SIMS

Carl Pettersson, Jakob Hallin, Gianluca Von Ehrenberg

Nov 02, 2023

CONTENTS:

CHAPTER

ONE

SIMS-WEBCRAWLER

1.1 main module

main.**crawl_all**(*data_dict*, *web_crawler*, *index*)

Crawl websites based on all values in a data dictionary. :param data_dict: The data dictionary containing key-value pairs. :param web_crawler: The WebCrawler instance for web crawling. :param index: The index used for generating temporary CSV file names. :return:

main.**crawl_with_id**(*id_val*, *web_crawler*, *index*)

Crawl specific websites based on an ID value. :param id_val: The ID value to search for on specific websites. :param web_crawler: The WebCrawler instance for web crawling. :param index: The index used for generating temporary CSV file names. :return:

main.**crawl_with_keys**(*data_dict*, *web_crawler*, *index*)

Crawl websites based on specific keys in a data dictionary. :param data_dict: The data dictionary containing key-value pairs. :param web_crawler: The WebCrawler instance for web crawling. :param index: The index used for generating temporary CSV file names. :return:

main.**empty_folder**(*path*)

Delete everything in a folder. (Be careful with this) :param path: The path to the folder to be emptied. :return:

main.**run**(*file*)

Main function to execute the web crawling and post-processing of an Excel file. :param file: The path to the input Excel file. :return:

main.**run_crawl**(*data*)

Run the web crawling process using a list of dictionaries. :param data: A list of dictionaries representing the data to be used for web crawling. :return:

main.**run_identify**(*file*)

Run the identification process on an Excel file and return the processed data. :param file: The path to the input Excel file. :return: List: A list of dictionaries representing the processed data.

main.**run_postprocess**(*file*)

Run the post-processing of an Excel file. :param file: The path to the input Excel file. :return: Creates a result.xlsx

1.2 postprocess module

class postprocess.**CSVHandler**

Bases: object

A class for handling CSV files and extracting information from them.

Methods:

has_only_pdf_urls(csv_file):
Check if a CSV file contains only PDF URLs and return a boolean indicating this and a list of PDF URLs.

find_all_pdf_urls(csv_file):
Find and return all PDF URLs in a CSV file.

static **find_all_pdf_urls**(*csv_file*)

Find and return all PDF URLs in a CSV file.

Args:
csv_file (str): The path to the CSV file to be processed.

Returns:
list: A list of PDF URLs found in the CSV file.

static **has_only_pdf_urls**(*csv_file*)

Check if a CSV file contains only PDF URLs and return a boolean indicating this and a list of PDF URLs.

Args:
csv_file (str): The path to the CSV file to be checked.

Returns:
tuple: A tuple containing a boolean (True if only PDF URLs, False otherwise) and a list of PDF URLs.

class postprocess.**ExcelProcessor**(*input_file*, *output_file*, *csv_directory*)

Bases: object

A class for processing Excel files, generating reports, and integrating data from CSV files.

Attributes:
input_file (str): The path to the input Excel file. output_file (str): The path to the output Excel file. csv_directory (str): The directory where CSV files are stored.

Methods:

_fill_row(row, fill_color):
Fill cells in a row with a specified fill color.

_add_to_row(sheet, row_index, num_cols, found_pdfs):
Add data to a row in an Excel sheet based on the presence of PDF URLs.

_process_csv_files(sheet, num_rows, num_cols):
Process CSV files associated with the Excel sheet and update cell fill colors accordingly.

process_excel():
Process the input Excel file, integrate data from CSV files, and generate an output Excel file.

process_excel()

Process an Excel file and create a result.xlsx file :return:

1.3 ui module

ui.**crawl_with_id**(*id_val*, *web_crawler*, *index*)

ui.**crawl_with_keys**(*data_dict*, *web_crawler*, *index*)

ui.**updateNumbers**()

1.4 WebCrawler module

class crawler.**WebCrawler**

Bases: object

A web crawler class for scraping and processing web content.

Attributes:

_MAX_RETRIES (int): Maximum number of retries for fetching a URL. _RETRY_DELAY (int): Number of seconds to wait between retries. driver (webdriver.Chrome): Selenium WebDriver for web scraping. link_queue (queue.Queue): Queue for managing URLs to be scraped. visited (set): Set to keep track of visited URLs. ignore_list (list): List of URLs to be ignored during crawling.

Methods:

set_ignorelist_url():
Read a list of URLs from 'WebCrawler/resources/ignoreUrls.csv' and set it as the ignore list.

setup_headless_chrome():
Set up and configure a headless Chrome WebDriver for web scraping.

get_html_content(url):
Fetch the HTML content of a URL, handling retries and waiting for page load.

has_product(html_content):
Check if the HTML content contains product-related information.

is_pdf(url):
Check if a URL points to a PDF file.

clean_html_content(html_content):
Clean the HTML content by removing specified elements and extracting text.

remove_all_children(element):
Remove all child elements from a BeautifulSoup element.

find_valid_links(html_content):
Find valid links in the HTML content.

is_valid_link(url):
Check if a URL is valid for crawling, considering file extensions and other criteria.

save_content_to_csv(content, csv_filename):
Save content to a CSV file in the 'temp_files' folder.

is_search_engine_url(url):
Check if a URL belongs to a search engine and should be skipped during crawling.

extract_search_engine_links(html_content):
Extract search engine result links from HTML content.

crawl_website_with_depth(csv_filename, depth_limit, start_url):
Crawl a website up to a specified depth, saving content to a CSV file.

close():
Close the Selenium WebDriver instance.

`clean_html_content`(*html_content*)

`close`()

`crawl_website_with_depth`(*csv_filename*, *depth_limit*, *start_url*)

`static extract_search_engine_links`(*html_content*)

`static find_valid_links`(*html_content*)

`get_html_content`(*url*)

`static has_product`(*html_content*)

`static is_pdf`(*url*)

`is_search_engine_url`(*url*)

`static is_valid_link`(*url*)

`static remove_all_children`(*element*)

`static save_content_to_csv`(*content*, *csv_filename*)

`static set_ignorelist_url`()

`static setup_headless_chrome`()

1.5 ExcelIdentifier module

`class excelidentify.ExcelManip`(*data_file*, *brands_file='resources/Brands.xlsx'*)

Bases: `object`

A class for processing Excel data and extracting information from it.

Attributes:
data_file (str): The path to the Excel file containing the data to be processed. brands_file (str): The path to the Excel file containing brand names (default: 'resources/Brands.xlsx'). brand_names (list): A list of brand names loaded from the brands_file.

Methods:

_load_brands(self):
Load brand names from the 'brands_file' Excel file and return them as a list.

_identify_brands(self, input_string):
Identify brand names in the input string and return the modified string with the brand removed and the identified brand name (if any).

_identify_rsk(s):
Identify 7-digit numbers in the input string and return the modified string with the numbers removed and a list of identified numbers (if any).

_identify_column(row, column_name):
Identify a specific column's value in a DataFrame row, remove it, and return the modified row along with the extracted value.

clean_row(row):
Clean and format a row by converting NaN values to empty strings and joining the row elements into a string.

pre_process(self):
Process the data from the 'data_file' Excel file and extract relevant information. Return the processed data as a list of dictionaries, where each dictionary represents a row of data.

`static clean_row(`*row*`)`

`pre_process()`

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX