Selenium을 활용한 베스트셀러 시각화

2021204045 이성민

본 리포트는 Selenium을 활용해 주요 온라인 서점 사이트(알라딘, 영풍문고, 교보문고, yes24)의 베스트셀러 목록에 자동으로 접근하여, 책과 작가 정보를 수집하고 이를 정규화하여 시각화하는 과정을 설명한다.

1. Selenium을 이용한 데이터 수집

```
service =Service('C:/Program Files/Google/Chrome/chromedriver-win64/chromedriver.exe')  # chromedriver의 경로 설정
driver =webdriver.Chrome(service=service, options=options)
```

웹드라이버 설정: webdriver.ChromeOptions() 및 webdriver.Chrome()을 사용해 웹드라이버를 설정하고, chromedriver의 위치를 지정한다. 코드 상의 디렉토리는 필자의 chromedriver 위치를 가리키므로, **사용자는 자신의 chromedriver 위치로 수정해야 한다.**

```
url_aladin ="https://www.aladin.co.kr/shop/common/wbest.aspx?BranchType=1&start=we"# 알라딘 베스트셀러 사이트
url_ypbooks ="https://www.ypbooks.co.kr/bestseller/week"# 영풍문고 베스트셀러 사이트
url_kyobobooks ="https://store.kyobobook.co.kr/bestseller/total/weekly"# 교보문고 베스트셀러 사이트
url_yes24 ="https://www.yes24.com/Product/Category/BestSeller?pageNumber=1&pageSize=24&categoryNumber=001"# yes24 베스트셀러 사이트

aladin_books = [] # 알라딘 책 목록
ypbooks_books = [] # 영풍문고 책 목록
kyobobooks_books = [] # 교보문고 책 목록
yes24_books = [] # yes24 책 목록

aladin_authors = [] # 알라딘 책의 작가 목록
ypbooks_authors = [] # 영풍문고 책의 작가 목록
kyobobooks_authors = [] # 교보문고 책의 작가 목록
yes24_authors = [] # yes24 책의 작가 목록


get_driver(url_aladin, aladin_books, aladin_authors)
get_driver(url_ypbooks, ypbooks_books, ypbooks_authors)
get_driver(url_kyobobooks, kyobobooks_books, kyobobooks_authors)
get_driver(url_yes24, yes24_books, yes24_authors)
```

사이트 URL 지정: 각 서점의 주간 베스트셀러 페이지 URL을 지정해 놓았다. 책 제목과 작가 정보를 담을 배열(books, authors)을 정의하여 각 순위의 정보를 순서대로 저장한다.

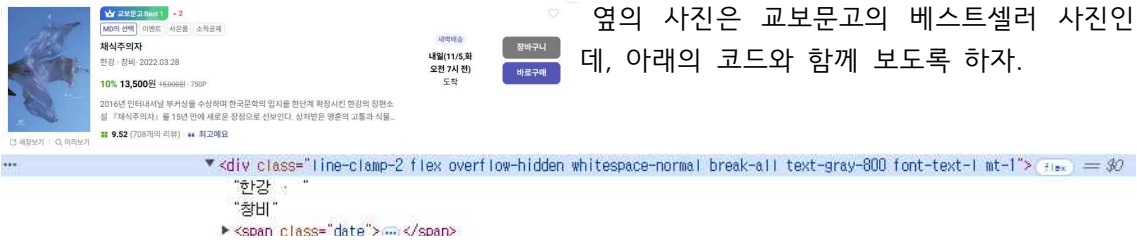
```
def get_driver(url, books, authors):
    driver.get(url) # 해당 url에 접속
    try:
        if url ==url_aladin:
            element =WebDriverWait(driver, 20).until(
                                                                    EC.presence_of_element_located((By.XPATH,
'//*[@id="Myform"]/div[3]/table/tbody/tr/td[3]/table/tbody/tr[1]/td[1]/div[1]/ul/li[2]/a[1]'))
            )
        elif url ==url_ypbooks:
            element =WebDriverWait(driver, 20).until(
                                                                    EC.presence_of_element_located((By.XPATH,
'//*[@id="content"]/div[2]/div[2]/section/div/ul/li[1]/div/div[1]/div/div[2]/a'))
            )
        elif url ==url_kyobobooks:
            element =WebDriverWait(driver, 20).until(
                                                                    EC.presence_of_element_located((By.XPATH,
'/html/body/div[1]/main/section/div/div/section/ol[1]/li[1]/div/div[2]/div[2]/a'))
            )
        elif url ==url_yes24:
```

```
            element =WebDriverWait(driver, 20).until(
                EC.presence_of_element_located((By.XPATH, '//*[@id="yesBestList"]/li[1]/div/div[2]/div[2]/a[1]'))  # 각
1위 베스트셀러가 로딩될 때까지 대기
            )
    except TimeoutException:
        print("오류: 지정된 XPath의 요소가 제한된 시간 안에 로딩되지 않았습니다.")
        sys.exit() # 20초 안에 로딩이 안되면 오류와 함께 프로그램 종료...
```

