IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/44AI/2024/4449324/
Vol. 2024 (22), Article ID 4449324

A Novel Approach for Efficient Web Data Collection*

Edyta FRĄSZCZAK

Military University of Technology, Warsaw, Poland

Correspondence should be addressed to: Edyta FRĄSZCZAK, edyta.fraszczak97@gmail.com

* Presented at the 44th IBIMA International Conference, 27-28 November 2024 Granada, Spain

Abstract

Websites are becoming increasingly popular and contain a vast amount of unstructured data. This data is often collected and transformed into vector representations, which can be utilized in various domains such as marketing, SEO, and cybersecurity. This article introduces a new solution called Web2Vec (Website To Vector), designed to extract features from websites and convert them into vector formats. These vectorized representations can be applied across different applications, improving analysis and decision-making processes in multiple fields.

Keywords: website data collection, structuring unstructured website data, website data vectorization, data mining

Introduction and research motivation

In today's digital age, the internet is a vast repository of information, hosting over a billion websites, with 201 million currently active and growing daily [1], [2]. These websites serve a variety of purposes, ranging from social media updates to business transactions, generating immense amounts of data. However, much of this data is unstructured and can vary greatly in format and content, making effective processing and organization challenging. As online information continues to expand, the need for tools that can systematically collect and structure this data becomes increasingly urgent. Existing methods, such as those used in search engine indexing, ranking, phishing detection, information warfare, and SEO, require constant adaptation to keep pace with these complexities [3], [4], [5], [6], [7], [8], [9]. These fields depend on accurate data structuring and analysis to function effectively in an ever-evolving digital landscape.

Processing data from websites is a complex, multi-stage operation that involves techniques such as web crawling, data mining, and data extraction [8], [10], [11], [12]. The first step is web crawling, where automated scripts or bots systematically browse websites to gather raw data. Common tools used in this phase include `Scrapy`, `BeautifulSoup`, and `Selenium`, which help navigate web pages and retrieve their content. Following web crawling, data mining takes place. This involves identifying useful patterns or insights within the raw data, often requiring advanced analytical skills to filter out irrelevant information and detect valuable insights. Finally, data extraction is the process of transforming this raw, often unstructured data into a structured format, making it suitable for analysis or integration into databases. This entire process is challenging because websites have diverse structures and formats that frequently change, requiring specialized skills and knowledge to manage effectively. Many existing tools, including Scrapy and similar frameworks, assist in the initial stages of web data collection by making crawling and parsing more efficient. These tools, along with various resources and guides for building such systems, provide essential support to researchers and developers working with complex web data [13], [14], [15], [16], [17].

Cite this Article as: Edyta FRĄSZCZAK, Vol. 2024 (22) "A Novel Approach for Efficient Web Data Collection" Communications of International Proceedings, Vol. 2024 (22), Article ID 4449324,
https://doi.org/10.5171/2024.4449324

This paper presents the possibilities of Web2Vec [18], a comprehensive library designed to convert websites into vector parameters for streamlined analysis. Web2Vec offers ready-to-use implementations of web crawlers built on `Scrapy`, making it accessible to both novice and experienced researchers. This tool is particularly beneficial for website analysis tasks across various fields, including SEO optimization, disinformation detection, and phishing identification. Website analysis is crucial in multiple domains. In SEO, it helps improve website rankings by analyzing content and structure, while in cybersecurity, it aids in identifying malicious or phishing sites by examining specific website characteristics. Web2Vec facilitates the efficient creation of datasets based on known safe and malicious websites, providing researchers and developers with structured data that supports their analysis and detection efforts in these areas. In a simplified way, the operation and results of the Web2Vec solution are shown in Figure 1. Here, a website, presented as an HTML file visualization, is described by a vector of parameters characterizing it. This processed vector can then be further analyzed and used by specialized software, enabling deeper analysis of the website's structure and content. 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 feature F_i

