JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지 Vol. 26, No. 11: 1729~1740, Nov. 2022

효과적인 데이터 수집을 위한 웹 크롤러 개선 및 동적 프로세스 설계 및 구현

왕태수[1] · 송재백[2] · 손다연[2] · 김민영[3] · 최동규[4] · 장종욱[5*]

Web crawler Improvement and Dynamic process Design and Implementation for Effective Data Collection

Tae-su Wang[1] · JaeBaek Song[2] · Dayeon Son[2] · Minyoung Kim[3] · Donggyu Choi[4] · Jongwook Jang[5*]

[1]Graduate Student, Department of Computer Engineering, Dong-Eui University, Busan, 47340 Korea
[2]Undergraduate Student, Department of Computer Engineering, Dong-Eui University, Busan, 47340 Korea
[3]Assistant Professor, Research Institute of ICT Fusion and Convergence, Dong-Eui University, Busan, 47340 Korea
[4]Senior Researcher, Institute of Smart IT, Dong-Eui University, Busan, 47340 Korea
[5*]Professor, Department of Computer Engineering, Dong-Eui University, Busan, 47340 Korea

요 약

근래 정보의 다양성과 활용에 따라 많은 데이터가 생성되었고, 데이터를 수집, 저장, 가공 및 예측 하는 빅데이터 분석의 중요성이 확대되었으며, 필요한 정보만을 수집할 수 있는 능력이 요구되고 있다. 웹 공간은 절반 이상이 텍스트로 이루어져 있고, 유저들의 유기적인 상호작용을 통해 수많은 데이터가 발생한다. 대표적인 텍스트 데이터 수집 방법으로 크롤링 기법이 있으나 데이터를 가져올 수 있는 방법에 치중되어 웹 서버나 관리자를 배려하지 못하는 크롤러가 많이 개발되고 있다. 본 논문에서는 크롤링 과정에서 발생할 수 있는 문제점 및 고려해야 할 주의사항에 대해 살펴보고 효율적으로 데이터를 가져올 수 있는 개선된 동적 웹 크롤러를 설계 및 구현한다. 기존 크롤러의 문제점들을 개선한 크롤러는 멀티프로세스로 설계되어 작업소요 시간이 평균적으로 4배정도 감소하였다.

ABSTRACT

Recently, a lot of data has been generated according to the diversity and utilization of information, and the importance of big data analysis to collect, store, process and predict data has increased, and the ability to collect only necessary information is required. More than half of the web space consists of text, and a lot of data is generated through the organic interaction of users. There is a crawling technique as a representative method for collecting text data, but many crawlers are being developed that do not consider web servers or administrators because they focus on methods that can obtain data. In this paper, we design and implement an improved dynamic web crawler that can efficiently fetch data by examining problems that may occur during the crawling process and precautions to be considered. The crawler, which improved the problems of the existing crawler, was designed as a multi-process, and the work time was reduced by 4 times on average.

키워드 : 웹 크롤링, 멀티프로세싱, 데이터마이닝, 빅데이터

Keywords : Web crawling, Multiprocessing, Data mining, Big data

Received 31 October 2022, **Revised** 3 November 2022, **Accepted** 8 November 2022

*** Corresponding Author** Jongwook Jang(E-mail:jwjang@deu.ac.kr, Tel:+82-51-890-1709)
Professor, Department of Computer Engineering, Dong-Eui University, Busan, 47340 Korea

Open Access http://doi.org/10.6109/jkiice.2022.26.11.1729 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

근래 정보의 다양성과 그 활용에 따라 많은 데이터가 생성되게 되었고, 이에 따라 데이터를 수집, 저장, 가공 및 예측하는 빅데이터 분석의 중요성이 확대되었다. 무수히 많은 데이터가 쏟아짐에 따라 이제는 빅데이터 속에서 필요한 정보만을 수집할 수 있어야 하고, 데이터 수집을 위한 많은 시도와 관련 기술 발전 등이 활발히 이루어지고 있다[1]. 현대 사회에서의 빅데이터에 대한 지속적인 관심과 실험적인 시도들은 다변화된 현대 사회를 더 정교하게 예측하고 효율적으로 작동하거나, 개인화된 사회 구성원들에게 적합한 정보를 제공하고 관리하려는 데에 목적을 두고 있다[2].

데이터가 많이 생성되는 곳 중 하나는 웹(Web)이며, 웹에서 발생하는 데이터들은 유저들의 거래, 검색, 업로드 등의 유기적인 웹 활동을 통해 많이 발생한다. 웹에서 발생한 많은 데이터는 원하는 형태로 가공되어 예측, 시각적 자료, 분석 등을 위한 재료로 이용된다. 이는 데이터 수집을 필연적으로 동반하게 되는데 수집 방법의 종류는 일반적 수집 데이터의 형태와 종류에 따라 로그 수집, FTP, HTTP, ETL, RDB 등의 수집 방법으로 분류할 수 있다[3].

웹 공간의 절반 이상은 텍스트로 이루어져 있고 뉴스, SNS, 사람과의 대화 또한 텍스트 데이터로 이루어져 있다. 이러한 텍스트 데이터를 쉽게 수집하기 위한 방법으로는 웹에서 HTTP를 불러와 텍스트 정보를 가져오는 크롤링 수집 기술이 있다[4]. 하지만 데이터 수집 방법에만 치중되어 웹 서버나 관리자를 배려하지 못하는 크롤러가 많이 개발되고 있다. 다른 사람의 물건을 함부로 가져오는 것이 잘못된 일인 것처럼, 손쉽게 데이터를 가져올 수 있다고 해도 다른 사람이 생성한 데이터를 가져오는 것에는 문제점과 주의해야 할 부분들이 존재한다.

본 논문에서는 웹 크롤링에 대한 전반적인 내용과 크롤링 과정에서 발생할 수 있는 문제점 및 고려해야 할 주의사항에 대해 살펴보고, 효율적으로 데이터를 가져올 수 있는 개선된 동적 웹 크롤러를 설계 및 구현한다.

Ⅱ. 관련 연구

2.1. 웹 크롤링 내용

웹 크롤링(Web Crawling)은 데이터를 수집하고 분류하는 것을 의미하며, 주로 인터넷상의 웹페이지(HTML, 문서)를 수집해서 분류하고 저장하는 것을 뜻한다. 비슷한 용어로 스크래핑(Scraping)이 있는데, 스크래핑은 웹이나 데이터베이스에서 데이터를 가져오는 것을 의미한다. 비슷한 용어로 볼 수도 있겠지만, 크롤링의 목적 링크를 따라 수많은 페이지에 도달하고 해당 메타 데이터와 콘텐츠를 분석하는 것으로 봤을 때 상이한 개념이라고 할 수 있다[2].

