송하윤 교수님
홍익대학교 학부생 연구 기회 프로그램 (UROP)
과제1 결과 보고서

연구 목적: D1과 D2를 비교하여 두 파일의 데이터들을 통합한 D 파일을 생성한다. (원본 문서: https://docs.google.com/document/d/1bS91vwepelrnxZgvfpuaFATvCIN0JUA9fsygy6Z_gJk/mobilebasic)

1) D1, D2, D: 구글 스프레트시트 링크

D1:
https://docs.google.com/spreadsheets/d/1bQyNNGMQsc4usqnakbWCnhlWed-iWhQh/edit?usp=sharing&ouid=108305840394117193693&rtpof=true&sd=true

D2:
https://docs.google.com/spreadsheets/d/1cddkF9V3IsA-U4IKwAqtefMadNz9ftesOiw4DW81qaM/edit?usp=drive_link

D:
https://docs.google.com/spreadsheets/d/1CMRIp-BQMHdmh94uCsjvIx-2dVjaYDW3/edit?usp=sharing&ouid=108305840394117193693&rtpof=true&sd=true

2) 중복 케이스 식별 로직(L)에 대한 자세한 설명

개괄적으로는 D1의 서점명과 D2의 서점명을 각각의 리스트로 저장하여 D1의 서점명이 D2에 서점명에 존재하는지 여부에 따라 코드를 작성하였다. 즉, 서점명을 key로 하여 중복 케이스를 식별하였다. 먼저, 중복 케이스가 없는 과정부터 실행하였다. 이 과정을 실행하기 위해 작성한 코드는 다음과 같다.

```
print("데이터 처리 전,",len(d2_df))
d2_name_list = d2_df['서점명']

num=len(d2_df)+1
#d1의 data -> d2의 data 과정 중 중복 데이터 없는 경우
for i in range(len(d1_name_list)):
  name=d1_name_list[i] #서점명

  if name not in d2_name_list: #중복 데이터가 없는 경우
    temp_dict={}
    temp_dict['번호']=num
    for k,v in result_dict.items():
      temp_dict[k]=v[i]
    d2_df = d2_df.append(temp_dict,ignore_index=True)
    num+=1

print("데이터 처리 후,",len(d2_df))
```

```
데이터 처리 전, 2528
데이터 처리 후, 3931
```

먼저, D2 파일을 pandas의 DataFrame으로 변환한 후, D2의 '서점명' 리스트를 저장하였다. 그러고나서 D1의 '서점명' 리스트 원소를 하나씩 비교하면서 D2에 서점명 리스트에 있는지(중복되는지)를 판별한 후, 중복되지 않는 경우에 result_dict으로부터 각 column과 그에 맞는 값을 저장하여, 이를 d2_df에 추가해 주었다. (result_dict은 D1을 읽어와서 key값을 D1의 column명으로, value를 각 column에 맞는 값들을 인덱스순으로 저장해 놓은 dictionary다.)

중복되지 않은 케이스를 처리한 후, 중복되는 케이스를 처리하였다. 코드는 다음과 같다.

```
[46]  d2_df=d2_df.set_index('서점명')
```

```
[47]  #d1 -> d2 과정 중 중복 데이터가 있는 경우

      col_list=list(result_dict.keys())
      col_list.pop(0)

      for i in range(len(d1_name_list)):
        name = d1_name_list[i]
        for j in range(len(col_list)):
          d2_df.loc[name,col_list[j]] = result_dict[col_list[j]][i]
```

col_list에 '서점명'을 제외한 D1의 나머지 column명들을 저장한 후, 위의 경우와 유사한 방식으로 중복 여부를 판단하여 중복된 경우, d2_df에 해당 name(서점명)을 갖는 row의 column에 대한 값을 D1의 값으로 저장을 해 준다. 이 경우, 기존의 D2 데이터 중 D1과 겹치지 않는 column은 그대로 유지되고, D1과 겹치는 데이터에 대해서만 값이 갱신된다.

정리하자면, 기존의 D2 파일을 pandas의 DataFrame으로 변환한 후, 해당 DataFrame에 중복되지 않는 경우에는 D1의 데이터를 추가해주고, 중복되는 경우에는 D1의 값을 이용하여 겹치는 column에 대해서만 값을 갱신해주며 겹치지

않는 column은 D1의 값으로 저장을 해준다. 이로써 기존의 D2파일에 새로운 데이터가 저장 및 갱신된 새로운 (DataFrame) d2_df를 얻은 후, 이를 excel로 저장하여 결과물을 저장하였다.

