대용량 문서 집합 간 전처리 모듈을 이용한 DeVAS의 확장

Extension of DeVAS using Preprocessing Module for Finding Similar Texts in Massive Document Repository

박선영
Park Sun-Young
부산대학교 컴퓨터공학과
parksy@pusan.ac.kr

ABSTRACT

본 보고서에서는 DeVAC에서 문서 간 유사도 탐색 기능을 위한 핵심 모듈 DeVAS에 대용량 문서 집합간 필터링 기능을 확장한 DeVAS Huge-edition(DeVASH)에 대해 논의한다. DeVASH에서 전처리를 위해 사용한 방법인 불용어 처리, 동시 등장 횟수 활용, 동시 등장 비율 활용 등에 대해 자세히 설명한다. 또한 각 환경변수와 실험에 대해 기술하고, 전처리 모듈을 적용하기 위한 실행 방법에 대해 자세히 설명한다. 또한 DeVAS와 DeVASH 프로그램의 전체 구조, 환경변수, 수행 시간 등을 비교하여 정리하였다. 끝으로 현재의 DeVASH의 한계점과 그 해결 방안, 앞으로 개선이 필요한 부분에 대해서도 기술하였다.

KEYWORDS Plagiarism, Similar Document, DeVAC

1 서 론

최근 표절이 사회적 문제가 됨에 따라 문서 간 유사도를 탐색하는 시스템의 필요성이 대두되었다. 유사도 탐색 시스템에서는 유사도 측정의 정확성과 탐색 시간이 가장 중요한 성능요소라고 할 수 있다. 유사도 탐색을 위한 여러 효과적인 방법 중 Attribute Counting 방법[?]과 Structured Metric 방법이 있는데[1], 이 두 방법의 장점을 혼합한 현재의 DeVAC[2]에서 사용되는 비교 모듈인 DeVAS는 1MB

문서의 수	문서쌍의 수	소요 시간(s)
10	45	1
100	4950	46
500	124750	1612
1000	499500	5521
5000	12497500	-

표 1. 입력 문서의 개수에 따른 검사해야할 문서쌍의 개수 변화와 검색 소요 시간. 문서쌍이 증가함에 따라 검사할 문서쌍의 개수와 예상 소요시간이 급격하게 증가하는 것을 알 수 있다.

내외의 비교적 대용량의 문서 간이라 할지라도 1 대 1 비교에는 빠른 시간 내에 유사도 측정이 가능하다. 하지만, 3 개 이상의 문서로 이루어진 문서 집합의 경우 존재하는 모든 pair 에 대한 검사를 수행하기 때문에 시간에 대한 복잡도가 $O(n^2)$ 을 따르게 되어 기하급수적으로 많은 계산 시간을 요구하게 된다. 예를 들어 한글 문서 5 천 개에 대하여 검사를 수행할 경우, 총 1250 만 개의 문서 쌍에 대하여 검사를 수행해야 하므로, 한 쌍에 10ms 가 걸린다고 가정해도 모두 다 수행하기 위해 40 시간이 넘는 시간이 소요되게 된다. 이를 해결하기 위하여 저자가 '대용량 문서 집합에서 유사 문서 탐색을 위한 효과적인 전처리 시스템의 설계'에서 제안하였던 방법을 DeVAS 에 적용하고자 한다. 이 방법을 사용하면 시스템 전체의 유사도 탐색 정확도를 최대한 유지하면서 동시에 계산 시간을 크게 줄이는 것이 가능하다.

