개인과제 보고서

Main.py

이 파일에서는 밑에 5가지의 파일의 함수들을 불러와 작동하는 파일로 만들었습니다.

따라서 main.py는 인포메이션 데스크 역할을 합니다.

특허 검색 가능한 프로그램 목록으로

1. **출원인 입력- 특허출력 – ai.py**
2. **발명가 입력- 특허 출력. – bi.py**
3. **키워드입력- 특허출력– ci.py**
4. **특정특허의 특허 설명– di.py**
5. **각 특허의 특정 정보 정리 – ei.py**

위의 5가지를 만들었습니다.

프로그램을 실행시키면 5가지 목록을 안내하고, 원하는 값(프로그램)을 입력받아 프로그램을 실행하는 구조를 세웠습니다.

목록 선택은 숫자이지만 (string)으로 입력받았습니다.

왜냐하면 int(input())으로 입력을 받았을 때, 제시된 숫자를 입력하지 않았을 경우 오류 핸들링이 길어지기 때문입니다.

처음 입력 값에 따라 위 5가지 프로그램으로 접근하게 됩니다.

1, 2, 4의 프로그램 실행 시 각 프로그램에 맞는 특정 목록(출원인, 발명가, 특허목록)도 출력할 수 있도록 설정하였습니다.

1,2,3 프로그램에서는 한글 or 영어를 선택할 수 있습니다.

언어를 선택하면, 그 언어에 맞게 입력하지 않는다면 예외 핸들링을 설정하였습니다.

또한 입력받은 이름이 한글인데, 한글이 없고 영어가 포함 or 영어, 숫자가 포함되어 있다면 "'한글' 또는 '영어'를 입력해 주세요."를 출력합니다. (Exception Handling)

이렇게 1 – 5번 목록의 안내 및 실행 코드를 구성하였습니다.

만약 특허 검색 조건(1-5 입력)에 부합하지 않는 입력 값이 입력된다면

'입력 번호를 확인 후 다시 시도해 주세요.'라는 문구를 출력하도록 했습니다.

ai.py 출원인입력- 특허출력

find_all_applicant(fn) 과 find_all_applicant_eng(fn) 함수는 파일에 있는 특허 출원인 이름 목록을 모두 찾아서 출력해 주는 함수입니다.

모든 함수에서 with open을 통해서 개인과 제 data 파일을 읽고 xml_read에 저장합니다.

그 후 BeautifulSoup으로 xml 파일로 취급하도록 변환했습니다.

(이 논리는 이후 모든 함수에서 사용했습니다.)

one_invent 변수는 2000개의 특허를 list of list로 만들어 저장한 변수입니다.

Ex)[[특허 1], [특허 2] , [특허 3], ... , [특허 2000]]

각 특허에서 특정 태그 정보를 뽑아내는 방법을 한참 고민하던 중 list of list에서 특허 하나씩 검색해가면서 찾아내는 방법으로 생각했습니다.

(이 변수는 다른 파일(.py)에서도 동일하게 사용하였습니다.)

이제 for 구문을 통해 앞에서부터 특허 하나씩 apl에 할당합니다.

하나의 특허가 들어있는 apl에서 출원 정보(부모) – 이름(자식) 논리로 출원인의 이름을 모두 찾습니다.

이때, 출원 정보(태그 : applicantinfo)를 찾는 이유는 ax2102:name 태그가 많기 때문에 출원인인지, 대리인인지, 발명가 인지 모르기 때문입니다.

따라서 name의 상위 태그를 출원 정보(부모) 태그를 찾고 그 속에 있는 이름을 찾기 위해서 구성하였습니다.

(이 논리는 이후 함수와 발명가 – 특허 찾기(bi.py)에도 적용하였습니다.

정리하면, 이 함수에서 특허 1의 출원 정보 태그 내의 -> 이름을 찾고 모든 이름값들을 리스트에 추가했습니다. (출원인이 한 명 이상인 경우가 있기 때문에 find_all을 사용했습니다.)

