

WEB MINING

Colorful world of web : Web Mining

Selahattin





1

What is Web Mining?

Step-by-Step Guide

2

3

Application of Web Mining

Why We need Web
Mining?

4

5

Github ?

1

What is Web Mining?

Information Extraction



Web mining involves extracting useful information from the vast amount of data available on the World Wide Web.

Here's a step-by-step guide to help you get started with web mining:

1. Understand the Basics:

- Learn about HTML, CSS, and JavaScript and understand how websites are structured

2. Choose a Programming Language:

- Python is a popular choice for web mining
- Other languages like JavaScript (with Node.js), Ruby, or PHP can also be used

3. Learn Web Scraping Libraries:

- BeautifulSoup - Scrapy - Selenium

4. Understand HTTP and Web Requests:

- Learn about HTTP methods (GET, POST, etc.) and status codes (200 OK, 404 Not Found, etc.).
- Familiarize yourself with making HTTP requests using libraries like Requests in Python.

5. Start with Simple Projects:

- Choose a simple website to scrape initially. Avoid scraping large or complex websites until you have gained more experience.
- Practice extracting different types of data such as text, images, links, etc.
- Experiment with different scraping techniques and libraries to understand their strengths and weaknesses.

6. Handle Parsing Challenges:

- Web pages can be poorly structured or inconsistently formatted, making parsing difficult. Learn how to handle such challenges using techniques like regular expressions.