2.2. 크롤링 대상

크롤링의 주요 대상은 다양한 형태로 존재하는 데이터이다. 데이터는 생성 형태에 따라 정형, 반정형, 비정형데이터로 구분된다. 또한 데이터는 생산 주체에 따라 기업(또는 기관/조직)과 사용자 생성 데이터로 분류할 수 있다. 조직이 생성하는 데이터는 언론이 제공하는 뉴스, 방송영상, 민간기업이 제공하는 제품 관련 정보, 영화·음악·게임 등의 미디어 산업체가 생성하는 콘텐츠형 데이터 등이 있다. 사용자 생성 데이터는 서비스와 제품 소비로 자동으로 만들어진 데이터(뉴스 기사 읽은 횟수, 온라인 쇼핑몰 방문자 수 등)와 뉴스 댓글, 상품평, 메신저·블로그·카페·유튜브·SNS 내 활동으로 생산된 데이터를 포함한다. 이러한 데이터는 사용자 단말을 통하여 읽을 수 있는 데이터이므로, 크롤링이나 스크랩핑 방법을 사용하여 수집할 필요가 있다[2].

2.3. 크롤링 방법

크롤링하는 방법에는 크롤링 명령어를 작성하는 경우와 데이터를 수집하고자 하는 사이트 혹은 회사에서 제공하는 API(Application Programming Interface)를 사용하는 방법이 있다. API는 여러 의미를 지니지만, 크롤링 같은 경우 데이터를 쉽게 수집하기 위한 도구라고 할 수 있다. 기관에서 제공하는 API를 사용하면 빠르게 데이터를 수집할 수 있고, 데이터 사용에 따른 법적·윤리적 문제가 거의 없다. 하지만 데이터를 수집하고자 하는 사이트에서 API를 제공하지 않는 경우, API를 활용하여 원하는 데이터를 수집할 수 없는 경우에는 직접 크롤링 명령어를 작성해서 데이터를 수집해야 한다[5].

2.4. 웹페이지 종류

웹페이지는 정적 웹페이지(Static Web page)와 동적 웹페이지(Dynamic Web Page)로 나눌 수 있다.

정적 웹페이지는 웹 서버에 이미 저장된 HTML 문서를 클라이언트에게 전송하는 웹페이지이며, 사용자는 서버에 저장된 데이터가 변경되지 않는 한 고정된 웹페이지를 보게 된다. 모든 사용자는 같은 결과의 웹페이지를 서버에 요청하고 응답받게 된다[6].

동적 웹페이지는 요청 정보를 처리한 후에 제작된 HTML 문서를 클라이언트에게 전송하며, 사용자는 상황, 시간, 요청 등에 의해 달라지는 웹페이지를 보게 된다. 사용자가 보는 대부분은 동적 웹페이지다[7].

2.5. 크롤링 과정

그림 1은 웹 크롤링 과정에 대한 전반적인 흐름이다. 먼저 데이터 수집을 위한 URL들을 리스트에 저장하고, 입력받아 웹 서버에 HTML을 요청하여 전송받은 후 파싱(Parsing)을 통해 웹페이지의 데이터를 사용자가 원하는 형식으로 가공하여 DB와 같은 저장공간에 적재한다[8].

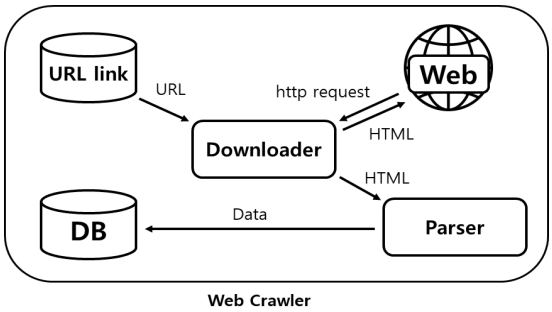

Fig. 1 Overall flow of Web Crawling process

2.6. URL 호출 방식

크롤링 과정에서 데이터 수집을 위해 웹 서버에 URL에 관한 결과를 호출하는 작업이 선행된다. 이때 사용되는 방식에는 동기(Synchronous) 방식과 비동기(Asynchronous) 방식이 있다.

그림 2는 동기, 비동기 처리방식에 대한 이미지다. 동기 방식은 웹사이트에 대한 URL 결과를 가져온 후 다음 결과를 가져오는 방식을 말한다. 즉, 하나의 웹사이트에서 랙(Lag) 발생 시 해당 웹사이트의 로딩이 끝날 때까지 다른 작업을 하지 않고 완료될 때까지 기다린다. 비동기 방식은 이전에 실행된 작업이 완료되지 않았음에도 다음 작업이 실행되는 방식을 말하며, 웹사이트 하나를 로딩하는 동안 다음 웹사이트를 불러오기 시작한다. 따라서 웹사이트마다 로딩 시작 시점은 다를 수 있어도, 하나의 로딩 시간에 상관없이 동시에 로딩하게 되므로 더 빠르게 웹사이트 결과를 얻을 수 있다. 또한, 그림 2를 통하여 작업 시작과 끝 사이의 간격을 비교해보면 비동기 처리가 더 짧고 수행 시간 또한 비동기 처리가 더 빠르다[9].

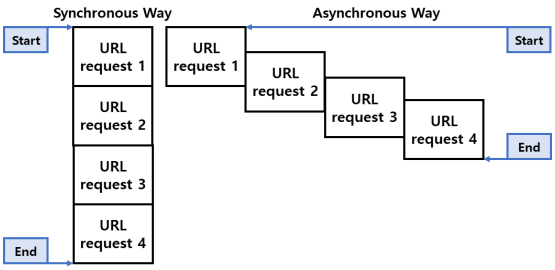

Fig. 2 Image for Synchronous, Asynchronous processing

2.7. Web Crawler Tools

웹에서 수많은 데이터가 발생하면서 데이터를 통해 활용하고자 하는 움직임이 많아지게 되었고, 웹 크롤링은 오늘날 많은 분야에 적용되고 있다.

