the same IP, usage of the old and same user-agent string. This hampers the overall performance of web crawlers. To tackle anti-crawling mechanisms and to increase the reach of our crawler, we have proposed a novel approach of pseudonymous and non-blocking crawling. The approach incorporates two major features 1) IP rotation, 2) User-agent rotation.

While crawling IP address of the crawler is exposed to site administrators and multiple requests from the same IP cause it to get blocked. Our crawler uses TOR expert bundle wrapper as a proxy and hence routes all the requests through TOR. The TOR expert bundle comes with two ports viz. 9050 and 9051. TOR routes all its requests through port 9050 and provides controlling capability through port 9051. Before making a connection to the server, a signal is sent to the TOR controller for the new IP address. It re-establishes the new circuit and thereby provides a new IP to the crawler.

Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:75.0) Gecko/20100101  
Firefox/75.0

**Fig. 2.** An example of an user-agent

User-agent strings transferred during web transactions send client system configuration details to the requested server [14]. If the user-agent is outdated or not set properly, then the server does not provide the content requested. We tested our crawler without setting an appropriate user-agent and observed a low harvest rate. To hide the real user-agent of the crawler, it selects a fake user-agent from the pool of current user-agents before requesting a website. An example of a user-agent is shown in Fig. 2. This approach eventually provides multiple unique identities to our crawler and thus reduces the probability of getting blocked.

### 3.2 Page Classifier

During crawling our crawler comes across various kinds of web pages. The key is to cover a wide range of web pages related to the domain of interest while discarding irrelevant web pages and increase the overall harvest rate. This section describes the feature construction of a web page and classification mechanism.

After fetching, the page classifier classifies the web page as relevant or irrelevant. If the page is categorized as relevant, then it is marked as a flag and stored in a database. The

downloaded web page is parsed and title and body content are extracted. Then extracted content is converted to lowercase, stemmed, tagged with part of speech followed by lemmatization and converted to vectors. The *CountVectorizer* is used to convert text into an encoded vector with the help of the learned vocabulary of a specific topic.

The feature space of feature vectors of the web page is given as:

$$\text{FSP} = \{T, B\} \quad (1)$$

where T, B are feature vectors corresponding to title and body text of the web page respectively. The resulting vector is fed to the ANN classifier which then decides if the page is relevant to the topic or not. The detailed process for page classification is described in Algorithm 1.

---

**Algorithm 1:** Page Classification

---

**Input:** Fetched HTML web page

**Output:** *pageRelevancy* score of web page

```

1 page = fetchPage()
2 title, body = extractTitleBody(page)
3 pageText = mergeText(title, body)
4 processedText = textPreprocessing(pageText)
5 pageVector = countVectorizer(processedText)
6 pageRelevancy = classifier(pageVector)

```

---

### 3.3 Link Classifier

In the context of CSAM, myriad web pages contain irrelevant hyperlinks which lead to pages which are the least interesting to Law Enforcement. This section sheds light upon the novel link selection approach combined with the page classifier, which selects links accurately.

The focused crawlers generally rely on the link context. topic-specific crawling works such as De Bra and Post [15] and Chakrabarti et al. [16]. applied the concept of link context in which the content of the entire page is treated as the context of the hyperlink embedded in it. SharkSearch [17] utilized the anchor text and some of the text in its neighborhood to determine the benefit of following the corresponding link. Iliou et al. [8] considered anchor text, terms within the URL and text window around the anchor text while calculating the relevancy of the hyperlink. Most of the sites encountered by our crawler are dark sites. Typically onion URLs contain automatically generated 16 (TOR version2) or 56 (TOR version3) character alpha-semi-numeric hashes which do not convey meaningful information. Hence crawler cannot rely on terms within the URLs and feature space of feature vector of the hyperlink is given as:

$$\text{FSL} = \{A, L\} \quad (2)$$

where A, L are feature vectors corresponding to anchor text and local context of the hyperlink respectively. This feature space is used by the link classifier to calculate *linkRelevancy*. This score is not sufficient to make a decision and thus the global context