해당 단계는 get_driver() 함수를 사용해 각 사이트에 자동으로 접속한 뒤, 로딩이 끝날 때까지 기다리는 과정이다. 이를 위해 각 사이트의 베스트셀러 1위 책 제목의 XPath를 사용하여 로딩 완료를 확인한다. 로딩이 20초 내에 완료되지 않으면 오류 메시지를 출력하고 프로그램이 종료된다. (참고로, **로딩 시간이 부족할 경우, 20을 더 큰 숫자로 변경해 시도**할 수 있다.)

```
    try:
        if url ==url_aladin:
            a =3
        else:
            a =1     # 알라딘만 1위가 Xpath 기준 3부터 시작하고, 나머지는 1부터 시작

        for i in range(a, a +num):
            if url ==url_aladin:
                                                                book_name        =driver.find_element(By.XPATH,
f'//*[@id="Myform"]/div[{i}]/table/tbody/tr/td[3]/table/tbody/tr[1]/td[1]/div[1]/ul/li[2]/a[1]').text
            elif url ==url_ypbooks:
                                                                book_name        =driver.find_element(By.XPATH,
f'//*[@id="content"]/div[2]/div[2]/section/div/ul/li[{i}]/div/div[1]/div/div[2]/a').text
            elif url ==url_kyobobooks:
                                                                book_name        =driver.find_element(By.XPATH,
f'/html/body/div[1]/main/section/div/div/section/ol[1]/li[{i}]/div/div[2]/div[2]/a').text
            elif url ==url_yes24:
                book_name =driver.find_element(By.XPATH, f'//*[@id="yesBestList"]/li[{i}]/div/div[2]/div[2]/a[1]').text
            books.append(book_name)   # 각 베스트셀러의 책 제목란의 Xpath를 인식해 books에 순위 순서대로 추가

        for i in range(a, a +num):
            if url ==url_aladin:
                                                              author_name        =driver.find_element(By.XPATH,
f'//*[@id="Myform"]/div[{i}]/table/tbody/tr/td[3]/table/tbody/tr[1]/td[1]/div[1]/ul/li[3]/a[1]').text
            elif url ==url_ypbooks:
                                                              author_name        =driver.find_element(By.XPATH,
f'//*[@id="content"]/div[2]/div[2]/section/div/ul/li[{i}]/div/div[1]/div/div[2]/ul[3]/li[1]/span').text
            elif url ==url_kyobobooks:
                                                              author_info        =driver.find_element(By.XPATH,
f'/html/body/div[1]/main/section/div/div/section/ol[1]/li[{i}]/div/div[2]/div[2]/div[2]').text
                author_name =author_info.split('·')[0].strip()  # 저자 이름만 추출.
            elif url ==url_yes24:
                                                              author_name        =driver.find_element(By.XPATH,
f'//*[@id="yesBestList"]/li[{i}]/div/div[2]/div[3]/span[1]/a').text
            authors.append(author_name)    # 각 베스트셀러의 책 작가란의 Xpath를 인식해 books에 순위 순서대로 추가
```

해당 단계는 해당 사이트의 Xpath를 따라 books와 authors 배열에 순위의 순서대로 내용을 넣는 과정이다. 대부분 사이트(영풍문고, 교보문고, yes24)는 1위의 Xpath가 li[1]부터 시작하지만 알라딘 사이트만 div[3]부터 시작하기 때문에 예외 처리하였다. 여기서 교보문고의 작가란에만 줄이 하나 추가된 것을 볼 수 있는데, 이는 해당 사이트의 Xpath 구조 때문이다.



