Regular paper | Journal of the KIECS. pp. 1277-1284, vol. 18, no. 6, Dec. 31. 2023, t. 122, pISSN 1975-8170 | eISSN 2288-2189
http://dx.doi.org/10.13067/JKIECS.2023.18.6.1277

웹페이지에서의 상품 데이터 추출을 위한 동적, 정적 크롤링 비교 및 활용

김상혁* · 김정훈* · 이승대**

Comparison and Application of Dynamic and Static Crawling for Extracting Product Data from Web Pages

Sang-Hyuk Kim* · Jeong-Hoon Kim* · Seung-Dae Lee**

요 약

본 논문에서는 소비자들이 편의점에서 진행 중인 행사상품에 대해 접근하기 쉬운 웹페이지를 제작하였다. 제작하는 과정에서 행사상품의 데이터를 추출하는 두 가지 크롤링 방식인 정적 크롤링과 동적 크롤링을 비교 및 활용하였다. 정적 크롤링은 홈페이지에서 정적인 데이터를 수집하는 추출 방식이고 동적 크롤링은 웹 페이지에서 동적으로 생성되는 페이지의 데이터를 수집하는 추출하는 방식이다. 두 크롤링에 대한 비교를 통해 행사상품 데이터를 추출하는 데에 있어 어떤 크롤링 방식이 더 효과적인 방식인지에 대해 연구하였다. 그 중 효과적인 정적 크롤링을 이용해 웹 페이지를 제작하였으며, 소비자들이 더 손쉽게 확인할 수 있도록 1+1, 2+1 상품들을 카테고리화 하였고 검색기능을 넣어 웹페이지를 제작하였다.

ABSTRACT

In this paper, a web page that is easy for consumers to access event products in progress at convenience stores was created. In the production process, static crawling and dynamic crawling, two crawling methods for extracting data from event products, were compared and used. Static crawling is an extraction method of collecting static data from a homepage, and dynamic crawling is a method of collecting data from pages dynamically generated from a web page. Through the comparison of the two crawlings, we studied which crawl method is more effective in extracting event product data. Among them, a web page was created using effective static crawling, and 1+1 and 2+1 products were categorized and a search function was added to create a web page.

키워드

API, Dynamic Crawling, Flask, Python, Static Crawling
에이피아이, 동적 크롤링, 플라스크, 파이썬, 정적 크롤링

Ⅰ. 서 론

소비자들이 편의점 상품을 알아보는 방법은 다양한 방법이 있다. 직접 근처 편의점에 가서 확인을 하는 법, 매장에 전화하여 확인하는 방법 등 여러 방법이 있다. 그 중 소비자들이 조금 더 편리하고 접근하기 쉬운 방법 중 하나인 PC를 이용해 편의점의 행사상품에 대한 정보를 나타내는 웹페이지를 제작하였고,

* 남서울대학교 전자공학과(ekzm5677@naver.com, wjdgns4519@naver.com)
** 교신저자 : 남서울대학교 전자공학과
· 접 수 일 : 2023. 10. 17
· 수정완료일 : 2023. 11. 14
· 게재확정일 : 2023. 12. 27
· Received : Oct. 17, 2023, Revised : Nov. 14, 2023, Accepted : Dec. 27, 2023
· Corresponding Author : Seung-Dae Lee
Dept. of Electronic Engineering, Namseoul University.
Email : seungdae@nsu.ac.kr

웹페이지를 제작함에 있어 행사상품에 대한 정보를 추출하는 과정에서 어떠한 방식이 가장 적절한 지에 대해 연구하였다. 그 과정에서 '웹 크롤링'이라는 방식을 찾았고, 정적 크롤링과 동적 크롤링 두 가지 방식으로 나뉜다는 것을 알았고 둘 중 어떠한 방식이 더 적합한 지에 대해 연구하였다.

본 논문에서는 소비자가 CU 편의점에서 제공하는 행사상품을 편리하게 찾아볼 수 있도록 웹 크롤링 기술로 추출한 상품 데이터를 데이터베이스에 저장 후 웹 서버에 조회하여 원하는 페이지를 제작한다. 그리고 검색 기능을 구현하여 사용자가 관련 제품을 쉽게 찾을 수 있도록 하였다.