of the parent page is taken into consideration. Finally, *linkScore* of each hyperlink is calculated using a static linear combination of *linkRelevancy* and *pageRelevancy*. The score i.e. *linkScore* of each hyperlink is computed as:

$$\text{linkScore} = \alpha \times \text{linkRelevancy} + (1 - \alpha) \times \text{pageRelevancy} \quad (3)$$

where  $\alpha$  is the relative weight assigned to hyperlink relevancy score. The value of  $\alpha$  plays an important role in selection of hyperlinks. Experimentally it is found that  $\alpha = 0.6$  yields satisfactory results. Then hyperlinks with *linkScore* greater than predefined *linkThreshold* are added to a queue. This queue is then sorted using *linkPrioritizing* and appended to URL frontier. The process of link classification is explained in Algorithm 2.

---

**Algorithm 2:** Link Classification
 

---

**Input:** Fetched HTML web page and *pageRelevancy* score  
**Output:** *linkScore* of hyperlinks

```

1 page = fetchPage()
2 relevantLinks = createQueue()
3 links = extractLinks(page)
4 for link in links do
5     anchorText = extractAnchorText(link)
6     anchorScore = classifier(anchorText)
7     localContext = extractLocalContext(link)
8     locoScore = classifier(localContext)
9     linkRelevancy = 0.5 × anchorScore + 0.5 × locoScore
    // Calculate static linear combination score using formula
    given in Eq.3
10    linkScore = statLinearCombination(linkRelevancy, pageRelevancy)
11    if linkScore > linkThreshold then
12        | relevantLinks.add(link)
    // sort relevantLinks in ascending order according to linkScore
13 linkPrioritizing(relevantLinks)
```

---

The ANN-based classifier forms the fulcrum of CrawlBot, which helps in the page classifier as well as the link classifier. The classifier is trained on the pornography dataset. The dataset contains 46,719 positive samples and 54,968 negative samples. The dataset is parsed, its textual content is extracted, tokenization, stop words and punctuation removal, part of speech tagging and lemmatization is applied. The *CountVectorizer* weighing scheme is applied to convert textual data into feature vectors. Then, the classifier is trained using a fully-connected Artificial Neural Network. Since other domains, besides CSAM, might be of interest, the classifier can be easily re-trained on a new dataset related to the topic of interest. The next section provides the evaluation results for assessing the effectiveness of the proposed crawler.



## 4 Evaluation

### 4.1 Selection of a Classifier

The performance of various classifier approaches on 5-fold cross validation with *CountVectorizer* as a feature weighting scheme is depicted in the Table 1. The data for mentioned approaches is the same and the same text preprocessing is applied to data. The number of features is kept constant for all approaches to assess the performance of each approach.

**Table 1.** Performance of classification algorithms

Weighting scheme	Classifier	5-fold cross validation accuracy (in %)
<i>CountVectorizer</i>	Multinomial NB	95.18
<i>CountVectorizer</i>	SVM	95.58
<i>CountVectorizer</i>	ANN	96.51

The Multinomial Naive Bayes classifier is used to classify relevant and irrelevant web pages. It uses the multinomial distribution on the features. The classifier is tested using *CountVectorizer* weighting scheme. This approach is not suitable as accuracy is not desirable. Then SVM classifier was tested which improved the accuracy over the previous approach. Further, *CountVectorizer* with ANN classifier is trained and this approach boosted up the accuracy to 96.51%. Comparing the accuracy values, we got significant improvement with ANN + *CountVectorizer* compared to Multinomial Naive Bayes + *CountVectorizer*. This shows the efficiency of the chosen classifier in the categorization task.

### 4.2 Nature of Darknet and It's Interconnectivity

The experiments carried out for a different number of web pages shows that the Dark Net and the Surface Net are interconnected for the CSAM domain. Table 2 represents the number of surface web links found while crawling the Dark Net. These results prove that several darknet links contain hyperlinks directed to surface web sites. The proposed crawler does not get obstructed by this interconnectivity and seamlessly crawls the Dark Net as well as the surface web during a single crawl session.

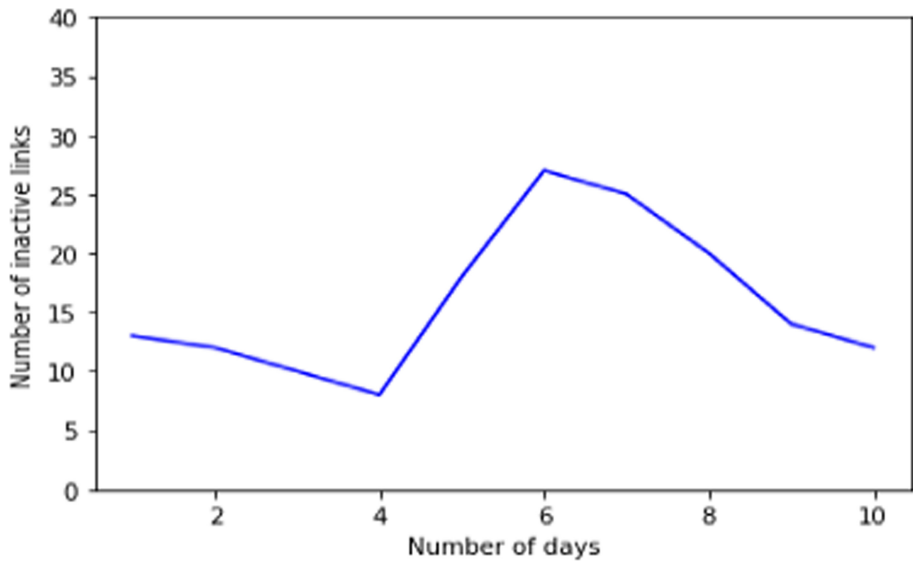
Another experiment conducted to monitor the nature of the Dark Net. The pro-posed crawler was run for ten consecutive days on the same set of the Dark Net links. The graph shown in Fig. 3 represents a very unstable nature of the Dark Net. The Dark Net links also known as the hidden services often go offline and recuperate under a new domain name [18]. Hence it is very important to acquire data of every new hidden service as soon as it appears to analyze its online activity. TOR Hidden Services related to CSAM captured by our crawler are given in Table 3. It shows the format of links of the Dark Net.

**Table 2.** Interconnectivity of the Dark Net and the Surface Net

Pages crawled	Links found	Surface net links	Dark net links
100	844	104	740
200	1391	112	1279
300	1858	123	1735
500	2048	125	1923

**Table 3.** Snapshot of crawled links

Onion links	Topic
z4c4kdaf42gem4x3fksjjan7ecjfq5zchoa7xpi7ujfiu2jqrzphqad.onion	Women abuse
s3icn6jrwkjov4pknz3iilcbuy7ahgeu7af7gqd7bjkf7gywbj44weid.onion	Pedophile
videocp3e3d65efnafl5jt3kqcavav755e7o7dvbnjcjynsmxhlrwpad.onion	Pedophile
nkmxbnup44toysy5na7zqoq2bomuwt34akapfp62lbscdptaiim7void.onion	CSAM



**Fig. 3.** Unstable nature of the dark web

**4.3 Harvest Rate**

This section presents the findings of experiments carried out to determine the link harvesting capability of the crawler. Many web pages contain hyperlinks that do not always point to related web pages and often misguide the crawler. The value of *linkThreshold* plays the main role in selecting relevant links efficiently. To explain this, Table 4 provides

the number of relevant and irrelevant links found by the crawler. The experiment was conducted by changing the value of *linkThreshold* while keeping the number of crawled pages to 200. The link-Threshold value 0.4 appears to be effective in properly distinguishing between related and non-related links. Hence it is obvious that the usage of the page classifier in combination with the link classifier and proper choice of *linkThreshold* improves the total effectiveness of the crawler.

