프로젝트 1: Information Retrieval System

2019년 3월 18일

제출 기한: 4월 1일 월요일 23시 59분

개요

프로젝트 1에서는 Inverted index, TF-IDF, PageRank를 이용한 Wikipedia 문서 검색 시스템을 만들어본다. 시스템은 우선 Inverted index를 이용하여 사용자가 입력한 검색어(query)를 포함하는 문서들을 모두 찾은 다음 TF-IDF score와 PageRank score로 정렬하여 상위에 있는 문서들을 사용자에게 제공한다.

요구 사항

1. 검색에 필요한 각종 테이블을 생성. (Wikipedia 문서 테이블, link 테이블, Inverted index 테이블 등)
2. 문서 텍스트를 담고 있는 테이블로부터 각 단어의 TF-IDF 계산.
3. 문서 간 link 정보를 담고 있는 테이블로부터 각 문서의 PageRank 계산.
4. Inverted index를 이용하여 사용자가 입력한 검색어(query)를 포함한 문서를 모두 검색.
5. 검색된 문서들을 TF-IDF score와 PageRank score를 기준으로 내림차순 정렬하여 상위 10개 출력.

상세 스펙

1. 제공된 SQL 덤프 파일(wiki.sql, link.sql)을 이용하여 테이블 생성
   A. **wiki** 테이블 스키마

컬럼명	데이터 타입	Primary Key 여부	설명
id	int(11)	O	Wikipedia 문서 고유 id
title	mediumtext	X	Wikipedia 문서 제목
text	mediumtext	X	Wikipedia 문서 내용

   B. **wiki** 테이블 예시

id	title	text
16583123	Albert_Brunies	Albert Brunies.Albert Abbie Brunies (January 19, 19...
16585069	Michel_Creton	Michel Creton.Michel Creton (17 August 1942 in Wa...
16629152	Habranthus	Habranthus.Habranthus (copperlily) is a genus of t...
16732148	Domestic_violence_in_Peru	Domestic violence in Peru.Domestic violence in Peru...

C. **link** 테이블 스키마

컬럼명	데이터 타입	Primary Key 여부	설명
id_from	int(11)	X	해당 링크가 포함된 문서 id
id_to	int(11)	X	해당 링크가 참조하는 문서 id

D. **link** 테이블 예시

id_from	id_to
17143	2727051
17154	36269085
17165	24610443
17165	3575319

2. **wiki** 테이블을 이용하여 Inverted index 테이블 생성

- 각 단어들이 포함된 문서의 id를 저장하는 Inverted index 생성.

- ~~단어는 wiki 테이블 text 컬럼에 저장된 텍스트에 NLTK Punkt Tokenizer를 적용하여 얻음. Tokenizer는 문장을 단어 단위로 parsing하고 문장 부호를 분리해 주는 툴.~~

- - NLTK Punkt Tokenizer의 경우 Sentence 단위의 tokenizer이기 때문에 단어 단위로 tokenizing 이 불가능한 것으로 판명되었습니다. 따라서 Punkt Tokenizer가 아닌 기본 tokenizer인 work_tokenize를 사용하시면 됩니다.

  사용 예시)

```
In [34]: from nltk.tokenize import word_tokenize

In [35]: sentence = 'i am iron man '

In [36]: word_tokenize(sentence)
Out[36]: ['i', 'am', 'iron', 'man']
```

- 사용에 대한 자세한 사항은 참고자료 4번 링크에서 확인

- 단어의 대/소문자는 구분함. (ex. PageRank와 pagerank는 서로 다른 단어로 취급)

- 테이블 스키마는 아래 예시처럼 구현할 수 있으며 다른 형태의 스키마도 허용.

A. Inverted index 테이블 스키마 예시

컬럼명	데이터 타입	Primary Key 여부	설명
term	varchar(1000)	X	개별 단어
id	int(11)	X	해당 단어가 포함된 문서 id

B. Inverted index 테이블 예시

term	id
finally	1739797
financed	416890
financial	416890
financial	1019791
financial	2266706
find	2266706

3. **wiki** 테이블을 이용하여 문서 내 각 단어의 TF-IDF 계산

- TF-IDF 공식은 강의 자료(Chapter 21) 7, 8페이지에 제시된 "the one used in textbook" 이용.

4. **link** 테이블을 이용하여 각 문서의 PageRank 계산

- 문서 간 link 관계를 나타낸 **link** 테이블을 이용하여 PageRank 알고리즘 구현.
- PageRank 공식은 강의 자료(Chapter 21) 13페이지에 제시된 것을 이용.
- Jump probability δ는 0.15로 설정.
- PageRank score의 수렴 판단 기준은 ε ($1.0e^{-8}$) 사용.
  즉 score 계산을 반복하다가 score 변화량의 총합이 ε보다 작을 때 계산을 중단.
- 알고리즘 실행 결과, 모든 문서들은 PageRank score를 가짐.