Table. 1 Web Crawling Tools

Web Crawler Tools	
Open Search Server	Spinn3r
Import.io	GNU Wget
Webhose.io	Norconex
Dexi.io	Zyte
Apache Nutch	UiPath
VisualScraper	ParseHub
80Legs	WebSphinx
OutWit Hub	Scrapy
Mozenda	Helium Scraper
Cyotek Webcopy	-

표 1은 인터넷에 배포되어있는 다양한 기능을 가진 웹 크롤링 도구 테이블이다[10]. 웹 크롤러 도구가 인터넷에 배포되기 전, 해당 기술은 전문가가 아니면 활용하기 힘든 기술이었다. 하지만, 현재는 프로그래밍 기술이 없이도 웹 크롤링을 할 수 있는 도구들이 인터넷에 많이 제공되고 있다. 간단한 웹사이트의 구조의 경우나 데이터의 형식이 가져오기 용이한 경우 API와 도구들을 사용하여 쉽게 데이터를 가져올 수 있지만, 반대의 경우에는 알고리즘 또는 프로그래밍을 적용한 크롤러 개발이

요구된다. 프로그래밍을 통해 크롤러를 개발하면, 주로 라이브러리를 활용하게 된다. JAVA는 “Jsoup”, C#은 “HtmlAgilityPack”, javascript는 “Playwrite”, Python은 “Requests, BeautifulSoup, Scrapy, Selenium” 등 여러 라이브러리가 사용된다.

Ⅲ. 기본 크롤러 & 크롤링

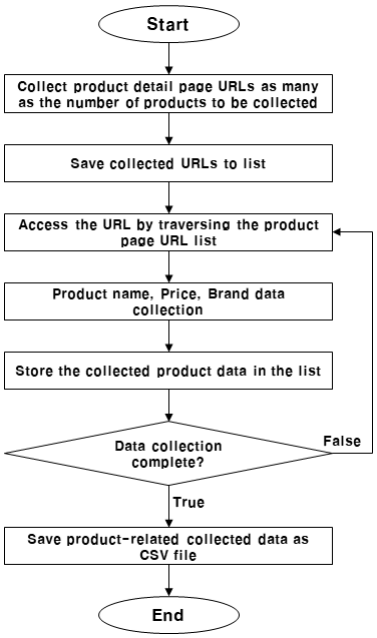

Fig. 3 Basic Crawler Flowchart

3.1. Basic Crawler

최근 웹사이트들은 정적 형태의 웹페이지가 아닌 동적 형태들로 구성된 경우가 많다. 그로 인해 웹페이지를 완전히 불러오지 못해 원하는 데이터를 가져오지 못하는 경우가 발생하였으며, 자바스크립트가 동적으로 만든 데이터를 크롤링하기 위해서는 HTML 코드와의 상호작용이 필요하다. 본 논문에서는 Python 언어로 동적 웹페이지의 HTML 코드와 상호작용할 수 있고 직접 브라우저를 제어하면서 크롤링할 수 있는 Selenium 라이브러리를 사용해 Basic Crawler를 구현했다.

그림 3은 Basic Crawler의 흐름도이다. 수집할 상품 상세 페이지 URL을 수집하여 list에 저장한 후, 해당 URL에 접속하여 상품명, 가격, 브랜드 정보를 수집 및 저장한다. 수집 목표 개수만큼 데이터가 수집되면 CSV 파일로 저장 후 종료된다.

그림 4는 Basic Crawler의 의사코드(Pseudocode)이다. 의사코드는 원하는 상품이 검색되는 전자상거래 페이지 접속, 상품 상세 페이지 URL 수집을 위한 While 반복문, 상품 관련 데이터 수집을 위한 for 반복문, 수집 데이터를 CSV 파일로 저장하는 saveToFile 코드로 구성되어 있다.

```
Access the e-commerce page where the desired product
is searched in order of popularity

req_num = get number of goods to collect
item_count = 1   #  Number of items collected
link_list = [ ]     # A set of links to the product details
page
product_list = [ ]  # A set of Product information

While item_count <= req_num :
  links = get attributes containing links

  For i in links :
    link_list.appand( About 'href' that exists in 'links' )
    item_count += 1

    If item_count < req_num and 'links' tour complete :
       Switch to next product list page

For product in link_list :
  Access product link stored in 'link_list'

  name = get Product name information from web pages
  price = get Product price information from web pages
  brand = get Product brand information from web pages

  product_list.appand( name, price, brand )

Save 'product_list' as '.csv' file
```

Fig. 4 Pseudocode of Basic Crawler

3.2. Crawling

표 2는 크롤링을 실험한 컴퓨터 환경을 나타낸다.

Table. 2 Crawling Experimental Environment Table

Experimental environment	
CPU	AMD Ryzen 5 4000
Memory	16 Gigabyte
GPU	NVIDIA GeForce GTX 1650 Ti
Internet Speed	100 Mbps

크롤링을 진행할 웹사이트는 5개의 e커머스 웹사이트로 지정하였고, 수집 데이터는 상품명, 가격, 브랜드 3개의 항목을 지정하였다. 크롤링 수집 데이터는 100, 400, 700개로 나누어 진행했다.

Table. 3 Basic Crawler Test Results Table

Web Site	URL Call Method	Library	Time Taken Crawl		
			Data 100	Data 400	Data 700
A	Synchronous	Selenium	0:10:53	0:49:49	1:05:03
B	Synchronous	Selenium	0:06:04	0:26:15	0:55:21
E	Synchronous	Selenium	0:03:31	0:12:12	0:16:05
C	Synchronous	Selenium	0:13:37	0:46:09	1:35:57
G	Synchronous	Selenium	0:02:59	0:09:16	0:15:37
N	Asynchronous	asyncio + aiohttp	X	X	X

표 3은 Basic Crawler 테스트 결과 테이블이다. 크롤링 작업 시 인터넷 속도에 따라 걸린 시간의 차이가 존재하기 때문에 3번씩 크롤링 작업을 진행하였으며, 걸린 시간의 평균값을 크롤링 작업시간으로 채택하였다. e커머스 사이트 중 'A'사와 'B'사 같은 경우 해외 e커머스 사이트이기 때문에 웹 서버에서 HTML 코드를 불러오는 속도가 느리고, 평균적으로 크롤링 작업시간이 국내 e커머스 사이트보다 길다.

비동기 방식의 처리 속도가 더 빠름에도 URL 호출 방식을 동기 방식으로 크롤링 작업을 진행하였다. 그 이유는 처음 Basic Crawler를 비동기 방식으로 URL을 호출하였을 때 크롤러로부터의 웹 서버에 많은 요청으로 인해 트래픽량이 비약적으로 증가하여 서버로부터 자주 차단되는 치명적인 문제점이 존재했기 때문이다('N'사 경우). IP 차단 횟수가 늘어날수록 차단이 풀리는 데 많은 시간이 소요되며, 비동기 방식이 속도는 가장 빠르나 대량의 데이터를 크롤링하기에는 부적합하다.