**Table 4.** Effectiveness of crawler in link harvesting

Values of <i>linkThreshold</i>	Relevant links	Irrelevant links	Total links
0.4	768	623	1391
0.5	723	668	1391
0.6	682	709	1391
0.7	659	732	1391
0.8	627	764	1391

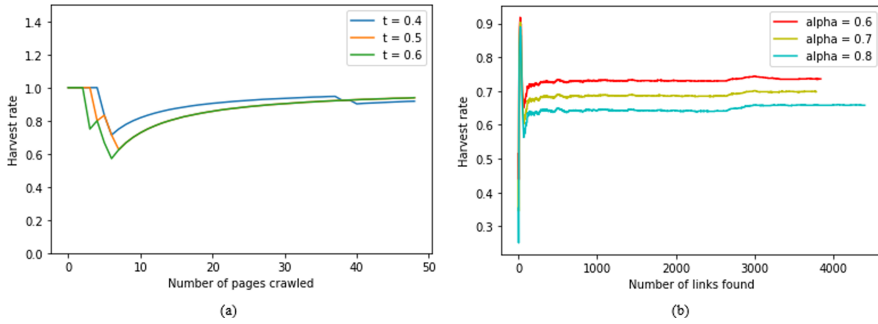
The performance of the focused crawler is generally measured by the harvest rate. It determines the fraction of the crawled web pages that are relevant to a given topic. To judge whether a given web page is related to a given topic or not, the crawler relies on the classifier. As discussed previously, the ANN-based classifier performs well with the *CountVectorizer* weighting scheme. We carried out three crawling sessions with different values of *pageThreshold* designated as ‘t’ in Fig. 4(a). The value of  $\alpha$  is set to 0.4 and the crawler was allowed to crawl 50 pages and the harvest rate of each session was recorded. The Harvest Rate is computed as:

$$HarvestRate = \frac{1}{N} \sum_{n=i}^N PageRelevancy_i \quad (4)$$

where N is the number of pages crawled and the *pageRelevancy* is a binary score of a web page. As discussed in Sect. 3.3, the overall link Score of a hyperlink depends upon the value of  $\alpha$ . The proper choice of  $\alpha$  ultimately determines the overall link harvest rate of the crawler. The different values of link harvest rate were recorded by changing  $\alpha$  value. This experiment was conducted by setting *linkThreshold* value to 0.4 and results are depicted in Fig. 4(b). The link harvest rate is a fraction of links found by the crawler which are relevant to a topic. The Harvest Rate for link is calculated as:

$$HarvestRatelink = \frac{1}{N} \sum_{n=i}^N LinkScore_i \quad (5)$$

where N is the number of links found during crawling and the *linkScore* is a binary score of a hyperlink. It is obvious from the graph shown in Fig. 4(b) that the value of  $\alpha = 0.6$  improves the harvest rate of links. Additionally, links collected by the crawler when manually investigated revealed that the values 0.7 and 0.8 gives some fraction of off-topic links.



**Fig. 4.** (a) The comparison of harvest rate for different values of  $t$ , (b) The comparison of link harvest rate for different values of  $\alpha$

## 5 Conclusion and Future Work

This work proposed a web crawler framework capable of seamlessly crawling the Surface Net as well as the Dark Net (predominantly TOR) within a single crawl. It employed a method wherein the crawler can automatically connect to the Surface Net or the Dark Net, based on the type of link it encounters. With the help of a pseudonymous crawling strategy, the crawler can navigate the web uninterrupted. The crawler utilized the trained ANN model which formed the foundation of both the link classifier and the page classifier. The page classifier in association with the link classifier helped to compute a relevancy score for a specific link. The evaluation results showed the effectiveness of the crawler and also how ANN-based classifier is superior over other classification approaches.

The proposed crawler can be employed by government bodies to track malicious content on the internet, gather specific types of information from websites and analyse social media, blogs and forum data for illicit material such as drugs, child pornography, and human trafficking. The crawler can be easily re-trained to explore other domains of the Dark Web as well.

Particularly, in the CSAM domain, many pages contain many images and thus crawler fails to classify the web page accurately. This bottleneck of the crawler will be eliminated in the future work with the incorporation of Image Classification which will greatly enhance the performance of the crawler which is currently developed to extract the content and determine its relevance using Text Classifier. The crawler can also be made more robust for crawling other dark web sites such as I2P and FreeNet.