5. 검색 결과 랭킹 방식

- 사용자가 입력한 검색어(query)를 포함한 문서들을 모두 찾은 다음 정렬하는 단계.
- 검색에서는 대/소문자를 구분하지 않음. (ex. "pagerank"를 검색할 경우, "PageRank", "Pagerank",

"pagerank" 등을 포함하는 모든 문서를 찾음)

- 먼저 문서들의 TF-IDF score를 구함.
  각 문서의 TF-IDF score는 문서 내 단어들 중 검색어와 일치하는 단어들의 TF-IDF 합과 같음.
- 다음으로 문서들의 PageRank score를 구함.
- 위에서 구한 **두 점수의 곱**을 **내림차순**으로 정렬. **상위 10개를 출력하여 검색 결과로 제공**.
- 두 개 이상 문서들의 순위가 동률인 경우, 문서 id로 오름차순 정렬.

6. Linux console에서 실행 가능한 검색 프로그램 제작
   - 검색 프로그램 처음 실행 시 inverted index 생성과 TF-IDF, PageRank 계산 수행.
   - 아래 A와 C에 설명된 프로그램 실행 입/출력 포맷, 검색 입/출력 포맷을 **필히 지켜야 함**.
   - 단일 단어 검색 (ex. pagerank)
     해당 단어를 포함한 모든 문서들은 검색 결과의 후보가 됨.
   - 복수 단어 검색 (ex. cool pagerank algorithm)
     복수 단어들 중 한 개 이상의 단어를 포함한 모든 문서들은 검색 결과의 후보가 됨.

A. 프로그램 실행 입/출력 포맷

입력	출력
python main.py	[학번]>

B. 프로그램 실행 입/출력 예시

```
i-love-d@tabase:~$ python main.py
building tables…
ready to search
2017-12345>
```

C. 검색 입/출력 포맷

입력	출력
	[문서 id], [문서 title], [TF-IDF score], [PageRank score]

검색어(query)	[문서 id], [문서 title], [TF-IDF score], [PageRank score] … (상위 10개 문서의 id, title, TF-IDF score, PageRank score 출력)

D. 검색 입/출력 예시

```
2017-12345> the president
37619696, Falsonar_Avia, 0.00019545, 0.00055346
6301341, Colina%2C_Chile, 0.00019947, 0.00037922
…
2017-12345>
```

E. 단일 단어 검색 입력 예시

```
2017-12345> mamamoo
```

F. 복수 단어 검색 입력 예시

```
2017-12345> red velvet
```

개발 환경

1. 개발 언어: **Python3**
2. 데이터베이스: MariaDB 10.3 (pymysql 패키지 이용)

IP	Port	Id	Password
s.snu.ac.kr	3306	ADB학번 (ex. ADB2017_12345)	ID와 동일

제출

1. main.py를 포함하는 소스코드와 requirements.txt 파일
   - NLTK, pymysql 이외의 package를 쓴다면 requirements.txt로 명시
   - Linux 환경에서 main.py가 실행되지 않거나 입/출력 포맷을 지키지 않은 경우 감점.
2. 리포트
   - 파일 형식: pdf, docx, hwp만 허용.

- 분량: 10페이지 이내.
- 필수 작성 항목은 다음과 같음. 필요에 따라 항목 추가 가능.

필수 작성 항목
개발 환경
구현 기능 소개 (간략하게)
PageRank 구현 코드 설명
주요 SQL문 설명
프로그램 실행 예시 (화면 캡쳐 포함)
평가 및 결론

3. 제출 방식

- main.py를 포함하는 소스코드, 리포트를 **하나의 파일로 압축**하여 메일로 제출.
- 압축 파일 명: PRJ1_학번_이름.zip (ex. PRJ1_2016-12345_홍길동.zip)
- E-mail 제목: [ADB]PRJ1_학번_이름 (ex. [ADB]PRJ1_2016-12345_홍길동)
- E-mail 주소: lecture@europa.snu.ac.kr
- 제출 기한: **4월 1일 월요일 23시 59분**

평가 항목

항목	배점	비고
Main.py 파일의 정상 실행 여부	10	실행 불가 또는 포맷 미준수 시 감점
프로그램 실행 입/출력 포맷 준수 여부	5	
검색 입/출력 포맷 준수 여부	5	
단일 단어 검색 가능 여부	10	별도의 테스트셋으로 채점
복수 단어 검색 가능 여부	20	
검색 결과 품질 평가	30	
Inverted index 생성 소요 시간	3	

TF-IDF, PageRank 계산 소요 시간	3	장시간 소요 시 감점
검색 소요 시간	4	
리포트	10	
총점	100	

주의 사항

1. 소스코드 카피, 보고서 표절 시 해당 프로젝트 전체 점수 0점
2. 과제 제출 기한을 넘길 시, E-mail 제출 기한을 기준으로 매 24시간 마다 10%씩 감점.
   (ex. 1일 지연 시 10%, 2일 지연 시 20%, 3일 지연 시 30%)
3. 제출 기한으로부터 3일이 넘으면 제출 불가. (4월 4일 23시 59분까지 제출 가능)
4. E-mail 반송, 첨부 파일 누락, 파일 실행 불가 시 마지막 제출일을 기준으로 위 감점 기준 적용.
5. E-mail 제출 시 첨부 파일 용량이 20MB를 초과하는 경우 조교에게 문의.

참고 자료

1. MariaDB
   https://mariadb.com/kb/en/mariadb/getting-started/
2. PyMySql
   https://pymysql.readthedocs.io/en/latest/
3. Mysql Workbench
   https://www.mysql.com/products/workbench/
4. NLTK Tokenizer
   https://www.nltk.org/api/nltk.tokenize.html