SoftwareX 29 (2025) 102070

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Web2Vec: A python library for website-to-vector transformation

Damian Frąszczak *, Edyta Frąszczak

Military University of Technology, gen. Sylwestra Kaliskiego 2, 00-908 Warsaw, Poland

ARTICLE INFO

Keywords:
Website processing
Website feature extraction
Website to vector
Data extraction
Web scraping

ABSTRACT

Web2Vec is a Python library designed to simplify website analysis by converting websites into vector representations through feature extraction from their content and structure. Utilizing Scrapy-based web crawlers, it automates data collection and supports both single-page analysis and large-scale crawling. This flexibility allows users to adapt the library to their specific needs, whether for quick, focused analysis or systematic data collection. Integrating over 200 website parameters into a single, easy-to-use framework, Web2Vec simplifies analytical tasks, making it a valuable resource across various fields. By serving as a centralized code repository for researchers, it eliminates the need to repeatedly implement similar code, providing an all-in-one integrator to streamline workflows and save time.

Metadata

This ancillary data table is required for the sub-version of the codebase. Please replace the italicized text in the right column with the correct information about your current code and leave the left column untouched.

Nr	Code metadata description	*Please fill in this column*
C1	Current code version	*0.1.3*
C2	Permanent link to code/repository used for this code version	https://github.com/damianfraszczak/web2vec
C3	Permanent link to reproducible capsule	https://codeocean.com/capsule/1087564/tree
C4	Legal code license	*MIT*
C5	The code versioning system used	*GIT*
C6	Software code languages, tools and services used	*python*
C7	Compilation requirements, operating environments and dependencies	*python, networkx, scrapy, beautifulsoup4 scipy, python-whois, dnspython, geoip2, tldextract, selenium, webdriver-manager, dnspython, pydantic, pydantic_settings*
C8	If available, link to developer documentation/manual	https://github.com/damianfraszczak/web2vec/blob/master/README.md https://web2vec.readthedocs.io/en/latest/index.html
C9	Support email for questions	damian.fraszczak@wat.edu.pl

1. Motivation and significance

In the actual world, also known as the digital age, the internet is a vast repository of information, with >1 billion websites, of which 201 million are active,[1] and new ones are added to this number every day. Websites serve various purposes, from sharing social media updates to conducting business transactions, resulting in the generation of massive amounts of data every day [1,2]. Extracting and processing information from these websites, commonly known as web crawling, scraping, website data extracting, and vectorization, are key techniques within the broader scope of website processing and web mining [3–6]. Those methods enable the automated retrieval of data from web pages and the conversion of unstructured web data into structured formats for storage and processing, making them extremely valuable for various applications such as data analysis, market research, and content aggregation [5,7,8]. Organizations can utilize those methods to gain insights, monitor trends, and make data-driven decisions. Additionally, these techniques play a crucial role in cybersecurity by detecting and preventing phishing websites and misinformation [9–12]. By identifying and cataloging legitimate and fraudulent websites, robust databases can be built to help differentiate between trustworthy and malicious sources, enhancing the

* Corresponding author.
E-mail addresses: damian.fraszczak@wat.edu.pl (D. Frąszczak), u64579@student.wat.edu.pl (E. Frąszczak).

[1] https://themeisle.com/blog/how-many-websites-are-there/

https://doi.org/10.1016/j.softx.2025.102070
Received 27 July 2024; Received in revised form 23 December 2024; Accepted 23 January 2025
Available online 31 January 2025

Table 1
Literature overview of data collection methods, technologies, datasets, and code availability.

Reference	Features extracted	Technologies used	Dataset	Code available
[20]	URL, HTML content, Third-party services, HTTP	Not provided	Public dataset	No
[21]	URL, HTTP	Not provided	Shared	No
[22]	URL	Not provided	Public dataset	No
[23]	URL	Not provided	Public dataset	No
[24]	URL, HTML content, Third-party services, HTTP	Python, BeautifulSoup	Shared	Yes
[26]	URL, HTML content, Third-party services	Java, JSoup	Not shared	No
[27]	URL, HTML content	Python, BeautifulSoup	Shared	No
[28]	URL, HTML content	Python, BeautifulSoup	Not shared	No
[29]	URL	Not provided	Public dataset	No
[30]	URL, HTML content	Not provided	Not shared	No
[31]	URL	Not provided	Shared	No
[32]	URL	Not provided	Public dataset	No
[33]	HTML	Python	Not shared	No
[34]	URL, HTML content	Not provided	Not shared	No
[35]	HTML content	Not provided	Not shared	No
[36]	URL	Not provided	Not shared	No
[37]	URL	Not provided	Shared	No
[38]	URL, HTML content	Python, Selenium	Shared	No

reliability of online information.