3) L을 포함한 전 과정에 대한 간결한 설명 매뉴얼

1. 데이터 추출

제공되는 데이터는 interactive하게 구현된 웹 페이지에 놓여있으므로, 이것을 추출해 가져오기 위해서는 javascript를 통한 이벤트의 발생, POST 등의 connection을 이용한 form 전송 등의 실제 유저가 웹페이지에 하는 상호작용을 구현할 것이 요구된다. 따라서 우리가 할 일은 Web scraping, 특수한 페이지를 대상으로 유저의 브라우저를 정의하고, 그 안에 있는 데이터를 추출해오는 작업이다. 여기서 우리는 Web crawling과 달리 robot.txt의 제약을 받지 않고, 이미 주어진 브라우저(Firefox, Safari, Chrome, etc)를 사용할 수 있으며, 그 페이지에 정확히 들어맞는 자바스크립트의 작성 등이 필요하게 된다. 즉, 그 특정 페이지에서 데이터를 추출하기 위한 목적으로 브라우저를 자동화하는 작업이 요구된다.

페이지는 DOM으로 제공되고 있기 때문에, 페이지의 html 문서를 분석하여 데이터가 놓여진 구조를 파악하였고, Javascript DOM을 이용하여 문서에 접근하며, 데이터 로드, 버튼 클릭 등의 이벤트를 발생시켜가면서 데이터에 접근하였다.

구체적으로는, 페이지 상에서 aVenue라고 분류되어 있는 element들을 최대한 가져오기 위해, 불러오기 버튼을 자동으로 누르는 interval을 구성하였고, 각 aVenue를 클릭하여 세부 정보(장소 타입, 주소, 영업 시간, 등)를 읽어들이고 제자리로 돌아오는 interval을 구성하였다. 즉, 총 1403개의 장소를 모두 로드시킨 후, 각 장소들을 클릭하여 정보를 추출하는 작업을 Javascript로 작성하였다.

마지막으로, 이런 Javascript를 페이지에 연결된 상태에서 실행시키기 위해 python의 selenium.webdriver 모듈을 사용하였다. (혹은, 브라우저의 개발자 모드 > 콘솔에서 직접 스크립트를 실행시킬 수도 있다.)

2. data 처리

D1 파일의 경우, json 파일을 그대로 읽어서 pandas의 DataFrame으로 변환한 후, excel파일로 저장하였다. D 파일의 경우, D1 파일과 D2 파일을 DataFrame으로 변환 과정을 먼저 수행하였다. 그러고나서 D1과 D2의 겹치는 column과 겹치지 않는 column 그리고, 겹치되 데이터 처리가 까다로운 column으로 구분하였다. 구분이 끝나면 각 column명에 따라 리스트를 정의하여 해당 리스트에 값을 넣어주는 과정을 작성하였다. 다만, 이 과정에서 D1의 데이터 양식과 D2의 데이터 양식이 다르기에 이 부분을 유의하면서 D2의 데이터 양식에 최대한 맞추며 값을 저장해 주었다. 예시로, D2의 전화번호/휴대폰번호 -> D1에서는 번호(010.. 혹은 042.. 등)로 되어 있는데, 이 경우에 010으로 시작하면 휴대폰번호로, 그렇지 않으면 전화번호로 저장해 주었다. 그리고, 처리가 까다로운 데이터에 대해서는 중복되는 column이지만 중복되지 않는다고 설정한 후 진행하였다. (예시, D2의 영업시작/영업마감은 시각이 입력되어 있으나, 이와 유사한 column인 D1의 운영시간은 tue-sat 10~19, 화-금

10-23 등 각기 상이한 상태로 입력이 되어 있어서 데이터 처리가 힘들다.) 기본적인 데이터 전처리 과정이 끝나면, 해당 데이터를 가지고 위에서 설명한 로직을 가지고 결과물 D파일을 추출한다.

본 데이터 처리 과정의 장점으로 꼽을 수 있는 점은 다음과 같다. 먼저, 각 column에 맞게 리스트를 나누는 과정이 매우 직관적으로 표현되어 있기에, 어떠한 파일이 주어지더라도 해당 파일에 적절하게 column의 리스트를 추출할 수 있다. 그리고 복잡한 과정을 거치지 않고, 리스트와 key 값만 가지고 중복되는 데이터와 그렇지 않은 데이터를 구분하여 데이터를 삽입 및 갱신할 수 있다는 점에서 간단하게 중복 데이터 처리를 수행할 수 있다는 점을 장점으로 꼽을 수 있다. 결론적으로, 직관성과 간결함을 가장 큰 장점으로 꼽을 수 있다. 따라서 크기가 작은 파일에 대해서는 효과적인 성능을 보여준다고 생각한다.

물론 개인적으로는 다음과 같은 한계점들을 보완할 필요가 있다고 생각한다. 일관성 없는 양식의 데이터에 대한 처리 방법의 부재, column의 의미성을 직접 비교해야 한다는 점, 리스트를 직접 column명에 맞게 정의해야 한다는 점, 단순히 index에 따라 값을 구별하다 보니 생기는 불필요한 오버헤드 등.

4) 위 과제를 수행하는데 사용된 코드 및 중간 과정에서 발생한 파일 일체

https://drive.google.com/drive/folders/1EqCRc-d4AHkT4XpgfLSz3_RLYL0CdXxc?usp=sharing