7. Data Storage and Analysis:

- Decide how you want to store the scraped data. Options include saving to a file (CSV, JSON, etc.), storing in a database, or integrating with other tools for analysis.
- Use data analysis and visualization libraries like Pandas, NumPy, and Matplotlib to analyze and visualize the scraped data.

Extracting all link from MSKU Website

```
import requests
from bs4 import BeautifulSoup

# URL to scrape
url = 'https://www.mu.edu.tr/'

# Send a GET request to the URL
response = requests.get(url)

# Parse the HTML content using BeautifulSoup
soup = BeautifulSoup(response.text, 'html.parser')

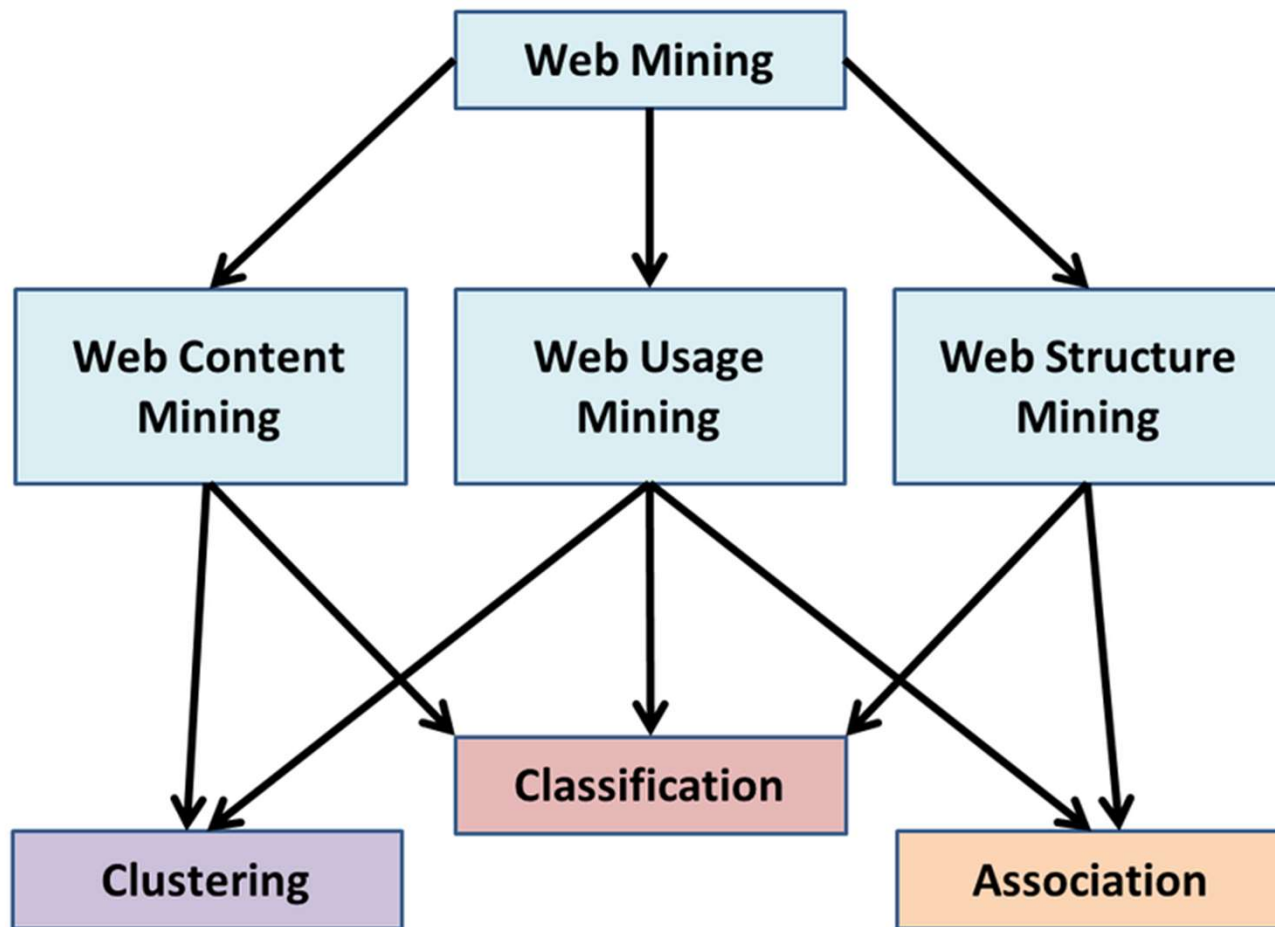
# Extract specific information
# Example: Extracting all the links on the page
links = soup.find_all('a')
for link in links:
    print(link.get('href'))
```

```
import scrapy

class MySpider(scrapy.Spider):
    name = 'example_spider'
    start_urls = ['https://example.com']

    def parse(self, response):
        # Extract specific information
        # Example: Extracting text content from all paragraphs
        paragraphs = response.css('p::text').extract()
        for paragraph in paragraphs:
            print(paragraph)

        # Follow links to other pages if needed
        # Example: Follow links to other pages
        for next_page in response.css('a::attr(href)').extract():
            yield response.follow(next_page, self.parse)
```