Numerous materials, such as books [13–15], academic articles [5,6,16], and technical resources[2,3,4] discuss how data is collected for research purposes, often specifying which available datasets were utilized [17–23]. Some papers [24] even provide code snippets to illustrate the feature extraction or data collection process. However, a significant gap remains: there are no dedicated open-source solutions available for extracting website parameters. Although researchers often rely on custom scripts and individualized methods to perform this task, they frequently end up "reinventing the wheel" [3,5,6,25]. This trend highlights the inefficiency of current practices, as valuable time is spent on technical implementation rather than on creative or analytical aspects of research. Table 1 summarizes a review of the literature, highlighting the current methods used for data collection, including the extracted features group, technologies employed, the datasets referenced, and whether code snippets are shared.

Furthermore, services like WHOIS,[5] SimilarWeb,[6] and the Google Search[7] Index do not provide free access through APIs, offering only HTML responses instead. These responses require parsing and analysis, often involving advanced techniques like rendering HTML documents with tools such as Selenium,[8] which adds complexity and demands specialized knowledge for effective data collection.

The introduction of *Web2Vec* aims to improve web data extraction by offering an open-source and accessible tool that integrates multiple techniques into a single platform. This unified approach simplifies the processes of data scraping, mining, and collection, which are essential for building datasets used in machine learning and other applications [39]. By streamlining these efforts, *Web2Vec* enhances efficiency, reduces redundancy, minimizes errors, and fosters collaboration within the research community. Furthermore, its integration of free services makes it especially valuable for non-commercial projects, accelerating research and improving the creation and sharing of datasets, which are critical resources in today's digital landscape.

[2] https://github.com/BLITBLAZERS/PHISHGUARDIAN/blob/main/Phishing%20Url%20Detection.ipynb

[3] https://github.com/gangeshbaskerr/Phishing-Website-Detection/blob/main/Phishing%20Website%20Detection_Models%20%26%20Training.ipynb

[4] https://github.com/gangeshbaskerr/Phishing-Website-Detection

[5] https://who.is/

[6] https://www.similarweb.com/

[7] Google Search Index is a database that organizes and ranks website information for search results.

[8] https://www.selenium.dev/

2. Software description

Web2Vec is a comprehensive library designed to convert websites into vector representation. It can be described mathematically as transforming the unstructured content of the page into a structured format that captures its key characteristics. It works by taking a website Was input and analyzing its features, such as text content, structure, links, HTTP response, HTML body, etc. Moreover, additional features from external services can be provided including metrics like page ranking, traffic data, social media mentions, or SEO-related metrics. Together with the features extracted directly from the website, a comprehensive set of parameters is created. Technically, each website $W$is represented by a set of features $\{F_i\}$where each feature $F_i$represents a specific characteristic of the website. These features are then mapped to a vector $V = [v_1, v_2, ..., v_n]$, where v_i represents the value associated with the featureF_i. Thus, Web2Vec converts websites into structured vectors that can be utilized for further computational tasks. This process can be achieved in two modes: based on pre-generated HTML snapshots or through active crawling of selected websites. For active crawling, Scrapy[9]-based crawlers have been implemented due to their efficiency in handling large-scale web scraping tasks, flexibility, and extensive support for managing complex extraction workflows [6]. Fig. 1 illustrates the overall process of processing individual web pages and crawling through multiple websites with Web2Vec components. The components provided as part of the solution are highlighted in bold, including a dedicated Scrapy spider and extractors, which are responsible for transforming web pages into their structured representations.

Provided ready-to-use components make *Web2Vec* highly accessible, even for less experienced researchers, and crucial for website analysis tasks like SEO [3], disinformation detection [11,40], and phishing identification [9].

2.1. Software functionalities

The purpose of *Web2Vec* is to serve as a centralized repository for a wide range of methods related to website processing, providing a single location where such solutions and techniques are collected and made accessible to enhance both academic research and industry applications. In the current version, our solution is capable of extracting 211 different parameters from websites (see summary presented in Table 2), providing extensive data for in-depth analysis.

[9] https://scrapy.org/

Table 2
Web2Vec extracting parameters summary.