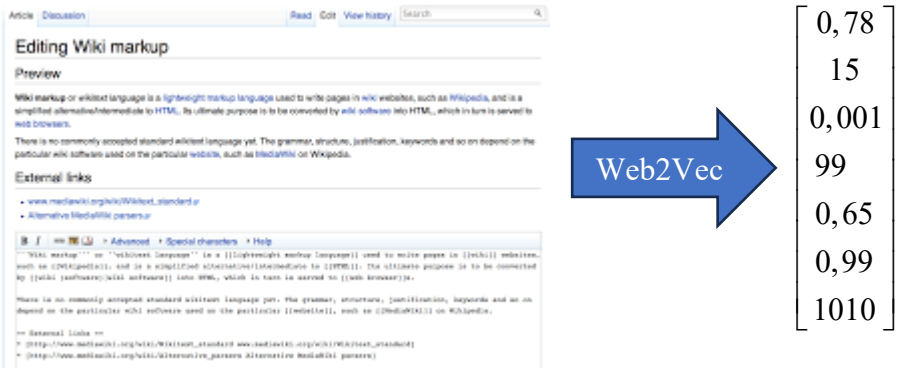

Figure 1 A visualization of the transformation of a website into vector parameters

Website data processing process

The process of converting website data into a structured format consists of three main stages [13], [14], [15], [16], [17]: fetching, extraction, and transformation, and has been presented in **Figure 2**. Each stage necessitates specific tools and techniques to effectively manage the unstructured nature of web data.

1. **Fetching Stage**

The first step in website data processing is the fetching stage, where the desired website is accessed and its raw data is retrieved. This stage involves making an HTTP GET request to the target URL using protocols and tools designed to interact with web servers, such as `curl` or `wget`. During this step, the server responds with the HTML content of the webpage, which includes both visible and hidden elements of the website's structure. This process is similar to how a web browser loads a page, but the focus here is on retrieving data for analysis rather than for display to users. The fetching stage lays the groundwork for further processing by acquiring the raw material needed to extract specific data.

2. **Extraction Stage**

Once the HTML page is retrieved, the extraction stage begins. In this phase, relevant data is extracted from the HTML code using various techniques. Regular expressions, HTML parsing libraries (such as `BeautifulSoup` in Python), and XPath queries are employed to locate and isolate the desired information. XPath is a language specifically designed for finding information within XML documents, but it is also effective for navigating and querying HTML structures. This stage allows us to filter out unnecessary content like ads or unrelated text, enabling us to focus solely on the data points that are important for our analysis. The extraction stage is crucial for pulling useful parameters from raw web content.

3. **Transformation Stage**

After gathering the necessary information, the transformation stage converts this data into a structured format, such as a table or database, making it suitable for analysis. This step often involves normalizing and standardizing data

formats to ensure consistency, which is crucial when dealing with datasets from various web sources. In many cases, the extracted data is further processed into vector parameters, representing specific features of the website in numerical form. This vectorization allows machine learning algorithms and other analytical models to interpret the website data efficiently, transforming qualitative elements into quantifiable metrics.

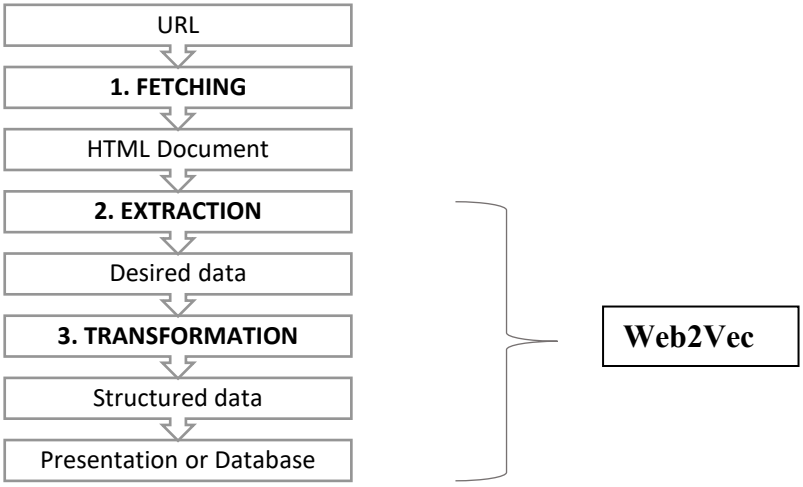

Figure 2 Web Scraping process with Web2Vec

By transforming website data into structured, vectorized formats, it becomes usable for various advanced applications, including:

