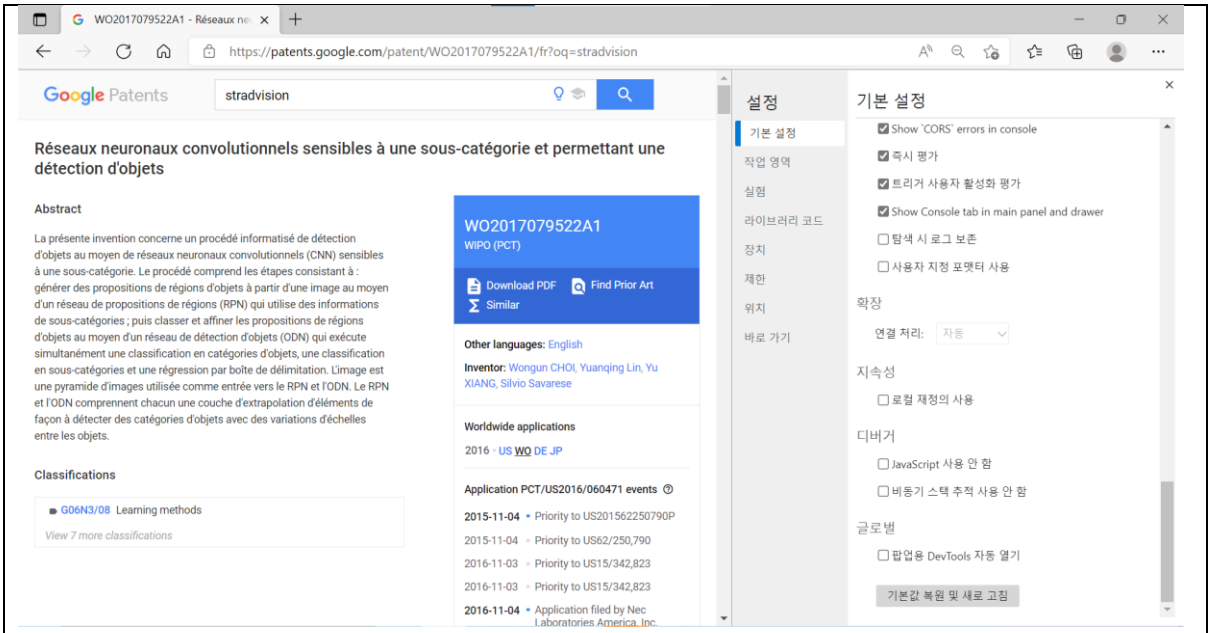
SV인턴 크롤링 가이드

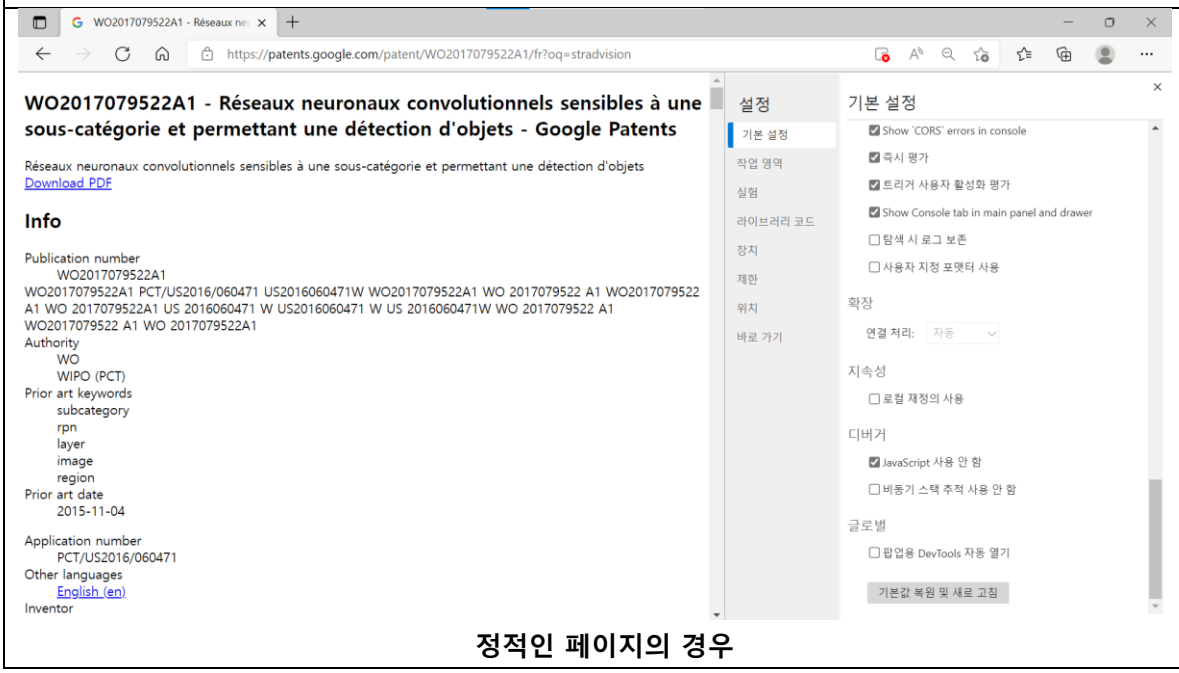
크롤링을 위해 참고해야 하는 파이썬 라이브러리

Beautifulsoup4, Selenium, Regex

