대용량 문서 집합에서 유사문서 탐색을 위한 후보 문서 쌍 검색 실험

박선영○, 정우근, 조환규
부산대학교 컴퓨터공학과
parksy@pusan.ac.kr, wkchung@pusan.ac.kr, hgcho@pusan.ac.kr

Experiment of Searching Candidate Text Pair for Searching Similar Texts among Massive Document Repository

Sun-Young Park○, WooKeun Chung, Hwan-Gue Cho
Dept. of Computer Science and Engineering, Pusan National University

요 약

문서 표절과 관련된 이슈가 급증함에 따라 유사 문서 탐색과 관련한 연구가 활발히 진행되고 있다. 특히 인터넷의 발달로 인해 일반 사용자가 수많은 전자 문서에 쉽게 접근할 수 있게 됨에 따라 대용량 문서 집합에 대한 탐색 속도와 정확성의 중요성도 커지고 있다. 대용량 문서 집합 내에서 빠른 시간 내에 유사 문서를 탐색하는 방법에는 전역 사전을 이용하여 후보 문서 쌍(유사할 가능성이 높은 문서의 쌍)를 추출한 후 찾아낸 후보 문서 쌍에만 정밀한 검사를 수행함으로써 검사 시간을 줄이는 방법이 존재한다. 이 때, 후보 문서를 찾아내기 위하여 전역 사전(Global DICtionary, GDIC)이라는 자료 구조를 이용하게 되는데, 이 전역 사전을 효과적으로 사용하면 후보 문서 쌍을 찾아내는 시간을 기존보다 더욱 줄일 수 있다. 본 논문에서는 전역 사전을 더욱 효과적으로 활용하여 후보 문서 쌍 검색 시간을 대폭 줄이는 방법에 대해 기술하며, 어느 정도의 성능 향상이 있는지 실험을 통해 측정하였다. 20,000건의 실험용 말뭉치 자료와 6263건의 실존하는 보고 문서에 대해 실험한 결과, GDIC 생성에서 2.5~4.6%, 후보 문서 쌍 탐색에서 1%~15.4% 정도의 성능이 향상된 것을 확인할 수 있었다. 추후 update query를 최소화하여 GDIC 생성 시간을 추가로 줄이는 방법에 대해 연구할 계획이다.

1. 서 론

최근 사회 각계의 표절 논란으로 인하여 연구, 창작 윤리의 중요성이 커지고 있다. 이에 한국 저작권 위원회는 내부에 표절 문제 전담기구인 '표절 위원회'까지 만든 상황이다[1]. 표절이란, 다른 사람의 저작물의 전부나 일부를 그대로 또는 그 형태나 내용에 다소 변경을 가하여 자신의 것으로 제공 또는 제시하는 행위를 의미한다[2]. 또한, 표절 중에서도 논문 표절, 문학 작품 표절, 보고서 표절 등 가장 빈번하게 발생하는 문서 표절에 대응하기 위하여 유사문서 탐색시스템과 관련한 많은 연구가 이루어지고 있다. 특히 인터넷의 발달과 웹 2.0의 등장, 보고서 매매 사이트 등의 등장으로 인해 일반인의 정보에 대한 접근성이 크게 높아지면서 과거에 비해 표절이 발생했을 가능성이 있는 문서의 범위가 크게 넓어졌다. 즉 유사문서 탐색시스템의 성능 척도 중 대용량 및 대량의 문서 집합 간에 탐색을 수행할 때의 성능과 정확성이 점점 더 중요한 이슈로 떠오를 것으로 예상된다. 본 논문에서는 대용량 문서 집합에서 빠른 성능을 낼 수 있기 위한 전처리 모델에 대해 소개하고, 이 모델에서 유사 문서 쌍을 찾아낼 때 걸리는 시간을 최소화하여 전체 시스템의 성능을 향상시키는 방법을 제안한다.