그리고 만들어진 출원인 이름 리스트를 출력합니다.이 방법을 for문을 통해 특허1 부터 특허 2000까지 반복하여 모든 출원인 이름을 출력했습니다.

find_apllicant(fn, name) 과 find_apllicant_eng(fn, name) 함수는 정해진 파일 명과 name을 이전 main.py에서 입력받습니다.

이 함수는 출원 관련 태그(부모)를 찾고, 출원인 이름(자식)을 찾고 리스트에 추가 후 입력받은 이름이 리스트에 있으면 그 특허의 이름을 찾는 함수입니다.

one_invent[0]부터 특허 하나하나씩 출원인 정보 (부모) – 이름(자식)을 찾습니다.

그 이름들을 apl_name0 리스트에 text로 추가합니다.

그 후 이 리스트에 입력했던 이름이 있으면, 이 특허의 이름과 출원 날짜를 찾습니다.

그리고 미리 선언해 두었던 inv (특허 제목 저장 변수 리스트), date(특허 출원 날짜 저장 변수 리스트)에 append 시킵니다. 만약 apl_name0에 입력했던 이름이 없다면, 다음 특허로 넘어갑니다.

이 과정을 2000개 특허 모두 반복하고, 입력했던 출원인의 특허 명과 출원 날짜를 모두 찾습니다.

만약, 입력받은 출원인의 이름이 특허 내에 없다면 이름을 확인해달라는 말을 출력합니다. (exception Handling)

결과를 출력하고 csv 파일로 저장합니다. (출원인 이름, 특허 명, 출원 날짜)

find_apllicant_eng(fn, name)에서는 모든 논리가 위의 함수와 동일하지만,

영어의 대, 소문자에 따라 입력받은 값을 찾지 못하는 경우가 있었습니다. 따라서 입력받은 영어 string을 대문자로 바꾸었습니다.

그 후 특허 안에서 출원인 영어 이름을 찾고, 그 이름도 모두 대문자로 바꾸어(.upper() 사용) 특정 이름을 찾도록 구성하였습니다.

이후에 그 이름이 존재한다면, 대문자로 출력하는 것이 아니고 입력받은 이름으로 출력하였습니다. 이 외의 논리는 같습니다.

bi.py 발명가 입력- 특허출력

find_all_inventor(filename), find_all_inventor_eng(filename) 함수는 발명가 이름을 모두 찾아서 발명가 목록을 출력하는 함수입니다.

위에서와 마찬가지로 with open과 beautifulsoup을 사용해서 파일을 xml로 읽습니다.

그 후 모든 특허를 list of list의 형태로 one_invent에 할당합니다.

for 문을 통해 one_invent[0]부터 하나씩 특허의 발명 정보(부모)-이름을 찾습니다.

하나의 특허에서 발명가 모든 사람의 이름을 찾습니다. (inv_name1)

Inv_name1 리스트에서 앞에서부터 하나씩 뽑아(inv_name2) 텍스트 형식으로 변환 후

inv_name0에 리스트 추가하고 출력했습니다.

다시 위의 for 문 (13 줄)로 돌아가 반복합니다.

이렇게 특허 1부터 특허 2000까지 반복하여 모든 특허의 발명가 이름을 출력했습니다.

find_inventor(filename, name), find_inventor_enge(filename, name)

위에서와 마찬가지로 with open과 beautifulsoup을 사용해서 파일을 xml로 읽습니다.

그 후 모든 특허를 list of list의 형태로 one_invent에 할당합니다.

for 문을 통해 one_invent[0]부터 하나씩 특허의 발명 정보(부모)-이름을 찾습니다.

하나의 특허에서 발명가 모든 사람의 이름을 찾습니다. (inv_name1)

Inv_name1 리스트에서 앞에서부터 하나씩 뽑아(inv_name2) 텍스트 형식으로 변환 후

inv_name0 리스트에 추가했습니다.