웹페이지의 user가 원하는 정보를 찾기 위해선 정적크롤링과 동적크롤링의 개념을 알아야한다.



동적인 페이지의 경우



정적인 페이지의 경우

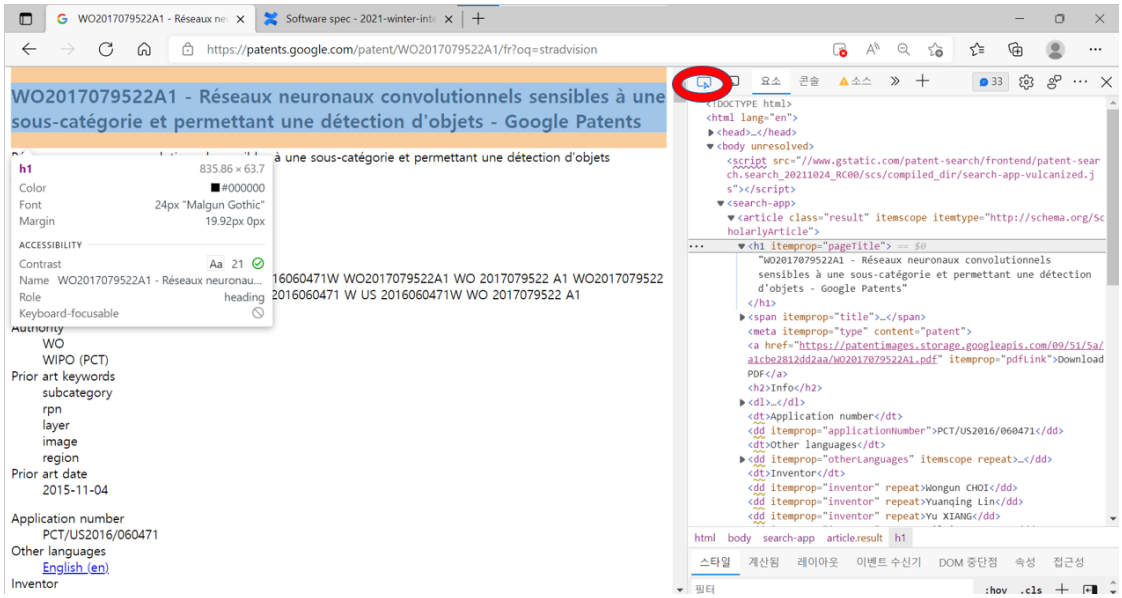
같은 페이지의 경우에도 보여지는 형태와 html의 구조가 달라지는 것을 볼 수 있다.