2. 관련 연구

대용량 문서 집합에서 유사한 문서 쌍을 빠르게 찾기 위한 연구에는 전역 사전(Global DICtionary, GDIC)을 이용한 전처리 모델이 있다[3]. 이 방법은 주어진 전체 문서 집합의 각 문서를 분석하여 문서에 출현하는 모든 단어를 3음절 단위로 분할하여 키(Key)를 생성하고 이 Key의 사전(Dictionary, DIC)을 만든 후, 모든 문서의 Key와 출현 문서 번호를 저장하여 GDIC를 생성한다. 생성된 DIC, GDIC는 DBMS에 저장한다. 이후 전처리 모델을 이용하여 각 문서에 대해 해당 문서와 비슷할 가능성이 있는 문서, 즉 각 문서에 대한 "후보 문서군"을 찾아낸다. 전처리 모델이 후보 문서군을 찾아내는 방법은 다음과 같다.

우선 문서 집합에서 특정 문서의 모든 Key에 대해 GDIC에서 얼마나 많은 문서에 활용되었는지를 검사하여 이 비율이 특정 값 T_{ratio} 이상일 경우, 해당 Key를 불용어로 처리하여 이후의 과정을 수행하지 않는다. 이 방법을 사용하면 '합니다', '습니다' 등과 같이 자주 사용되는

단어들에 대한 검사를 수행하지 않음으로써 전처리에 필요한 시간을 대폭 단축할 수 있으며 문서의 종류에 따라 dynamic한 처리가 가능하다는 장점이 있다. 가령 Java 관련 문서 집합에서는 'class', 'method'와 같은 단어는 자주 등장할 가능성이 크고, 치과 관련 문서라면 '치아', '치료' 등의 단어가 자주 등장할 것인데, 이러한 단어들은 위에 언급한 집합 내에서는 T_{ratio}가 매우 높게 나오므로 반드시 불용어 처리된다. 일반적으로 문서의 개수가 많아질수록 T_{ratio}값을 낮추었을 때 찾아내야 할 유사 문서 쌍을 누락시키지 않으면서 빠른 시간 내에 전처리를 수행할 수 있다. 이전 연구에서 찾은 적절한 값은 문서의 수가 20,000개 내외일 때 0.002~0.005, 6,000개 안팎일 때 0.005~0.010 정도로 측정되었다. 즉 이러한 값을 사용할 경우 후보 문서군을 찾아낼 때 전체 문서 집합 내에서 50~100개 내외의 일부 문서에서만 사용되는 소수의 Key만을 사용하게 되어 전처리 속도가 빨라지게 된다.

각 문서의 불용어가 아닌 Key(유효 Key)에 대해 해당 문서와 다른 문서에서 몇 번 사용되었는지를 검사한다. 각 등장 횟수가 두 번째 환경 변수인 N_{match} 이상이면서, 그 합이 S_{match}보다 크다면 해당 문서 쌍은 후보 문서군에 포함된다. 이전 연구에서 찾아낸 두 변수의 적정 범위는 N_{match}의 경우 2~4, S_{match}의 경우 7~13이다. 예를 들어 N_{match}=3, S_{match}=10으로 설정했을 경우, A문서와 B문서에서 어떤 유효 Key가 각각 5번, 8번 등장했다면 (A,B) 는 후보 문서쌍에 포함되는 것이다.

마지막으로 두 문서 사이의 유효 Key의 일치 비율이 전체 Key의 개수의 일정 비율(C_{ratio}) 이상이면 후보 문서쌍에 포함한다. 이 방법은 문서 집합 내에 길이가 짧은 문서가 포함된 경우 매우 좋은 성능을 낸다. 만약 전체 Key가 500개인 문서와 300개인 문서 중, 유효 Key가 총 100개 정도 일치한다면 두 문서를 후보 문서쌍에 포함하는 것이다. 실험을 통해 찾아낸 C_{ratio}의 적정 범위는 0.02~0.08 정도이다.