정적 크롤링은 request 모듈을 이용해 상품 리스트 API를 호출하고 Beautiful Soup 라이브러리를 사용하여 파싱한 데이터를 추출하는 방식을 사용하였다[1]. 동적 크롤링은 Selenium Module 및 Chrome Driver를 이용하였다[2]. 그리고 행사상품 페이지에 접근 후 더 보기 버튼을 클릭하여 모든 제품을 표시한 후 정적 크롤링과 마찬가지로 Beautiful Soup 라이브러리를 사용하여 상품 리스트를 추출하는 방식을 사용하여 웹페이지 구현에 있어서 어떤 크롤링 방식이 더 효과적인지 연구하였다[3].

본 논문의 구성은 다음과 같다. 2장에서는 실행 프로그램에 대한 순서도와 동적 크롤링과 정적 크롤링을 비교 분석하고 3장에서는 2장에서 도출된 결과를 바탕으로 웹페이지를 제작하였으며 제작된 웹페이지의 결과 분석을 통해 4장에서 결론을 맺었다.

II. 웹 페이지 구현 및 크롤링 비교

2.1 프로그램 기획

그림 1은 웹페이지를 구현하는 절차를 나타내며 관련 순서는 다음과 같다.

단계 1] 먼저 CU 편의점의 행사상품 데이터를 추출하기 위해 해당 페이지에서 정적, 동적 크롤링을 진행한다.

단계 2] CU 편의점 홈페이지에서 크롤링을 이용해 추출한 데이터를 데이터베이스 프로그램인 Sqlite에 저장한다.

단계 3] 데이터베이스 저장 후 상품에 대한 정보를 저장하고 있는 데이터베이스 테이블이 존재하는지 확인한다.

단계 4] 데이터베이스에 테이블이 존재하면 테이블을 이용한다.

단계 5] 데이터베이스의 테이블을 웹서버에 호출하여 웹페이지에 표시한다.

단계 4에서 데이터베이스에서 테이블이 존재하지 않는다면 테이블을 재생성하고 이용한다.

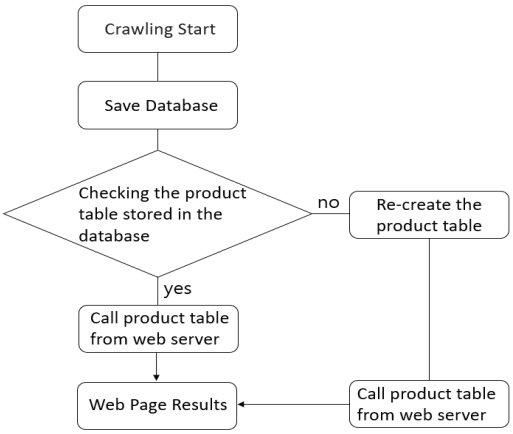

그림 1. 순서도
Fig. 1 Flowchart

2.2 웹 크롤링 및 서버 코드 구성

웹 크롤링이란 웹 사이트에서 원하는 정보를 추출하는 것을 의미한다[4-5]. 웹은 기본적으로 HTML 형태로 되어 있다[6]. 소스는 개발자가 정형화된 형태로 관리하기 때문에 일정한 규칙이 있으며, 이러한 규칙을 분석해서 원하는 정보들만 뽑아오는 것이 웹 크롤링 작업이다. 본 논문에서 사용하는 웹 크롤링 방식은 2가지인데 동적 크롤링과 정적 크롤링이다.

첫째, 동적 크롤링이란 웹 페이지에서 동적으로 생성되는 페이지의 데이터를 효과적으로 추출하는 방식이다. 본 논문에서는 웹 브라우저를 제어하여 웹페이지에 동적으로 접근할 수 있는 도구인 Selenium Module 및 Chrome Driver를 이용해 행사 상품 페이지에 접근 후 '더보기' 버튼을 클릭하여 모든 상품을 표시한 후 BeatifulSoup를 이용하여 상품 리스트를 Parsing하여 데이터를 추출하였다.