Features Type	Number of Features	Example Features
URL Lexical features	83	is_ip, count_dollar_parameters
HTML Body	40	has_forms, contains_obfuscated_scripts
HTTP Response	22	uses_https, missing_x_frame_options
Whois	19	creation_date, expiration_date
SimilarWeb	16	country_rank, traffic_sources
URLHaus	8	threat, tags
SSL certificate	8	is_verified, is_trusted
PhishTank	6	is_phishing, verified
DNS	3	ttl, record_type
Geolocalization	2	country_code, asn
GoogleIndex	2	is_indexed, position
PageRank	1	position
OpenPhish	1	is_phishing

Web2Vec is now available in the official Python packages repository, PyPI. To install it, use the command: pip install web2vec. This way, the community has easy access to the latest version of the library, which allows for quicker bug resolution and implementation of new methods. By making the code available to the public, it becomes easier to maintain this project with the help of the open-source community.

2.2. Sample code snippets analysis

After installing the library, it needs to be configured. It comes with a default set of parameters, but it is important to update them because, for certain types of gathered files, a temporary directory is used. For external feeds or a database, it is recommended to set the directory to a non-temporary location to improve effectiveness. It is important to note that the library can be configured using environmental variables, .env files, or directly through the code. A sample configuration of the .env file is shown in Listing 1. This configuration sets the default library output path (WEB2VEC_DEFAULT_OUTPUT_PATH) and the API key for the OpenPageRank[10] service (WEB2VEC_OPEN_PAGE_RANK_API_KEY).

This process can be accomplished directly in the code using the os.environ module, as demonstrated in Listing 2. Additional information about library configuration can be found in the documentation.

The library offers two modes of usage. The first involves crawling websites and extracting parameters. To facilitate this, the Scrapy process is invoked with a dedicated spider object – Web2VecSpider. It accepts three main parameters and the sample crapping and extracting script is presented in Listing 3:

- start_urls: an array of websites to process.
- allowed_domains: an array of domains to process for links.
- Extractors: an array of data extractors to use.

It is configured to process http://quotes.toscrape.com/, visit only quotes.toscrape.com, and extract all available website parameters by indicating a utility array of all available extractors - w2v.ALL_EXTRACTORS.As a result, in the crawling directory, each processed page will be stored in a separate file as JSON with the following keys and sample content as presented in:

- url: the processed URL.
- title: website title extracted from HTML.
- html: HTTP response text attribute.
- response_headers: HTTP response headers.
- status_code: HTTP response status code.
- extractors: a dictionary with extractor's results.

Listing 4

The second approach follows Python's best practices to extract website parameters by directly invoking extractors. To facilitate this, all modules are exported in the library's main module, providing immediate access to all available techniques. Users can import the module using import web2vec as w2v and then utilize it with that import. For example, to get data from the *SimilarWeb* service for a given domain get_similar_web_features method should be invoked as presented in Listing 5.

It is worth mentioning that each extractor response is a Python standard dataclass, which makes it easy to understand the available parameters and their meanings based on their names. For example, to gather geographic-related information for a given URL, the schema presented in Listing 6 will be returned as the resulting object. Library users immediately know that it returns the country code as a string object and the ASN number is an int object.

3. Illustrative examples

The presented solution offers a convenient method for collecting and extracting features from web pages. To showcase the usefulness of *Web2Vec*, several examples using Jupyter Notebook have been prepared, some of which are partially demonstrated in this paper. One such example involves updating or building a new dataset of phishing websites. Often, solutions such as [18–23] rely on publicly available datasets, for example phishing websites [41]. However, static datasets have limitations, including outdated data and missing recent threats. Therefore, they should be regularly updated, or new parameters should be evaluated. This is where the proposed solution provides a remedy for this issue. Appendix A presents sample source code that takes a list of legitimate and phishing websites and processes them to extract their features. It transforms the websites into vectors that can be further utilized in machine learning methods. Fig. 2a presents a sample input content, and Fig. 2b shows a sample of the output data. This example extracts 139 different features, demonstrating the simplicity of use and showing that features can be easily extracted with minimal effort.

Another notable example is the utilization of the built-in method for creating a visual representation of related web pages in the form of a directed graph – see Appendix C. This functionality enables comprehensive analysis using libraries designed for processing graph structures [42,43]. This visualization capability improves the understanding of relationships and connections between various web pages, providing valuable insights for research and analytical purposes. A sample result of the visualization is presented in Appendix D. It provides a network of websites in a graph and a summary in a table.