- **Search Engine Optimization (SEO):** Improving website structure and keyword usage to boost search engine rankings.
- **Disinformation Detection:** Identifying and monitoring patterns in misleading or harmful content.
- **Phishing Detection and Cybersecurity:** Identifying signs of harmful websites is essential for preventing fraud and enhancing online safety.
- **Market Research and Trend Analysis:** Analysing product pages, customer reviews, and relevant content to gather insights on competitors and market trends.

Website to Vector representation

Web2Vec is a versatile library designed to transform website data into vector parameters, enabling users to extract and analyze web content efficiently. This comprehensive tool integrates web crawlers and data extractors powered by Scrapy, making it accessible even to users with limited technical expertise. Web2Vec aims to offer a full-featured repository for various website processing methods, simplifying the traditionally complex process of transforming web pages into structured, vectorized data.

Available on PyPI, Web2Vec can be easily installed using the command `pip install web2vec`, granting the community straightforward access to the latest version. This accessibility encourages community involvement in resolving bugs, implementing new methods, and enhancing the tool over time. As an open-source project, Web2Vec benefits from public collaboration, which helps ensure its reliability and continuous improvement.

One of the primary goals of Web2Vec is to streamline the data extraction process for researchers, providing a user-friendly interface and reducing setup complexity. By offering pre-configured web crawlers and extractors, Web2Vec eliminates the need for extensive setup and specialized knowledge, allowing users to begin data collection and analysis quickly. The library also integrates with a wide range of Python libraries, supporting seamless data extraction and making it ideal for researchers seeking efficient, robust solutions for website analysis. It supports the extraction of a wide range of parameters from websites, providing a comprehensive and detailed dataset for research and analysis. The current 0.3.1 version provides access to the parameters presented in **Table 1**.

Table 1 Web2Vec available extracted parameters

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`

It is worth mentioning that the library includes official documentation [19] with a wide range of examples utilizing Jupyter Notebooks, making it easy to get started and use effectively.

Each feature type is provided by a different type of extractor—an object responsible for processing HTML or interacting with external services to deliver numerical features. Each result is returned as a `dataclass` object, a Python structure specifically designed for data storage. This approach makes it easy to access and view the parameters collected by a given extractor and the sample content of those `dataclasses` is presented in **Figure 3**.

```python
@dataclass
class HtmlBodyFeatures:
    contains_forms: bool
    contains_obfuscated_scripts: bool
    contains_suspicious_keywords: bool
    body_length: int
    num_titles: int
    num_images: int
    num_links: int
    script_length: int
    special_characters: int
    script_to_special_chars_ratio: float
    script_to_body_ratio: float
    body_to_special_char_ratio: float
    iframe_redirection: int
    mouse_over_effect: int
    right_click_disabled: int
    num_scripts_http: int
    num_styles_http: int
    num_iframes_http: int
    num_external_scripts: int
    num_external_styles: int
    num_external_iframes: int
```

```python
@dataclass
class HttpResponseFeatures:
    redirects: bool
    redirect_count: int
    contains_forms: bool
    contains_obfuscated_scripts: bool
    contains_suspicious_keywords: bool
    uses_https: bool
    missing_x_frame_options: bool
    missing_x_xss_protection: bool
    missing_content_security_policy: bool
    missing_strict_transport_security: bool
    missing_x_content_type_options: bool
    is_live: bool
    server_version: Optional[str] = None
    body_length: int = 0
    num_titles: int = 0
    num_images: int = 0
    num_links: int = 0
    script_length: int = 0
    special_characters: int = 0
    script_to_special_chars_ratio: float = 0.0
    script_to_body_ratio: float = 0.0
    body_to_special_char_ratio: float = 0.0
```

Figure 3 Dataclass representations of some of the extractor results

Sample usage: building a website-related dataset

This example involves a dataset that contains both legitimate and phishing websites. For each website, Web2Vec will gather the necessary parameters to create a comprehensive training and validation set for the selected machine learning models. This dataset will help in developing models that can effectively distinguish between genuine and phishing websites by utilizing various numerical features extracted from web content, metadata, and structural elements.