그림 3은 동적 크롤링으로 데이터를 추출하는 코드

이다. 기본적인 URL은 [CU 편의점] - [상품서비스] - [행사상품]을 사용하였다. 크롤링 속도 비교를 위해 시간을 측정할 수 있는 모듈을 작성한다.

```
def start(self):
try:
    startTime = time.time()

  print("### DYNAMIC CRAWLER START ###")

   self.cur.execute("DELETE FROM product")
   self.driver.get(HTTP_HOST)

  i = 1
  while True:
    time.sleep(10)
    page_source = self.driver.page_source
    html = BeautifulSoup(page_source, 'html.parser')
    nextBtn = len(html.select(".prodListBtn .prodListBtn-w"))
    if(nextBtn > 0):
       xpath = '//*[@id="contents"]//*[@class="prodListBtn-w"]/a'
       self.driver.find_element("xpath", xpath).click()

       print(f"Click the More button : {i}th")
       i += 1
     else:
        break
```

그림 2. 동적 크롤링 코드
Fig. 2. Dynamic crawling code

크롤링을 할 URL로 이동을 하고 While 툴을 이용해서 '더보기' 버튼을 반복적으로 클릭하는 루프를 설정하고 오류방지를 위해 6초간의 대기 후 다음 클릭이 실행되도록 설정한다. 그리고 로직이 실행되고 더 이상의 버튼이 보이지 않으면 Break 툴을 이용해 루프를 종료한다.

```
product_list = html.select("ul li.prod_list")

for item in product_list:
    image_url = item.select_one(".prod_wrap .prod_img img").attrs['src']
    product_name = item.select_one(".prod_wrap .prod_text .name p").get_text()
    product_price = item.select_one(".prod_wrap .prod_text .price strong").get_text
                    ().replace(",", "")
    category = item.select_one(".badge span").get_text()

    print(f"ID : {self.inputId}")
    print(f"product name : {product_name}")
    print(f"product price : {product_price}won")
    print(f"product image URL : {image_url}")
    print(f"product category : {category}")
    print("")

  self.cur.execute("INSERT INTO product VALUES(?, ?, ?, ?, ?, ?)", (self.inputId,
   "CU", product_name, image_url, product_price, category))

    self.inputId += 1

  endTime = time.time()

print("### DYNAMIC CRAWLER FINISH ###")
print(f"Operation time : {round(endTime - startTime, 2)}s₩n")
```

그림 3. 동적 크롤링 코드
Fig. 3 Dynamic crawling code

그림 3은 상품 정보에 대해 BeautifulSoup 라이브러리를 사용해서 HTML로 구성된 문서에서 데이터를 추출하는 코드이다. 이를 Parsing하여 상품 정보를 추출한다. 조회된 상품 정보가 있다면 각 상품의 이미지 URL, 상품명, 가격, 카테고리를 출력하고 데이터베이스에 저장한다.

	id	store	product_name	proc
	Filter	Filter	Filter	Filter
1	1	CU	46cm) Ultra-fine bristle toothbrush	//...
2	2	CU	Median) tartar care toothpaste	//...
3	3	CU	Median) Sensitive toothbrush	//...
4	4	CU	2080)toothbrush	//...
5	5	CU	2080)toothpaste	//...
6	6	CU	2080) Sensitive fine hair	//...
7	7	CU	Elastin) Perfume Shampoo	//...
8	8	CU	Perio) Total 7 toothpaste	//...
9	9	CU	Perio) Easy Clinic Toothbrush	//...
10	10	CU	Saffron) Fabric softener 1L	//...
11	11	CU	Tech) laundry detergent 1.4kg	//...
12	12	CU	Seoul) Specialty Cup Latte Mile	//...
13	13	CU	Good) Organic Long Liner 18pcs	//...
14	14	CU	Oral-B) Fresh toothbrush	//...
15	15	CU	Good) cotton wool blade 16P	//...

그림 4. 데이터베이스 Sqlite 저장
Fig. 4 Save Database Sqlite

데이터베이스에 저장할 정보는 그림 4와 같이 제품 ID, 제조사명("CU"), 제품명, 제품 이미지 URL, 제품 가격, 제품 카테고리가 있다.