이후 찾아낸 문서들과의 정밀 검사를 통하여 유사도를 검사한다. 정밀 검사에는 내용기반 유사문서 탐색 시스템인 DeVAC(Document eVolution Analysis Center)의 탐색 모듈을 사용한다[4]. 이 모듈은 유사문서를 탐색하기 위한 많은 방법 중 'fingerprint' 방식과 'BLAST' 방법을 효과적으로 결합하여 유사 문서를 탐색한다[5][6][7]. 이 방법은 비교적 용량이 큰 문서에 대해서도 1:1 비교의 경우에는 높은 성능을 내며 내용의 일부 변경이나 삽입, 삭제 등을 가하더라도 매우 정확하게 유사 문서를 찾아낸다[8][9]. 즉 최소한의 계산 시간으로 전처리를 수행하여 비교할 문서쌍의 개수만 최소화하면 전체 탐색 시간을 더욱 줄일 수 있다.

3. 전처리 성능 측정

앞선 연구에서 약 6200건 규모의 문서 집합에 대한 실험에서 GDIC 생성 시간과 전처리 시간, 그리고 실제 검사 시간 등의 분포는 <표 1>과 같다. 지면 상 4가지 실험 조건에 대해서만 표를 작성하였지만 실험을 진행한 27회의 모든 조건에서 위와 비슷한 결과가 나왔다. 핵심은 GDIC 생성 시간이 전체 탐색 시간 비중의 50%가 넘는다는 것인데, 이는 GDIC를 한번 생성해 두면 해당 문서 집합에 대해서는 재사용이 가능하다는 점을 고려하더라도 너무 많은 시간을 소요하고 있다고 판단되어 반드시 개선해야 할 부분 중 하나이다. 정밀 검사 역시 적지 않은 시간을 소요하지만, 이 시간을 줄이기 위해서는 환경 변수의 조정을 통해 후보 문서 쌍을 더 줄여야 하고, 후보 문서 쌍을 지나치게 줄이면 sensitivity가 떨어지게 된다. 본 연구에서는 GDIC를 사용한 전처리 시스템에서 DIC 및 문서의 내용 데이터를 DB에 저장하지 않고 파일 시스템에서 그대로 읽어옴으로써 GDIC 생성 시간을 줄인다. 또한 전처리를 수행하는데 걸리는 시간이 10분 이상 걸리는데, 이 또한 전처리 과정을 프로그램 내에서 수행하지 않고 최적화된 Query를 생성하여 이를 통해 전처리 수행 시간을 줄이기 위한 방법을 찾아낸다.

<표 1> 약 6200건의 보고서에 관한 실험 데이터. 단위는 초. 괄호 안은 전체 탐색 시간에 대한 비율(%). GDIC 생성 시간은 모든 조건에 대해 동일하며 전체 탐색 시간에서의 비중을 살펴 보면 공통적으로 GDIC생성 시간이 전체 탐색 시간에서 매우 큰 비중을 차지하고 있음을 알 수 있다.

실험 조건		GDIC 생성 시간	전처리 시간	검사 시간	전체 탐색 시간
T_{ratio}	C_{ratio}				
N_{match}	S_{match}				
0.003	0.02	12473	2189	5103	14,662.02
2	7	(63.1)	(11.1)	(25.8)	(100)
0.003	0.05	12473	733	1414	13,206.05
3	10	(85.3)	(5.0)	(9.6)	(100)
0.004	0.08	12473	781	1825	13,254.08
3	10	(82.7)	(5.1)	(12.1)	(100)
0.005	0.02	12473	938	4052	13,411.02
2	7	(71.4)	(5.4)	(23.2)	(100)

3.1 GDIC 생성 시간 개선 방법

앞서 설명한 바와 같이 GDIC는 입력된 문서 집합의 문자 선별(Select TeXt, STX)데이터와 DIC, 그리고 이를 통해 생성된 GDIC 전부를 포함하고 있으며, 이 모든 데이터가 DB에 저장된다. 이 중 STX는 특수문자 제거와 한자 변환 등의 연산을 통해 순수한 문자열만 선택한 뒤 파일로 저장한 것이고, 이 데이터에서 Key 추출 과정을 통해 파일로 저장한 것이 DIC가 된다. STX와 DIC는 전처리 뿐 아니라 후보 문서 쌍 추출 후 DeVAC 모듈을 이용한 정밀 검사에도 필요한 데이터이기 때문에 GDIC 구성 전에 미리 생성하는데, 일반적으로 STX파일이 원본 문서 파일(TXT)의 70~80%, DIC파일이 150~200% 정도라는 점을 고려하면, 문서의 크기가 커질수록 디스크 접근으로 인한 시간 비용도 커지게 된다.