Presented in this paragraph example involves a dataset that contains both legitimate and phishing websites. For each website, Web2Vec will gather the necessary parameters to create a comprehensive training and validation set for the selected machine learning models. This dataset will help in developing models that can effectively distinguish between genuine and phishing websites by utilizing various numerical features extracted from web content, metadata, and structural elements. It is important to note that this example is available in the official documentation, which allows users to download, run, and analyse the results. The documentation also provides clear instructions for setting up and using the tool effectively. Additionally, Web2Vec supports integration with external services that may require API keys. In such cases, users must provide the necessary authentication details; otherwise, the extractor will not return data from those services. This flexibility ensures that users can customize their datasets according to their analysis needs, enhancing Web2Vec's usability for both research and industry applications.

Firstly, the list of websites to process should be created, a sample content is presented in Figure 4. It includes two columns: "domain" and "is_phish," which indicate whether a website is legitimate or phishing. For Web2Vec "domain" value is crucial, while "is_phish" is for machine learning models.

```
Domain,is_phish
sourceforge.net,0
indianexpress.com,0
bspoke.cn,1
cannada.site,1
```

Figure 4 A sample CSV file containing URLs of websites to process. It includes two columns: "domain" and "is_phish," which indicate whether a website is legitimate or phishing.

Next web2vec should be configured and run and a sample of those actions presented in **Listing 1**. The process begins by initializing a set of Web2Vec extractors designed to gather various details, such as HTML content, HTTP response attributes, URL structure, SSL certificate information, and phishing indicators. The code then loads website data from a file named "phish_websites.csv" and iterates through each entry, constructing the URL and applying the extractors. For each website, only numerical features are retained, and an "is_phish" label is added to the extracted data to indicate whether the site is classified as phishing or legitimate. Finally, the collected data, which includes the domain name and the extracted parameters, is written to a new CSV file named "output.csv." This output file provides a structured dataset containing the processed parameters of each website, making it ready for analysis or use in machine learning tasks, such as training a phishing detection model. A sample content of this file is presented in **Figure 5**.

```
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)
```

Listing 1 Web2Vec setup, execution, and storage of vectorized website data.

```
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
```

Figure 5 Sample content of vectorized website data after processing

Afterward, any machine learning model can be applied to this prepared dataset. An example of this process is shown in Listing 2. This code begins by loading the dataset, "output.csv," into a pandas DataFrame. Within this dataset, the "is_phish" column serves as the target label, indicating whether a site is fraudulent. The "Domain" column, which is not relevant for training, is also excluded from the feature set X. Consequently, "is_phish" is assigned as the target variable y. The dataset is divided into two sets: 80% for training and 20% for testing. To ensure reproducibility, a random seed of 42 is used. A decision tree classifier is then created and trained on the training data to learn the patterns that distinguish phishing sites from legitimate ones. After the training process, the model makes predictions on the test set, and its performance is evaluated. The accuracy of these predictions is reported, along with a detailed classification report that includes metrics such as precision, recall, and F1 score. This information provides valuable insights into the model's performance for both classes. This workflow demonstrates how machine learning can effectively be applied to detect phishing attempts.

```
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))
```

Listing 2 Decision tree classifier setup, execution, and validation based on the prepared dataset

Conclusions

Web2Vec offers a streamlined method for extracting various parameters from websites. It captures elements such as HTML content, HTTP responses, URL structures, SSL certificates, and external classifications. This organized approach enables the automatic transformation of unstructured website data into a vectorized dataset, where each website is represented by a set of numerical features. These features can be easily utilized in machine learning models for tasks such as phishing detection, SEO optimization, and other forms of website analysis. In the example presented, Web2Vec was utilized to create a dataset for training a phishing detection model, demonstrating its practical application and potential to tackle complex cybersecurity challenges. The results indicate that this vectorized approach, when combined with machine learning, can generate high-quality datasets that enable accurate and effective classification.