데이터베이스 저장파일은 SQlite를 이용하여 데이터파일이 보다 안전하고 효과적으로 검색 및 관리할 수 있게 하였다. SQLite는 MySQL나 PostgreSQL와 같은 데이터베이스 관리 시스템이지만, 서버가 아니라 응용 프로그램에 넣어 사용하는 비교적 가벼운 데이터베이스이다[7].

둘째, 정적 크롤링이란 홈페이지에서 정적인 데이터를 수집하는데 효과적인 추출 방식이다. 정적인 데이터란 변하지 않는 데이터를 의미한다. 즉 한 페이지 안에서 원하는 정보가 모두 드러날 때 정적 데이터라고 할 수 있다.

본 논문에서는 Request 모듈을 이용해 상품 리스트 API를 호출하여 데이터를 추출하였다. CU 편의점 홈페이지 행사 상품 페이지에서 개발자 도구를 열고 Network 탭에서 호출하는 API 목록을 확인할 수 있고, 이 중 원하는 API를 찾아서 호출하는 방법으로 상품 리스트를 불러올 수 있다[8].

CU 편의점 홈페이지 행사 상품 페이지의 경우 'http://cu.bgfretail.com/event/plusAjax.do' API를 호출하여 상품 리스트를 가져오고 있다.

```
def request(self, index = 1):
response = requests.get(f"https://cu.bgfretail.com/event/plusAjax.do?pageIndex=
    {index}")

if(response.status_code == 200):
page_source = response.text

html = BeautifulSoup(page_source, 'html.parser')

if len(html.select('#nothing')) == 0:
    product_list = html.select("ul li.prod_list")

    for item in product_list:
        image_url = item.select_one(".prod_wrap .prod_img img").attrs['src']
        product_name = item.select_one(".prod_wrap .prod_text .name p").get_text()
        product_price = item.select_one(".prod_wrap .prod_text .price
                strong").get_text().replace(",", "")
        category = item.select_one(".badge span").get_text()
```

그림 5. 정적 크롤링 코드
Fig. 5 Static crawling code

그림 5는 제품 정보에 대한 HTTP를 호출하는 코드이다. Request.Get을 이용해 CU 홈페이지에 Get요청을 보내어 응답 객체 Response를 받는다. 만약 응답 코드가 200이면 성공적으로 응답을 받은 것이기 때문에 페이지 소스 코드를 가지고 온다. 가져온 페이지 소스 코드를 BeautifulSoup 라이브러리를 사용해서 파싱하여 상품 정보를 추출한다. 조회된 상품 정보가 있다면 각 상품의 이미지 URL, 상품명, 가격, 카테고리를 출력하고 동적 크롤링과 마찬가지로 데이터베이스 Sqlite에 저장한다. 데이터베이스에 저장할 정보 또한 동적과 마찬가지로 제품 ID, 제조사명("CU"), 제품명, 제품 이미지 URL, 제품 가격, 제품 카테고리가 있다. 각 제품마다 제품 ID가 증가하며, 다음 제품의 제품 ID는 이전 제품의 제품 ID보다 1 큰 값을 가진다 [9].

```
def start(self):
    try:
        startTime = time.time()
        print("### STATIC CRAWLER START ###")
        self.cur.execute("DELETE FROM product")
        index = 1
        while True:
        result = self.request(index)
        if result == False: break
        else: index += 1

        endTime = time.time()

        print("### STATIC CRAWLER FINISH ###")
        print(f"Operation time| : {round(endTime - startTime, 2)}s₩n")

        except KeyboardInterrupt:
        self.stop()
        raise
```

그림 6. 정적 크롤링 데이터 추출
Fig. 6 Extracting static crawling data

그림 6은 정적 크롤링으로 데이터를 가져오기 위한 코드이다. 우선 시작 시간과 종료 시간을 측정하여 총 작동 시간을 알기 위해 코드를 설정한다. Cur.Execute 함수를 사용하여 새로운 크롤링 작업을 시작할 때 마다 중복 데이터를 방지하고 새로운 데이터만 수집하기 위해 이전에 수집된 데이터를 비워주고 나서 크롤링을 시작할 페이지 인덱스를 1로 설정한다. While루프를 실행하여 페이지를 크롤링하고 인덱스를 1씩 더해가면서 크롤링할 페이지가 더 이상 없어지면 False를 반환하고 While루프를 빠져나가 크롤링이 종료된다.

2.3 동적 크롤링과 정적 크롤링의 비교

동적 크롤링은 크롬 드라이버를 실행하여 기존의 CU 편의점 웹페이지에서 직접 로딩 과정을 거쳐 데이터를 추출하기 때문에 전체적인 과정과 코드 전개 자체가 상대적으로 복잡하다. 또한 동적 크롤링은 웹페이지의 구조가 변경되면 크롤링 코드도 변경해야 하는 번거로움이 있다.

동적 크롤링은 크롬 드라이버를 실행하여 기존의 CU 편의점 웹페이지에서 직접 로딩 과정을 거쳐 데이터를 추출하기 때문에 전체적인 과정과 코드 전개 자체가 상대적으로 복잡하다. 또한 동적 크롤링은 웹페이지의 구조가 변경되면 크롤링 코드도 변경해야 하는 번거로움이 있다. 하지만 CU 홈페이지 구조상 상품 리스트 API를 호출할 수 있고 상품 리스트 API 사용 시. 동적 페이지에서도 정적 크롤링으로 데이터 추출이 가능하고 추출과정에서도 동적 크롤링의 단점을 보완할 수 있다.

```
ID : 1147
product name : Hbaf)Caramel Peanut Pretzel
product price : 3900won
product image URL : //tqklhszfkvzk6518638.cdn.ntruss.com/product/8809617944866.jpg
product category : 2+1

ID : 1148
product name : Hbaf) Butter Coconut Peanut
product price : 3900won
product image URL : //tqklhszfkvzk6518638.cdn.ntruss.com/product/8809617947614.jpg
product category : 2+1

ID : 1149
product name : Samyang) Real Crispy Chicken
product price : 5900won
product image URL : //tqklhszfkvzk6518638.cdn.ntruss.com/product/8801073910621.jpg
product category : 2+1

### STATIC CRAWLER FINISH ###
Operation time : 319.76s
```

그림 7. 동적 크롤링 시간 결과
Fig. 7 Dynamic crawling time results

```
ID : 1147
page Index : 29
product name : Hbaf)Caramel Peanut Pretzel
product price : 3900won
product image URL : //tqklhszfkvzk6518638.cdn.ntruss.com/product/8809617944866.jpg
product category : 2+1

ID : 1148
page Index : 29
product name : Hbaf) Butter Coconut Peanut
product price : 3900won
product image URL : //tqklhszfkvzk6518638.cdn.ntruss.com/product/8809617947614.jpg
product category : 2+1

ID : 1149
page Index : 29
product name : Samyang) Real Crispy Chicken
product price : 5900won
product image URL : //tqklhszfkvzk6518638.cdn.ntruss.com/product/8801073910621.jpg
product category : 2+1

### STATIC CRAWLER FINISH ###
Operation time : 201.66s
```

그림 8. 정적 크롤링 시간 결과
Fig. 8 Static crawling time results

웹 페이지를 제작함에 있어 필요한 데이터를 가지고 오기 위해 CU 홈페이지 행사상품 페이지에서 동/정적 크롤링을 이용하였다. 그림 8과 그림 9와 같이 추출한 데이터의 결과값(상품 리스트)에는 차이가 발생하지 않고 Time 모듈을 이용하여 시간을 측정한 결과, 동적 크롤링으로는 약 319초, 정적 크롤링으로는 약 201초로 추출 시간에서 큰 차이가 발생하였다.

필요한 데이터만 요청 가능하고 보다 더 구조화가 되어있어 데이터 추출에 있어 정확하고 효율적인 측면이 부각된다는 점과 앞서 말했듯이 정적 크롤링이 동적 크롤링에 비해 데이터 추출 속도에 있어 빠른 속도를 나타낸다는 장점을 극대화할 수 있다.