[10] https://www.domcop.com/openpagerank/what-is-openpagerank

```
export WEB2VEC_DEFAULT_OUTPUT_PATH=
export WEB2VEC_OPEN_PAGE_RANK_API_KEY=XXXXX
```

Listing 1. A sample .env file configuration for Web2Vec.

```
import os

os.environ['WEB2VEC_DEFAULT_OUTPUT_PATH'] = '/home/admin/crawler/output'
os.environ['WEB2VEC_OPEN_PAGE_RANK_API_KEY'] = 'XXXXX'
```

Listing 2. A sample Web2Vec configuration via Python os.environ module.

```
import web2vec as w2v
from scrapy.crawler import CrawlerProcess

process = CrawlerProcess(
    settings={
        "FEEDS": {
            os.path.join(w2v.config.crawler_output_path, "output.json"): {
                "format": "json",
                "encoding": "utf8",
            }
        },
        "DEPTH_LIMIT": 1,
        "LOG_LEVEL": "INFO",
    }
)

process.crawl(
    w2v.Web2VecSpider,
    start_urls=["http://quotes.toscrape.com/"], # pages to process
    allowed_domains=["quotes.toscrape.com"], # domains to process for links
    extractors=w2v.ALL_EXTRACTORS, # extractors to use
)
process.start()
```

Listing 3. A sample scrapping script developed with Web2Vec.

These examples emphasize the practicality and versatility of *Web2Vec* in real-world applications, demonstrating its ability to simplify the intricate process of web data extraction and analysis.

4. Impact

Web2Vec fills a crucial gap in the landscape of website analysis tools by addressing the lack of dedicated open-source solutions for extracting website parameters. While researchers often rely on custom scripts or ad hoc methods, this approach leads to redundancy and inefficiency, as similar tools are repeatedly developed independently. By providing an all-in-one, integrated framework, *Web2Vec* shifts the focus from technical implementation to the creative and analytical aspects of research. This streamlined approach not only enhances productivity but also fosters collaboration and resource-sharing within the research community. One of the most significant advantages of *Web2Vec* is its ability to easily create and share datasets, which is increasingly crucial in both academic and industry applications.

A critical application of *Web2Vec* is in the field of cybersecurity. By integrating with dedicated phishing or disinformation detection services and offering solutions for comprehensive web data analysis, *Web2Vec* helps in identifying and mitigating online threats. Its capability to visualize related web pages as directed graphs enhances the understanding of phishing networks, making it essential for cybersecurity professionals.

In addition, this library is designed to serve as a repository for new methods, encouraging researchers to contribute their techniques to the project. It will allow them to quickly disseminate their results and make their work easier and more noticeable. The solution is part of the doctoral research of one of the authors, ensuring its continued development and relevance, at least for the duration of the research.

5. Conclusions

Web2Vec is a Python library available on PyPI, designed to provide a versatile solution for website analysis and data extraction. It supports both single-page analysis and large-scale crawling, allowing users to adapt it to their specific needs—whether performing a quick, focused analysis or deploying it as a robust tool for systematic data collection. Integrating over 200 website parameters into a single, user-friendly framework simplifies complex analytical tasks, making it a valuable resource across various fields. Through ongoing development and community engagement, it acts as a code repository for transforming websites into vector representations, enabling seamless integration of new website features and indicators while minimizing the technical effort required for their implementation. Currently, work is underway to integrate tools for analyzing technologies used on websites, such as BuildWith[11] or WhatWeb.[12]

[11] https://github.com/richardpenman/builtwith
[12] https://github.com/urbanadventurer/WhatWeb

```
{
    "url": "http://quotes.toscrape.com/",
    "title": "Quotes to Scrape",
    "html": "HTML body, removed too big to show",
    "response_headers": {
        "b'Content-Length'": "[b'11054']",
        "b'Date'": "[b'Tue, 23 Jul 2024 06:05:10 GMT']",
        "b'Content-Type'": "[b'text/html; charset=utf-8']"
    },
    "status_code": 200,
    "extractors": [
        {
            "name": "DNSFeatures",
            "result": {
                "domain": "quotes.toscrape.com",
                "records": [
                    {
                        "record_type": "A",
                        "ttl": 225,
                        "values": [
                            "35.211.122.109"
                        ]
                    },
                    {
                        "record_type": "CNAME",
                        "ttl": 225,
                        "values": [
                            "ingress.prod-01.gcp.infra.zyte.group."
                        ]
                    }
                ]
            }
        }
    ]
}
```


Listing 4. A sample of the processed website file content using the DNS extractor.

```
import web2vec as w2v

domain_to_check = "down.pcclear.com"
entry = w2v.get_similar_web_features(domain_to_check)
print(entry)
```

Listing 5. A sample of direct method invocation to gather specific parameters.

```
@dataclass
class URLGeoFeatures:
    url: str
    country_code: str
    asn: int
```

Listing 6. The URLGeoFeatures dataclass definition, representing geographic information associated with a URL.

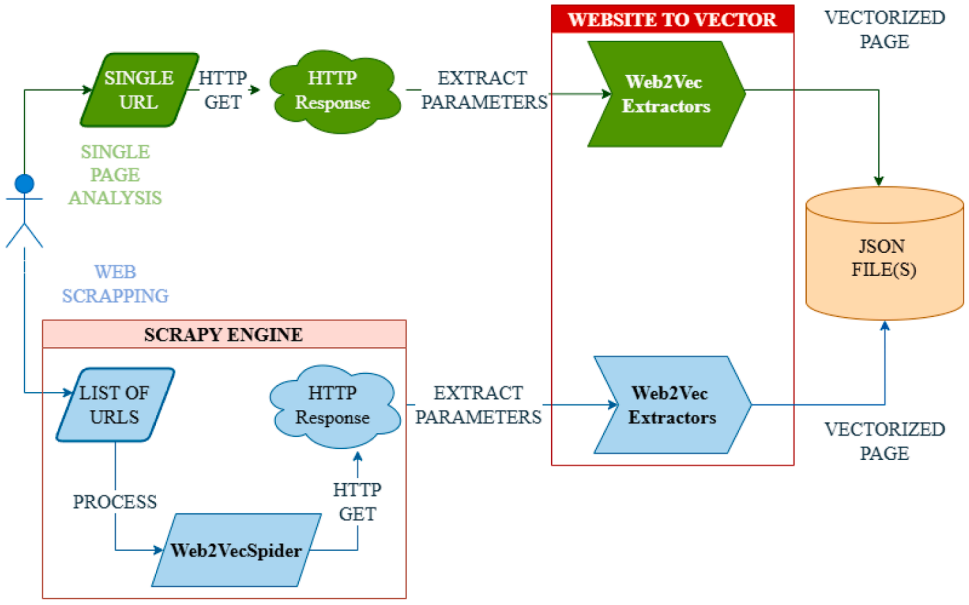


Fig. 1. Overview of components and processes for website processing and crawling using Web2Vec.

```
Domain,is_phish
sourceforge.net,0
indianexpress.com,0
temu.com,0
indeed.com,0
messenger.com,0
```

a)

```
Domain,HTML_contains_forms,HTML_contains_obfuscate
sourceforge.net,1,0,1,10231,9,22,164,25,255,0.0980
indianexpress.com,0,0,0,216,1,0,0,0,22,0.0,0.0,9.8
temu.com,0,0,0,384,0,0,0,24,305,0.0786885245901639
indeed.com,1,0,0,399,2,2,12,2,6,0.3333333333333333
messenger.com,1,0,0,885,3,5,42,17,62,0.27419354838
```

b)

Fig. 2. A sample of input content – a) a list of legitimate and phishing websites, b) a list of websites with extracted parameters.

CRediT authorship contribution statement

Damian Frąszczak: Writing – review & editing, Writing – original draft, Software, Project administration, Methodology, Investigation, Conceptualization. **Edyta Frąszczak:** Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Listing for new dataset generation

```
import csv

import pandas as pd
import web2vec as w2v

websites_parameters = {}

extractors = [
    w2v.HtmlBodyExtractor(),
    w2v.HttpResponseExtractor(),
    w2v.UrlLexicalExtractor(),
    w2v.CertificateExtractor(),
    w2v.OpenPhishExtractor(),
]
data = pd.read_csv("phish_websites.csv")
for domain, is_phish in zip(data['Domain'], data['is_phish']):
    url = f"https://{domain}"
    websites_parameters[domain] = w2v.process_extractors(url, extractors,

use_only_numerical=True)
    websites_parameters[domain]["is_phish"] = is_phish

with open('output.csv', 'w', newline='') as file:
    fieldnames = ["Domain"] + list(
        next(iter(websites_parameters.values())).keys())
    writer = csv.DictWriter(file, fieldnames=fieldnames)
    writer.writeheader()
    for domain, parameters in websites_parameters.items():
        row = {"Domain": domain}
        row.update(parameters)
        writer.writerow(row)
```

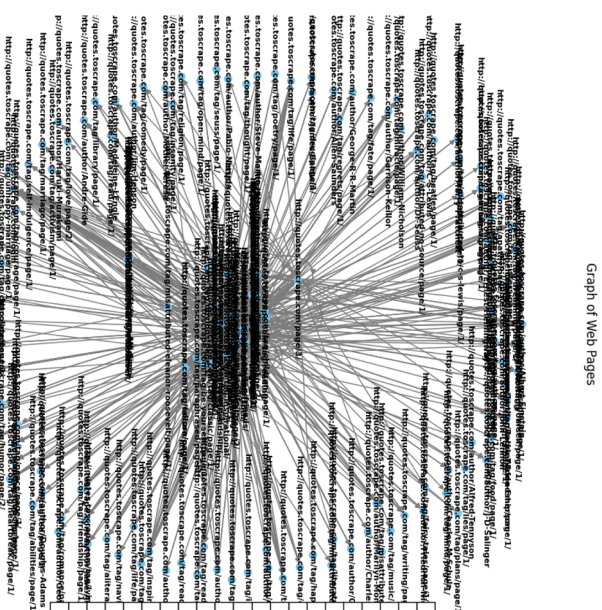
Appendix B. Dataset usage to train ML model

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report

df = pd.read_csv('output.csv')

X = df.drop(columns=['is_phish', 'Domain'])
y = df['is_phish']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

Appendix C. Visualizing website connections and doing network analysis

```
import networkx as nx
from matplotlib import pyplot as plt

def visualize_graph_with_centrality(graph: nx.Graph):
    """Visualize the graph of web pages with centrality."""
    degree_centrality = nx.degree_centrality(graph)

    sorted_nodes = sorted(
        degree_centrality.items(), key=lambda item: item[1], reverse=True
    )

    table_data = [["Node", "Degree Centrality"]]
    for node, centrality in sorted_nodes[:20]:
        table_data.append([str(node), f"{centrality:.4f}"])
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(24, 12))

    pos = nx.spring_layout(graph)
    nx.draw(
        graph,
        pos,
        with_labels=True,
        node_size=50,
        node_color="skyblue",
        font_size=8,
        font_weight="bold",
        edge_color="gray",
        ax=ax1,
    )
    ax1.set_title("Graph of Web Pages")

    ax2.axis("off")
    table = ax2.table(cellText=table_data, cellLoc="center", loc="center")
    table.auto_set_font_size(False)
    table.set_fontsize(10)
    table.scale(1.5, 1.5)
    ax2.set_title("Top 10 Degree Centrality of Nodes")

    plt.show()


G = w2v.build_graph(w2v.config.crawler_output_path,
allowed_domains=["quotes.toscrape.com"])
visualize_graph_with_centrality(G)
```

Appendix D. Result of visualizing website connections and doing network analysis



Top 10 Degree Centrality of Nodes

Node	Degree Centrality
http://quotes.toscrape.com/	0.5389
http://quotes.toscrape.com/tag/inspirational/	0.5030
http://quotes.toscrape.com/tag/life/	0.4850
http://quotes.toscrape.com/tag/books/	0.4850
http://quotes.toscrape.com/tag/love/	0.4671
http://quotes.toscrape.com/tag/humor/	0.4671
http://quotes.toscrape.com/tag/reading/	0.4311
http://quotes.toscrape.com/tag/friends/	0.3892
http://quotes.toscrape.com/tag/friendship/	0.3892
http://quotes.toscrape.com/tag/books/page/1/	0.3713

References

[1] Kavassalis P, Lelis S, Rafea M, Haridi S. What makes a web site popular? Commun ACM 2004;47(2):50–5. https://doi.org/10.1145/966389.966415.

[2] Algosaibi AA, Albahli S, Khasawneh SF, Melton A. Web evolution - the shift from information publishing to reasoning. Int J Artif Intell Appl 2017;8(6):11–28. https://doi.org/10.5121/ijaia.2017.8602.

[3] Zhang J, Qiang J, Zhou C. New Horizons in Web search, web data mining, and web-based applications. Appl Sci 2024;14(2):530. https://doi.org/10.3390/app14020530.