2 전처리 방법

2.1 성능 요소

전처리 모델의 성능 요소는 Specificity, Sensitivity, 그리고 시간 비용을 들 수 있다. Specificity 는 시스템이 필터링 한 후 문서의 총 개수 중 실제 유사 문서 수의 비율로, 높을수록 적은 개수의 후보 중 정답이 많이 포함된 상태이므로 짧은 시간 내에 효율적으로 탐색을 수행할 준비가 되었다는 뜻이다. Sensitivity 는 필터링 후 결과에 포함된 유사 문서의 개수를 전체 문서 중 유사문서의 수로 나눈 것으로, Sensitivity 가 100% 가 아니라면 필터링 후 정답이 누락된 것을 의미한다. 따라서 필터링을 수행할 때에는 Sensitivity 를 최우선으로 고려하여야 하고, Specificity 는 높을수록 시스템 성능이 향상되므로 이 또한 고려 대상이 된다. 시간 비용은 전처리 완료 후 후보로 등록된 문서의 개수와 비례하여 증가한다. 여기에 전처리 자체에 소요되는 시간을 더하면 구할 수 있다. 위에서 언급한 것처럼 Specificity 가 높으면 시간 비용은 감소한다.

2.2 전역 사전 (Global Dictionary)

DeVASH 에서 전처리를 위해 전역 사전 (Global DICtionary, GDIC) 을 사용한다. 이 사전은 입력되는 모든 문서들의 key 사전 (DIC 파일) 를 분석하여 각 key 에 대한 index 를 생성하며, 해당 key 가 각 문서에서 몇 번 등장하는지만 기록한다. 사전을 관리하기 위해서 mySQL DBMS 를 사용하고 있다. GDIC 가 완성되면 전처리 기능을 사용할 수 있다.

2.3 불용어 처리

DeVASH 에서는 여러 개의 입력 문서에 대한 검사 후보 문서의 수를 최소화하기 위하여 각각의 문서마다 전처리를 수행한다. 또한 전처리 과정을 수행하기 전에 불용어 처리 과정을 우선 수행한다. 즉 원본 문서 A 에 존재하는 키 중 GDIC 에서 일정 비율 이상 사용된 키는 불용어로 처리하여 이후 과정에서 사용하지 않음으로써 무의미한 부분이 비슷한 문서가 후보 문서로 등록되는 것을 최소화한다. 예를들어 '그', '이', '것이다' 등 사용 빈도는 높지만 유사도 측정과는 큰 관련이 없는 단어들을 전처리 과정에서 제거하는 것이다. 이를 통해 전처리 성능을 크게 끌어올릴 수 있다.

T_{ratio}	문서의 수	정확도(%)
1.0000	20000	100.00
0.2000	719	100.00
0.1000	407	100.00
0.0100	53	100.00
0.0050	31	100.00
0.0040	24	100.00
0.0030	16	100.00
0.0020	10	100.00
0.0010	6	88.00
0.0005	1	12.00

표 2. T_{ratio}에 따른 전처리 결과와 정확도. T_{ratio}가 낮을수록 전처리 후 문서 개수가 감소하고 수행 시간도 적지만, 0.0020 미만으로 내려갈 경우 정확도가 급격히 하락한다.

실험 결과 특정 단어가 GDIC에서 사용된 비율 T_{ratio} 가 0.0020 정도까지 내려가더라도, 즉 0.2% 정도의 필수적인 단어만 후보 문서 등록 과정에 사용하여도 전처리 정확도는 유지되는 것으로 나타났다. Specificity가 가장 높고 Sensitivity가 100%로 유지되는, 가장 효과적인 T_{ratio}의 범위는 0.0020 $\sim$ 0.0050 정도로 볼 수 있다.

위에서 언급한 방법을 사용해서 불용어를 처리하게 되면 불용어로 판단되는 단어는 DB에 저장된 문서의 주제에 따라 다르게 나타난다. 이 방법은 불용어 사전을 만들기 위한 시간과 노력을 필요로 하지 않는다는 장점이 있다. 또한 비슷한 주제의 문서에서 반복적으로 사용될 수 있는 단어들을 동적으로 찾아내어 불용어로 처리하는 효과를 갖는다. 가령 치의예과 학생들의 보고서에서 "치아", "충치" 등의 단어는 유사성 검사에서 별 의미를 갖지 못하는데, 위에서 제시한 방법을 사용하면 위와 같은 단어들은 자동적으로 불용어 처리된다. 이 방법의 단점은 DB에 저장된 문서의 개수가 너무 적을 경우 불용어 처리 효과가 떨어진다는 것인데, 문서의 개수가 적다면 굳이 전처리를 하지 않고 모두 검사하면 되므로 문제가 되지 않는다.

