

Crawler

استاد: دکتر منصوری زاده

دانشجویان: محمد میرزایی 9912358039

مهدی براتی 9912358008

سید علی امامی 9912358005

۲	توضیحات کلی
۲	توضیحات کد
۲	راه اندازی کد
۳	تعریف کلاس WebCrawler
۳	fetch_content
۴	save_content
۴	extract_links
۵	index_content
۵	Crawl
۶	save_index
۶	load_index
۷	display_index
۷	استفاده از خزنده
۸	صفحات پیدا شده توسط crawler (basu.ac.ir)
۸	گوشه ای از اجرای برنامه
۹	Posting list ایجاد شده دارای نام فایل های متنی (page_files_posting_list.json)
۱۰	گوشه ای از محتوای فایل web_links_posting_list
۱۱	تست posting list بر اساس WEB LINK با استفاده از کد retrieve link
۱۱	تست posting list بر اساس page file با استفاده از کد retrieve page name
۱۳	Retrieve Link Code
۱۳	کلاس PostingListSearcher
۱۳	متد __init__
۱۳	متد load_posting_list
۱۴	متد search
۱۵	متد display_results
۱۶	استفاده از کلاس PostingListSearcher
۱۷	Retrieve Page Name

توضیحات کلی

همانطور که در صورت تمرین خواسته شده بود ، کد مربوط به خزشگر (Crawler) برای بررسی محتوای سایت ها و استخراج دیتای آنها و سپس انجام عملیات indexing بر روی این دیتاها انجام شده است. همچنین دو قطعه کد دیگر اضافه شده که براساس posting list اضافه شده ، این دو کد میتوانند عملیات جستجوی query را انجام دهند و نتیجه را برگردانند. به دلیل اینکه خزشگر پس از استخراج محتوای سایت ها فایل های متنی ایجاد میکند ، دو نوع پردازش کوئری مختلف ایجاد کردیم که یکی از آنها پس از پردازش فایل های متنی را بازمیگرداند که کوئری ما در آن وجود دارد و دیگری پس از پردازش کوئری لینک هایی که آن کوئری در آن وجود دارد را به ما بازمیگرداند ، همچنین در عملیات بازگرداندن نتیجه برای اینکه شبیه یک موتور جستجو عمل کنیم ، frequency (تعداد تکرار) را دخیل کردیم تا سایت هایی در رتبه بالاتر قرار گیرند که تعداد بیشتری از آن کلمه را دارند.

در کد crawler سایت <https://basu.ac.ir> قرار داده شده است ، همچنین کد به نحوی پیاده سازی شده که تا عمق خاصی لینک ها را دنبال میکند.

توضیحات کد

راه اندازی کد

ابتدا کتابخانه های مورد نیاز دانلود می شوند:

```
1 import os
2 import requests
3 from bs4 import BeautifulSoup
4 from urllib.parse import urljoin, urlparse
5 from collections import defaultdict
6 import nltk
7 import re
8 import json
9
10 # Initialize the nltk resources
11 nltk.download('punkt')
12
```

کتابخانه‌های مورد استفاده شامل requests برای دریافت محتوای وب، BeautifulSoup برای تجزیه و تحلیل HTML، urllib برای مدیریت URLها، defaultdict برای ایجاد فهرست‌های کلمات، nltk برای پردازش زبان طبیعی و json برای ذخیره داده‌ها به صورت فایل JSON هستند.

تعریف کلاس WebCrawler

```
1 class WebCrawler:
2     def __init__(self, start_url, max_depth=2):
3         self.start_url = start_url
4         self.max_depth = max_depth
5         self.visited_urls = set()
6         self.web_links_index = defaultdict(list)
7         self.page_files_index = defaultdict(list)
8         self.output_dir = 'crawled_pages'
9         self.web_links_index_file = 'web_links_posting_list.json'
10        self.page_files_index_file = 'page_files_posting_list.json'
11
12        if not os.path.exists(self.output_dir):
13            os.makedirs(self.output_dir)
14
```

fetch_content

```
1 def fetch_content(self, url):
2     try:
3         response = requests.get(url, verify=False) # Bypass SSL/TLS certificate verification
4         response.raise_for_status()
5         return response.text
6     except requests.RequestException as e:
7         print(f"Error fetching {url}: {e}")
8         return ""
```