옆의 사진은 교보문고의 베스트셀러 사진인데, 아래의 코드와 함께 보도록 하자.

/html/body/div[1]/main/section/div/div/section/ol[1]/li[2]/div/div[2]/div[2]/div[2]/text()[1]
/html/body/div[1]/main/section/div/div/section/ol[1]/li[2]/div/div[2]/div[2]/div[2]/text()[2]
/html/body/div[1]/main/section/div/div/section/ol[1]/li[2]/div/div[2]/div[2]/div[2]/span

각 줄의 Xpath 목록이다. 마지막을 보면 text()로 끝나는데, text()를 통해 텍스트 노드를 직접 접근하는 것은 Selenium에서 지원하지 않는다. 요소(element) 단위로만 XPath를 통해 접근하기 때문에, text() 직전인 div[2]까지 접근할 수 있다. 그러나 이렇게 코드를 짜게 되면....

```
================================== 교보문고 ==================================
1위: 채식주의자 - 한강 · 창비
· 2022.03.28
2위: 소년이 온다 - 한강 · 창비
· 2022.03.28
3위: 작별하지 않는다 - 한강 · 창비
· 2022.03.28
4위: 흰 - 한강 · 창비
· 2022.03.28
5위: 서랍에 저녁을 넣어 두었다 - 한강 · 창비
· 2022.03.28
6위: 희랍어 시간 - 한강 · 창비
· 2022.03.28
7위: 디 에센셜: 한강(무선 보급판) - 한강 · 창비
· 2022.03.28
8위: 트렌드 코리아 2025 - 한강 · 창비
· 2022.03.28
9위: 어른의 행복은 조용하다 - 한강 · 창비
· 2022.03.28
10위: 유랑하는 자본주의자 - 한강 · 창비
· 2022.03.28
```

이렇게 원하지 않는 출판사와 출판 날짜 정보까지 같이 출력된다.

```
            elif url ==url_kyobobooks:
                                                    author_info     =driver.find_element(By.XPATH,
f'/html/body/div[1]/main/section/div/div/section/ol[1]/li[{i}]/div/div[2]/div[2]/div[2]').text
                author_name =author_info.split('·')[0].strip()  # 저자 이름만 추출.
```

그래서 '·'(작가와 출판사를 구분하는 기호)를 기준으로 앞에 있는 단어만 추출하면 작가의 이름만을 추출할 수 있다.

```
        if url ==url_aladin:
                                                 print("\n====================================     알라딘
====================================")
        elif url ==url_ypbooks:
                                                print("\n====================================     영풍문고
====================================")
        elif url ==url_kyobobooks:
                                                print("\n====================================     교보문고
====================================")
        elif url ==url_yes24:
                                                 print("\n====================================     yes24
====================================")

        for i in range(0, len(books)):
            print(f"{i+1}위: {books[i]} - {authors[i]}")
```