2.4 match count 측정

불용어를 제거한 후 남은 key 중에서, GDIC 안에 일정 횟수 이상 동시에 등장하는 key가 존재한다면, 그 key를 가지고 있는 문서는 정밀한 검사를 할 필요성이 있다고 판단할 수 있다. 즉 후보 문서로 등록할 수 있다. 다만 한 쪽 문서에서만 여러 번 등장하는 단어가 후보 문서로 등록되는 것을 방지하기 위하여 원본 A, 대상 B 문서 각각에 N_{match} 번, 그리고 A, B 문서를 더해서 S_{match} 번 이상 등장해야만 후보 문서로 등록하도록 하였다.

여러 T_{ratio}에 대해 N_{match}, 그리고 S_{match}를 달리하면서 전처리 후보 문서의 개수와 전처리 정확도를 측정해 본 결과, 가장 효과적인 N_{match}는 2 $\sim$ 4, 그리고 S_{match}는 7 $\sim$ 12 정도로 판단되었다.

N_{match}	1	2	3	4	5
S_{match}	4	7	10	12	15
T_{ratio}					
0.001	10.6(100)	6.2(90)	5.4(90)	4.4(90)	3.8(68)
0.005	90.0(100)	24.4(100)	16.4(100)	13.0(100)	9.0(88)
0.010	183.4(100)	38.2(100)	20.0(100)	15.4(100)	10.2(100)
0.020	351.6(100)	60.2(100)	26.4(100)	18.0(100)	12.6(100)
0.030	497.8(100)	95.6(100)	44.4(100)	30.0(100)	23.6(100)
0.050	994.6(100)	226.0(100)	86.0(100)	49.0(100)	24.8(100)
0.100	1584.6(100)	286.0(100)	92.2(100)	49.4(100)	24.8(100)
0.200	2952.8(100)	393.0(100)	130.4(100)	74.8(100)	44.0(100)

표 3. N_{match}, S_{match} 에 따른 전처리 후 문서의 개수. N_{match}, S_{match} 가 클수록 전처리 후 문서의 개수가 감소한다. 괄호 안은 Sensitivity 를 나타낸 것으로 N_{match}, S_{match} 가 너무 크면 유사 문서임에도 불구하고 전처리 결과에 포함되지 않는 경우가 발생한다.

2.5 match ratio 측정

두 번째 방법은 불용어를 제거한 key 중 입력 문서와 각 대상 문서 간 key 일치 비율을 검사하여 C_{ratio} 이상일 경우 해당 문서를 후보 문서에 등록한다. 이 방법은 크기가 매우 작으면서 유사도가 높은 문서쌍에 대해 효과적으로 작용한다. T_{ratio} 와 C_{ratio} 를 달리하여 얻은 실험 결과는 다음과 같다.

T_{ratio}	0.001	0.002	0.003	0.004	0.005	0.010
C_{ratio}						
0.005	40.2	108.6	205.2	301.0	442.8	1095.0
0.010	13.4	28.2	49.2	64.4	90.2	240.4
0.020	7.8	12.0	21.0	22.6	26.8	46.6
0.030	6.6	10.0	14.6	15.4	16.0	23.4
0.040	6.6	8.8	12.2	13.4	14.0	17.4
0.050	6.4	6.8	11.2	11.8	12.0	14.6
0.100	5.8	6.0	7.2	8.4	9.4	10.4
0.200	5.4	5.8	6.0	6.0	6.0	6.2

표 4. C_{ratio} 에 따른 전처리 후 문서 개수. C_{ratio} 가 높을수록 전처리 후 문서 개수가 감소한다.