این متد محتوای صفحه وب را دریافت می‌کند.

save_content

```
1 def save_content(self, url, content, count):
2     file_path = os.path.join(self.output_dir, f'page_{count}.txt')
3     with open(file_path, 'w', encoding='utf-8') as file:
4         file.write(content)
5     return file_path
```

این متد محتوای دریافت شده را به یک فایل ذخیره می‌کند.

extract_links

```
1 def extract_links(self, content, base_url):
2     soup = BeautifulSoup(content, 'html.parser')
3     links = set()
4     for a_tag in soup.find_all('a', href=True):
5         link = urljoin(base_url, a_tag['href'])
6         parsed_link = urlparse(link)
7         if parsed_link.netloc == urlparse(self.start_url).netloc:
8             links.add(link)
9     return links
```

این متد لینک‌های موجود در صفحه را استخراج می‌کند.

index_content

```
1 def index_content(self, content, url, file_path):
2     soup = BeautifulSoup(content, 'html.parser')
3     visible_texts = soup.stripped_strings
4     full_text = " ".join(visible_texts)
5     tokens = nltk.word_tokenize(full_text)
6     for token in tokens:
7         token = token.lower()
8         if re.match(r'\w+', token):
9             self.web_links_index[token].append(url)
10            self.page_files_index[token].append(file_path)
11
```

این متد محتوای صفحه را فهرست‌بندی می‌کند.

Crawl

```
1 def crawl(self, url, depth):
2     if depth > self.max_depth or url in self.visited_urls:
3         return
4
5     self.visited_urls.add(url)
6     content = self.fetch_content(url)
7     if content:
8         file_path = self.save_content(url, content, len(self.visited_urls))
9         self.index_content(content, url, file_path)
10        links = self.extract_links(content, url)
11        for link in links:
12            self.crawl(link, depth + 1)
13
```

این متد وظیفه اصلی خزیدن وب را بر عهده دارد.

save_index

```
1 def save_index(self):
2     with open(self.web_links_index_file, 'w', encoding='utf-8') as file:
3         json.dump(self.web_links_index, file, ensure_ascii=False, indent=4)
4     with open(self.page_files_index_file, 'w', encoding='utf-8') as file:
5         json.dump(self.page_files_index, file, ensure_ascii=False, indent=4)
6
```

این متد فهرست‌ها را در فایل JSON ذخیره می‌کند.

load_index

```
1 def load_index(self):
2     if os.path.exists(self.web_links_index_file):
3         with open(self.web_links_index_file, 'r', encoding='utf-8') as file:
4             self.web_links_index = json.load(file)
5     if os.path.exists(self.page_files_index_file):
6         with open(self.page_files_index_file, 'r', encoding='utf-8') as file:
7             self.page_files_index = json.load(file)
8
```

این متد فهرست‌ها را از فایل JSON بارگذاری می‌کند.

display_index

```
1 def display_index(self):
2     print("Web Links Index:")
3     for word, links in self.web_links_index.items():
4         print(f'{word}: {links}')
5
6     print("\nPage Files Index:")
7     for word, files in self.page_files_index.items():
8         print(f'{word}: {files}')
```