[4] Gheisari M, et al. Data mining techniques for web mining: a survey. Artif Intell Appl 2023;1(1):3–10. https://doi.org/10.47852/bonviewAIA2202290.
[5] Diouf R, Sarr EN, Sall O, Birregah B, Bousso M, Mbaye SN. Web scraping: state-of-the-art and areas of application. In: 2019 IEEE International Conference on Big Data (Big Data). IEEE; 2019. p. 6040–2. https://doi.org/10.1109/BigData47090.2019.9005594.
[6] Khder M. Web scraping or Web crawling: state of art, techniques, approaches and application. Int J Adv Soft Comput Its Appl 2021;13(3):145–68. https://doi.org/10.15849/IJASCA.211128.11.
[7] Magdziarz K, Frąszczak D. The architecture concepts for building highly scalable crawling cluster for data-driven on-page optimization. In: Proceedings of the 40th International Business Information Management Association (IBIMA). International Business Information Management; 2023. p. 451–9. https://doi.org/10.6084/m9.figshare.21909273.V1.
[8] Haddaway N. The use of web-scraping software in searching for grey literature. Grey J 2015;11:186–90.
[9] Frąszczak E, Fraszczak D. A review of a website phishing detection taxonomy. In: Proceedings of the 43rd International Business Information Management Association Conference (IBIMA). International Business Information Management; 2024. p. 455–64. https://doi.org/10.6084/m9.figshare.26345473.
[10] Alkhalil Z, Hewage C, Nawaf L, Khan I. Phishing attacks: a recent comprehensive study and a new anatomy. Front Comput Sci 2021;3:563060. https://doi.org/10.3389/fcomp.2021.563060.
[11] Frąszczak D. Fake news source detection – The State of the art survey for current problems and research. In: Proceedings of the 37th International Business Information Management Association (IBIMA). International Business Information Management; 2021. p. 11381–9. https://doi.org/10.6084/m9.figshare.16545675.
[12] Meel P, Vishwakarma DK. Fake news, rumor, information pollution in social media and web: a contemporary survey of state-of-the-arts, challenges and opportunities. Expert Syst Appl 2020;153:112986. https://doi.org/10.1016/j.eswa.2019.112986.
[13] Mitchell RE. Web scraping with python: collecting more data from the modern web. 2nd edition. Sebastopol, CA: O'Reilly Media; 2018.
[14] Bonzanini M. Mastering social media mining with python: acquire and analyze data from all corners of the social web with python. Birmingham, UK: Packt Publishing; 2016.
[15] Chapagain A. Hands-on web scraping with python: perform advanced scraping operations using various python libraries and tools such as selenium, regex, and others. Birmingham: Packt Publishing; 2019. Limited.
[16] Kumar M, Bhatia R, Rattan D. A survey of web crawlers for information retrieval. WIREs Data Min Knowl Discov 2017;7(6):e1218. https://doi.org/10.1002/widm.1218.
[17] Ubing AA, Kamilia S, Abdullah A, Jhanjhi N, Supramaniam M. Phishing website detection: aan improved accuracy through feature selection and ensemble learning. Int J Adv Comput Sci Appl 2019;10(1). https://doi.org/10.14569/IJACSA.2019.0100133.
[18] Gundla SC, et al. A feature extraction approach for the detection of phishing websites using machine learning. J Circuit Syst Comput 2023:2450031. https://doi.org/10.1142/S0218126624500312.
[19] Moedjahedy J, Setyanto A, Alarfaj FK, Alreshoodi M. CCrFS: ccombine correlation features selection for detecting phishing websites using machine learning. Future Internet 2022;14(8):229. https://doi.org/10.3390/fi14080229.
[20] Vrbančič G, Fister I, Podgorelec V. Swarm intelligence approaches for parameter setting of deep learning neural network: ccase study on phishing websites classification. In: Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics. Novi Sad Serbia: ACM; 2018. p. 1–8. https://doi.org/10.1145/3227609.3227655.
[21] Dutta AK. Detecting phishing websites using machine learning technique. PLOS ONE 2021;16(10):e0258361. https://doi.org/10.1371/journal.pone.0258361.
[22] Minh Linh D, Hung HD, Chau HMinh, Vu QSy, Tran T-N. Real-time phishing detection using deep learning methods by extensions. Int J Electr Comput Eng IJECE 2024;14(3):3021. https://doi.org/10.11591/ijece.v14i3.pp3021-3035.