III. 웹 페이지 제작 및 구성

이 웹페이지를 구성하기 위해서는 웹 서버를 구성해야한다. 웹 서버는 파이썬 웹 프레임워크 중 하나인 Flask 모듈을 이용하여 제작을 하였다.

```
@app.route('/', methods=['GET'])
def index():
return render_template('index.html')
```

그림 9. 인덱스 코드
Fig. 9 Index code

그림 9는 Flask모듈에서 Get 요청이 들어왔을 때 'Index.Html' 파일을 렌더링해서 반환하는 함수이다. Flask에서 지원하는 함수를 이용하여 템플릿 파일을 렌더링해서 Html 코드로 변환해준다. 이를 통해 서버 측에서 동적으로 Html페이지를 생성해서 클라이언트에게 보여준다.

```
@app.route('/', methods=['GET'])
def index():
return render_template('index.html')


@app.route('/api/getProduct', methods=['GET'])
def getProduct():
input_search = request.args.get("search") if request.args.get("search") else ""
input_category = request.args.get("category") if request.args.get("category") else ""
input_event_type = request.args.get("event_type") if request.args.get("event_type")
else ""
con = sqlite3.connect(DB_PATH, isolation_level=None)
cur = con.cursor()
cur.execute("SELECT name FROM sqlite_schema WHERE type = 'table' AND name
NOT LIKE 'sqlite_%' AND name = 'product';")
result = cur.fetchall()

if len(result) == 0:
cur.execute("CREATE TABLE product ( id INTEGER NOT NULL PRIMARY KEY
AUTOINCREMENT, store VARCHAR(50) NOT NULL, product_name VARCHAR(255)
NOT NULL, product_image VARCHAR(255) NULL, price INTEGER NOT NULL, category
VARCHAR(255) NOT NULL);")

sql = f"select * FROM product where product_name LIKE '%{input_search}%' AND
category LIKE '%{input_category}%' AND category LIKE '%{input_event_type}%';"
cur.execute(sql)
result = cur.fetchall()

con.close()

return result
```

그림 10. 상품 정보를 불러오는 API 코드
Fig. 10 API code to retrieve product information

그림 10은 Flask를 사용하여 상품 정보를 불러오는 API를 구현한 코드이다. Get 방식으로 엔드 포인트를 호출하면 Request 객체를 통해 검색어와 카테고리 정보를 받아서 해당하는 상품 정보를 앞서 설명한 Sqlite 데이터베이스에 연결하여 검색하고 그 결과를 반환한다[10].

본 논문은 정적 크롤링을 이용하여 추출한 데이터를 가지고 웹사이트를 구현했기 때문에 정적 크롤링으로 추출한 데이터를 저장한 데이터베이스가 연결된다. 그 후 상품 ID, 가게 이름, 상품 이름, 상품 이미지 경로, 가격, 카테고리 정보를 저장하고 있는 데이터베이스 테이블이 존재하는 지 확인 후 존재하지 않을 경우에는 다시 생성한다. 입력받은 검색어와 카테고리를 사용하여 데이터베이스에서 해당하는 상품정보를 검색한다. 검색 쿼리문에서는 입력받은 검색어가 상품명에 포함되고 입력받은 카테고리가 해당 상품의 카테고리에 포함되는 상품 정보를 찾는다. 검색된 결과는 모두 가져와서 Result 변수에 저장된다. 이 과정

을 거친 후 검색 결과를 반환하면 정보를 화면에 표시할 수 있게 된다.

Ⅳ. 결 론

본 논문에서는 정적과 동적 크롤링의 비교를 통해 더 효과적인 크롤링 방식인 정적 크롤링을 통해 추출한 행사상품 데이터를 이용해 CU 편의점 행사상품 웹페이지를 제작하였다. 기존의 CU 편의점 행사상품 페이지는 동적 크롤링으로 추출하는 것이 더 적합한 방식이지만 크롬 드라이버를 설치함과 구조 변경에 의한 코드 변경과 같은 단점이 있었다. 하지만 상품 리스트 API가 호출 가능함에 따라 정적 크롤링으로 데이터를 추출함이 가능해지므로 동적 크롤링의 단점을 보완할 수 있다. 정적 크롤링은 API 사용 시 필요한 데이터만 요청 가능하고 더 구조화가 되어있어 데이터 추출에 있어 정확도와 효율성 향상할 수 있고 무엇보다 동적에 비해 빠른 속도로 추출할 수 있다.