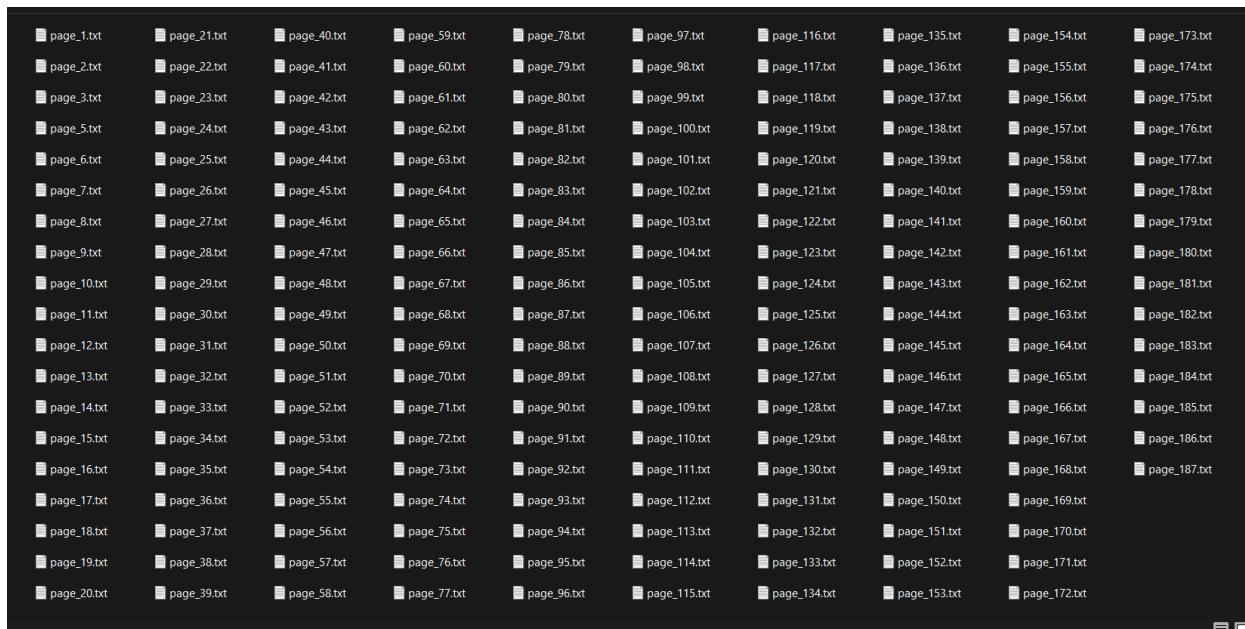
این متد فهرست‌ها را نمایش می‌دهد.

استفاده از خزنده

```
1 # Usage
2 start_url = 'https://basu.ac.ir/'
3 crawler = WebCrawler(start_url)
4 crawler.crawl(start_url, 0)
5 crawler.save_index() # Save the posting lists to separate files
6 crawler.display_index()
```

در نهایت، از کلاس WebCrawler برای خزیدن وب استفاده می‌شود. این کد از آدرس داده شده شروع به خزیدن کرده و تا عمق تعیین شده ادامه می‌دهد، سپس فهرست‌ها را در فایل‌های JSON ذخیره کرده و آن‌ها را نمایش می‌دهد.

صفحات پیدا شده توسط crawler (basu.ac.ir)



کد crawler ما بایک عمق خاصی صفحات را پیمایش نموده و محتویات آن را در فایل های متنی بالا ذخیره کرده است.

