사용한 파이썬 라이브러리

① re : 정규식 라이브러리로써 파싱할 때 가장 속도가 빠르다.

② urllib.request.urlopen : Html을 String으로 변환해준다.

③ urllib.error.HTTPError : 웹 사이트를 열 때 발생하는 오류를 알려준다.(예시 404 Page not found)

코드 내부 변수 및 함수 구조

변수

inner\_html (list of string) : 한동안 문제였던 <jframe> 태그에 숨어있던inner html의 URL을

저장하는 리스트이다.

pics (list of string) : inner html(블로그 내부)의 사진들의 URL을 저장하는 리스트이다.

text (list of string) : inner html(블로그 내부)의 문자열들을 저장하는 리스트이다.

num\_pic(int) : inner html(블로그 내부)의 사진들의 개수를 저장하는 정수이다.

target\_url(String) : 네이버 블로그의 상위 URL을 저장하는 문자열이다.(https://blog.naver.com)

html, link, links : html을 파싱한 결과를 저장하는 문자열이다.

함수

web\_parse(String url) : url 문자열을 기준으로 해당 사이트 Html코드 전체를 반환한다.

re.findall(’정규식’,HTML) : HTML 코드 중 정규식 조건에 맞는 문자열을 반환한다.

get\_targeturl(String url) : inner html에 접속하려면 뭐시기.com 문자열이 필요하다. .com 문자열을 반환한다.

순서

① 파싱할 웹 페이지의 주소를 지정한다.

② 파싱한 결과로 <iframe [모든 문자열]> 안에 src=으로 시작하는 url을 읽어온다.

(inner html을 찾기 위함이다. 예시 /PostView.njn? 예시를 보면 앞에 naver.com이 없다. target\_url로 완전한 URL을 얻을 수 있다.)

③ inner html을 파싱한다.

④ inner html의 이미지 파일을 파싱한다.(추후 네이버 스티커, 광고 문자열 등을 추가할 계획이다.)

⑤ 사진이 몇 개 있는지 센다. (pic\_url 리스트에 원소가 몇 개 있는지 보면 된다.)

업데이트 내역

4.13

①변경된 점들

1. 다양한 형태의 네이버 블로그를 파싱하기 위한 코드 수정

2. 원활한 테스트를 위해서 url txt (input.txt)을 읽어 오고 파싱 결과를 output.txt 파일에 씀

②추가사항

1. 다양한 형태의 네이버 블로그를 파싱하기 위한 코드 수정

다음과 같은 형태의 블로그를 파싱할 수 있도록 수정함

형태 1 https://blog.naver.com 으로 시작하는 네이버 블로그(ex https://blog.naver.com/invu1657/221473146800)

기존과 동일하다.

형태 2 https://m.blog.naver.com 으로 시작하는 네이버 블로그

inner\_html이 없어서 그냥 파싱하면 된다.

형태 3 http://로 시작하는 네이버 블로그(ex http://o-goon.com/221287199411)

프록시 서버 형태로 주정되며 형태 1과 파싱방법이 약간다름

(frame에 blog.naver.com url 이 있고 이 url을 가지고 형태 1 코드를 실행하면 된다.)

2. 원활한 테스트를 위해서 url txt (input.txt)을 읽어 오고 파싱 결과를 output.txt 파일에 씀

처음에 input.txt와 output.txt 파일 위치를 적어줘야한다. 코드 전처리 부분에 있다.

4.10

①변경된 점들

1. 변수명 및 함수명 정리 작업

3.28

①변경된 점들

1. 정규식 2차 수정

②추가사항

1. 정규식 2차 수정(노란색 부분이 변경된 것들)

사진에 경우 url, 가로길이, 세로길이 파싱하여 저장한다.

pics += re.findall('<img src="(.\*?)".\*?data-width="(.\*?)" data-height="(.\*?)".\*?class="se-image-resource" />', html)

스티커는 그대로 url

sticker += re.findall('<img.\*? src="(.\*?)".\*?class="se-sticker-image" />',html)

텍스트는 그대로 텍스트 그 자체

text += re.findall('<span.\*?>(.\*?)</span>',html)

쓸대없는 문자열들이 많이 빠졌지만 그래도 몇몇은 남아있음

3.27

①변경된 점들

1. 블로그의 텍스트 파싱 기능 추가

2. 범용성을 높힌 정규식으로 교체

3. URL 읽어오기 기능을 추가하기 위한 밑작업

②추가사항

1. 텍스트 파싱 + 2. 범용성 높은 정규식

범용성을 높힌 정규식 덕분에 필요없는 html 속성들도 같이 딸려 파싱된다.(속도 저하에 원인)

pics += re.findall('<img src="(.\*?)" (.\*?) class="se-image-resource" />', html)

sticker += re.findall('<img src="(.\*?)" (.\*?) class="se-sticker-image" />',html)

text += re.findall('<span(.\*?)>(.\*?)</span>',html)

유형 1(이걸 기준으로 한다 하면)

<span style="" class="se-fs- se-ff- " id="SE-8bc8444a-b906-4fe9-97a0-39e64fba4e67">열심히 사진을 찍었습니다^-^ㅎㅎ</span>

유형 2(class와 id가 다르고)

<span style="" class="se-fs- se-ff-nanumbarungothic " id="SE-28b8065b-db89-4e33-99d5-89521d79ff2b">그림말풍선은 2017년 초에 아이디어가 떠올랐고</span>

유형 3(아마도 다른곳에서 ctrl+c v 해보이는 텍스트들은 세부 설정 없이 그냥 span태그만 있음)

<span>파이썬 공식홈페이지에 설치형 버전이 있던데, 그림에 표시한거 다운받아서 설치했다.<br></span>

이런식으로 블로그마다 조금씩 차이가 나서 모든 유형을 포함하는 정규식을 사용하다보니 쓸대없는 se-ffnanumb 같은 코드들도 같이 딸려옴