위의 작업을 수행하면 이러한 출력본을 얻을 수 있다. 원하는 대로 순위부터 책 제목, 작가 이름까지 정상적으로 출력된 모습을 볼 수 있다.

```
================================== 알라딘 ==================================
1위: 소년이 온다 - 한강
2위: 작별하지 않는다 - 한강
3위: 채식주의자 (리마스터판) - 한강
4위: 나의 히어로 아카데미아 41 (트리플 특장판) - 호리코시 코헤이
5위: 흰 - 한강
6위: 팬텀 버스터즈 1 - 네오쇼코
7위: 김대리의 취향 니트 - 바늘이야기 김대리
8위: 서랍에 저녁을 넣어 두었다 - 한강
9위: 희랍어 시간 - 한강
10위: 모우어 - 천선란

================================== 영풍문고 ==================================
1위: [국내도서]채식주의자(개정판) - 한강
2위: [국내도서]소년이 온다 - 한강
3위: [국내도서]작별하지 않는다 - 한강
4위: [국내도서]서랍에 저녁을 넣어 두었다 (한강-문학과지성시인선438) - 한강
5위: [국내도서]흰(개정판) - 한강
6위: [국내도서]트렌드 코리아 2025 - 2025 대한민국 소비트렌드 전망 - 김난도,전미영,최지혜,권정윤,한다혜,이혜원,이준영,이향은,추예린,전다현
7위: [국내도서]디 에센셜 한강 (무선 보급판) - 한강
8위: [국내도서]급류(오늘의 젊은 작가40) - 정대건
9위: [국내도서]희랍어 시간 - 한강
10위: [국내도서]모순 - 양귀자

================================== 교보문고 ==================================
1위: 채식주의자 - 한강
2위: 소년이 온다 - 한강
3위: 작별하지 않는다 - 한강
4위: 흰 - 한강
5위: 서랍에 저녁을 넣어 두었다 - 한강
6위: 희랍어 시간 - 한강
7위: 디 에센셜: 한강(무선 보급판) - 한강
8위: 트렌드 코리아 2025 - 김난도 외
9위: 어른의 행복은 조용하다 - 태수
10위: 유랑하는 자본주의자 - 유랑쓰 임현주

================================== yes24 ==================================
1위: 소년이 온다 - 한강
2위: 채식주의자 - 한강
3위: 작별하지 않는다 - 한강
4위: 서랍에 저녁을 넣어 두었다 - 한강
5위: 흰 - 한강
6위: 희랍어 시간 - 한강
7위: 트렌드 코리아 2025 - 김난도
8위: 나의 히어로 아카데미아 41 트리플 특장판 - 호리코시 코헤이
9위: 하루 한 장 나의 어휘력을 위한 필사 노트 - 유선경
10위: 넥서스 - 유발 하라리
```

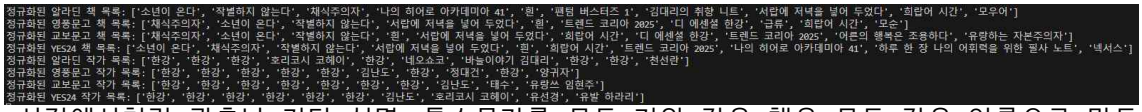
2. 정규화

우리는 지금까지 총 4개의 도서 사이트에서 베스트셀러의 책 제목과 작가 이름을 자동으로 추출하고, 배열에 넣어 출력하는 작업을 수행하였다. 그러나 최종 목표인 시각화에 데이터를 이용하기 위해선 아직 부족하다. 분명 같은 책인데도 불구하고 도서 사이트에 따라 이름이 조금씩 달라 다른 책으로 인식하기 때문이다. 작가 이름 또한 문제이다. 작가 이름이 여러 명인 책의 경우 영풍문고는 모든 작가의 이름을 적었고, 교보문고에선 대표 작가 한 명과, ' 외' 라는 단어로 다른 작가 이름을 대체하였고, yes24에선 대표 작가 한 명의 이름만 기입해놓았다. 따라서 우리는 같은 책, 작가이지만 다른 제목을 같게 만드는 정규화 작업을 진행할 것이다.

```python
# 책 제목 정규화 함수
def normalize_title(title):
    # 불필요한 부분 제거: [], (), 특정 단어 등
    title =re.sub(r"\[.*?\]|\(.*?\)", "", title)  # 대괄호와 소괄호 내용을 제거
    title =re.sub(r"리마스터판|개정판|트리플 특장판|무선 보급판| - .*", "", title)  # 기타 설명 제거
    title =title.replace(":", "") # : 제거
    title =title.strip()  # 앞뒤 공백 제거
    return title

# 작가 정규화 함수
def normalize_authors(author):
    if " 외"in author:
        author =author.replace(" 외", "") # " 외"를 지움
    return author.split(",")[0]  # ','가 나오면 앞쪽 글자만 사용
```

먼저 책 제목 정규화 함수에선 [국내도서], (개정판)과 같이 괄호 안에 있는 내용을 제거한다.

또한 기타 설명을 제거하고, ‘:’ 같은 특수 문자까지 지움으로써 책 이름의 정규화를 이룰 수 있다. 작가 정규화 함수에선 “ 외” 라는 글자가 나온다면(작가 이름 중간에 외 자가 들어갈 수 있기에 외 직전 띄어쓰기까지 포함한다.) 빈칸으로 대체하고, 다른 작가의 이름이 나온다는 기호인 ‘,’가 나온다면 그 기호의 앞쪽 글자만 사용함으로써 작가 이름의 정규화를 이룰 수 있다.