گوشه ای از اجرای برنامه

دنبند: ['crawled_pages\\page_186.txt', 'crawled_pages\\page_186.txt']
 دن: ['crawled_pages\\page_186.txt']
 دن سربیم: ['crawled_pages\\page_186.txt']
 دن سرب: ['crawled_pages\\page_186.txt']
 دودوخ: ['crawled_pages\\page_186.txt']
 عرسا: ['crawled_pages\\page_186.txt']
 رغل: ['crawled_pages\\page_186.txt', 'crawled_pages\\page_186.txt']
 راه اگب واخ: ['crawled_pages\\page_186.txt']
 رج ختیم: ['crawled_pages\\page_186.txt']
 زدی امیم: ['crawled_pages\\page_186.txt']
 vpn: ['crawled_pages\\page_186.txt', 'crawled_pages\\page_186.txt']
 دورول ادیج: ['crawled_pages\\page_186.txt', 'crawled_pages\\page_186.txt']
 دورول ا: ['crawled_pages\\page_186.txt']
 دل امیتا: ['crawled_pages\\page_186.txt']
 نوی سام وتا: ['crawled_pages\\page_186.txt', 'crawled_pages\\page_186.txt']
 97/7/14: ['crawled_pages\\page_186.txt']
 لی ویج: ['crawled_pages\\page_186.txt', 'crawled_pages\\page_186.txt']
 داغ: ['crawled_pages\\page_186.txt']
 زدیق اکم: ['crawled_pages\\page_186.txt']
 روسیورب: ['crawled_pages\\page_186.txt', 'crawled_pages\\page_186.txt']
 سنه: ['crawled_pages\\page_186.txt', 'crawled_pages\\page_186.txt']
 لوارک: ['crawled_pages\\page_186.txt', 'crawled_pages\\page_186.txt']
 کیم انید: ['crawled_pages\\page_186.txt']
 یقدا امیت: ['crawled_pages\\page_186.txt']
 یاه تراهم: ['crawled_pages\\page_186.txt']
 یخ داو: ['crawled_pages\\page_186.txt']
 لدعیم: ['crawled_pages\\page_186.txt']
 دح او: ['crawled_pages\\page_186.txt']
 دازام: ['crawled_pages\\page_186.txt']
 یرایخا: ['crawled_pages\\page_186.txt']
 هویلخت: ['crawled_pages\\page_186.txt', 'crawled_pages\\page_186.txt']

Posting list ایجاد شده دارای نام فایل های متنی) (page_files_posting_list.json

```
334120      "crawled_pages\\page_186.txt"
334121    ],
334122    "دینامیک": [
334123      "crawled_pages\\page_186.txt"
334124    ],
334125    "تصادفی": [
334126      "crawled_pages\\page_186.txt"
334127    ],
334128    "مهارت های": [
334129      "crawled_pages\\page_186.txt"
334130    ],
334131    "واحد": [
334132      "crawled_pages\\page_186.txt"
334133    ],
334134    "معدل": [
334135      "crawled_pages\\page_186.txt"
334136    ],
334137    "واحد": [
334138      "crawled_pages\\page_186.txt"
334139    ],
334140    "مازاد": [
334141      "crawled_pages\\page_186.txt"
334142    ],
334143    "اختیاری": [
334144      "crawled_pages\\page_186.txt"
334145    ],
334146    "تخلیه": [
334147      "crawled_pages\\page_186.txt",
334148      "crawled_pages\\page_186.txt"
334149    ],
```

قسمتی از محتوای فایل page file posting list

- Posting list ایجاد شده دارای لینک سایت است ، به این صورت که **vocabulary** موردنظر در کدام لینک ها وجود دارد.

web_links_posting_list

```
334096 ],  
334097 "97/7/14": [  
334098     "https://basu.ac.ir/%D8%A2%D8%B1%D8%B4%DB%8C%D9%88-%D8%A7%D8%B7%D9%84%D8%A7%D8%B9%DB%8C%D9%",  
334099 ],  
334100 "تحويل": [  
334101     "https://basu.ac.ir/%D8%A2%D8%B1%D8%B4%DB%8C%D9%88-%D8%A7%D8%B7%D9%84%D8%A7%D8%B9%DB%8C%D9%",  
334102     "https://basu.ac.ir/%D8%A2%D8%B1%D8%B4%DB%8C%D9%88-%D8%A7%D8%B7%D9%84%D8%A7%D8%B9%DB%8C%D9%",  
334103 ],  
334104 "غذا": [  
334105     "https://basu.ac.ir/%D8%A2%D8%B1%D8%B4%DB%8C%D9%88-%D8%A7%D8%B7%D9%84%D8%A7%D8%B9%DB%8C%D9%",  
334106 ],  
334107 "امكانيدير": [  
334108     "https://basu.ac.ir/%D8%A2%D8%B1%D8%B4%DB%8C%D9%88-%D8%A7%D8%B7%D9%84%D8%A7%D8%B9%DB%8C%D9%",  
334109 ],  
334110 "پروفيسور": [  
334111     "https://basu.ac.ir/%D8%A2%D8%B1%D8%B4%DB%8C%D9%88-%D8%A7%D8%B7%D9%84%D8%A7%D8%B9%DB%8C%D9%",  
334112     "https://basu.ac.ir/%D8%A2%D8%B1%D8%B4%DB%8C%D9%88-%D8%A7%D8%B7%D9%84%D8%A7%D8%B9%DB%8C%D9%",  
334113 ],  
334114 "منى": [  
334115     "https://basu.ac.ir/%D8%A2%D8%B1%D8%B4%DB%8C%D9%88-%D8%A7%D8%B7%D9%84%D8%A7%D8%B9%DB%8C%D9%",  
334116     "https://basu.ac.ir/%D8%A2%D8%B1%D8%B4%DB%8C%D9%88-%D8%A7%D8%B7%D9%84%D8%A7%D8%B9%DB%8C%D9%",  
334117 ],  
334118 "كراول": [  
334119     "https://basu.ac.ir/%D8%A2%D8%B1%D8%B4%DB%8C%D9%88-%D8%A7%D8%B7%D9%84%D8%A7%D8%B9%DB%8C%D9%",  
334120     "https://basu.ac.ir/%D8%A2%D8%B1%D8%B4%DB%8C%D9%88-%D8%A7%D8%B7%D9%84%D8%A7%D8%B9%DB%8C%D9%",  
334121 ],  
334122 "ديناميك": [  
334123     "https://basu.ac.ir/%D8%A2%D8%B1%D8%B4%DB%8C%D9%88-%D8%A7%D8%B7%D9%84%D8%A7%D8%B9%DB%8C%D9%",  
334124 ],  
334125 "معلومات": [
```

دو کد retrieve link و retrieve page name به ترتیب فایل های web link posting list و page file posting list را استفاده میکند و سپس کوئری را به عنوان ورودی از ما گرفته و سپس با جستجو در posting list ها نتایج را با اولویت تکرار بیشتر، نمایش میدهند.

تست **posting list** بر اساس **WEB LINK** با استفاده از کد **retrieve link**
 کوئری پروفیسور را به عنوان ورودی دادیم و خروجی زیر را نمایش داد.

```
C:\Users\moham\Desktop\New folder>python retrieve_link.py
Enter your search query: روسفوپ
Word: "روسفوپ" found in:
- https://basu.ac.ir/%D8%A2%D8%B1%D8%B4%DB%8C%D9%88-%D8%A7%D8%B7%D9%84%D8%A7%D8%B9%DB%8C%D9%87-%D9%87%D8%A7?sort=createDate- (frequency: 2)

C:\Users\moham\Desktop\New folder>
```

کلمه "پروفیسور" ۲ بار در لینک نمایش داده شده، آمده است (frequency = 2)
 این تصویر از همان لینکی است که در بالا به ما داده شده است و همانطور که قابل مشاهده است کلمه پروفیسور ۲ بار آمده است.



تست **posting list** براساس **page file** با استفاده از کد **retrieve page name**

