Hindawi
Mobile Information Systems
Volume 2023, Article ID 9757813, 1 page
https://doi.org/10.1155/2023/9757813

Retraction

Retracted: Analysis of Application Data Mining to Capture Consumer Review Data on Booking Websites

Mobile Information Systems

Received 17 October 2023; Accepted 17 October 2023; Published 18 October 2023

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope

(2) Discrepancies in the description of the research reported

(3) Discrepancies between the availability of data and the research described

(4) Inappropriate citations

(5) Incoherent, meaningless and/or irrelevant content included in the article

(6) Peer-review manipulation

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] Y. Tsai, C. Lin, and M. Lee, "Analysis of Application Data Mining to Capture Consumer Review Data on Booking Websites," *Mobile Information Systems*, vol. 2022, Article ID 3062953, 15 pages, 2022.

Hindawi
Mobile Information Systems
Volume 2022, Article ID 3062953, 15 pages
https://doi.org/10.1155/2022/3062953

Research Article

Analysis of Application Data Mining to Capture Consumer Review Data on Booking Websites

Yao-Hsu Tsai,[1] **Chien-Cheng Lin**,[1] **and Min-Hsien Lee**[2]

[1]*Master Program of College of Tourism, Chung Hua University, Hsinchu 30012, Taiwan*
[2]*Department of Food and Beverage Management, Yuanpei University of Medical Technology, Hsinchu 30015, Taiwan*

Correspondence should be addressed to Min-Hsien Lee; minhsien@mail.ypu.edu.tw

Received 12 April 2022; Accepted 15 July 2022; Published 26 August 2022

Academic Editor: Yang Gao

The rapid development of the Internet has led to the prevalence of big data analysis. Data mining is crucial to extracting potentially valuable information from big data and has therefore received considerable attention from researchers. Python is a common programming language used in data mining. Because of its rich database and robust capacity for scientific calculations, Python is considered an irreplaceable tool for data mining. This study adopted Python to perform a data mining analysis on visitor comments on Booking.com. The study was divided into several stages, namely, data source selection, data acquisition, data saving, data preprocessing, indexing of comments on Booking.com through the Python-based Scrapy framework, and user operation simulation through Selenium to analyze the performance of the spider program. Data mining can be used to identify useful information, which can serve as references for consumers to make purchase decisions. Extraction of data from booking sites through spider programs enables site administrators to attract more visitors. Analysis of extracted data also facilitates the elimination of misjudged comments and helps hotels improve their service quality, hardware, and personnel training.

1. Introduction

1.1. Background and Relevant Research. Web crawling plays an essential role in data inquiry and mining. Research on web crawling began in the 1990s, and the technology has matured significantly since then. Web crawling uses a spider, which automatically visits and downloads websites. Spiders can be used to quickly download a large volume of data, abolishing the need for various manual tasks. An example of a spider is the Internet Research Lab spider, which can extract considerable amounts of data from millions of websites. A report indicated that this program has been used to acquire data from 60 million websites [1]. The Mercator crawling system developed by Heydon and Najork [2] in Compaq utilizes the synchronization function of Java to achieve multithread crawling. To improve crawling efficiency, the Mercator system uses multiple optimization strategies, such as DNS caching and delayed saving [3–6]. The Fish-Search crawling algorithm based on the content and ratings in a link was first proposed by De Bra and Post [7]. This algorithm mimics the foraging and reproduction behaviors of fish and searches for websites related to a selected topic on the assumption that relevant websites are logically associated with each other [8–12]. The algorithm uses a binary model to determine whether a website is associated with the selected topic. Page et al. [13] invented the PageRank algorithm in the late 1990s. It is used to rank websites in a search engine and is currently used in the Google search engine. It proposed a search strategy that constructs a probability model according to website content and URL structure characteristics and uses the model to calculate a usefulness score for each link to control the crawling process [14–16]. In Taiwan, research on web crawling technology started relatively late but has developed rapidly [17]. Starting in 2003, various domestic conferences on data mining have been held, which contributed to the prevalence of data mining in studies related to crawling. Among scholars exploring big data analysis, Jiang et al. [18] proposed a genetic algorithm–XGBoost machine learning classifier, which combines histogram of oriented gradients

and local binary pattern descriptors to describe pedestrian features and form a novel static pedestrian image analysis method. Gao et al. [19] created an intelligent transportation system to address the complexity and information redundancy problems in existing models. The system is composed of a queue length sensing model based on vehicle-to-everything communication. Their results confirmed that the model accuracy is proportional to the penetration rate of connected vehicles, and the queue length can be sensed even in a low penetration rate environment. In mixed traffic environments, the model performs equally well for connected and unconnected vehicles. Moreover, its performance is equivalent to that of the probability distribution model under a high penetration rate, and the performance remains favorable under low penetration rates. Guo et al. [20] asserted that in wireless sensor networks (WSNs), resource consumption is highly correlated with the execution of missions, which must consume computing and communication bandwidth. They proposed self-adapted task scheduling strategies for WSNs and adopted a dynamic alliance discrete particle swarm optimization algorithm with an efficient particle position code and fitness function. Wang et al. [21] developed a novel self-adapted intelligent routing scheme for WSNs. Compared to other WSN routing schemes based on ant colony optimization, it exhibits more favorable network performance in terms of energy consumption, energy efficiency, and packet delivery latency. Wan et al. [22] maintained that in WSNs, highly dense node distribution leads to transmission collision and energy dissipation caused by redundant data. To address this problem, they proposed an energy-efficient sleep scheduling mechanism with a similarity measure for WSNs. This mechanism puts sensors into the state of sleep or awaken to effectively reduce energy consumption. Under the premise of maintaining data accuracy, the proposed mechanism achieves favorable performance in clustering accuracy and energy efficiency.

Word-of-mouth is the process of informal human-to-human communication about sellers, products, and services that share their experiences and pros and cons with other consumers. WOM communicates messages through people's direct face-to-face communication in daily life, so the delivery process is short-lived and disappears after it is sent. Villanueva et al. [23] found that consumers' trust value in electronic WOM is higher than in traditional marketing channels. Since eWOM is a condition that consumers write down and use online and does not disappear immediately; on the contrary, other consumers can still see these messages after a long period of time [24]. On the Internet, when buyers search for goods or services, consumer opinions are stored for a long time and seen by most buyers, and become the reference point for buyers when purchasing goods or services [25–29]. Although eWOM is different from advertising, operational writers and messages that exaggerate eWOM must also be prevented. How to distinguish the authenticity of unfamiliar information that consumers have no way of understanding depends on the review method of the website. For example, Booking websites (agoda, expedia) can only be written by consumers who book accommodation. Use data mining analysis methods, from the breadth, depth, and scale of data collected and the ability to analyze data, to solve problems that actually occur [30]. Internet technology shares the second dissemination of product reviews, such as Twitter, Facebook, LINE, Wechat, WhatsApp, countless blogs, emails, etc.