그리고 inv_name0에 입력받은 name의 값과 일치하는 게 있는지 확인합니다.

일치하는 값이 있다면, 그 특허의 특허 명 과 초록(astrtcont)를 찾습니다.

각각 inv, describ라는 리스트 형식의 변수에 추가하고 특허 개수(co)에 +1 했습니다.

일치하지 값이 없다면 다음 특허로 넘어가고 반복합니다.

이렇게 2000개의 특허를 다 검사한 후

특허 개수(co)가 0이면, '이름을 확인해 주세요.'를 출력합니다.

특허 개수가 1 이상이면, 입력한 발명가의 특허가 존재한다는 의미입니다.

따라서 입력한 발명가의 특허 개수, 특허 명, 각 특허의 초록을 출력하고

csv 파일에 저장합니다.

find_inventor_eng(filename, name) 함수도 위와 동일한 구조입니다.

하지만 find_apllicant_eng(fn, name) 함수와 마찬가지로 영어의 대, 소문자에 따라 입력받은 값을 찾지 못하는 경우가 있기 때문에, find_apllicant_eng 함수와 마찬가지로 입력받은 발명가의 이름과 특허 영어명을 대문자로 바꾼 후 일치 여부를 판단하였습니다.

나머지 구조는 동일하게 작성했습니다!

ci.py 키워드입력 – 특허 출력

find_key_title(filename) 함수는 키워드(단어)를 입력 후 키워드가 포함된 특허를 출력하는 함수입니다.

함수 안에서 한글 or 영어로 언어를 선택할 수 있습니다.

한글, 영어를 선택하여 들어가면 각 언어로 키워드를 입력을 받습니다.

이때 한글 언어로 선택하였을 경우, 한글, 숫자 키워드를 입력하면 특허를 찾을 수 있습니다.

만약, 영어를 입력했을 경우 두 가지의 경우가 있습니다.

첫 번째, 영어가 포함된 특허가 있을 경우에는 출력 하니다.

두 번째, 입력한 영어가 포함되지 않았다면 문자열 분류(regex_kor)를 통해 "특허 키워드에 영어가 포함되어 있습니다. 확인해 주세요"를 출력하는 예외 핸들링을 넣었습니다.

각 언어에서 특정 키워드가 포함되지 않았을 경우에는 "'(입력한 단어)' 단어를 포함하는 특허가 없습니다. "를 출력합니다.

한글로 언어를 선택했을 경우, 모든 특허의 제목을 리스트 형식(find_all)으로 찾습니다.

그리고 특허 제목이 저장된 리스트에서 for 문을 통해 하나씩 text 화하고 입력받은 키워드와 비교합니다.

만약 입력 한 키워드가 특허 제목에 있을 경우에는 t라는 변수 특허 제목을 저장하고 위에서 선언한 리스트인 data_key_title 리스트에 추가합니다. 그리고 특허 개수(co)를 1개 추가합니다.

모든 특허(2000개) 제목에 키워드가 있는지 확인한 후 co의 수를 통해 키워드가 포함된 특허가 있는지 확인합니다.

입력한 키워드가 특허 제목에 없을 경우에는(co = 0) 키워드에 영어가 포함되어 있는지 우선 확합니다.

키워드에 영어가 포함되어 있을 경우, '입력하신 특허 키워드에 영어가 포함되어 있습니다. 확인해 주세요.'를 출력합니다.

키워드에 영어가 없이 한글, 숫자로 이루어진 경우에는 "'입력한 단어'단어를 포함하는 특허가 없습니다.' "를 출력합니다.

입력한 키워드가 포함된 특허 제목이 있을 경우(co>= 1)에는 "'keyword'가 포함된 특허는 총 co 개입니다."라는 문구와 특허 제목의 리스트를 출력합니다.

그리고 이 경우에만 키워드가 포함된 특허 제목이 있기 때문에 csv 파일로 저장합니다.

di.py : 특정 특허의 특허 설명

find_titles(filename, name) 함수는 파일 내에 있는 특허 제목을 모두 출력해 주는 함수입니다.