DeVASH 에서는 위에서 언급한 방법들을 한꺼번에 사용하여 전처리 성능을 최대화한다. 성능 측정을 위해 사용한 변수들은 모두 전달인자 파일에 기록되어 있으므로 이것을 수정하면 전처리 환경을 수정할 수 있다.

3 DeVASH 사용법

DeVASH를 사용하기 위해서는 크게 두 가지 작업을 수행하여야 한다. 첫 번째는 GDIC를 생성하는 것이고, 두 번째는 검사를 수행하는 것이다. 이 작업은 아직 자동화가 되지 않아서 DeVAC 시스템의 도움을 받아 수동으로 수행하여야 한다.

3.1 GDIC 생성

GDIC를 생성하기 위해 우선 GDIC에 저장할 파일들을 DeVAC Manager로 변환하여 dvc파일을 생성하여야 한다. dvc생성이 끝나면 이 파일을 DeVAC 홈페이지의 표절 검사 메뉴를 통해 업로드한다. 여기서 업로드만 할 뿐 검사는 하지 않아도 된다. 검사 창에서 파일을 선택하여 업로드 버튼을 클릭하고 잠시 후 파일의 목록이 표시되면 업로드가 완료된 상황이므로, 이 때 종료하면 된다(물론 검사가 필요하다면 검사를 수행하여도 된다). 업로드가 완료되면 Linux 서버에 접속하여 업로드된 폴더명을 알아내어야 한다. 디렉토리 이름은 날짜를 포함한 업로드 시각으로 정해지므로(예 : 100226123040) 서버의 $var/www/html/user/($ 사용자계정명$)/spatial_group$ 에 들어간 후 최신의 시각을 찾으면 된다. 해당 디렉토리의 full path를 포함하여 filelist.txt 의 경로명을 저장해 둔다.
(예 : $var/www/html/user/admin/spatial\_group/100226123040/filelist.txt$)
이후 GDIC 생성 모듈을 실행하여야 한다. 생성 모듈은 총 4개로 이루어져 있다. 신규 데이터를 위한 데이터베이스 및 Table 생성 모듈, STX, DIC 저장 모듈, GDIC 정보 생성 모듈, 카운트 모듈이다. GDIC를 생성하기 위해서는 이후 언급할 순서대로 작업을 수행하면 된다. 데이터베이스 및 Table 생성 모듈을 실행하면서 데이터베이스 이름을 넘겨주면 해당하는 데이터베이스와 테이블이 모두 생성된다. 이후 위에서 저장해 둔 filelist.txt의 full path와 앞서 만든 데이터베이스명을 STX, DIC 저장 모듈에 전달하면 STX, DIC 정보가 모두 DB에 저장된다. 그 상태에서 데이터베이스명을 GDIC 정보 생성 모듈에 전달하면 GDIC 정보가 생성되고, 카운트 모듈에 전달하면 key 카운트가 수행된다.

3.2 유사도 검사 수행

GDIC가 완성되었다면, 유사도 검사를 수행할 수 있다. DB내부 검사와 DB 대 file 검사를 수행할 수 있는데 두 버전이 따로 컴파일되어 있으며, 실행 방법도 조금 차이가 있다. DB내부 검사는 mySQL 내에서의 데이터베이스 명을 전달인자로 받고, DB대 file 검사는 데이터베이스 명 뿐 아니라 file list를 추가로 받는다. 프로세스가 시작되면 우선 선택된 DB 내의 모든 데이터를 메모리에 올린 후 전처리 수행 및 검사, 결과 파일 생성까지 한번에 실행된다. 결과 파일은 DeVAS의 양식과 완전히 동일하므로 DeVAC Manager로 쉽게 확인할 수 있다.