현재까지는 GDIC를 생성할 때 데이터의 무결성을 고려하여 STX와 DIC 데이터를 함께 저장했지만 파일 경로를 저장하고 각 파일에 쓰기 제한과 함께 접근 제어를 걸어둠으로써 GDIC 생성 시 STX와 DIC를 이중으로 복사하지 않고도 시스템을 정상적으로 가동할 수 있다. 이

경우 불필요한 디스크 접근이 발생하지 않으므로, 전체 STX와 DIC의 용량에 비례하여 GDIC 생성 시간도 단축된다.

3.2 후보 문서 쌍 검색 속도 개선 방법

앞서 2절에서 설명한 전처리 과정에 대한 의사코드는 <표 2>와 같다.

<표 2> 문서 하나에 대한 전처리 과정 의사 코드. 한 문서에 포함된 모든 Key에 대해 불용어 처리 과정을 거친 후, N_{match}, S_{match}를 이용해 후보 문서 쌍을 찾아내고, C_{ratio}를 이용해 추가로 후보 문서 쌍을 찾아낸다.

```
DIC = readDB_DIC(document_name);

matchCount=0; // KEY match count in documents

for(Key in DIC)
{
    if(Key.usedcount/DocRepository.size > Tratio)
        countiue;  /* Do not use the KEY(skip) */

    /* read same keys in GDIC */
    candidateKey = readDB_GDIC(Key);

    /* for preprocessing method that using C_ratio */
    matchCount+=min(Key.count, cKey.count);

    for(cKey in candidateKey)
    {
        if(Key.count > Nmatch &
          cKey.count > Nmatch &
          Key.count+cKey.count > Smatch)
        {
          cadidatePair.append(Key.getDocument(),
                              cKey.getDocument() );
        }
    }
}

/* Preprocessing that using C_ratio. Append when
condition satisfied */
if(matchCount / min(Key.getDocument().Keysize(),
                    cKey.getDocument().Keysize() ) )
    cadidatePair.append(Key.getDocument(),
                              cKey.getDocument() );
```

위의 방식대로 전처리를 진행할 경우, 성능적 측면에서 몇 가지 개선할 여지를 찾을 수 있다. 이 방식은 Key를 이용하여 데이터베이스에 접근한 후, 후보가 될 수 있는 다른 Key들과 count를 일일이 비교한 후 후보 문서쌍에 해당하는 문서들을 찾아낸다. 한 문서에 대해서는 그 문서에 포함된 Key에 대해서만 연산을 수행하기 때문에 많은 시간이 필요하지 않으나, 문서쌍 전체에 대해 전처리를 수행하게 되면 문서쌍에 포함된 문서의 개수만큼 위의 절차대로 진행하게 된다. 문서의 개수를 N, 한 문서에 포함된 Key의 개수를 K라고 했을 때 $O(N \cdot K^2)$을 따르게 되어 많은 시간을 소요하게 된다. 이 부분에 대한 최적화를 위하여 DBMS를 최대한 활용할 수 있다. 즉 최적화된 SQL Query를 작성하여 전체 문서 집합에서 검사가 필요한 후보 문서군을 한 번에 뽑아내는 것이다. 위 소스코드의 출력은 cadidatePair 인데, 개선된 방법에서는 위와 동일한 출력을 내기 위해 우선 <표 3>과 SQL Query를 수행한다.