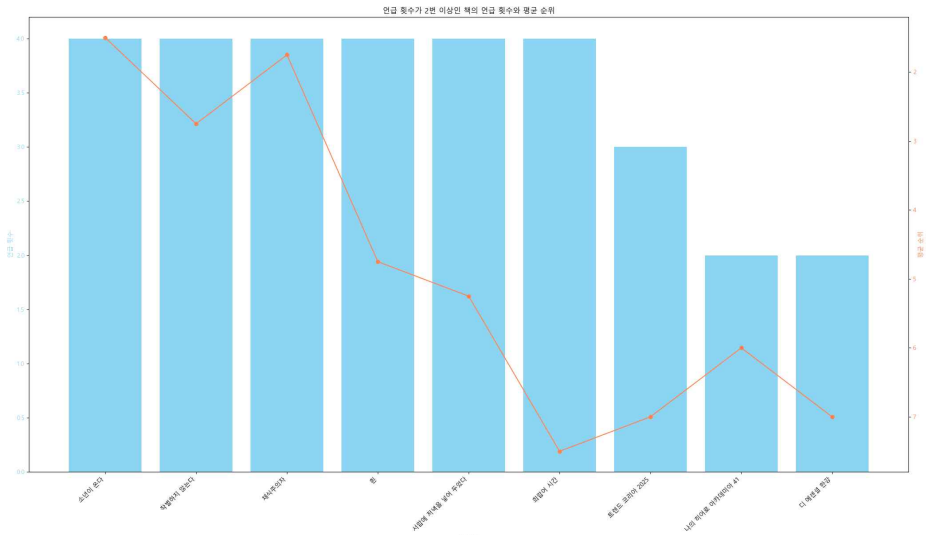
사진에서처럼 괄호나 기타 설명, 특수문자를 모두 지워 같은 책은 모두 같은 이름으로 만드는데 성공하였고, 작가 이름 또한 대표 작가 이름 한 명만 남김으로써 정규화를 이루는데 성공하였다.

3. 시각화

원하는 모든 정보를 배열에 넣고, 정규화하는데까지 성공했다면 이제 시각화를 할 차례이다.

```
# 모든 순위 데이터 통합
all_books =aladin_books_normalized +ypbooks_books_normalized +kyobobooks_books_normalized +yes24_books_normalized

# 책별 언급 횟수 계산
book_counts =Counter(all_books)

# 책별 순위를 저장할 딕셔너리
book_ranks =defaultdict(list)

# 각 책의 순위를 리스트에 저장
for i, book in enumerate(aladin_books_normalized):
    book_ranks[book].append(i +1)  # 1위부터 시작

for i, book in enumerate(ypbooks_books_normalized):
    book_ranks[book].append(i +1)

for i, book in enumerate(kyobobooks_books_normalized):
    book_ranks[book].append(i +1)

for i, book in enumerate(yes24_books_normalized):
    book_ranks[book].append(i +1)

# 각 책의 평균 순위 계산
average_ranks = {book: np.mean(ranks) for book, ranks in book_ranks.items()}

# 언급 횟수가 2번 이상인 책만 선택
filtered_books = [book for book, count in book_counts.items() if count >=2]

# 언급 횟수에 따라 정렬
filtered_books.sort(key=lambda x: book_counts[x], reverse=True)

# 정렬된 책의 언급 횟수와 평균 순위
filtered_counts = [book_counts[book] for book in filtered_books]
filtered_avg_ranks = [average_ranks[book] for book in filtered_books]

# 시각화
fig, ax1 =plt.subplots(figsize=(12, 8))

# 첫 번째 y축 - 언급 횟수
ax1.bar(filtered_books, filtered_counts, color='skyblue', label='언급 횟수')
ax1.set_xlabel("책 제목")
```