본 논문에서 사용한 정적 크롤링은 웹페이지를 제작함에 있어 동적 크롤링에 비해 많은 장점을 나타내었다. 편의점 상품 데이터뿐만 아니라 추출하고 싶은 데이터가 API로 호출이 가능할 시 장점이 비교적 더 많은 정적 크롤링을 더 활용하는 것을 권장한다. 본 웹페이지는 편의점에 직접 가지 않고 본인이 원하는 행사 상품을 검색하면 그 상품이 어떤 행사를 진행중인지 알 수 있다.

향후 연구에서는 수집하려는 데이터의 형태에 따라 유동적으로 정적 크롤링과 동적 크롤링을 혼합하는 알고리즘을 적용한다면 일반 웹페이지에서 데이터를 수집 및 가공하기 위한 신뢰성이 있으면서 유연한 데이터 형태를 추출할 수 있을 것으로 판단된다.

References

[1] J. Choi and S. Kim, “Early Detection Assistance System for Rare Diseases based on Patient’s Symptom Information,” *J. of the Korea Institute of Electronic Communication Sciences*, vol. 18, no. 2, 2023, pp. 373-378.

[2] S. Yu and S. Park, “Avocado Classification and Shipping Prediction System based on Transfer Learning Model for Rational Pricing,” *J. of the Korea Institute of Electronic Communication Sciences,* vol. 18, no. 2, 2023, pp. 329-335.

[3] J. Lee, “Building an SNS Crawling System Using Python,” *J. of the Korea Industrial Information Systems Research*, vol. 23, no. 5, 2018, pp. 69-75.

[4] S. Kwon, J. Lee, and C. Lee, “A Study on the Legal Perception of Web Crawling in the Data Economy Era,” *Korean Journal of Industrial Security*, vol. 11, no. 3, 2021, pp. 73-100.

[5] J. Kim and E. Kim, “WCTT: Web Crawling System based on HTML Document Formalization,” *Korea Institute Of Information and Communication Engineering*, vol. 26, no. 4, 2022, pp. 495-502.

[6] J. Jeon and S. Lee, “Trends on Standardizations of HTML5 based Web Platform Technology,” *Electronics and Telecommunications Research Institute*, vol. 27, no. 4, 2012, pp. 83-95.

[7] H. Ko, M. Kim, S. Lee, and H. Lee, “Django based ChatBot System Using KakaoTalk API,” *The Korea internet of Things Society,* vol. 4, no. 1, 2018, pp. 31-36.

[8] Y. Chi, S. Moon, E. Shin, and H. Kim, “Study on Effective Web Services for Data Acquisition, Analysis, and Visualization,” *Jeonbuk National University Cultural Convergence Research Center Archiving,* vol. 4, no. 2, 2021, pp. 113-122.

[9] T. Wang, J. Song, D. Son, M. Kim, D. Choi, and J. Jang, “Web crawler Improvement and Dynamic process Design and Implementation for Effective Data Collection,” *J. of the Korea Institute of Information and Communication Engineering,* vol. 26, no. 11, 2022, pp. 1729-1740.

[10] K. Kim and Y. Zhao, “System Development of the Traffic Accident Prediction using Weather,” *J. of the Korea Institute of Electronic Communication Sciences,* vol. 16, no. 1, 2021, pp. 101-108.

저자 소개

김상혁(Sang-Hyuk Kim)

2017년 3월 ~ 현재 : 남서울대학교 전자공학과 4학년 재학
2024년 2월 남서울대학교 전자공학과 졸업 예정

※ 관심분야 : 전자회로, 통신시스템

김정훈(Jeong-Hoon Kim)

2017년 3월 ~ 현재 : 남서울대학교 전자공학과 4학년 재학
2024년 2월 남서울대학교 전자공학과 졸업 예정

※ 관심분야 : 전자회로, 전자기기

이승대(Seung-Dae Lee)

1990년 단국대학교 전자공학과 졸업(공학사)
1992년 단국대학 대학원 전자공학과 졸업(공학석사)
1999년 단국대학교 대학원 전자공학과 졸업(공학박사)
1995년 ~ 현재 남서울대학교 전자공학과 교수

※ 관심분야 : 유무선통신시스템, 네트워크 보안