References

- R. Gupta, 'How Many Websites Are There? The Web in 2024', Themeisle Blog. Accessed: Jun. 16, 2024. [Online]. Available: https://themeisle.com/blog/how-many-websites-are-there/
- P. S. Sharma, D. Yadav, and R. N. Thakur, 'Web Page Ranking Using Web Mining Techniques: A Comprehensive Survey', *Mob. Inf. Syst.*, vol. 2022, pp. 1–19, May 2022, doi: 10.1155/2022/7519573.
- E. Frąszczak and D. Fraszczak, 'A review of a website phishing detection taxonomy', in *Procedings of the 43th International Business Information Management Association Conference (IBIMA)*, Madrid, Spain, 2024. doi: 10.6084/m9.figshare.26345473.
- K. Magdziarz and D. Frąszczak, 'The Architecture Concepts for Building Highly Scalable Crawling Cluster For Data-Driven On-Page Optimization', 2023, *figshare*. doi: 10.6084/M9.FIGSHARE.21909273.V1.
- D. Frąszczak, '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)*, Cordoba, Spain: International Business Information Management, 2021, pp. 11381–11389. doi: http://dx.doi.org/10.6084/m9.figshare.16545675.
- J. L. Ledford, *SEO search engine optimization bible*. Indianapolis, Ind: Wiley, 2008.
- E. S. ENGE STEPHAN STRICCHIOLA, JESSICA, *ART OF SEO: mastering search engine optimization*. S.l.: O'REILLY MEDIA, INC, USA, 2022.
- R. Diouf, E. N. Sarr, O. Sall, B. Birregah, M. Bousso, and S. N. Mbaye, 'Web Scraping: State-of-the-Art and Areas of Application', in *2019 IEEE International Conference on Big Data (Big Data)*, Los Angeles, CA, USA: IEEE, Dec. 2019, pp. 6040–6042. doi: 10.1109/BigData47090.2019.9005594.
- D. Frąszczak, 'Detecting rumor outbreaks in online social networks', *Soc. Netw. Anal. Min.*, vol. 13, no. 1, p. 91, Jun. 2023, doi: 10.1007/s13278-023-01092-x.
- I. S. H. Almaqbali, F. M. A. Al Khufairi, M. S. Khan, A. Z. Bhat, and I. Ahmed, 'Web Scrapping: Data Extraction from Websites', *J. Stud. Res.*, Jul. 2020, doi: 10.47611/jsr.vi.942.
- A. Chapagain, *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, Limited, 2019.
- R. E. Mitchell, *Web scraping with Python: collecting more data from the modern web*, Second edition. Sebastopol, CA: O'Reilly Media, 2018.
- D. M. Thomas and S. Mathur, 'Data Analysis by Web Scraping using Python', in *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*, Coimbatore, India: IEEE, Jun. 2019, pp. 450–454. doi: 10.1109/ICECA.2019.8822022.
- V. Singrodia, A. Mitra, and S. Paul, 'A Review on Web Scrapping and its Applications', in *2019 International Conference on Computer Communication and Informatics (ICCCI)*, Coimbatore, Tamil Nadu, India: IEEE, Jan. 2019, pp. 1–6. doi: 10.1109/ICCCI.2019.8821809.
- V. Morina and S. Sejdiu, *Evaluating and comparing web scraping tools and techniques for data collection*. 2022.

- G. L. Hajba, ‘Using Beautiful Soup’, in *Website Scraping with Python*, Berkeley, CA: Apress, 2018, pp. 41–96. doi: 10.1007/978-1-4842-3925-4_3.
- N. Haddaway, ‘The Use of Web-scraping Software in Searching for Grey Literature’, *Grey J.*, vol. 11, pp. 186–190, Oct. 2015.
- D. F. Frąszczak Edyta, *web2vec: Website to vector representation library*. Python. Accessed: Aug. 12, 2024. [Online]. Available: https://github.com/damianfraszczak/web2vec
- ‘Welcome to Web2Vec documentation! — web2vec 0.1.2 documentation’. Accessed: Jul. 27, 2024. [Online]. Available: https://web2vec.readthedocs.io/en/latest/