3.3. Basic Crawler의 문제점

Basic Crwler를 통해 크롤링을 진행했을 때 다음과 같은 문제점이 발생할 수 있다.

① 단일 프로세스 작업으로 인한 느린 작업속도
② Request Error가 날 때 비즈니스 로직과 함께 프로그램이 정지됨.
③ Request 처리가 길어지게 되면 다른 Request 처리가 안 되어 데이터 처리 시간이 지연됨 (Critical Path 생김)[11].

3.4. 웹 크롤링 작업 시 주의 및 고려사항

웹 크롤링은 사이트에 해로운 영향을 미치지 않도록 책임감 있게 수행해야 하는 작업이다. 웹 크롤러는 사람보다 훨씬 빠르고 깊이 있는 데이터를 검색할 수 있으므로 잘못된 스크래핑 방식은 사이트 성능에 약간의 영향을 미칠 수 있다. 대부분의 웹사이트는 크롤링 방지 메커니즘이 없을 수 있으나, 일부 사이트는 공개 액세스를 신용하지 않기 때문에 웹 크롤링이 차단될 수 있는 조치를 사용한다[12]. 다음 내용들을 통해 웹 크롤링 과정에서 주의 및 고려해야 할 점들을 숙지하여 웹 크롤링을 진행하는 것이 좋다.

- Robots.txt

웹사이트의 스크래핑 규칙은 robots.txt 파일에서 찾을 수 있으며, 파일은 웹 크롤러의 크롤링 가능 범위를 정의한 것이다. 즉, 웹사이트에서 자동 추출이 허용되지 않는 부분 또는 봇이 페이지를 요청할 수 있는 빈도를 식별한다. 많은 웹사이트에서 robots.txt 파일을 제공하고 있고, 기본 도메인 뒤에 robots.txt를 작성해 찾을 수 있다(예 : http://www.web_site.com/robots.txt).

그림 5는 robots.txt 파일 예시 이미지다. User-agent는 대상 크롤러(*:모든 크롤러), Disallow는 허용하지 않는 경로, Allow는 허용하는 경로에 관한 내용이다. 대부분의 사람들은 robots.txt에 대해 신경 쓰지 않지만, 규칙을 존중하려고 노력해야 하고 규칙을 따를 계획이 없더라도 최소한 규칙을 살펴봐야 한다[13].

```
User-agent: Googlebot
User-agent: Twitterbot
User-agent: Yeti
Disallow: *
Allow: /
Allow: /top
Disallow: /search
Allow: /v/
Allow: /l/
Allow: /channel/
Allow: /category/
Allow: /guide/
Allow: /live/

User-agent: *
Disallow: /
```

Fig. 5 Examples of robots.txt file contents

- Highly volatile HTML

웹사이트의 HTML 태그는 id 또는 class를 포함할 수 있다. HTML id는 고유한 id를 지정하지만, class는 고유하지 않다. 즉, 웹사이트 개발자나 관리자에 의해 class 이름이나 요소를 변경하면 코드가 손상되거나 잘못된 결과가 발생할 수 있다. class 대신 변경될 가능성이 낮은 id를 타겟 요소로 지정하는 것이 좋다.

- User agent Spoofing

User agent는 사용 중인 웹 브라우저를 서버에 알려주는 도구이며, 일부 웹사이트는 User agent를 제공하지 않는 한 콘텐츠를 표시하지 않는다. 웹사이트는 정품 사용자를 차단하고 싶지 않지만, 웹 브라우저에서 만들어진 모든 요청에는 User agent Header가 포함되어 있어 같은 User agent로 초당 100~200개의 요청을 보내면 봇이 감지될 수 있다. User agent를 사람처럼 보이게 하고 봇 탐지를 우회하는 유일한 방법은 User agent를 가짜로 만드는 것이다. 대부분의 웹 크롤러에는 기본적으로 User agent가 없어 직접 추가해야 한다. 차단 방지를 위해 요청 시 urllib 모듈은 기피하고 requests 모듈을 사용하여 헤더 정보를 수정해 요청하는 것이 좋다. User agent 변경 후에도 봇이 차단되는 경우 요청 헤더를 더 추가해야 하며, 웹 크롤러가 기능을 위해 Cookie에 의존하지 않는 한 쿠키를 보내지 않는 것이 좋다. 여러 브라우저에 대해 유사한 헤더 조합을 만들고 각 요청 간에 해당 헤더를 순환하여 차단될 가능성을 줄일 수 있다.

- Honeypot

허니팟(honeypot)은 비정상적인 접근을 탐지하기 위해 의도적으로 설치해 둔 시스템을 의미하며, 크롤러 또는 스크래퍼를 감지하는 수단으로 사용된다. 일부 웹사이트의 경우 일반 사용자에게는 보이지 않지만, 웹 스크래퍼 또는 스파이더 봇에 의해 추출될 수 있는 링크인 허니팟을 설치한다. 봇 감지를 위한 허니팟 링크는 CSS 스타일이 display:none을 갖거나 페이지의 배경색과 혼합되도록 위장된 색상으로 표시된다. 크롤러가 허니팟 링크를 방문하면 추가 조사를 위해 IP 주소에 플래그를 지정하거나 즉시 차단할 수 있다.

- Preventing Server Overload

웹 크롤러는 데이터를 빠르게 수집하지만, 사람은 빠르게 탐색할 수 없기 때문에 웹 서버에서 봇을 쉽게 감지할 수 있다. 즉, 크롤링 속도가 빠를수록 요청이 많아져 크롤러와 서버 모두에게 좋지 않음을 뜻한다. 서버 과부하 방지를 위해서는 사람의 행동을 모방하여 크롤러를 실제처럼 보이게 만들어야 한다. 요청 사이에 프로그래밍 방식 절전 호출을 넣고, 적은 수의 웹페이지를 크롤링한 후 시간지연을 추가하고, 가능한 낮은 동시 요청 수를 선택하며, 클릭 사이에 10~20초의 지연을 두고 웹사이트에 많은 부화를 주지 않도록 처리하는 것이 좋다.

- Check Blocking

만약 크롤링 과정에서 차단되었다면, HTTP Error Code를 통해 왜 차단되었는지 알아야 한다. 서버로부터의 차단은 잘못된 방식의 크롤링을 했다는 것을 의미하기 때문이다.

표 4는 웹 서버로부터 차단되었을 때 발생할 수 있는 HTTP Error Code들이다. 404, 403, 408과 같은 상태 코드가 자주 나타나면 차단되었음을 알 수 있으며, 해결을 위해 요청에 대한 예외 처리를 해야 한다.

Table. 4 HTTP Error Code

HTTP Error Code	Error Message
403	Forbidden: Occurs when the server rejects the request. It can also occur if the administrator blocks the user or the server does not have index.HTML.
404	Not Found: The most common error code, meaning that there are no resources to find.
408	Request Timeout: The code used when the request times out.

- IP rotation

User agent를 임의로 지정하더라도 모든 요청은 동일한 IP 주소에서 발생하기 때문에 단일 IP 주소에서 비정상적으로 많은 요청이 있는 경우 서버가 이를 감지할 수 있다. 그래서 프록시 시스템을 사용할 수 있도록 IP Pool을 만들고 여러 IP에 소수의 요청을 분산해야 한다. IP 주소를 변경할 수 있는 방법에는 토르(The Onion Router), VPN(Virtual Private Network), 공유프록시 등이 있다.

- Avoiding Infinite Loops

크롤링 진행 중 무한 루프를 발생시키는 특정 도메인에 갇혀 빠져나오지 못하는 경우가 발생할 수 있다. 크롤러 개발 시 문제점 발생 예상 지점에 대한 주기적인 모니터링이 필요하고, 문제점 발생 시 우회하거나 접근하지 않도록 해야 한다[14].

Ⅳ. 개선된 크롤러 설계 및 구현

4.1. Basic Crawler 문제점 개선 방법

표 5는 Basic Crawler의 문제점 개선을 위한 방법 테이블이다.

- Multiprocessing

멀티프로세싱은 다수의 프로세서가 서로 협력적으로 일을 처리하는 것을 의미한다. 멀티쓰레드가 아닌 멀티프로세싱을 적용하는 이유는 파이썬에서는 GIL(Global Interpreter Lock)이라는 동작으로 인해 사실상 여러 개의 쓰레드(Thread)가 동일한 자원에 대해 접근할 수 없기 때문이다. 결국, 기대한 바와 달리 하나의 쓰레드가 종료됨에 따라 다른 스레드가 진행되는 것이다. GIL 때문에 오히려 멀티 쓰레딩이 싱글 쓰레딩보다 I/O 작업량이 증가함에 따라 시간 소요가 커질 수 있다[15]. Basic Crawler의 문제점들은 단일 프로세스라서 발생하는 문제점들이기 때문에 멀티프로세싱을 사용하면 이러한 문제점들을 해결할 수 있다. Selenium은 지속적으로 렌더링을 해주기 때문에 동적 크롤링에 적합한 것이며, 그만큼 속도도 느린 편이다. 즉, 데이터 수집량 또한 낮다. 멀티프로세싱을 통한 크롤링을 진행하면 시간 대비 데이터 수집량을 높일 수 있다. 또한 단일 프로세스는 예상치 못한 오류 발생 시 프로그램이 정지되는데, 멀티프로세싱 적용 시 오류가 발생해도 해당 프로세스는 종료되지만 프로그램은 종료되지 않는다. 멀티프로세싱을 사용하면 수집된 URL들이 다수의 프로세스에 할당되어 request 처리 지연에 따른 Critical path 발생을 방지할 수 있다.

Table. 5 How to Improve the Problems of Basic Crawler

Problem		Improvement method	
Basic Crawler Problem	Slow Work	Multiprocessing	Headless, Disable-GPU, Remove image loading
	Stop the program if an error occurs		Window-size Settings
	Critical path		-
IP Blocking		User-Agent Settings, Language Settings	

- Headless

브라우저(Chrome 등)를 이용해 인터넷을 브라우징할 때 기본적으로 창이 뜨고 HTML파일을 불러오고, CSS 파일을 불러와 어떤 내용을 화면에 그려야 할지 계산을 하는 작업을 브라우저가 자동으로 진행해준다. 하지만 이와 같은 방식을 사용할 경우 사용하는 운영체제에 따라 크롬이 실행될 수도, 실행이 되지 않을 수도 있다. 예를 들어 우분투 서버와 같은 OS에서는 '화면' 자체가 존재하지 않기 때문에 일반적인 방식으로는 크롬을 사용할 수 없다. 이를 해결해 주는 방식이 바로 Headless 모드다. 브라우저 창을 실제로 운영체제의 '창'으로 띄우지 않고 대신 화면을 그려주는 작업(렌더링)을 가상으로 진행해주는 방법으로 실제 브라우저와 동일하게 동작하지만, 창은 띄우지 않는 방식으로 동작할 수 있다. Headless 기능을 사용하면 자원을 적게 잡아먹기 때문에 속도를 높일 수 있다. 다만, 드라이버가 백그라운드에서 실행돼서 브라우저가 보이지 않고, 다른 작업을 하더라도 방해가 되지 않는다는 장점이 있지만, 간혹 Headless를 하고 브라우저 접근 시 access denied나 원하는 요소를 제대로 찾지 못할 수 있다.

- Disable-GPU

브라우저들은 CPU(Central processing unit)의 부담을 줄이고 좀 더 빠른 렌더링(Rendering)을 위해 WebGL 방법으로 GPU(Graphics Processing Unit)를 통해 그래픽 가속을 사용한다. 해당 기능은 불필요한 GPU 가속 기능을 제거함으로써 CPU에 대한 의존율이 증가해 Selenium 작동 속도를 높여준다.

- Remove Image Loading

이미지 로딩을 하지 않으면 네트워크 사용량이 줄고 페이지 로딩 속도가 빨라진다. 하지만 이미지 클릭 시 문제가 발생할 수 있어 주의해야 한다.

Fig. 6 Image Loading Disabled Example Image

그림 6은 이미지 로딩을 하지 않았을 때의 예시 이미지다.

- Window-Size

크롤링할 때 특정 웹사이트들은 브라우저의 크기에 따라 HTML 값들이 변경되어 오류가 발생할 수 있다. 이때 브라우저의 크기를 특정 크기로 고정시키면 오류를 방지할 수 있다[16]. 많은 웹사이트들이 반응형으로 만들어져 크기에 따라 요소들이 다르게 나타날 수 있으니 적절히 지정하여 사용해야 한다.

- User-Agent

웹 서버는 크롤러가 접속하여 정보를 가져가는 것을 선호하지 않기 때문에 특정 정보들에 대한 확인을 통해 봇과 실제 사용자를 구분하는데, 이때 주로 확인 하는 것이 User-Agent 값이다. 해당 값은 서버로부터의 차단 방지를 위해 임의로 지정해주어야 한다. 또한, Headless 기능 사용 시 Headless 사용 여부가 서버에 기록되기 때문에 User-Agent 기능을 필수적으로 사용해야 차단을 방지할 수 있다.

- Language Settings

이 기능은 서버에서 해당 언어의 설정을 자동으로 바꿔주는 기능이다. 크롤러로 접속할 때는 언어 설정이 없고, Headless 모드에서는 언어 설정이 되어있지 않아 봇으로 추측이 가능하다. 즉, 차단의 원인이 될 수 있다.

4.2. 개선된 크롤러 설계 및 구현

Basic Crawler에서 표 6의 개선 방법들을 적용하여 개선된 크롤러를 설계 및 구현한다.

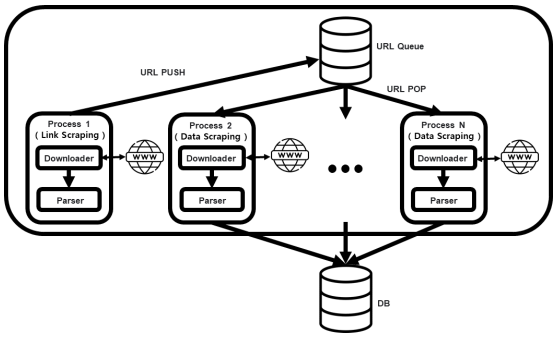

Fig. 7 Improved Crawler Configuration Diagram

그림 7은 개선된 크롤러에 대한 구성도이다. 개선된 크롤러는 기본적으로 멀티프로세스로 설계되었다. '프로세스 1'은 웹페이지 URL 링크를 수집하는 프로세스이며, '2-N 번 프로세스'들은 수집된 링크를 받아 페이지들을 순회하면서 데이터를 가져오는 프로세스다.

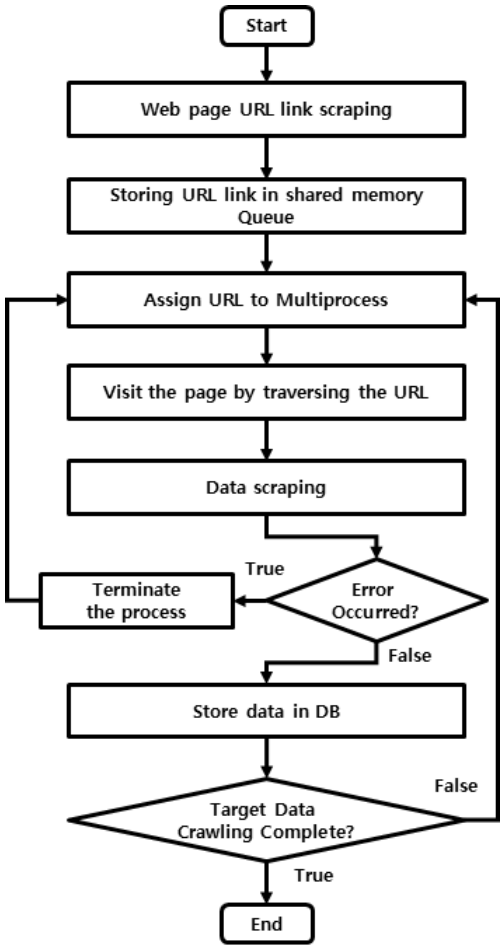

Fig. 8 Improved Crawler Flow Chart

그림 8은 개선된 크롤러에 대한 흐름도이다. 수집된 웹페이지 URL 링크들은 공유메모리 큐(Queue)에 저장 후 멀티프로세스에 할당된다. URL 링크를 전달받은 프로세스는 링크를 순회하면서 데이터 수집 후 DB에 저장한다. 오류 발생 시 해당 프로세스는 종료되며, 크롤링이 완료되면 프로그램이 종료된다.

표 6은 멀티프로세싱으로 구현해놓은 개선된 크롤러의 의사 코드다.

Table. 6 Improved Crawler Pseudocode

```
# Multiprocessing

Initialize 'process list'
Create 'save url process'
Add 'save url process' to 'process list'
Start 'save url process'

For _ in range(maximum count of processes)
  create 'get data process'
  add 'get data process' to 'process list'
  start 'get data process'

For 'process' in 'process list'
  join 'process'

# Save url process
  initialize 'url Queue'
  for _ in range(maximum count of page)
    go to url
    get elements with an href attribute
    for element in elements
      push 'element with an href attribute' for 'url Queue'

# Get data process
  while 'url Queue' not empty
    get url for 'url Queue'
      go to url
      get data
      save data for database
```

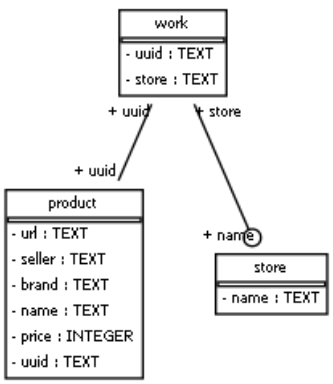

Fig. 9 Improved Crawler Database ERD

그림 9는 수집된 데이터를 저장하기 위한 데이터베이스의 ERD(Entity Relationship Diagram)이다. DB 테이블은 work, store, product 3개로 구성되어 있다.

store 테이블은 크롤링 사이트를 구별하기 위한 테이블이며, name 컬럼(Column)은 해당 테이블의 PK(Primary Key : 기본키)로, work 테이블에서 FK(Foreign Key : 참조키)로 사용된다.

work 테이블은 크롤링 결과물을 작업별로 구별하기 위한 것이다. 같은 store의 크롤링 작업이어도 데이터 수집 개수를 똑같이 설정했을 때 작업을 구별할 필요가 있다. 그래서 uuid 컬럼을 PK로, store 컬럼을 FK로 사용한다. UUID(Universally Unique IDentifier)는 네트워크 상에서 고유성이 보장되는 id를 만들기 위한 표준규약으로, 128bit 숫자이며, 32자리의 16진수로 표현된다. 이는 8-4-4-4-12자리 패턴으로 붙임표(-)를 집어넣어 5개의 그룹으로 구분한다[17]. 본 uuid 컬럼에는 4버전의 난수값(무작위 UUID 생성)으로 생성된 레코드가 저장된다.

product 테이블은 상품 데이터를 저장하기 위한 테이블이다. url(상품 링크), seller(판매자), brand(상품 브랜드), name(상품명), price(가격) 컬럼은 데이터 수집을 위한 것이며, 작업에 따라 같은 데이터를 저장할 수 있으므로 작업별로 제품을 구별하기 위해 uuid 컬럼을 FK로 사용한다. 최종적으로 product 테이블에서 url과 uuid를 묶어서 PK로 사용한다.

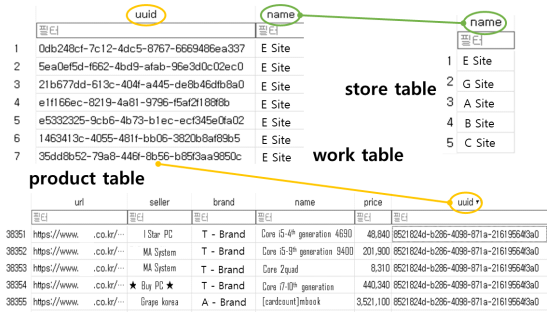

Fig. 10 DB Table Example Image

그림 10은 DB 테이블의 예시 이미지다. uuid 컬럼의 고유한 레코드들을 통해 크롤링 결과를 작업별로 구별하여 상품에 대한 정보들을 검색할 수 있다.

Ⅴ. 개선된 크롤러 성능

5.1. Core 개수에 따른 성능

개선된 크롤러의 성능 확인을 위해 프로세스 개수를 20개로 설정하여 크롤링 작업을 진행하였고 그에 따른 높은 성능을 기대했지만, 기대치만큼의 크롤링 속도가 나오지는 못하였다. 개선된 크롤러는 멀티프로세스 구조를 가지고 있으므로 성능 검증을 위해 실험환경에서의 최적의 프로세스 개수인 Best 프로세스를 찾고자 하였다. 첫 번째 실험에서는 Best 프로세스를 찾기 전, 높은 성능의 CPU 코어를 파악하고자 하였다.

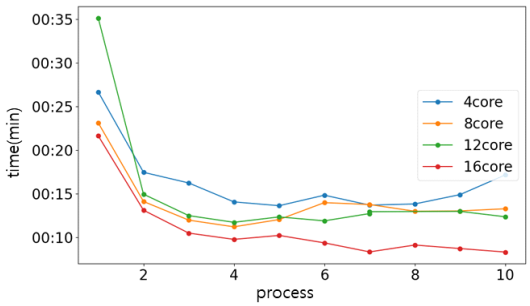

Fig. 11 Crawling Time Graph according to the Number of processes per core

그림 11은 'E'사 사이트를 기준으로 Core 별 프로세스 개수에 따른 크롤링 소요 시간 그래프이다. 그래프를 통해 전반적으로 16 Core의 성능이 높은 것을 알 수 있다.

5.2. Process 개수 별 성능

두 번째 실험에서는 프로세스 개수 별 성능 비교를 통해 Best 프로세스를 찾고자 하였다.

그림 12는 프로세스 개수 별 크롤링 소요 시간 그래프이다. 사이트는 'E'사, 크롤링 데이터는 700개, CPU 16 Core로 진행하였다. 그래프 추이를 살펴보면 프로세스가 증가할수록 크롤링 소요 시간이 감소하지만, 특정 프로세스 개수 이상부터는 비슷한 성능을 보인다. 또한, 프로세스 개수가 10 이상일 경우 처리시간이 증가하는데, 그 이유는 Context Switching으로 인해 여러 프로세스 간에 캐시 공유가 발생하여 성능이 저하되기 때문이다[18]. 이를 통해 프로세스 개수의 증가가 성능개선에 항상 비례하지 않으며, 크롤링 작업 환경에 맞는 최적의 프로세스 개수를 찾는 것이 자원 낭비를 방지할 수 있음을 알 수 있다[19]. 본 실험에서의 Best 프로세스 개수는 7개이다.

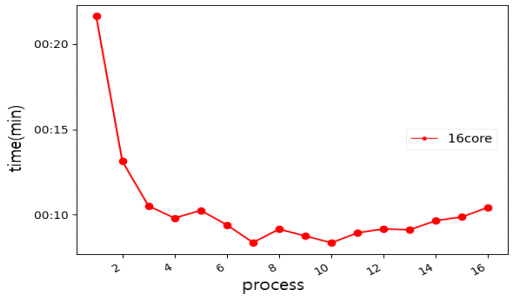

Fig. 12 Crawling Time Graph by Process Count

5.3. Basic Crawler & 개선된 크롤러 성능 비교

Basic Crawler의 문제점들을 개선한 크롤러의 성능을 확인하기 위해 16 Core, 프로세스 개수 7로 설정하여 수집 데이터 100, 400, 700개의 크롤링 작업 결과를 측정하였다.

Table. 7 Basic Crawler and Improved Crawler operation time table

Web Site	Basic Crawler Time Taken to Crawl			Improved Crawler Time Taken to Crawl		
	Data 100	Data 400	Data 700	Data 100	Data 400	Data 700
A	0:10:53	0:49:49	1:05:03	0:02:03	0:06:36	0:16:16
B	0:06:04	0:26:15	0:55:21	0:01:41	0:03:22	0:06:42
E	0:03:31	0:12:12	0:16:05	0:01:44	0:04:52	0:07:42
C	0:13:37	0:46:09	1:35:57	0:03:46	0:10:05	0:18:40
G	0:02:59	0:09:16	0:15:37	0:00:49	0:01:57	0:03:01

표 7은 대표적인 e커머스들의 Basic Crawler와 개선된 크롤러의 크롤링 소요 시간 테이블이다. 단일 프로세스인 Basic Crawler의 작업시간보다 멀티프로세스로 설계된 개선된 크롤러의 작업시간이 전체적으로 월등히 적게 소요됨을 알 수 있다.

Ⅵ. 결 론

본 논문에서는 웹 크롤링 과정에서의 발생 가능한 문제점과 설계 시 고려해야 할 주의사항에 대해 살펴보고, 문제점들을 해결할 수 있는 방법들을 적용한 개선된 동

적 웹 크롤러를 설계 및 구현했다. 기본적으로 단일 프로세스로 설계된 Basic Crawler의 경우 대표적으로 비효율적인 3가지 문제점들이 존재하였다. 문제점은 각각 ① 단일 프로세스 크롤링 작업으로 인한 느린 작업속도, ② 기본적인 예외 처리가 되지 않았을 경우, 한 번의 request error가 날 때 다음 request 및 비즈니스 로직과 함께 프로그램 정지됨, ③ 하나의 request 처리가 길어지게 되면 다른 request 처리가 안 되기 때문에 데이터 처리 시간이 지연되는 Critical Path 발생이 있다. 이 3가지 문제점들을 개선할 수 있는 멀티프로세싱 기법을 기반으로 각각의 문제점들을 개선할 수 있는 여러 옵션을 추가하여 개선된 동적 웹 크롤러를 설계 및 구현하였다. 크롤링 작업은 5개의 대표적인 국내외 e커머스 웹사이트들을 대상으로 비정형데이터 중 하나인 상품에 대한 텍스트 데이터 100, 400, 700개를 수집하였다. 개선된 크롤러의 성능 측정은 CPU 코어 개수에 따른 크롤링 성능 측정 실험과 프로세스 개수 별 크롤링 성능 측정 실험으로 진행되었다. 실험을 통해 얻은 Best 프로세스 개수는 7개이며, 실험 결과를 통해 프로세스 개수의 증가는 성능개선에 항상 비례하지 않기 때문에 자원 낭비 방지를 위해 크롤링 작업 환경에 맞는 최적의 프로세스 개수를 찾아야 함을 알 수 있다. Basic Crawler와 개선된 크롤러의 성능 비교 결과, 개선된 크롤러의 작업시간이 평균적으로 4배 정도 감소하였다.

ACKNOWLEDGEMENT

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the Grand Information Technology Research Center support program(IITP-2022-2016-0-00318) supervised by the IITP(Institute for Information & communications Technology Planning & Evaluation).

Also, this research was supported by the BB21plus funded by Busan Metropolitan City and Busan Institute for Talent & Lifelong Education (BIT).

Finally Thank you Ms. Bomin Kim(20180020 @office.deu.ac.kr) for completing the grammar inspection of this paper.

REFERENCES

[1] J. H. Kim and E. G. Kim, "WCTT: Web Crawling System based on HTML Document Formalization," *Journal of the Korea Institute of Information and Communication Engineering*, vol. 26, no. 4, pp. 495-502, Apr. 2022.

[2] Samsung Display Newsroom. Collect only the information you want! Leverage crawling and big data analytics [Internet]. Available: https://news.samsungdisplay.com/22907.

[3] DATA ON-AIR. Data collection methods and techniques [Internet]. Available: https://dataonair.or.kr/db-tech-reference/d-guide/data-practical/?mod=document&uid=378.

[4] Y. -R. Suh, K. P. Koh, and J. Lee, “An analysis of the change in media’s reports and attitudes about face masks during the COVID-19 pandemic in South Korea: a study using Big Data latent dirichlet allocation (LDA) topic modelling,” *Journal of the Korea Institute of Information and Communication Engineering*, vol. 25, no. 5, pp. 731 – 740, May 2021.

[5] C. Y. An, S. W. Moon, E. H. Shin, and H. Kim, "Study on Effective Web Services for Data Acquisition, Analysis, and Visualization," *Journal of D-Culture Archives (JDCA)*, vol. 4, no. 2, pp. 113-122, Oct. 2021.

[6] J. S. Han, J. S. Kim, I. B. Kim, and H. I. Lee, "Building Personal Blogs with Static Site Generators," in *Proceeding of the Korea Contents Association Comprehensive Conference*, Daejeon, Korea, pp. 475-476, 2021.

[7] J. S. Yoo, S. Y. Heo, and S. W. Park, "Forgery detection system of dynamic web page using snapshot," in *Proceeding of the Korean Institute of Information Scientists and Engineers*, Pyeongchang, Korea, pp. 1612-1614, 2019.

[8] KDB VELOG. Web Data Crawling [Internet]. Available: https://velog.io/@kimdukbae/.

[9] Cosmos Project. Python Basic: Python coroutine, coroutine [Internet]. Available: https://cosmosproject.tistory.com/474.

[10] 101 Help. 25 Best Free Web Crawler Tools [Internet]. Available: https://ko.101-help.com/25gaji-coegoyi-muryo-web-keurolreo-dogu-baa8db87e8/.

[11] Exmemory Tistory. A productive web crawler structure for collecting large amounts of data [Internet]. Available: https://exmemory.tistory.com/.

[12] ScrapeHero. How to Scrape Websites Without Getting Blocked [Internet]. Available: https://www.scrapehero.com/how-to-prevent-getting-blacklisted-while-scraping/.

[13] HACKERNOON. Web Scraping Tutorial with Python: Tips and Tricks [Internet]. Available: https://hackernoon.com/web-scraping-tutorial-with-python-tips-and-tricks-db070e70e071.

[14] 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, Jun. 2019.
[15] Tigercow Door Tistory. Multi-processing and Multi-threading Source [Internet]. Available: https://doorbw.tistory.com/205.
[16] Yeko90 Tistory. Learn-headless-and-multiple-function-with python-basic-selenium-addargument [Internet]. Available: https://yeko90.tistory.com.
[17] Apache Commons. A Universally Unique Identifier (UUID) [Internet]. Available: https://commons.apache.org/sandbox/commons-id/uuid.html.
[18] C. Li, C. Ding, and K. Shen, “Quantifying The Cost of Context Switch,” in *Proceedings of the 2007 workshop on Experimental computer science*, San Diego: CA, USA, pp. 218, 2007.
[19] T. -S. Hur, J. -H. Kim, and S. -H. Baek, "Recruitment collector using multiple processes based on Python," in *Proceedings of the Korean Society of Computer Information Conference*, Jeju, Korea, pp. 229-230, 2019.

왕태수(Tae-su Wang)

동의대학교 e-비즈니스학과 경영학사
現 동의대학교 컴퓨터공학과 석사과정
※관심분야 : 영상처리, 데이터분석, 딥러닝, 데이터마이닝

송재백(JaeBaek Song)

現 동의대학교 컴퓨터공학과 학사과정
※관심분야 : 백엔드, 딥러닝, 데이터분석

손다연(Dayeon Son)

現 동의대학교 컴퓨터공학과 학사과정
※관심분야 : 이미지처리, 백엔드, 데이터분석

김민영(Minyoung Kim)

동의대학교 컴퓨터공학과 공학박사
現, 동의대학교 ICT융복합연구소 조교수
※관심분야 : 유무선통신시스템, IoT, 자동차네트워크, 데이터통신

최동규(Donggyu Choi)

동의대학교 소프트웨어융합학과 공학석사
現, 동의대학교 스마트IT연구소 선임연구원
※관심분야 : 딥러닝, 헬스케어, 의료 데이터분석, 영상처리

장종욱(Jongwook Jang)

부산대학교 컴퓨터공학과 공학박사
한국전자통신연구소 연구원
現, 동의대학교 컴퓨터공학과 교수
※관심분야 : 유무선통신시스템, 자동차네트워크, 블록체인, 딥러닝