**Acknowledgement.** The authors are very grateful to the Center of Excellence in Complex and Nonlinear Dynamical Systems (COE-CNDS) under TEQIP-III funding for providing the infrastructure necessary to develop the crawler. We would also like to thank the members of the lab for their worthy feedback and comments.

## References

1. Liggett, R.: Commercial child sexual abuse markets on the dark web. White Paper. School of Criminal Justice, Michigan State University.
2. McCoy, D., Bauer, K., Grunwald, D., Kohno, T., Sicker, D.: Shining light in dark places: Understanding the Tor network. In: Borisov, N., Goldberg, I. (Eds.) *Privacy Enhancing Technologies. PETS 2008. Lecture Notes in Computer Science*, vol. 5134. Springer, Berlin, Heidelberg. <https://doi.org/10.1007/978-3-540-70630-45> (2008)
3. Finklea, K.: Dark Web. Congressional Research Service (2017)
4. Thorn: <https://www.thorn.org>
5. Spotlight: <https://www.thorn.org/spotlight/>
6. Safer: <https://getsafier.io/about-safer/>
7. Zhao, F., Zhou, J., Nie, C., Huang, H., Jin, H.: SmartCrawler: a two-stage crawler for efficiently harvesting deep-web interfaces. In: *IEEE Transactions on Services Computing*, vol. 9, no. 4, pp. 608–620, IEEE. <https://doi.org/10.1109/TSC.2015.2414931> (2016)
8. Iliou, C., Kalpakis, G., Tsikrika, T., Vrochidis, S., Kompatsiaris, I.: Hybrid focused crawling on the surface and the dark web. *EURASIP J. Inf. Secur.* **2017**(1), 1–13 (2017). <https://doi.org/10.1186/s13635-017-0064-5>
9. Ali, A., et al.: TOR vs I2P: a comparative study. In: 2016 IEEE International Conference on Industrial Technology (ICIT), pp. 1748–1751. IEEE, Taipei. <https://doi.org/10.1109/ICIT.2016.7475027> (2016)
10. Mani, A., Wilson-Brown, T., Jansen, R., Johnson, A., Sherr, M.: Understanding Tor usage with privacy-preserving measurement. In: *IMC 2018: Proceedings of the Internet Measurement Conference 2018*, pp. 175–187. ACM, New York. <https://doi.org/10.1145/3278532.3278549> (2018)
11. Pant, G., Srinivasan, P., Menczer, F.: Crawling the web. In: *Web Dynamics*. Springer, Berlin (2004)
12. ACHE Crawler Documentation, Release latest. New York University (April 2019)
13. Li, Y., Wang, Y., Du, J.: E-FFC: an enhanced form-focused crawler for domain-specific deep web databases. *J. Intell. Inform. Syst.* **40**, 159–184 (2013)
14. Kline, J., Cahn, A., Barford, P., Sommers, J.: On the structure and characteristics of user agent string. In: *IMC 2017: Proceedings of the 2017 Internet Measurement Conference*, pp. 184–190. ACM, New York. <https://doi.org/10.1145/3131365.3131406> (2017)
15. De Bra, P., Post, R.D.J.: Information retrieval in the world-wide web: making client-based searching feasible. *Comput. Netw. ISDN Syst.* **27**(2), 183–192 (1994)
16. Chakrabarti, S., van den Berg, M., Dom, B.: Focused crawling: a new approach to topic-specific web resource discovery. *Comput. Netw.* **31**(11–16), 1623–1640 (1999). [https://doi.org/10.1016/S1389-1286\(99\)00052-3](https://doi.org/10.1016/S1389-1286(99)00052-3)
17. Hersovici, M., Jacovi, M., Maarek, Y., Pelleg, D., Shtalhim, M., Ur, S.: The shark-search algorithm. An application: tailored Web site mapping. *Comput. Netw. ISDN Syst.* **30**(1–7), 317–326 (1998). [https://doi.org/10.1016/S0169-7552\(98\)00038-5](https://doi.org/10.1016/S0169-7552(98)00038-5)
18. Chertoff, M., Simon, T.: The impact of the dark web on internet governance and cyber security. In: *Global Commission on Internet Governance Paper Series: No. 6* (2015)