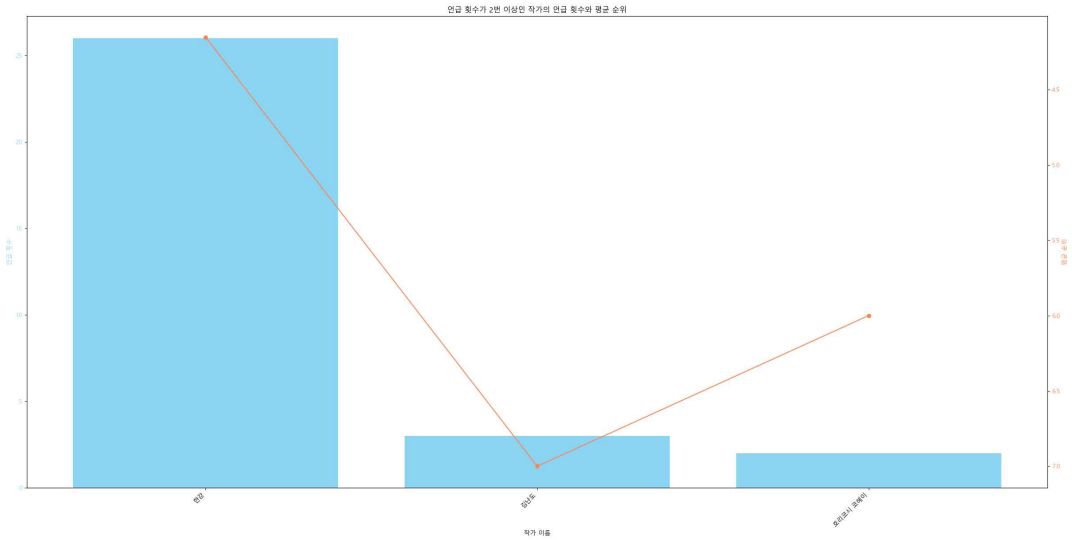
```
ax1.set_ylabel("언급 횟수", color='skyblue')
ax1.tick_params(axis='y', labelcolor='skyblue')

# x축 tick 위치 설정
ax1.set_xticks(np.arange(len(filtered_books)))  # 필터된 작가 수에 맞춰 tick 위치 설정

# x축 tick 레이블 설정
ax1.set_xticklabels(filtered_books, rotation=45, ha="right")

# 두 번째 y축 - 평균 순위
ax2 =ax1.twinx()
ax2.plot(filtered_books, filtered_avg_ranks, color='coral', marker='o', label='평균 순위')
ax2.set_ylabel("평균 순위", color='coral')
ax2.invert_yaxis()  # 순위는 낮을수록 상위이므로 축 반전
ax2.tick_params(axis='y', labelcolor='coral')

# 그래프 제목과 레이블 추가
plt.title("언급 횟수가 2번 이상인 책의 언급 횟수와 평균 순위")
```