Web Content Mining:

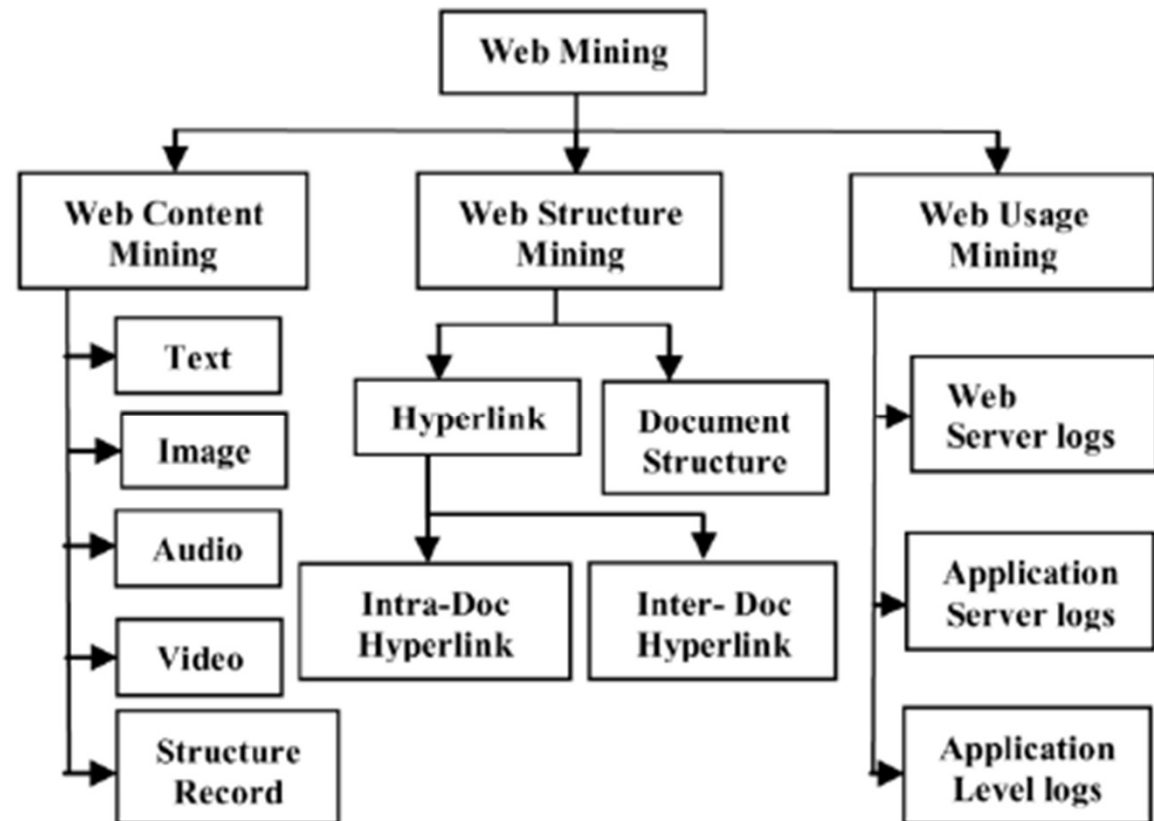
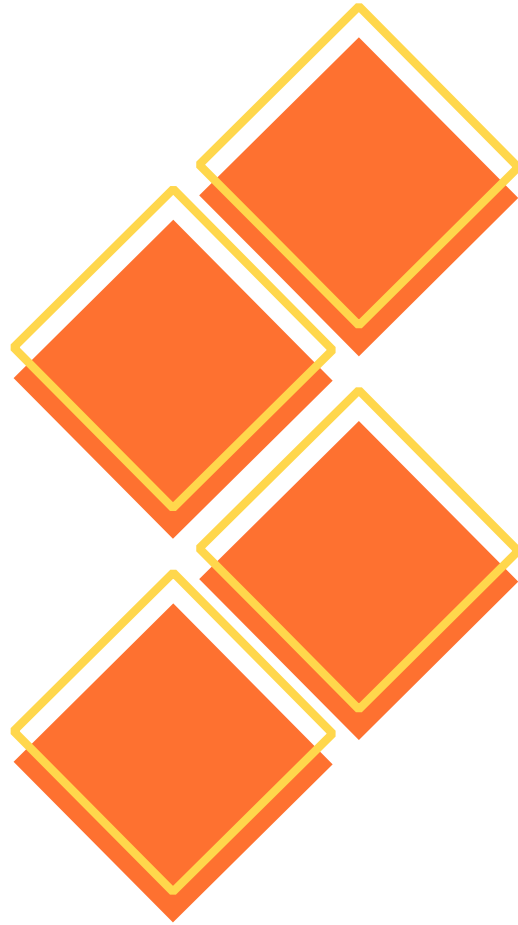
Web content mining is the application of extracting useful information from web content.