[23] Chu W, Zhu BB, Xue F, Guan X, Cai Z. Protect sensitive sites from phishing attacks using features extractable from inaccessible phishing URLs. In: 2013 IEEE International Conference on Communications (ICC) June. IEEE; 2013. p. 1990–4. https://doi.org/10.1109/ICC.2013.6654816.
[24] Hannousse A, Yahiouche S. Towards benchmark datasets for machine learning based website phishing detection: aan experimental study. Eng Appl Artif Intell 2021;104:104347. https://doi.org/10.1016/j.engappai.2021.104347.
[25] Morina V, Sejdiu S. Evaluating and comparing web scraping tools and techniques for data collection. 2022.
[26] Rao RS, Pais AR. Detection of phishing websites using an efficient feature-based machine learning framework. Neural Comput Appl 2019;31(8):3851–73. https://doi.org/10.1007/s00521-017-3305-0.
[27] Opara C, Chen Y, Wei B. Look before you leap: ddetecting phishing web pages by exploiting raw URL and HTML characteristics. Expert Syst Appl 2024;236:121183. https://doi.org/10.1016/j.eswa.2023.121183.
[28] Yoon J-H, Buu S-J, Kim H-J. Phishing webpage detection via multi-modal integration of HTML DOM graphs and URL features based on graph convolutional and transformer networks. Electron (Basel) 2024;13(16):3344. https://doi.org/10.3390/electronics13163344.
[29] Kocyigit E, Korkmaz M, Sahingoz OK, Diri B. Enhanced feature selection using genetic algorithm for machine-learning-based phishing URL detection. Appl Sci 2024;14(14):6081. https://doi.org/10.3390/app14146081.
[30] Yang P, Zhao G, Zeng P. Phishing website detection based on multidimensional features driven by deep learning. IEEE Access 2019;7:15196–209. https://doi.org/10.1109/ACCESS.2019.2892066.
[31] Rao RS, Vaishnavi T, Pais AR. CatchPhish: detection of phishing websites by inspecting URLs. J Ambient Intell Humaniz Comput Feb. 2020;11(2):813–25. https://doi.org/10.1007/s12652-019-01311-4.
[32] Karim A, Shahroz M, Mustofa K, Belhaouari SB, Joga SRK. Phishing detection system through hybrid machine learning based on URL. IEEE Access 2023;11:36805–22. https://doi.org/10.1109/ACCESS.2023.3252366.
[33] Pandey P, Mishra N. Phish-Sight: a new approach for phishing detection using dominant colors on web pages and machine learning. Int J Inf Secur 2023;22(4):881–91. https://doi.org/10.1007/s10207-023-00672-4.
[34] Li Y, Yang Z, Chen X, Yuan H, Liu W. A stacking model using URL and HTML features for phishing webpage detection. Future Gener Comput Syst 2019;94:27–39. https://doi.org/10.1016/j.future.2018.11.004.
[35] Jung G, Han S, Kim H, Kim K, Cha J. Don't read, just look: main content extraction from web pages using visual features. arXiv; 2021. https://doi.org/10.48550/ARXIV.2110.14164.
[36] Sameen M, Han K, Hwang SO. PhishHaven—An efficient real-time AI phishing URLs detection system. IEEE Access 2020;8:83425–43. https://doi.org/10.1109/ACCESS.2020.2991403.
[37] Sahingoz OK, Buber E, Demir O, Diri B. Machine learning based phishing detection from URLs. Expert Syst Appl 2019;117:345–57. https://doi.org/10.1016/j.eswa.2018.09.029.
[38] Chiew KL, Tan CL, Wong KS, Yong KSC, Tiong WK. A new hybrid ensemble feature selection framework for machine learning-based phishing detection system. Inf Sci 2019;484:153–66. https://doi.org/10.1016/j.ins.2019.01.064.
[39] Vrbančič G, Fister I, Podgorelec V. Datasets for phishing websites detection. Data Brief 2020;33:106438. https://doi.org/10.1016/j.dib.2020.106438.
[40] Frąszczak D. Detecting rumor outbreaks in online social networks. Soc Netw Anal Min 2023;13(1):91. https://doi.org/10.1007/s13278-023-01092-x.
[41] L.M. Rami Mohammad, "Phishing Websites." UCI Machine Learning Repository, 2012. doi: 10.24432/C51W2X.
[42] Frąszczak D. RPaSDT—Rumor propagation and source detection toolkit. SoftwareX 2022;17:100988. https://doi.org/10.1016/j.softx.2022.100988.
[43] Frąszczak D, Frąszczak E. NetCenLib: a comprehensive python library for network centrality analysis and evaluation. SoftwareX 2024;26:101699. https://doi.org/10.1016/j.softx.2024.101699.