1.2. Research Objectives. This study used Python to code a crawling program that collects and analyzes visitor comments on the Booking.com website. The crawler enables the exploration of the ratings given by visitors and the weights of these ratings. The research objectives were as follows:

(1) To conduct a literature review on web crawling and provide a reference for subsequent researchers.

(2) To use a Python-based spider program to extract visitor comments on booking sites, analyze the ratings given by visitors, and simulate the amount of data extracted and time spent by the spider program versus the data extracted and time spent by manual data extraction.

1.3. Key Technology. The list of Python-related technology used in this study was as follows:

(1) Data Acquisition: Technologies related to web crawling and databases were used for data acquisition. Specifically, the Python-based Scrapy framework was used to acquire data, MongoDB was used to save data, and PyMongo was used for data conversion and other related tasks.

(2) Data Preprocessing: Data were preprocessed in batches. The Pandas, NumPy, and Matplotlib tools in Python provide strong matrix calculation functions, facilitating the rapid preprocessing of the vast amount of data.

2. Theoretical and Technological Bases

2.1. Theoretical Background and Web Crawling. Informal comments on products or services are communicated to other consumers after consumers use them (WOM); virtual, online commenting on products and services is called electronic word-of-mouth [31]. Word-of-mouth is a non-commercial positive and negative comment on a product or service between people. Consumers' word-of-mouth comments share their experiences, ideas, and information about a product with other people. For example, talk about the quality of the meal or the experience of using a certain product recently, discuss the gossip about sports events, share the quality of travel accommodation and colleagues' opinions on the company, etc. Due to the rapid growth of social media and CGM's huge amount of review materials, it has affected the hospitality industry [32–35]. In 2017, the global online travel business accounted for 43% of the total travel market, and the proportion of online bookings increased by 10.5% per year growth [36]. Consumer-generated content on social media and the Internet is quickly inspiring

people to develop so-called big data analytics to solve real-life problems. Although a few studies have employed new data sources to address important research questions in the hospitality industry, big data analytics techniques have not been systematically applied in these studies. Therefore, using traditional methods to understand the reviews of Internet users in the field of food and travel cannot achieve good results. Use big data analysis methods to solve real problems from the breadth, depth, and scale of data collected and the ability to analyze data [30].

Web crawling technology first emerged in 1996. It has multiple names, including web data mining, web knowledge discovery, and online data mining. It is a product of data mining and web technology and involves the extraction and organization of valuable data from a variety of websites. The extracted data are converted, analyzed, and transformed to identify potentially useful information, examine existing information, evaluate current situations, and make predictions. Web data mining possesses considerable commercial and scientific research value [37–41].

The development of big data analysis and the prevalence of web technology have accentuated the value of web data mining. Currently, web data mining is considered one of the most widespread applications in the field of data mining. It can be used to effectively analyze valuable information, such as user behavior and preferences, and the analysis results can be integrated with other commercial strategies to help companies generate substantial benefits [42].

Web data mining has several unique characteristics:

(1) Complexity: Web data are diverse and complex. The rapid development of the Internet leads to the presentation of diverse and intertwined data by various industries in numerous formats, including text, videos, audio, and images. Therefore, how to quickly, logically, and accurately acquire target data is the main topic for data mining.

(2) Dynamicity: The development of web technology has facilitated the diversification of the types of online data. Advanced technology enables large volumes of data to be uploaded to the Internet each day. The update and repeated computation of data are essential to web data management. Ensuring that collected data is up-to-date is a major challenge of web data mining.

(3) Heterogeneity: When the Internet was first introduced, the structures of websites were not standardized, leading to substantial differences between data sources. Data mining must overcome the challenge of acquiring data from websites of varying structures.

2.2. Web Data Mining Process

(1) Web data acquisition: this stage involves the analysis of website content to acquire useful information. Raw data needed for data mining are acquired at this stage, and such data feature complex and diverse structures. Since the collected raw data have yet to be processed, they contain a large amount of repeated information.

(2) Web data preprocessing: the stage involves the initial processing of collected data. Algorithms and analysis methods are used to sort the data. These methods include noise reduction, lattice reduction, and discretization. Through preprocessing, raw data are converted to more useful data, thereby reducing the computation time for subsequent data mining and the cost of data analysis.

(3) Data conversion and integration: in this stage, preprocessed data are formatted and saved in databases for subsequent data mining. Specifically, preprocessed data are loaded into databases with preconfigured structures. This expedites the subsequent process of data extraction, addition, deletion, adjustment, and inquiry, thereby enhancing the efficiency of data mining.

(4) Mode identification: categorization algorithms, correlation analysis, clustering analysis, and other statistical methods are used to mine the data saved in structured databases, the results of which can be used to determine suitable mining models.

(5) Data analysis: existing data mining tools are used to perform analysis. The analysis results are converted to graphs for direct visualization, assessment, and reasonable description of the mined information.

2.3. Common Web Data Mining Technology.

As mentioned in the previous section, various data extraction methods and technologies are used during data mining. Several common web data mining tools are as follows:

(1) Programming Languages: Common languages used for data mining are Python, Java, and C#.

(2) Python Tools: Common tools used with Python programs include the Selenium framework, the Requests library, the Scrapy framework, the Pandas library, and multiprocessing.

3. Web Crawling System Requirements

This study examined a fairly common data mining target, an online Booking website, and various aspects of this target were analyzed, compared, and explained. All data used in this study came from open data on the Internet, and an account login was not required to view the data. The data source was the official Chinese website of Booking.com (https://www.booking.com/). Booking.com provides online booking services for over 120,000 accommodations worldwide, in addition to offering various discounts and special deals. In this study, a spider program was used to extract data for 12 regions in Taiwan (namely Taipei City, Kaohsiung City, Taichung City, Tainan City, Puli Township, Liuqui, Luodong Township, Taitung City, Keelung City, Jiaoxi Township, Taoyuan City, and Yilan County).

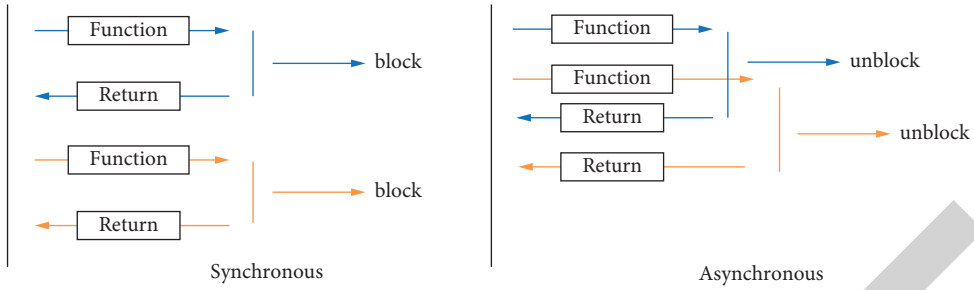

Figure 1: Synchronous and asynchronous processes.

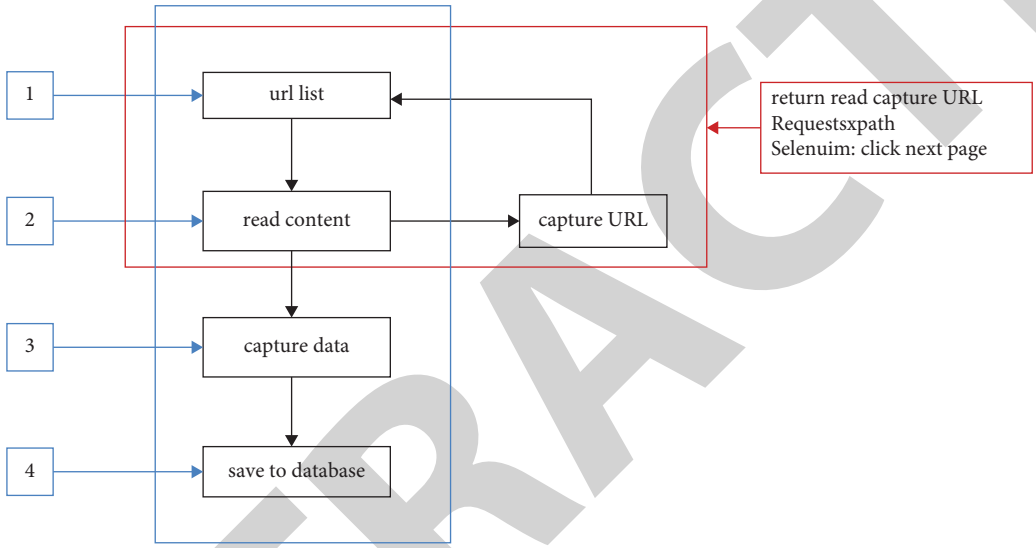

Figure 2: Flowchart of web crawling through Scrapy.

4. Overall Design of Web Crawling Programs

4.1. Program Design and Algorithm. The slowest process of a spider program is request sending. When a program uses the request, get method to send a request to a web page, the spider program cannot control the amount of time needed by the target server to return a response. The program can encounter problems such as disconnection, slow connection speed, and unstable connection. Two common strategies used to optimize the request sending process are to configure the length of the wait time and send multiple requests repeatedly.

4.1.1. Concepts of Synchronization and Asynchronization. Before the concept of synchronization and asynchronization is introduced, the concept of a blocked process should be described. A blocked process is paused because it lacks the authorization to use specific resources (e.g., those of a central processing unit).

As shown in Figure 1, if Function 2 needs to receive Return 1 from Function 1 before receiving Return 2, the overall process is synchronous. By contrast, if Function 2 does not need to receive Return 1 from Function 1 before receiving Return 2, the overall process is asynchronous. In the synchronous process, Function 1/2 is blocked until response.

4.1.2. Scrapy Framework. Scrapy is a framework designed to crawl web page data and extract structured information. It only requires a small amount of code to rapidly acquire web page data. Scrapy uses the asynchronous Twisted framework, which is written with Python and supports numerous Internet protocols, including UDP, TCP, and TLS. This allows for high-speed web crawling. Figure 2 shows a flowchart of web crawling through Scrapy. First, a list is created to include all websites to be processed, followed by the processing of the returned content. Next, target data are extracted from the returned content. Finally, the extracted data are saved in a database.

Figure 3 shows the flowchart for optimization using a queue structure, which prioritizes the more essential components. Specifically, components 1 and 4 in Figure 2 are optimized through the queue structure.

The flowchart of the Scrapy framework is divided into five components (Figure 4). The scheduler is also a queue structure for describing the request items saved by the

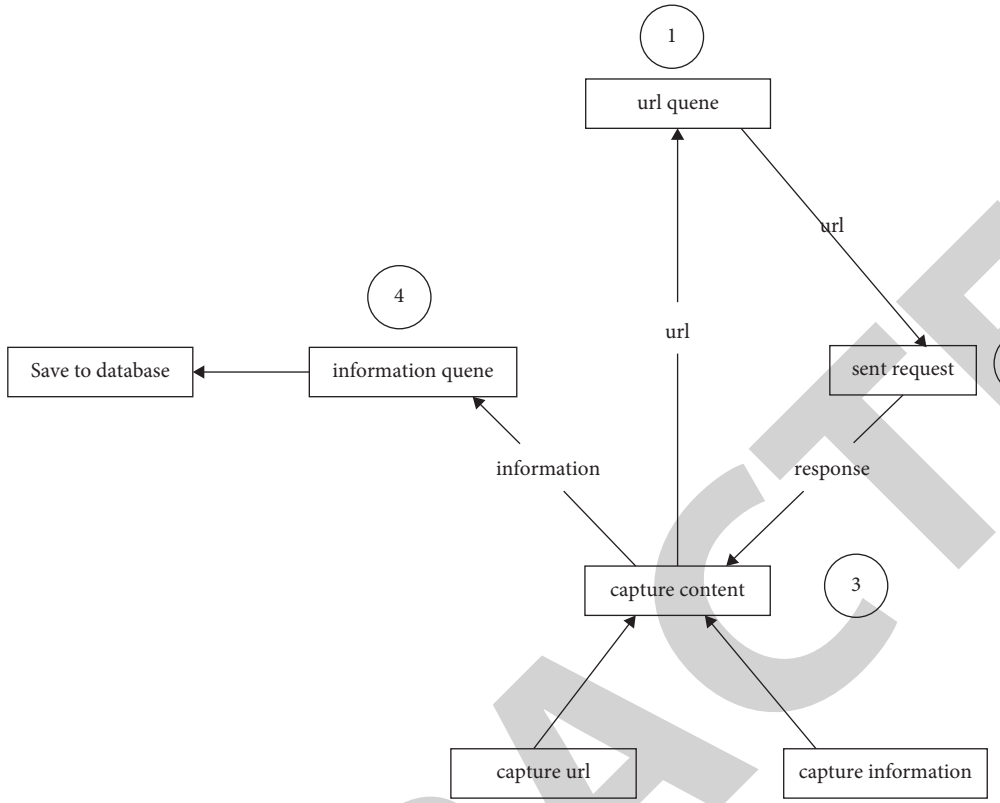

Figure 3: Optimized flowchart of scrapy.

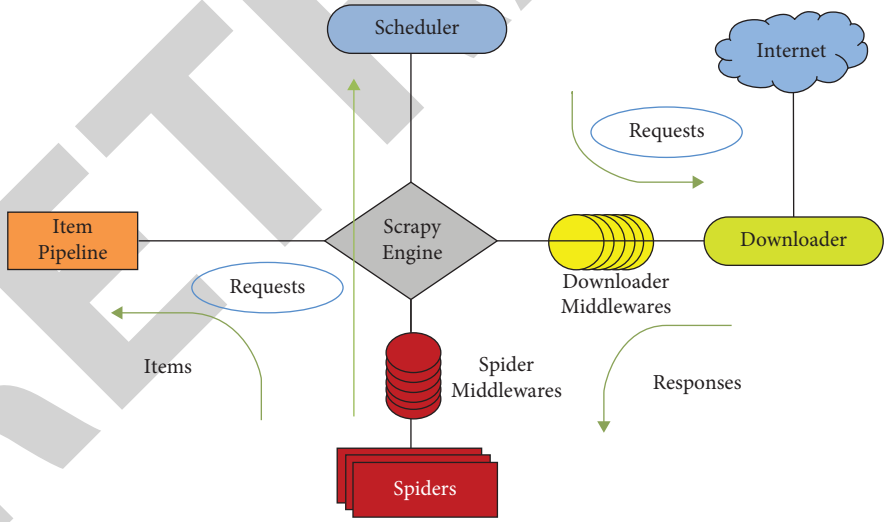

Figure 4: Flowchart of the Scrapy framework.

developer. A URL is sent from the scheduler to the downloader, which then sends a request to the target server and processes the responses from the server. The received responses are then sent from the downloader to the spiders, which complete two tasks, namely, data extraction and URL extraction. The extracted URLs are combined to form request items, which are saved by the scheduler. The extracted data are sent to the item pipeline for processing.

The Scrapy engine is considered the core of the framework and mainly serves to coordinate all components and modulate data transfer processes. The downloader middleware filters and modifies the request items sent from the scheduler to the downloader. The spider middleware adjusts the responses sent from the downloader to spiders. Accordingly, the Scrapy framework is based on task specifications and high-efficiency processing to minimize the response time of each process. (Algorithm 1).

(1) Take the input as IURL from the user
(2) Process (URL)
{ Get the main URL in lower case
if the web page exists
apply timestamp
else display error message and
exit } Initialize the queues "todo" = URL and done'
initialize $n = 1$; Extract links(todo[k]) {
While(todo ! empty) do If the page has frames then { For($i = 1$ to no.of frames) do
Extract links(URLi) } If the page has forms then Omit them
Else { While(!EOF URL) do { Parse the document for anchor tags <a> Get the
child URL – CURL If the domains of the URL and CURL are equal
Then Enter CURL in to "todo" queue
Else Extract links(todo[++k]) } } Enter the url in to "done" queue and increment n }
Initialize the array select[n]6. Display the web graph
selection() { For (all web graph nodes) do { If (web graph node is selected)
Select[i] = 1 Else Select[i] = 0 } } download(select[n])
{ For($i = 1$ to n) If select[i] = 1 then Save done[i] in temp folder }
Update (URLs) Exit

Algorithm 1: Web Crawling Algorithm [43].

Table 1: Web crawling and anticrawling strategies.

	Web crawling	Anticrawling
Strategies	Sending requests to websites and acquiring data	When the view count of a website increases drastically during a specific period, all views are from the same IP address, and all user agents are Python-based, the manager limits the access from the IP address to the website
	Simulating a user agent and acquiring a proxy IP	When the view count is abnormal, all users are required to log in to their accounts before viewing the website
	Registering an account and visiting a website through cookies or tokens	A complete account database is established, and each account must have clearance to review specific information
	Mimicking user operations by restricting the request sending frequency	A verification code is used to determine whether website visitors are real people
	Passing the required authentication (e.g., OpenCv authentication)	Dynamic loading pages are introduced, in which data are loaded through JavaScript to increase the difficulty of website analysis
	Using Selenium and PhantomJS to fully mimic the browsing behavior of real users	

```
from scrapy import signals
class BookingspiderSpiderMiddleware(object):
    @classmethod
    def from_crawler(cls, crawler):
    s = cls()
    crawler.signals.connect(s.spider_opened, signal=signals.spider_
    return s
def process spider input (self, response, spider):
    return None
```

Figure 5: Middlewares.py.

```
class BookingspiderPipeline(object):
    def process item(self, item, spider):
        return item
```

Figure 6: Piplines.py.

```
BOT_NAME = 'BookingSpider'
SPIDER_MODULES = ['BookingSpider.spiders']
NEWSPIDER_MODULE = 'BookingSpider.spiders'
ROBOTSTXT_OBEY = True
```

Figure 7: Settings.py.

```
def parse(self, response):
    result01=response.xpath("/html/head/title/text()")
    print(result01)
```

Figure 8: Start_urls.

```
['scrapy.extensions.corestats. CoreStats',↵
 'scrapy.extensions.telnet TelnetConsole',↵
 'scrapy.extensions.logstats.LogStats']↵
['scrapy.downloadermiddlewares.robotstxt.RobotsTxtMiddleware',↵
 'scrapy.downloadermiddlewares.httpauth.HttpAuthMiddleware',↵
 'scrapy. downloadermiddlewares.downloadtimeout. DownloadTimeoutMiddleware',↵
 'scrapy.downloadermiddlewares.defaultheaders. Default HeadersMiddleware',↵
 'scrapy.downloadermiddlewares.useragent. User Agent Middleware',↵
 'scrapy.downloadermiddlewares.retry. RetryMiddleware',↵
 'scrapy, downloadermiddlewares.redirect.MetaRefreshMiddleware',↵
 'scrapy.downloadermiddlewares.httpcompression. HttpCompressionMiddleware',↵
 'scrapy.downloadermiddlewares.redirect.RedirectMiddleware',↵
 'scrapy. downloadermiddlewares.cookies. CookiesMiddleware',↵
 'scrapy.downloadermiddlewares.httpproxy. HttpProxyMiddleware',↵
 'scrapy, downloadermiddlewares.stats. DownloaderStats']↵
['scrapy.spidermiddlewares.httperror.HttpErrorMiddleware',↵
 'scrapy.spidermiddlewares.offsite. OffsiteMiddleware',↵
 'scrapy.spidermiddlewares.referer. RefererMiddleware',↵
 'scrapy.spidermiddlewares.urllength. UrlLengthMiddleware',↵
 'scrapy.spidermiddlewares.depth. DepthMiddleware']↵
```

Figure 9: Scrapy crawl bookingtw.

4.2. Anticrawling Strategies and Response Measures. Web crawling simulates the process of request sending and data acquisition. A simple spider might place an excessive burden on the target server when the request sending frequency is overly high, crashing the website. Therefore, website managers adopt various measures against web crawling. Table 1 provides several common web crawling and anticrawling strategies.

5. Scrapy Operation

5.1. Establishment of Scrapy Items. The command Scrapy startproject mySpider is entered. The command is "scrapy startproject BookingSpider."

The item name is BookingSpider. In the template, the path is "d:\python37\lib\site-package\Scrapy\templates\project," whereas, in this study, the designated path was set to "d:\python37\lib\site-package\Scrapy\templates\project." The Scrapy framework then suggests that the commands cd BookingSpider and Scrapy genspider example example.com can be used to initiate the generated spider. The main directory crawlerBooking presents the structure of the generated BookingSpider file including _init_.py, _init_.py, items.py, middlewares.py, pipelines.py, settings.py, and scrapy.cfg.

The BookingSpider file includes a folder named spiders and a document named Scrapy.cfg. The scrapy.cfg is the configuration file for the entire spider program. settings.py then designates BookingSpider.settings as the configuration file. deploy is used to initiate the Scrapyd synchronization function. This function enables synchronization of the Scrapy framework in an online server or user machine.

The _init_.py is empty. The items.py is used to define the crawling target to import scrapy.

As shown in Figure 5, middlewares.py provides a template, which can be used to code the middleware and downloader.

As shown in Figure 6, piplines.py is used to create a process_item empty function for data processing and saving.

As shown in Figure 7, settings.py is used to configure the entire item content.

5.2. Definition of Spider. Entering the command Scrapy genspider bookingtw booking.com reveals that bookingtw has been created in BookingSpider.spiders. It presents the changes in the files in the BookingSpider folder following the execution of the genspider command. In bookingtw.py, the name attribute indicates the name of the spider program. In bookingtw.py, the parse method is used to process start_urls responses. Notably, the parse method must be used for crawling.

5.3. Data Extraction I. A complete spider uses methods such as XPath to extract data. An initial value is assigned for start_urls (https://booking.com/reviews/tw/city/t-ai-pei.zh-cn.html). Subsequently, the parse is modified. A spider was used for testing; hence, the parsed content was as shown in Figure 8.

```
D:\codeSpace\crawlerBooking\BookingSpider>Scrapy crawl
bookingtw
[<Selector xpath='/html/head/title/text()' data='\n30 best hotels in
Taipei according to 479,337 comments on Booking.com'>]
D:\codeSpace\crawlerBooking\BookingSpider>
#
def parse(self, response):
result01=response.xpath("/html/head/title/text()").extract_first()
print(result01)
#
ITEM_PIPELINES = {
  'BookingSpider.pipelines.BookingspiderPipeline': 300,
}
#
class BookingtwSpider(Scrapy.Spider):
   name = 'bookingtw'
   allowed_domains = ['booking.com']
   start_urls = ['https://booking.com/reviews/tw/city/t-ai-pei.zh-
cn.html']
   def parse(self, response):
      item = {}
      item["content"] =
response.xpath("/html/head/title/text()").extract_first()
      item["count"] = 1
      yield item
#
class BookingspiderPipeline(object):
   def process_item(self, item, spider):
      print(item)
      return item
```

Figure 10: Pipeline transfer data.

Entering the command Scrapy crawl bookingtw initiates the spider test (Figure 9). It presents the data log recorded after the spider is activated.

Under normal circumstances, users do not need to view the entire log. LOG_LEVEL can be added in settings.py to control the log display. LOG_LEVEL has four levels; in ascending hierarchical order, these levels are debug, information, warning, and error. After LOG_LEVEL is set to a specific level, only data at the said level and above are displayed by the log. For example, setting LOG_LEVEL to "warning" causes the lag to display only warning and error data, whereas debug and information data are hidden. Therefore, adding LOG_LEVEL = "WARNING" to settings.py yields the following result. Result01 is composed of multiple selector items. The source code is modified as follows. This enables data extraction from the selector.

5.4. Data Extraction II. To store data using pipelines, they must be activated in settings.py.

A value in ITEM_PIPELINES is 300. This value denotes the priority of the pipeline; a low value indicates high priority, meaning that data pass through pipelines with low values first. Next, the spider uses yield to transfer data.

The source code of a pipeline is shown in Figure 10.

The Scrapy crawl bookingtw command is executed again. It presents the execution results. Notably, the process_item process is necessary for pipelines for processing and saving data. This ends the basic process of the Scrapy framework.

6. Testing with Booking.Com

The overall web crawling process was conducted according to the procedures introduced in the previous section.

6.1. Configuration of Scrapy Parameters. Parameters were configured using settings.py.

The source code is explained as follows:

(1) Activating the pipeline, hotel review booking.pepelines.HotelReviewBookingPipeline, was activated. This pipeline was set to the highest priority.
(2) Parameters were configured according to robots.txt.
(3) The delay was set to 5 s during initialization.
(4) The maximum download delay was set to 60 s.
(5) HTTP caching was activated and configured.

6.2. Program Initiation. The command line was used to start the program. This was achieved through the cmdline packet in Scrapy. The source code is as follows:

6.3. Definition of Target Items. The item types to be crawled and saved were defined using the source code shown in Figure 11.

Description:

(1) The name of each geographical region was defined using city_name.
(2) The name of each hotel was defined using target.
(3) The rating of each hotel was defined using score.
(4) The date of each comment was defined using date.
(5) The overall rating was defined using overall_comment.
(6) Positive comments were defined using positive_comment.
(7) Negative comments were defined using negative_comment.

6.4. Definition of Spider. The spider was defined using the steps listed in the second subsection of the previous section. Relevant procedures are described as follows (Figure 12).

```
ITEM_PIPELINES = {
   'hotel_review_booking.pipelines.HotelReviewBookingPipeline': 1,
}

ROBOTSTXT_OBEY = True

AUTOTHROTTLE_START_DELAY = 5

AUTOTHROTTLE_MAX_DELAY = 60

HTTPCACHE_ENABLED = True
HTTPCACHE_EXPIRATION_SECS = 0
HTTPCACHE_DIR = 'httpcache'
HTTPCACHE_IGNORE_HTTP_CODES = []
HTTPCACHE_STORAGE = 'Scrapy.extensions.httpcache.FilesystemCacheStorage'
#
from Scrapy.cmdline import execute

execute(['Scrapy', 'crawl', 'hotel_review_booking'])
#
class HotelReviewBookingItem(Scrapy.Item):
   city_name = Scrapy.Field()
   target = Scrapy.Field()
   score = Scrapy.Field()
   date = Scrapy.Field()
   overall_comment = Scrapy.Field()
   positive_comment = Scrapy.Field()
   negative_comment = Scrapy.Field()
```

Figure 11: Definition of the target items.

(1) URLs linking to the comment pages shared by the target geographical regions were defined. The attribute of the URLs was defined using base_url.

(2) road_map was configured to define the review page and comment page of each geographical region.

(3) String shifting and time were defined.

(4) The initial request was defined. In start_request, the URLs to be crawled were spliced before returning to yield.

(5) The method for acquiring all comments for a hotel in a geographical region was defined as get_one_hotel_review_lists. The XPath coding for acquiring the comment framework of the hotel was defined as//ul [@class = 'rlp-main-hotels__container].

(6) The method for acquiring all comments for a geographical region was defined as get_one_review_list

(7) The method for acquiring a single hotel comment was defined as get_one_review_entity.

6.5. Definition of the Pipeline. The data acquired from web crawling were saved in a csv file. The source code is explained as follows (Figure 13).

(1) The Pandas database was used to rapidly save data.

(2) Parameters of the csv file were defined.

(3) The definition of saving data in the csv file was presented in the first line.

(4) Items sent by the spider were saved in the defined format.

(5) The acquired data were saved in the csv files.

(6) Appendix A presents the content of the csv file.

```
import re
import Scrapy
from Scrapy.selector import Selector
from Scrapy.http import Request
from hotel_review_booking.items import HotelReviewBookingItem
class MyCrawler(Scrapy.Spider):
    name = 'hotel_review_booking'
    allowd_domain = 'booking.com'
    pure_base_url = 'https://www.booking.com'
    base_url = 'https://www.booking.com/reviews/'
    road_map = {
      "t-ai-pei":
          {'url': 'tw/city/t-ai-pei.zh-cn.html?',
           'pages': 18, },
       "kao-hsiung":
          {'url':'tw/city/kao-hsiung.zh-cn.html?',
           'pages':20,            },
       "tai-chung":
          {'url': 'tw/city/tai-chung.zh-cn.html?',
           'pages': 20,           },
       "pu-li":
          {'url': 'tw/city/pu-li.zh-cn.html?',
           'pages': 20,           },
       "tainan":
          {'url': 'tw/city/tai-nan.zh-cn.html?',
           'pages': 20,           },
       "t-ai-tung.":
          {'url': 'tw/city/t-ai-tung.zh-cn.html?',
           'pages': 20,           },
       "pai-sha-wei":
          {'url': 'tw/city/pai-sha-wei.zh-cn.html?',
           'pages': 20,           },
       "lo-tung":
          {'url': 'tw/city/lo-tung.zh-cn.html?',
           'pages': 20,           },
       "chi-lung":
          {'url': 'tw/city/chi-lung.zh-cn.html?',
           'pages': 20,           },
       "chiao-hsi":
          {'url': 'tw/city/chiao-hsi.zh-cn.html?'
          'pages': 20,            },
       "taoyuan":
```

(a)

Figure 12: Continued.

```
        {'url': 'tw/city/taoyuan.zh-cn.html?',
        'pages': 20,            },
    "yilan":
        {'url': 'tw/city/yilan.zh-cn.html?',
        'pages': 20,            },
}
hotel_list_offset_str_1 = 'offset='
hotel_list_offset_str_3 = '&'
count = 1
pattern = r'(\d{4})year(\d{1,2})month(\d{1,2})day'
pattern_compiled = re.compile(pattern)
def start_requests(self):
    # max_pages = 27  # 20
    for k, v in self.road_map.items():
        city_name = k
        url = self.road_map[city_name]['url']
        pages = self.road_map[city_name]['pages']
        for index in range(0, pages):
            if index == 0:
                offs = ''
            else:
                offs = self.hotel_list_offset_str_1 + str(index * 30) + self.hotel_list_offset_str_3
            page_url = self.base_url + url + offs
            yield Request(page_url, callback=self.get_one_hotel_review_lists, meta={'city_name': city_nam
e})
    def get_one_hotel_review_lists(self, response):
        print(self.count)
        self.count += 1
        item['target'] = str_target
        item['date'] = str_date
        item['score'] = str_score
        item['city_name'] = city_name
        item['overall_comment'] = str_overall_comment
        item['positive_comment'] = str_positive_comment
        item['negative_comment'] = str_negative_comment
        return item
```

(b)

Figure 12: Definition of the Spider.

```
from hotel_review_booking.items import HotelReviewBookingItem
import codecs
import pandas as pd
import os
class HotelReviewBookingPipeline(object):
    def __init__(self):
        # replace data dir with your owns.
        self.base_data_path = r'/home/hucl/PycharmProjects/Scrapy_hotel_review/datas/'
        self.file_name = r'hotels.csv'
        self.pd_file = pd.DataFrame(
            columns={"target", "city_name", "date", 'score', 'overall_comment', 'positive_comment', 'negative_c
omment'})
    def process_item(self, item, spider):
        str_city_name = item['city_name']
        str_target = item['target']
        str_date = item['date']
        str_score = item['score']
        str_overall_comment = item['overall_comment']
        str_positive_comment = item['positive_comment']
        str_negative_comment = item['negative_comment']
        new_record = {"target": str_target,
                      "date": str_date,
                      "city_name": str_city_name,
                      "score": str_score,
                      "overall_comment": str_overall_comment,
                      "positive_comment": str_positive_comment,
                      "negative_comment": str_negative_comment}
        self.pd_file = self.pd_file.append(new_record, ignore_index=True)
        return item
def close_spider(self, spider):
        print(len(self.pd_file))
        self.pd_file = self.pd_file.reset_index(drop=True)
        self.pd_file.to_csv(self.base_data_path + self.file_name, sep="\t", index=False, header=True, encoding
="utf-8")
```

Figure 13: Definition of the pipeline.

Table 2: Performance comparison.

Method	Number of executed sequences	Amount of extracted data	Time consumption
Manual extraction	1	26561 lines	5 h
Simulated extraction	4	82278 lines	57 min

7. Conclusions

7.1. Program Execution Results and Analysis. After the source code was completed, 12 geographical regions in Taiwan (i.e., Taipei City, Kaohsiung City, Taichung City, Tainan City, Puli Township, Liuqui, Luodong Township, Taitung City, Keelung City, Jiaoxi Township, Taoyuan City, and Yilan County) were selected to verify the developed program and compare the speed and effectiveness with manual data collection. A manual extraction approach took approximately 5 h to extract 26561 lines of comments related to Taipei City, whereas the developed Scrapy method only spent 57 min to collect 82278 lines of comments (Table 2). This verified that in the absence of anticrawling measures, the Scrapy method was considerably more efficient in data extraction.

The results of program execution were not optimized because of limitations in time and resources. However, when the Python-based program was used to simulate the operation of real users to extract Chinese and English comments and ratings from the booking site, it outperformed the

manual extraction method in data acquired and time spent. Moreover, the developed program can be used to simultaneously extract data from multiple booking sites, thereby enabling a wider scope of research applications.

7.2. Research Contributions. Schindler and Bickart [44] researched online shopping and found that most consumers who shop online will browse online reviews, the main purpose of which is to collect consumers with product experience to provide purchase intentions. The current state of biased consumer online reviews includes extremely positive and negative reviews. Positive reviews express liking for the product and highly recommend buying it. Gretzel et al. [45] researched that 77.9% of users were most concerned about staying in the travel process when they viewed online reviews on Tripadvisor. A whopping 92.3% of users use travel websites to collect information on accommodation and alternatives, and to avoid uncomfortable hotels [46–49]. According to Tripadvisor.com, most potential hotel guests browse hotel reviews online, and consumer online reviews can influence potential purchase intentions [45]. For example, Travelindustrywire.com 84% of hotel guests are influenced by hotel reviews. Therefore, using traditional methods to understand the comments of Internet users in the field of food and travel cannot achieve good results [50].

This study attempted to resolve a problem concerning data overload during the analysis of booking site comments. A Python-based program was developed to simulate the behavior of real users when collecting comments and ratings on booking sites. The use of this program can facilitate the extraction of hidden information in such data for making management decisions. The program execution results verified that the developed program outperformed the manual extraction method in terms of the number of comments collected, the relevance of the ratings, and the time consumed. In addition, this study compiled relevant research on data mining to provide a reference for subsequent researchers interested in this topic.

7.3. Future Research Directions. Three directions for future research are proposed. First, the parameters of similar Python-based spider programs should be further optimized. Although parameter optimization was not an objective of this study, increasing the number of comments crawled under the premise of sufficient computation resources and time can enhance the overall research results. Second, the developed program can be applied to recommending customized products and information inquiry services. Finally, other booking sites can be analyzed to provide further insight into web crawling.

Data Availability

The datasets used and/or analyzed during the current study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

References

[1] H.-T. Lee and D. Leonard, "IRLbot: Scaling to 6 Billion Pages and beyond," in *Proceedings of the 17th International World Wide Web Conference*, pp. 427–439, Beijing, China, April 2008.

[2] A. Heydon and M. Najork, "Mercator: a scalable, extensible web crawler," *World Wide Web*, vol. 2, no. 4, pp. 219–229, 1999.

[3] B. Cao, J. Zhao, Z. Lv, and P. Yang, "Diversified personalized recommendation optimization based on mobile data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 4, pp. 2133–2139, 2021.

[4] B. Cao, W. Zhang, X. Wang, J. Zhao, Y. Gu, and Y. Zhang, "A memetic algorithm based on Two_Arch2 for multi-depot heterogeneous-vehicle capacitated arc routing problem," *Swarm and Evolutionary Computation*, vol. 63, Article ID 100864, 2021.

[5] Z. Lv, D. Chen, and H. Lv, "Smart city construction and management by digital twins and BIM big data in COVID-19 scenario," *ACM Transactions on Multimedia Computing, Communications, and Applications*, 2022.

[6] M. Burner, "Crawling towards eternity: building an archive of the world wide web, web techniques magazine," *Journal of Vocational Behavior*, vol. 2, no. 5, pp. 125–130, 1997.

[7] P. De Bra and R. Post, "Information retrieval in the World-Wide Web: making client-based searching feasible," *Computer Networks and ISDN Systems*, vol. 27, no. 2, pp. 183–192, 1994.

[8] H. Sheng, Y. Zhang, W. Wang et al., "High confident evaluation for smart city services," *Frontiers in Environmental Science*, 2022.

[9] H. Sheng, R. Cong, D. Yang, R. Chen, S. Wang, and Z. Cui, "UrbanLF: a comprehensive light field dataset for semantic segmentation of urban scenes," *IEEE Transactions on Circuits and Systems for Video Technology*,p. 1,2022.

[10] B. Cao, J. Zhang, X. Liu et al., "Edge-cloud resource scheduling in space-air-ground-integrated networks for internet of vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 8, pp. 5765–5772, 2022.

[11] P. R. J. Dhanith, B. Surendiran, and S. P. Raja, "A word embedding based approach for focused web crawling using the recurrent neural network," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 6, no. 6, pp. 122–132, 2021.

[12] S. Neelakandan, A. Arun, R. Ram Bhukya, B. M. Hardas, T. Ch Anil Kumar, and M. Ashok, "An automated word embedding with parameter tuned model for web crawling," *Intelligent Automation & Soft Computing*, vol. 32, no. 3, pp. 1617–1632, 2022.

[13] L. Page, S. Brin, and R. Motwani, *The Page Rank Citation Ranking: Bring Order to the Web*, Stanford University, Stanford, CA, 1998.

[14] C. C. Aggarwal, F. Ai-Garawi, and P. S. Yu, "Intelligent crawling on the world wide web with arbitrary predicates," in *Proceedings of the 10th International World Wide Web Conference*, pp. 79–82, Hong Kong, China, May 2001.

[15] F. M. J. Mehedi Shamrat, Z. Tasnim, A. K. M. Sazzadur Rahman, N. I. Nobel, and S. A. Hossain, "An effective implementation of web crawling technology to retrieve data

from the world wide web," *International Journal of Scientific & Technology Research*, vol. 9, no. 1, pp. 1251–1256, 2020.

[16] D. Yu, Z. Ma, and R. Wang, "Efficient smart grid load balancing via fog and cloud computing," *Mathematical Problems in Engineering*, vol. 2022, pp. 1–11, 2022.

[17] S. H. Hong, S. K. Lee, and J. H. Yu, "Automated management of green building material information using web crawling and ontology," *Automation in Construction*, vol. 102, pp. 230–244, 2019.

[18] Y. Jiang, G. Tong, H. Yin, and N. Xiong, "A pedestrian detection method based on genetic algorithm for optimize XGBoost training parameters," *IEEE Access*, vol. 7, Article ID 118310, 2019.

[19] K. Gao, F. Han, P. Dong, N. Xiong, and R. Du, "Connected vehicle as a mobile sensor for real time queue length at signalized intersections," *Sensors*, vol. 19, no. 9, p. 2059, 2019.

[20] W. Guo, N. Xiong, H. C. Chao, S. Hussain, and G. Chen, "Design and analysis of self-adapted task scheduling strategies in wireless sensor networks," *Sensors*, vol. 11, no. 7, pp. 6533–6554, 2011.