그렇기 때문에 크롤링 할 때 원하는 정보가 나타나지 않는다면 해당 문제일 확률이 높다.

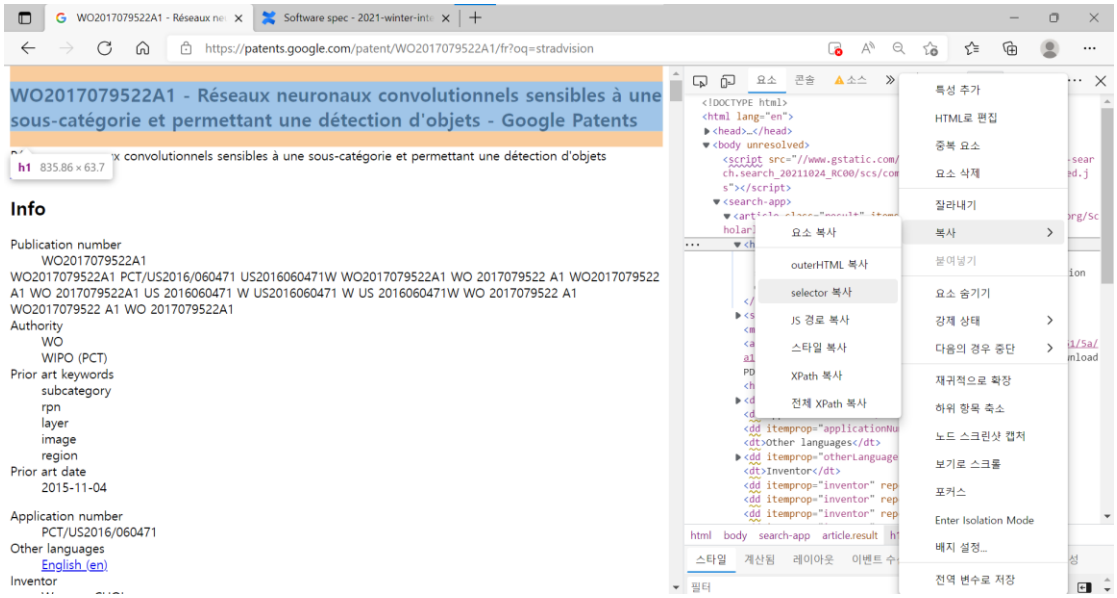
'beautifulsoup을 활용한 웹크롤링' 예제들을 살펴보면 많은 도움이 될 것이다.

1. 정적인 페이지 크롤링 (beautifulsoup)

Chrome 혹은 Microsoft edge를 사용해서 인터넷에 접속한 이후 키보드의 f12를 누르면 아래 사진과 같은 화면이 나타난다. 이 때 빨간색 동그라미를 체크하고, 필요한 정보가 있는 곳에 마우스를 가져가면 html 구조 중 어디에 해당되는 부분인지 보여준다.