<표 3> 문서 집합 전체에서 후보 문서 군 추출 시간을 단축하기 위한 SQL Query. 출력은 불용어가 처리와 N_{match}가 동시에 적용된 결과가 출력된다.

```
SELECT
        dic1.*
FROM
        (SELECT idx
         FROM dic_count
         WHERE (sum_loc/6263)>0.003 ) as count1,
        (SELECT loc, word_idx
         FROM dic
         WHERE cnt>3) dic1
WHERE
        count1.idx=dic1.word_idx;
```

위의 Query를 통해 Key의 번호와 해당 Key가 존재하는 문서의 index가 추출되면 Key별로 S_{match} 를 적용하여 조건을 만족하는 모든 문서 조합에 대해 중복을 제거하고 후보 문서 쌍에 등록하면 된다.

5. 실 험

5.1 실험 데이터

실험에 사용한 데이터는 '21세기 세종 계획'에서 제공하는 '연구/교육용 1,000만 어절 현대국어 균형 말뭉치'[10]를 200~700어절 정도의 19,867개의 파일로 분할한 문서 집합과 1999~2000년 정부의 정책 연구와 관련한 보고용 문서 6263건을 사용하였다. 말뭉치 데이터의 총 용량은 277.70MB, 보고서 데이터의 총 용량은 1494.32MB이다.

5.2 GDIC 생성 시간 측정

GDIC 생성 시간 측정 실험은 각 데이터를 10개로 분할하여 기존 방법과 STX, DIC를 경로만 저장하는 방법에 대해 각각 2회씩 측정한 후 그 평균을 구하였다. 디스크 접근 시간을 줄임으로써 GDIC 생성 시간이 크게 줄어들 것으로 예상했으나, 실험 결과는 생성 시간 2.5~4.6%정도 감소로 예상보다 미미한 성능 향상을 보여주었다. 이는 GDIC 생성 시간 중 STX와 DIC 데이터에 대한 디스크 접근 시간이 차지하는 비중이 크지 않다는 것을 의미한다. GDIC 생성 시간을 세부적으로 검토해 본 결과 GDIC 데이터 중 Key의 count 값을 저장할 때 사용되는 update query문을 수행할 때 많은 시간이 소요되었다. 이 부분에 대한 추가적인 조사 및 개선이 필요하다고 판단된다.

<표 4> GDIC 생성 시 STX, DIC 데이터 전체를 저장한 경우(기존)와 경로만 저장한 경우(개선) 생성 시간 비교표. 약 2.5~4.6% 정도 성능이 향상되었으며 이는 STX, DIC 데이터에 대한 디스크 접근 시간에 해당하는 것이다.

그룹	말뭉치 데이터		보고서 데이터	
	기존 방법	개선된 방법	기존 방법	개선된 방법
1	621.2	601.1	1274.2	1243.1
2	598.8	582.4	1321.3	1264.1
3	587.1	577.7	1127.7	1091.7
4	631.4	612.3	1216.8	1174.0
5	611.3	600.5	1339.7	1291.3
6	617.0	602.3	1262.6	1135.6
7	553.2	545.5	1294.4	1221.8
8	628.9	603.0	1207.4	1158.2
9	609.5	596.1	1187.0	1137.3
10	661.7	649.8	1248.2	1190.0
합계	6120.1	5970.7	12479.3	11907.1
기존 대비 소요 시간	97.5%		95.4%	

5.3 후보 문서 쌍 검색 속도 측정

전처리 시간 측정에는 이미 언급한 보고서 데이터를 사용하였다. <표 1>의 실험 조건에 대해 개선된 방법을 적용하여 전처리 시간의 변화를 측정하였다. 전처리 결과가 동일하기 때문에 검사시간은 거의 동일할 것으로 예측하였고, 이를 검증하기 위해 검사 시간도 측정하였다.

<표 4> 후보 문서 쌍 추출 시 최적화된 query를 사용하여 데이터를 추출하였을 때 전처리 시간과 검사 시간 비교. 전처리 결과가 같으므로 검사 시간은 동일하여야 함. 전처리 시간의 경우, 기존 전처리 시간이 비교적 길게 측정되었던 첫 번째 실험의 경우과 마지막 실험은 각각 14.6%, 8.7%정도 전처리 시간이 단축되었으나 나머지 두 경우에 대해서는 1~3% 정도로 성능 향상 폭이 비교적 작았다. 검사 시간은 거의 동일하게 측정되었다.