[21] X. Wang, Q. Li, N. Xiong, and Y. Pan, "Ant colony optimization-based location-aware routing for wireless sensor networks," in *Wireless Algorithms, Systems, and Applications. WASA 2008. Lecture Notes in Computer Science*, Y. Li, D. T. Huynh, S. K. Das, and DZ. Du, Eds., vol. 5258, Berlin, Heidelberg, Springer, 2008.

[22] R. Wan, N. Xiong, and N. T. Loc, "An energy-efficient sleep scheduling mechanism with similarity measure for wireless sensor networks," *Human-centric Computing and Information Sciences*, vol. 8, no. 1, p. 18, 2018.

[23] J. Villanueva, S. Yoo, and D. M. Hanssens, "The impact of marketing-induced versus word-of-mouth customer acquisition on customer equity growth," *Journal of Marketing Research*, vol. 45, no. 1, pp. 48–59, 2008.

[24] M. Breazeale, "Forum - word of mouse - an assessment of electronic word-of-mouth research," *International Journal of Market Research*, vol. 51, no. 3, pp. 1–19, 2009.

[25] J. Mou, P. Duan, L. Gao, X. Liu, and J. Li, "An effective hybrid collaborative algorithm for energy-efficient distributed permutation flow-shop inverse scheduling," *Future Generation Computer Systems*, vol. 128, pp. 521–537, 2022.

[26] X. Zenggang, L. Xiang, Z. Xueming et al., "A service pricing-based two-stage incentive algorithm for socially aware networks," *Journal of Signal Processing Systems*, 2022.

[27] F. Meng, Y. Zheng, S. Bao, J. Wang, and S. Yang, "Formulaic language identification model based on GCN fusing associated information," *PeerJ Computer Science*, vol. 8, p. e984, 2022.

[28] F. Meng, X. Xiao, and J. Wang, "Rating the crisis of online public opinion using a multi-level index system," *The International Arab Journal of Information Technology*, vol. 19, no. 4, pp. 597–608, 2022.

[29] J. Ward and A. Ostrom, "Motives for posting negative word of mouth communications on the internet," *Advances in Consumer Research*, vol. 29, no. 1, p. 429, 2002.

[30] V. Mayer-Schönberger and K. Cukier, *Big Data: A Revolution that Will Transform How We Live, Work, and Think*, John Murray, Greenland, 2013.

[31] W. Ozuem, C. A. Pinho, and Y. Azemi, "User-generated content and perceived customer value," in *Competitive Social Media Marketing Strategies*IGI Global, Hershey, Pennsylvania, 2016.

[32] J. Yan, H. Jiao, W. Pu, C. Shi, J. Dai, and H. Liu, "Radar sensor network resource allocation for fused target tracking: a brief review," *Information Fusion*, vol. 86-87, pp. 104–115, 2022.

[33] Y. Xi, W. Jiang, K. Wei, T. Hong, T. Cheng, and S. Gong, "Wideband RCS reduction of microstrip antenna array using coding metasurface with low Q resonators and fast optimization method," *IEEE Antennas and Wireless Propagation Letters*, vol. 21, no. 4, pp. 656–660, 2022.

[34] F. Zhang, J. Zhai, X. Shen, O. Mutlu, and X. Du, "POCLib: a high-performance framework for enabling near orthogonal processing on compression," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 2, pp. 459–475, 2022.

[35] B. L. Browning and S. R. Browning, "Improving the accuracy and efficiency of identity-by-descent detection in population data," *Genetics*, vol. 194, no. 2, pp. 459–471, 2013.

[36] S. Stockfelt, "We the minority-of-minorities: a narrative inquiry of black female academics in the United Kingdom," *British Journal of Sociology of Education*, vol. 39, no. 7, pp. 1012–1029, 2018.

[37] S. Goel, M. Bansal, A. K. Srivastava, and N. Arora, "Web Crawling-Based Search Engine Using Python," in *Proceedings of the 2019 3rd International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pp. 436–438, Coimbatore, India, June 2019.

[38] C. W. Na and B. W. On, "A proposal on a proactive crawling approach with analysis of state-of-the-art web crawling algorithms," *Journal of Internet Computing and Services*, vol. 20, no. 3, pp. 43–59, 2019.

[39] W. Zheng, L. Yin, X. Chen, Z. Ma, S. Liu, and B. Yang, "Knowledge base graph embedding module design for visual question answering model," *Pattern Recognition*, vol. 120, Article ID 108153, 2021.

[40] W. Zheng, X. Liu, X. Ni, L. Yin, and B. Yang, "Improving visual reasoning through semantic representation," *IEEE Access*, vol. 9, Article ID 91476, 2021.

[41] W. Zheng, X. Liu, and L. Yin, "Sentence representation method based on multi-layer semantic network," *Applied Sciences*, vol. 11, no. 3, p. 1316, 2021.

[42] K. Aggarwal, "An efficient focused web crawling approach," in *Software Engineering. Advances in Intelligent Systems and Computing*, M. Hoda, N. Chauhan, S. Quadri, and P. Srivastava, Eds., vol. 731, Singapore, Springer, 2019.

[43] B. V. Babu, M. S. P. Babu, and Y. C. Prasad, "An algorithm for effective web crawling mechanism of a search engine," *Oriental Journal of Computer Science and Technology*, vol. 1, no. 1, pp. 49–54, 2008.

[44] R. M. Schindler and B. Bickart, "Published word of mouth: referable, consumer-generated information on the internet," *Online Consumer Psychology: Understanding and Influencing Consumer Behavior in the Virtual World*, vol. 32, pp. 35–61, 2005.

[45] Y. A. Park, U. Gretzel, and E. Sirakaya-Turk, "Measuring web site quality for online travel agencies," *Journal of Travel & Tourism Marketing*, vol. 23, no. 1, pp. 15–30, 2007.

[46] PhoCusWright, *Technology and Independent Distribution in the European Travel Industry*, European Technology & Travel Services Association, Brussels, BE-BRU, Belgium, 2010.

[47] S. Zhao, F. Li, H. Li et al., "Smart and practical privacy-preserving data aggregation for fog-based smart grids," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 521–536, 2021.

[48] H. Zhu, M. Xue, Y. Wang, G. Yuan, and X. Li, "Fast visual tracking with siamese oriented region proposal network," *IEEE Signal Processing Letters*, vol. 29, pp. 1437–1441, 2022.

[49] J. Li, K. Xu, S. Chaudhuri, E. Yumer, H. Zhang, and L. Guibas, "GRASS: Generative recursive autoencoders for shape structures," *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 1–14, 2017.

[50] J. Yang, S. Li, J. Su, and X. Yu, "Continuous nonsingular terminal sliding Mode control for systems with mismatched disturbances," *Automatica*, vol. 49, no. 7, pp. 2287–2291, 2013.