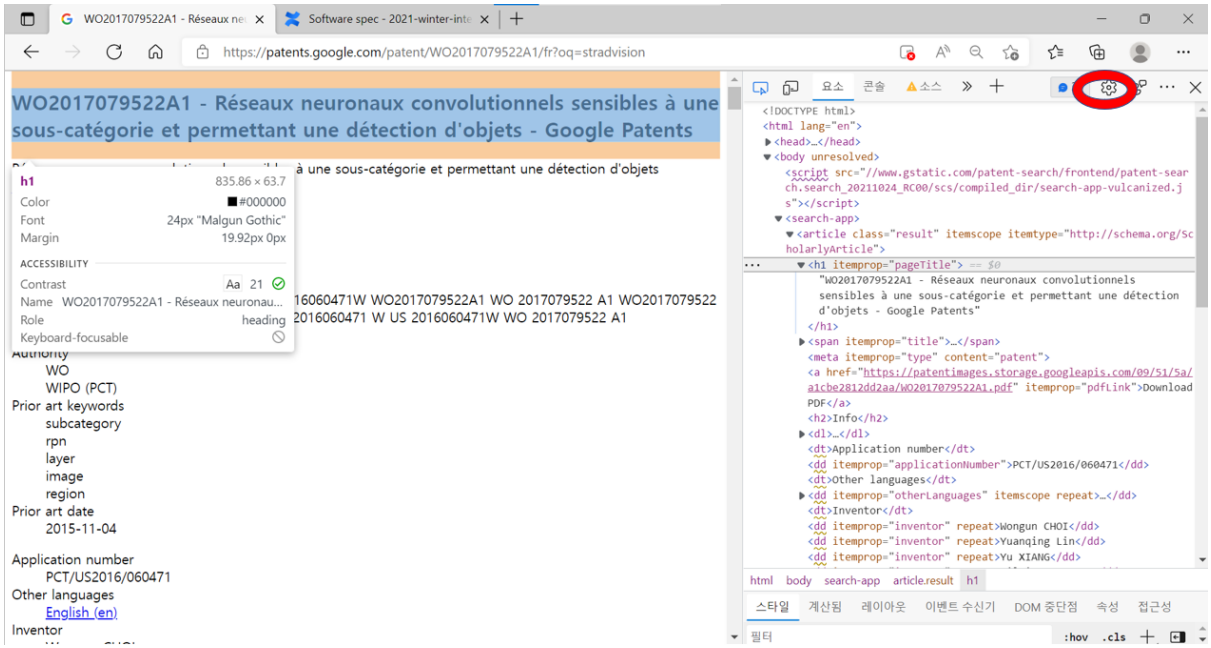
그 이후에 아래 사진처럼 한다면 selector 경로를 복사할 수 있다.



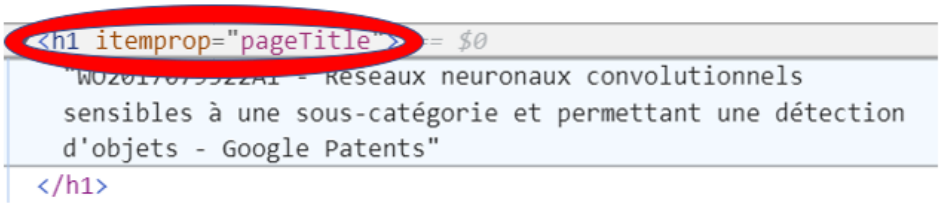
이를 통하여 soup.select 함수를 사용할 수 있다.

주로 select를 사용하여 프로그래밍을 하는 것이 직관적으로 쉽게 다가올 것이다.

!! 다만 beautifulsoup만을 활용하기에 정적인 페이지를 가져오므로 selector경로를 복사할 때에는 설정의 javascript사용해제를 체크해야 동일한 경로가 된다. !!



만일 위의 사진에서 제목에 관한 정보를 가져오고 싶다면 문자열 slicing 혹은 regex라이브러리를 사용하면 된다. 다만 이 때에는 다른 페이지의 정보는 문자열의 개수가 다를 수 있기에 이를 고려해서 프로그래밍을 해야 한다.



다만 selector 경로가 동일하기에 의도한 것보다 많은 데이터들이 찾아지는 경우가 있다. 이 때에는 beautifulsoup의 find 또는 find_all 함수를 사용해야 한다.