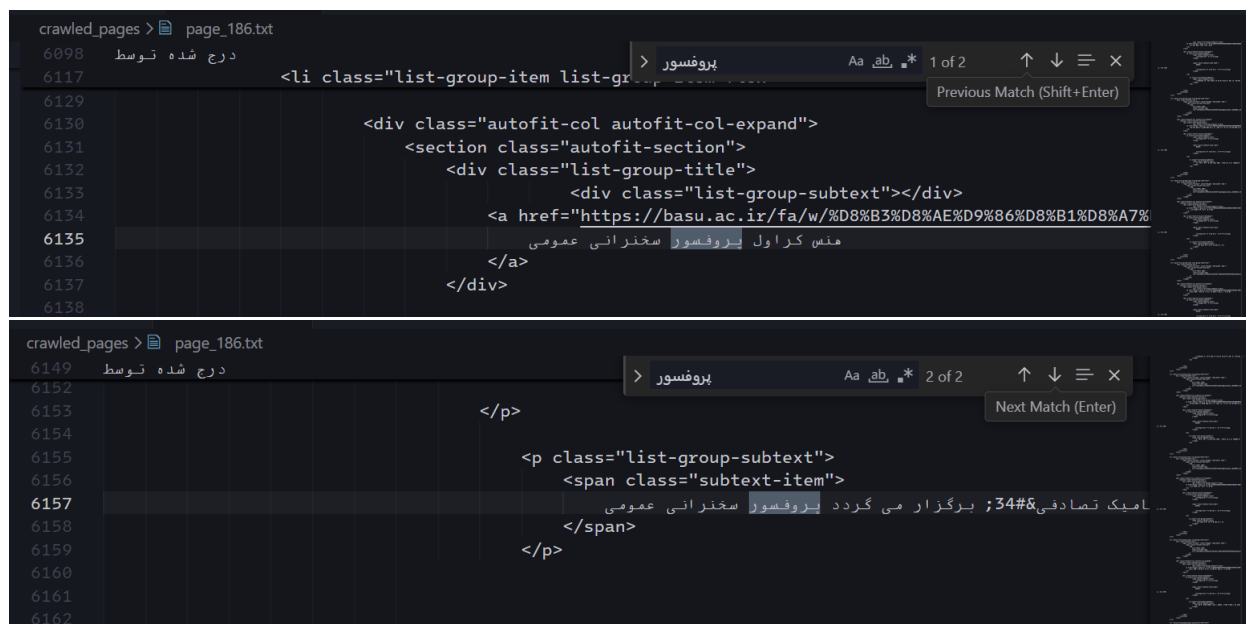
```
Microsoft Windows [Version 6.0.6002.18005]
(c) Microsoft Corporation. All rights reserved.

C:\Users\moham\Desktop\New folder>python retrieve_page_name.py
Enter your search query: روسفوپ
Word: "روسفوپ" found in:
- crawled_pages\page_186.txt (frequency: 2)

C:\Users\moham\Desktop\New folder>
```

با توجه به نتیجه بدست آمده کلمه پروفیسور در فایل crawled_page\page_186.txt به تعداد ۲ بار (frequency = 2) آمده است.

محتوای فایل crawled_page\page_186.txt که کلمه پروفیسور در آن آمده .



The image shows two screenshots of a text editor displaying HTML code from a file named 'crawled_page\page_186.txt'. The editor has a dark theme and a search bar at the top right. The search term 'پروفیسور' is entered, and the results are shown in a list on the right side of the editor. The first screenshot shows the search results for 'پروفیسور' in the HTML code, with the first match highlighted. The second screenshot shows the search results for 'پروفیسور' in the HTML code, with the second match highlighted. The search bar shows '1 of 2' and '2 of 2' respectively.

```
6098 درج شده توسط
6117 <li class="list-group-item list-group-item-text">
6129
6130 <div class="autofit-col autofit-col-expand">
6131 <section class="autofit-section">
6132 <div class="list-group-title">
6133 <div class="list-group-subtext"></div>
6134 <a href="https://basu.ac.ir/fa/w/%D8%B3%D8%AE%D9%86%D8%B1%D8%A7%
6135 هنس کراول پروفیسور سخنرانی عمومی
6136 </a>
6137 </div>
6138

6149 درج شده توسط
6152 </p>
6153
6154 <p class="list-group-subtext">
6155 <span class="subtext-item">
6156
6157 ایک تصادفی#34; برگزار می گردد پروفیسور سخنرانی عمومی
6158 </span>
6159 </p>
6160
6161
6162
```

در این تمرین کد crawler و همچنین دو کد جانبی برای جستجوی بهتر پیاده سازی شده است.

Retrieve Link Code

PostingListSearcher کلاس

این کلاس برای جستجوی کلمات در فایل فهرست پستینگ طراحی شده است.

متد __init__

این متد سازنده کلاس است که فایل فهرست پستینگ را می‌گیرد و آن را بارگذاری می‌کند.

```
1 def __init__(self, posting_list_file):
2     self.posting_list_file = posting_list_file
3     self.index = self.load_posting_list()
4
```

posting_list_file : مسیر فایل فهرست پستینگ را دریافت می‌کند.

self.index : فهرست پستینگ را با استفاده از متد load_posting_list بارگذاری می‌کند.

متد load_posting_list

```
1 def load_posting_list(self):
2     with open(self.posting_list_file, 'r', encoding='utf-8') as file:
3         return json.load(file)
4
```

این متد فایل فهرست پستینگ را باز کرده و به صورت JSON بارگذاری می‌کند.

```

1  def search(self, query):
2      query_words = query.lower().split()
3      result = {}
4
5      for word in query_words:
6          if word in self.index:
7              result[word] = self.index[word]
8          else:
9              result[word] = []
10
11     return result
12

```

این متد کلمات مورد جستجو را در فهرست پستینگ جستجو می کند و نتایج را برمی گرداند.

Query : عبارت جستجو را دریافت می کند.

query_words : کلمات موجود در عبارت جستجو را به صورت لیست جدا می کند.

Result : نتایج جستجو را در یک دیکشنری ذخیره می کند.

متد display_results

```
1 def display_results(self, results):
2     for word, files in results.items():
3         file_counter = Counter(files)
4         sorted_files = sorted(file_counter.items(), key=lambda item: item[1], reverse=True)
5
6         print(f'Word: "{word}" found in:')
7         for file, frequency in sorted_files:
8             print(f' - {file} (frequency: {frequency})')
9         if not files:
10             print(f' - No results found for "{word}".')
11
```

این متد نتایج جستجو را نمایش می‌دهد.

Results : نتایج جستجو که توسط متد search تولید شده است.

file_counter : شمارش تعداد تکرار هر فایل برای هر کلمه را انجام می‌دهد.

sorted_files : فایل‌ها را بر اساس تعداد تکرار مرتب می‌کند و آن‌ها را نمایش می‌دهد.

استفاده از کلاس PostingListSearcher



```
1 posting_list_file = 'web_links_posting_list.json'
2 searcher = PostingListSearcher(posting_list_file)
3
4 query = input("Enter your search query: ")
5 results = searcher.search(query)
6 searcher.display_results(results)
7
```

برای استفاده از این کلاس، ابتدا فایل فهرست پستینگ را مشخص کرده و یک شیء از کلاس ایجاد می‌کنیم.

سپس یک عبارت جستجو را از کاربر دریافت کرده و جستجو را انجام می‌دهیم.

Query : عبارت جستجو را از کاربر دریافت می‌کند.

Results : نتایج جستجو را با استفاده از متد search دریافت می‌کند.

display_results : نتایج جستجو را نمایش می‌دهد.

Retrieve Page Name

این کد نیز همانند retrieve link عمل میکند و همان بخش هارا نیز دارد با این تفاوت که نتیجه ای که میدهد نام فایل های متنی است.

```
1 import json
2 from collections import Counter
3
4 class PostingListSearcher:
5     def __init__(self, posting_list_file):
6         self.posting_list_file = posting_list_file
7         self.index = self.load_posting_list()
8
9     def load_posting_list(self):
10         with open(self.posting_list_file, 'r', encoding='utf-8') as file:
11             return json.load(file)
12
13     def search(self, query):
14         query_words = query.lower().split()
15         result = {}
16
17         for word in query_words:
18             if word in self.index:
19                 result[word] = self.index[word]
20             else:
21                 result[word] = []
22
23         return result
24
25     def display_results(self, results):
26         for word, files in results.items():
27             file_counter = Counter(files)
28             sorted_files = sorted(file_counter.items(), key=lambda item: item[1], reverse=True)
29
30             print(f'Word: "{word}" found in:')
31             for file, frequency in sorted_files:
32                 print(f' - {file} (frequency: {frequency})')
33             if not files:
34                 print(f' - No results found for "{word}".')
35
36 # Usage
37 posting_list_file = 'page_files_posting_list.json' # Update this path if needed
38 searcher = PostingListSearcher(posting_list_file)
39
40 query = input("Enter your search query: ")
41 results = searcher.search(query)
42 searcher.display_results(results)
43
```