실험 조건		전처리 시간		검사 시간	
T_{ratio} N_{match}	C_{ratio} S_{match}	기존 방법	개선된 방법	기존 방법	개선된 방법
0.003	0.02	2189	1871	5103	5090
2	7				
0.003	0.05	733	711	1414	1433
3	10				
0.004	0.08	781	776	1825	1811
3	10				
0.005	0.02	938	869	4052	4012
2	7				

실험 결과 첫 번째 실험 조건과 마지막 실험 조건에 대해서는 각각 14.6%, 8.7% 정도의 전처리 성능 향상을 보였으나 나머지 두 실험 조건에 대해서는 1~3%정도로 비교적 작은 수준의 성능 향상이 있었다. 최적화 query에서 두 실험 조건의 가장 큰 차이는 N_{match} 이다. 최적화 query가 가장 효율적으로 적용되는 변수 값에 대한 연구도 추가로 필요할 것으로 생각된다.

6. 결론 및 추후 연구

유사문서 탐색 시스템에서 대용량 문서 집합에 대응하기 위하여 만든 전처리 모델의 연산 시간의 분포를 분석한 결과 GDIC 생성 시간과 후보 문서 쌍 추출 연산에 대해 개선할 여지를 찾을 수 있었다. GDIC 생성 시 STX, DIC의 경우 파일 경로만 저장함으로써 생성 시간을 2.5~4.6% 단축할 수 있었고, 후보 문서 쌍 추출 연산을 소스코드 대신 최적화된 SQL문으로 처리한 결과 1.0~15.4% 정도 단축할 수 있었다. 실험 결과에서 예상보다 성능 향상 정도가 적었던 부분에 대해 분석하였고, 이에 따라 추후에는 GDIC를 생성 할 때 사용하는 update query를 최소화하여 GDIC 생성 시간을 더욱 줄이는 방법에 대한 연구와 최적화 query의 환경 변수와의 연관성에 대해서도 연구하여 개선점을 찾을 것이다. 또한, 비표절 대량 문서 집합과 표절 발생 문서 집합 간의 유사도 분포를 조사하여 유사도에 따른 표절 발생 확률 모델에 대한 연구를 진행할 계획이다.

참고문헌(References)

[1] 한국 저작권 위원회, http://www.copyright.or.kr/

[2] 특허청, http://www.kipo.go.kr/

[3] 박선영, 김지훈, 김선영, 김형준, 조환규, 대용량 문서 집합에서 유사 문서 탐색을 위한 효과적인 전처리 시스템의 설계, 한국정보과학회 2009 가을 학술발표논문집 제36권 제2호(A), pp. 76-77, 2009.

[4] 류창건, 김형준, 박수현, 조환규, DeVAC(Document eVolution Analyzing Center), http://devac.cs.pusan.ac.kr/

[5] J. L. Donaldson, A. Lancaster, and P. H. Sposato, A plagiarism detection system, In Proceedings of the Twelfth SIGCSE Technical Symposium on Computer Science Education, pp. 21-25. 1981.

[6] Schleimer, S., Wilkerson, D. S., and Aiken, A. Winnowing: local algorithms for document fingerprinting. In Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data. 76-85. June 09-12, 2003.

[7] Cameron, M., Williams, H. E., and Cannane, A. Improved Gapped Alignment in BLAST. IEEE/ACM Trans. Comput. Biol. Bioinformatics 1, 3, 116-129. Jul. 2004.

[8] 류창건, 김형준, 조환규, 한글 말뭉치를 이용한 한글 표절 탐색 모델 개발, 정보과학회논문지: 컴퓨팅의 실제 및 레터, 제14권, 제2호, pp. 231-235, 2008.

[9] 김형준, 조환규, 정렬을 이용한 내용기반 문서탐색 시스템의 전처리 과정 개선, 2008 한국컴퓨터종합학술대회 논문집, 제35호, 제1권(C), pp. 354-358, 2008.

[10] 21세기 세종 계획, http://www.sejong.or.kr/