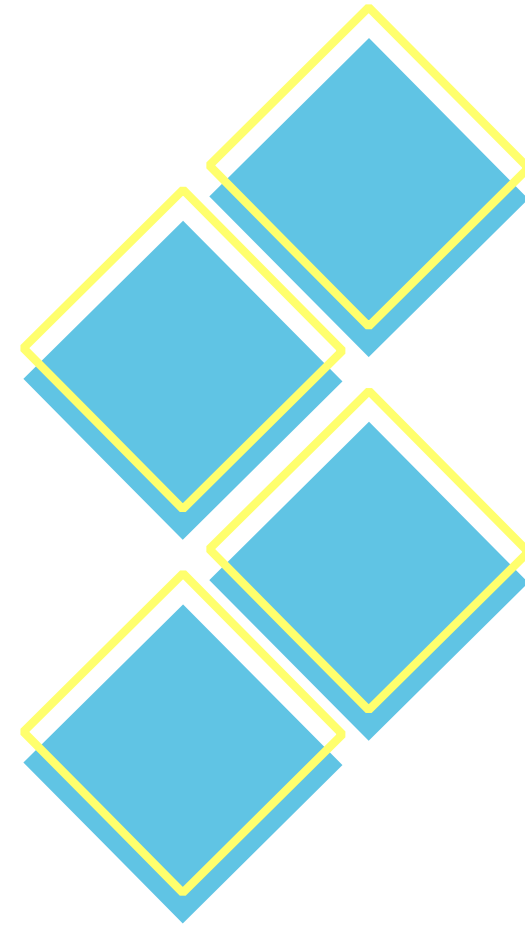
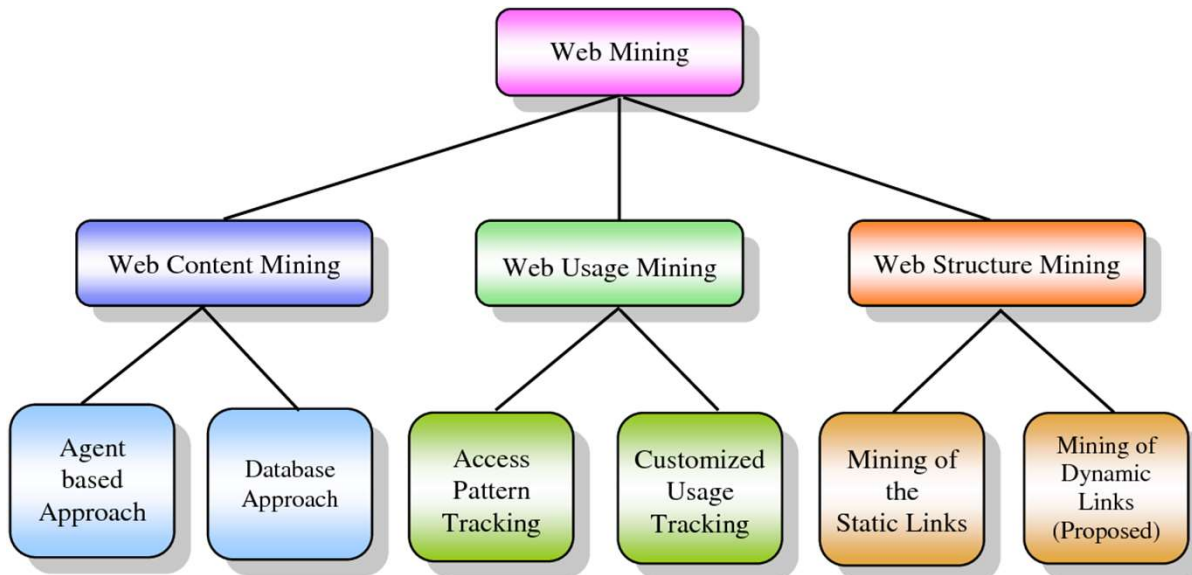
Web Structure Mining:

Web structure mining is the application of discovering structural information from web structure.

Web Usage Mining:

Web usage mining is the application of identifying or discovering patterns in web usage data.





web-mining

☆ Star

Here are 62 public repositories matching this topic...

Language: All ▾

Sort: Most stars ▾

clips / **pattern**

☆ Star 8.6k ▾

<> Code ⓘ Issues 🔗 Pull requests

Web mining module for Python, with tools for scraping, natural language processing, machine learning, network analysis and visualization.

python machine-learning natural-language-processing sentiment-analysis wordnet web-mining
network-analysis

Updated on Jul 12, 2023 ● Python

deepwn / **deepMiner**

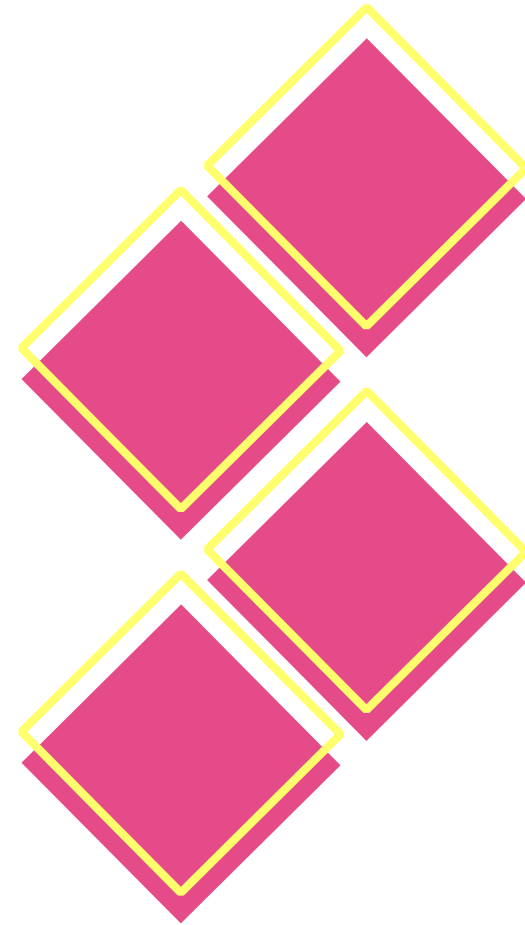
☆ Star 541 ▾

<> Code ⓘ Issues 🔗 Pull requests











deepMiner webminer proxy (update for cryptoNight R)