해당 차트는 4개의 사이트에서 2번 이상 언급된 책을 대상으로 하며, 파란색 막대그래프는 각 책의 언급 횟수를, 주황색 꺾은선 그래프는 해당 책의 평균 순위를 나타낸다. 4개의 사이트에서 조사하였고 2번 이상 순위에 언급된 책을 나열한 차트이니 이 차트에서 언급 횟수, 즉 파란색 막대그래프는 2~4의 값을 보인다. 결과적으로, 한강 작가의 책인 **'소년이 온다'**, **'작별하지 않는다'**, **'채식주의자'**가 높은 언급 횟수와 평균 순위를 보이며 상위권에 위치하고 있다. 이는 노벨 문학상 수상 이후 한강 작가의 작품에 대한 독자들의 높은 관심을 반영한다. 이 외에도 경제서적 **'트렌드 코리아 2025'**와 인기 만화 **'나의 히어로 아카데미아'**가 뒤를 잇고 있다.

```
# 작가 통합
all_authors    =aladin_authors_normalized    +ypbooks_authors_normalized    +kyobobooks_authors_normalized
+yes24_authors_normalized

# 작가별 언급 횟수 계산
author_counts =Counter(all_authors)

# 작가별 순위를 저장할 딕셔너리
author_ranks =defaultdict(list)
```

```python
# 각 작가의 순위를 리스트에 저장 (임의의 순위 할당)
for i, author in enumerate(aladin_authors_normalized):
    author_ranks[normalize_authors(author)].append(i +1)

for i, author in enumerate(ypbooks_authors_normalized):
    author_ranks[normalize_authors(author)].append(i +1)

for i, author in enumerate(kyobobooks_authors_normalized):
    author_ranks[normalize_authors(author)].append(i +1)

for i, author in enumerate(yes24_authors_normalized):
    author_ranks[normalize_authors(author)].append(i +1)

# 각 작가의 평균 순위 계산
average_ranks = {author: np.mean(ranks) for author, ranks in author_ranks.items()}

# 언급 횟수가 2번 이상인 작가만 선택
filtered_authors = [author for author, count in author_counts.items() if count >=2]

# 언급 횟수에 따라 정렬
filtered_authors.sort(key=lambda x: author_counts[x], reverse=True)

# 정렬된 작가의 언급 횟수와 평균 순위
filtered_counts = [author_counts[author] for author in filtered_authors]
filtered_avg_ranks = [average_ranks[author] for author in filtered_authors]

# 시각화
fig, ax1 =plt.subplots(figsize=(12, 8))

# 첫 번째 y축 - 언급 횟수
ax1.bar(filtered_authors, filtered_counts, color='skyblue', label='언급 횟수')
ax1.set_xlabel("작가 이름")
ax1.set_ylabel("언급 횟수", color='skyblue')
ax1.tick_params(axis='y', labelcolor='skyblue')

# x축 tick 위치 설정
ax1.set_xticks(np.arange(len(filtered_authors)))  # 필터된 작가 수에 맞춰 tick 위치 설정

# x축 tick 레이블 설정
ax1.set_xticklabels(filtered_authors, rotation=45, ha="right")

# 두 번째 y축 - 평균 순위
ax2 =ax1.twinx()
ax2.plot(filtered_authors, filtered_avg_ranks, color='coral', marker='o', label='평균 순위')
ax2.set_ylabel("평균 순위", color='coral')
ax2.invert_yaxis()  # 순위는 낮을수록 상위이므로 축 반전
ax2.tick_params(axis='y', labelcolor='coral')

# 그래프 제목과 레이블 추가
plt.title("언급 횟수가 2번 이상인 작가의 언급 횟수와 평균 순위")
```



작가 이름 또한 정규화를 거쳤기 때문에 시각화 작업을 진행할 수 있다. 마찬가지로 정규화된 작가 데이터를 기준으로 2번 이상 언급된 작가를 대상으로 시각화하였으며, 파란색 막대그래프는 언급 횟수, 주황색 꺾은선 그래프는 평균 순위를 나타낸다. 한강 작가가 압도적인 언급 횟수와 순위로 상위권을 차지하고 있으며, 그 뒤를 잇는 것은 **'트렌드 코리아 2025'**를 집필한 **김난도** 작가이다. 만약 정규화 과정을 거치지 않았다면 해당 지표는 '김난도', '김난도 외', '김난도, 타 작가, 타 작가, ...'' 등의 다양한 표기법으로 인해 동일 작가임에도 여러 개의 항목으로 분산되었을 것이다. 인기 만화 **'나의 히어로 아카데미아'**를 집필한 **호리코시 코헤이**가 그 뒤를 잇고 있다. 흥미로운 것은, **'트렌드 코리아 2025'**와 **'나의 히어로 아카데미아'**를 비교했을 때, '트렌드 코리아 2025'가 더 높은 언급 횟수를 기록한 반면, 평균 순위에서는 '나의 히어로 아카데미아'가 우세함을 확인할 수 있다. 이는 만화책이 특정 서점(예: 알라딘, yes24)에서 높은 판매 경향을 보이는 것을 시사한다. 또한 경제서적은 사이트에 관계 없이 비교적 넓게 고루 분포하고 있음을 확인할 수 있다. 이 모든 실행 과정은 같이 첨부된 동영상으로 확인할 수 있다.

4. 결론

이 리포트는 Selenium을 사용해 각 서점의 베스트셀러 정보를 자동으로 수집하고, 정규화하여 시각화한 작업을 상세히 설명한다. 정규화된 데이터는 동일한 책과 작가를 일관되게 인식하게 하여, 특정 도서 및 작가의 인기도와 순위를 시각적으로 확인할 수 있게 한다.