2. 동적인 페이지 크롤링 (selenium + beautifulsoup)

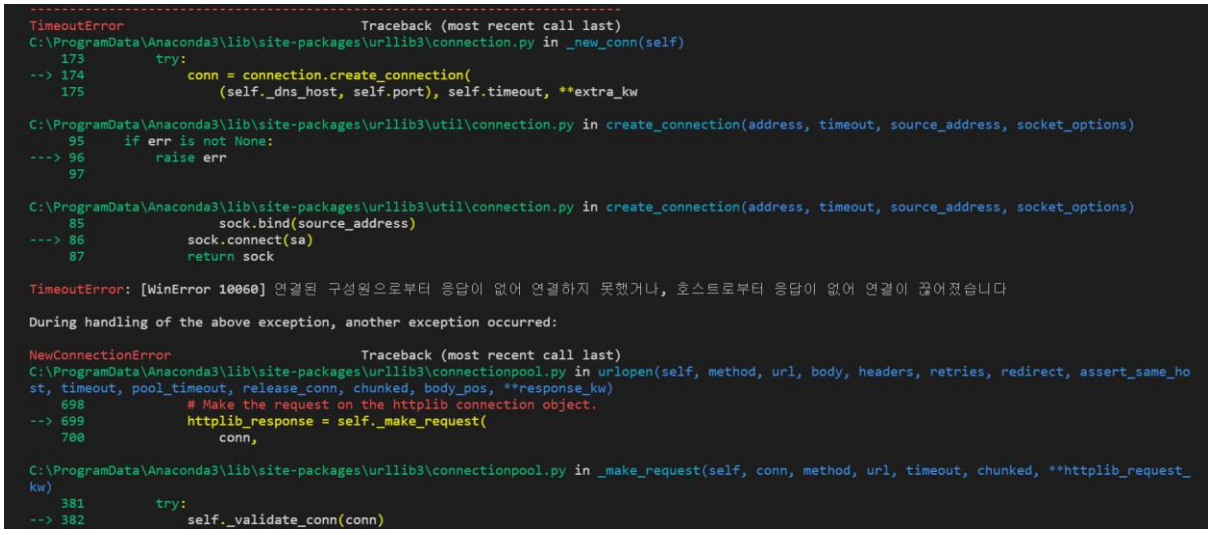
먼저 'selenium을 활용한 동적크롤링'으로 검색해서 관련된 자료를 보는 것을 추천한다.

Chromedriver를 설치하고 selenium을 활용한다면 동적인 페이지도 모두 가져올 수 있다. 다만 이 때에는 정적인 페이지와 html의 구조가 다르기 때문에 selector 경로가 다르게 표시된다. 이 때에는 Javascript 사용해제 부분을 체크해제하고 크롤링을 진행해야 한다.

전체적인 크롤링 과정은 위의 정적인 페이지 크롤링과 동일하다 다만 selenium과 chromedriver를 사용해서 직접 페이지에 접속해 모든 동적인 데이터를 가져온 이후에 크롤링을 진행한다고 생각하면 이해가 수월할 것이다.

사용법은 어렵지 않기에 검색한다면 쉽게 사용할 수 있을 것으로 보인다.

3. 발생한 문제사항



다만 코드가 길어지고 크롤링 해야하는 페이지가 많아진다면 응답없음이 뜨게 된다. 뜨는 원인은 단위 시간당 페이지에 접속하는 횟수가 너무 많다면 공격으로 판단해 차단한다는 것이었다. 코드 파일

의 마지막 부분을 살펴보면 error가 발생하는 부분은 request 부분이었다. 때문에 모든 request에 exception 처리를 하고, 예외가 발생한다면 크롤링된 부분 이후부터 크롤링 할 수 있도록 함수의 input output으로 어디까지 크롤링이 진행되었는지 정보가 추가된다면 가능할 것으로 보인다. 이 때에는 time.sleep()를 통해 천천히 크롤링하도록 해야할 것이다. 시간이 부족하기에 마지막 14000여개 자율주행 키워드로 검색했을 때에는 중간중간 계속 connection error가 발생해 크롤링 된 곳까지 확인하고 다시 크롤링하는 식으로 마무리했다. 이에 대한 개선이 이루어진다면 좋을 것이다.