javascript website wasm emscripten mining miner web-mining monero xmr cryptonight
coinhive pool-proxy webminer deepminer

Updated on Mar 21, 2019 ● C



Filter by

- <> Code ...
-  Repositories 2.9k
-  Issues 7k
-  Pull requests 1k
-  Discussions 214
-  Users 287
-  Commits 2k
-  Packages 1
-  Wikis 3k
-  Topics 6
-  Marketplace 0

Languages

-  Jupyter Notebook
-  Python
-  HTML
-  JavaScript
-  Java
-  R
-  PHP
-  CSS
-  TypeScript
-  C#
-  More languages...

2.9k results (192 ms)

Sort by: Best match ...

[ptwobrussell/Mining-the-Social-Web-2nd-Edition](#) StarThe official online compendium for **Mining the Social Web**, 2nd Edition (O'Reilly, 2013) HTML ·  2.9k · Updated on Jul 12, 2022[ptwobrussell/Mining-the-Social-Web](#) StarThe official online compendium for **Mining the Social Web** (O'Reilly, 2011) JavaScript ·  1.2k · Updated on Oct 18, 2013[mikhaillassen/Mining-the-Social-Web-3rd-Edition](#) StarThe official online compendium for **Mining the Social Web**, 3rd Edition (O'Reilly, 2018) Jupyter Notebook ·  897 · Updated on Dec 8, 2022[clips/pattern](#) Star**Web mining** module for Python, with tools for scraping, natural language processing, machine learning, network analysis and visualization. python  machine-learning  natural-language-processing  sentiment-analysis  wordnet Python ·  8.6k · Updated on Jul 12, 2023[devanshbatham/ParamSpider](#) Star Sponsor**Mining** URLs from dark corners of **Web** Archives for bug hunting/fuzzing/further probing osint  fuzzing  parameter  bugbounty  content-discovery Python ·  2.1k · Updated on Oct 7, 2023[minernl/Miningcore.WebUI](#) StarWebUI made for <https://github.com/minernl/miningcore>

Pattern

build no longer available coverage 68% pypi v3.6 License BSD 3-Clause

Pattern is a web mining module for Python. It has tools for:

- Data Mining: web services (Google, Twitter, Wikipedia), web crawler, HTML DOM parser
- Natural Language Processing: part-of-speech taggers, n-gram search, sentiment analysis, WordNet
- Machine Learning: vector space model, clustering, classification (KNN, SVM, Perceptron)
- Network Analysis: graph centrality and visualization.

It is well documented, thoroughly tested with 350+ unit tests and comes bundled with 50+ examples. The source code is licensed under BSD.



Example

This example trains a classifier on adjectives mined from Twitter using Python 3. First, tweets that contain hashtag #win or #fail are collected. For example: "\$20 tip off a sweet little old lady today #win". The word part-of-speech tags are then parsed, keeping only adjectives. Each tweet is transformed to a vector, a dictionary of adjective → count items, labeled WIN or FAIL. The classifier uses the vectors to learn which other tweets look more like WIN or more like FAIL.

```
from pattern.web import Twitter
from pattern.en import tag
from pattern.vector import KNN, count

twitter, knn = Twitter(), KNN()

for i in range(1, 3):
    for tweet in twitter.search('#win OR #fail', start=i, count=100):
        s = tweet.text.lower()
        p = '#win' in s and 'WIN' or 'FAIL'
        v = tag(s)
        v = [word for word, pos in v if pos == 'JJ'] # JJ = adjective
        v = count(v) # {'sweet': 1}
        if v:
            knn.train(v, type=p)

print(knn.classify('sweet potato burger'))
print(knn.classify('stupid autocorrect'))
```





THANK YOU