이 함수에는 ai.py, bi.py 와 동일한 논리로 특허목록을 출력할 수 있습니다.

특허 목록을 출력하여 특허를 선택한 후 그 특허의 제목을 입력할 수 있도록 하기 위해 만들었습니다.

find_claims(filename, name) 함수는 특정 특허 제목을 입력하여 그 특허의 설명(claims)을 출력하는 함수입니다.

2000개의 특허 리스트(one_invent)에서 하나씩 뽑아서 각 특허의 제목을 찾으며 입력받은 특허 제목과 같은 값이 있는지 찾습니다.

특허 제목이 같은 특허도 여러 개 존재하는 특허가 있어 2000개 모두 탐색하는 방향으로 설정했습니다.

입력한 특허 제목과 같은 특허 제목이 있으면, inv_title_real 리스트에 추가합니다.

그리고 그 특허의 설명(claim)을 모두 찾고 text 형식으로 cl 리스트에 추가합니다.

여기서는 특허의 개수를 co가 아닌 특허 제목의 리스트 길이로 구분하였습니다.

특허 리스트(inv_title_real)의 길이가 1 이상이면, 입력한 특허가 존재하는 것이므로 특허 제목과 특허 설명을 각각 출력합니다.

이때 특허가 2개 이상이면, 출력할 때는 [특허 1, 특허 2 ..., 특허 n] 리스트로 특허 제목을 출력하고 그 특허의 설명으로는 [[특허 1설명] , [특허 2 설명]] 리스트 of 리스트로 만들었습니다!

그리고 그 특허(들)의 설명을 출력하고 csv 파일로 저장합니다.

첫 번째 행에 파일에 대한 간단한 설명, 두 번째 행에 열에 대한 간단한 분류, 그리고 세 번째 행부터 특허 명, 특허 설명으로 쓰도록 하였습니다.

이때 특허 1, 특허 1 설명

특허 2, 특허 2 설명으로 각 행에 하나의 특허로 행을 구분하도록 했습니다.

ei.py : 본 데이터내의 특허 정보 정리

write_Invent(filename) 함수는 파일에 있는 데이터에서 특허 명(특허 제목), 출원인, 발명가, 특허 초록, 특허 설명을 출력하고 csv 파일로 저장하는 함수입니다.

처음에는 위의 파일들과 같은 논리로 파일을 xml 파일로 변환 후 거의 모든 자식을 포함하는 태그(''soapenv:body')를 사용해서 모든 특허를 list of list로 만듭니다.

그리고 앞에 특허부터 co: 특허 개수(밑에서 순번으로 사용) 출원 정보 -> 이름(출원인) , 발명 정보 -> 이름(발명가) , 초록, 특허 설명 태그를 통해 각 리스트에 저장했습니다.

이때, 출원인, 발명가, 특허 설명은 여러 개가 존재하는 경우도 있기 때문에 find_all을 사용하였고,

이를 for 문을 통해 text로 정리 후 리스트에 추가했습니다.

각 리스트에 저장된 정보를 정리된 특허 리스트(sub_info)에 추가하였습니다.

이제 sub_info에는 순서대로 특허 명(특허 제목), 출원인, 발명가, 특허 초록, 특허 설명이 들어갑니다.

이렇게 만든 리스트를 반복하며 total_info에 추가하여 정리된 전체 특허를 만들었습니다.

그리고 csv 파일로 저장했습니다.

맨 첫 번째 행에 파일 설명을 작성하고

두 번째 행에 열 별로 구분하는 설명을 적었습니다.

그리고 세 번째 행부터 total_info에서 첫 element부터 끝 element까지 반복하도록 ragne를 통해 숫자를 증가시키며 쓰도록 하였습니다!

그리고 다 쓰게 되면 예외 처리로 그전까지 썼던 개수로 '(co) 개의 특허 정보 생성 완료'를 출력하도록 했습니다.

감사합니다.