4 DeVAS와 DeVASH 비교

4.1 프로그램 전체 구조

DeVASH는 DeVAS의 extension이므로 DeVAS에 존재하는 모든 기능을 가지고 있다. 여기에 추가로 GDIC 생성 모듈과 DB 접근 모듈, 전처리 모듈이 추가되어 있다. 앞서 언급한 대로 출력 파일의 포멧 역시 완전히 동일하다. 단 기존의 DeVAS와 전달 인자가 다르기 때문에 DeVAS와 다른 위치에 저장되어 있다.

4.2 전달 인자

DeVAS의 경우 전달인자가 두 개이다. 하나는 비교할 파일의 목록이고, 하나는 환경변수 목록이다. 전달된 파일 목록을 읽어 해당하는 파일의 내용을 모두 가져와서, 환경 변수에 따라 유사도 검사를 수행한다. DeVASH의 경우는 DB 버전은 환경변수 목록과 비교할 DB이름을 전달인자로 받고, DB와 file 비교 버전은 환경변수 목록, 비교할 DB이름, 비교할 파일 이름을 전달받는다. 전처리를 위해 추가된 환경 변수는 지정된 파일(filter.cfg)에서 불러들이므로 이 파일만 수정하면 전처리 환경을 수정할 수 있다.

4.3 수행 시간

DeVASH는 DeVAS에 비해 검사에 소요되는 시간이 일반적으로 10%, 적절한 필터링을 거칠 경우 1% 미만으로 매우 짧은 시간 내에 검사가 가능하다. 단 전처리를 위해 GDIC를 생성하고 이를 메모리에 적재하는 시간이 추가로 소요되기 때문에 전체 수행 시간이 1%까지 떨어지지는 않는다. 데이터에 따라 차이는 있지만 GDIC의 생성 시간과 메모리 적재 시간이 검사 시간을 초과하는 경우도 발생한다.

5 실험

5.1 실험 방법

실험에 사용된 데이터는 1999 $\sim$ 2009년의 정부 정책 연구와 관련한 보고용 문서 6263건의 문서(총 1.52GB)을 사용하였다. 전처리를 수행하지 않고 유사도 탐색을 수행했을 경우의 연산 시간을 구하기 위하여, 6263건을 10개의 그룹으로 나누어 각각에 대해 탐색 시간을 측정한 후 6263건에 대한 전체 탐색 시간을 추정하였다. 전처리 수행의 경우 $T_{ratio} = 0.005$, $N_{match} = 2$, $S_{match} = 7$, $C_{ratio} = 0.8$ 일 때 전처리 및 탐색에 필요한 모든 시간을 측정하였으며, 두 경우의 시간과 유사도 500 이상의 문서쌍의 개수를 전처리를 수행하지 않은 경우와 비교하였다.

5.2 실험 결과

전처리를 수행하지 않았을 경우, 6263건에 대해 144886.9초(40.2시간)의 시간이 걸릴 것으로 계산되었으며, 유사도 500 이상의 문서쌍은 1303개 정도로 추산되었다. 전처리를 수행하였을 경우, 6263건

에 대하여 전역 사전을 생성하는데 12473.2초, 사전을 메모리에 적재하는데 151초, 전처리 및 탐색에 6054초 등 합계 18679초(5.2시간)이 소요되었고, 유사도 500 이상의 문서쌍은 1147개를 찾아냈다.

	수행 시간(초)	유사 문서쌍(개)
전처리 미 수행	144886.9	1303
전처리 수행	18679.0	1147
비율	12.9%	88.0%

표 5. 전처리 여부에 따른 수행 시간 및 탐색한 유사도 500 이상의 문서쌍의 개수. 전처리 미 수행의 경우 연산을 통한 추정치이다. 수행 시간의 경우 전처리 미 수행시의 12.9% 정도의 연산 시간으로 탐색이 완료되는 것을 확인할 수 있다. 찾아낸 문서쌍의 경우, 88.0% 정도의 신뢰도를 보여준다.

6 DeVASH의 한계와 개선점

DeVASH와 DeVAS 모두 크기가 큰 입력 파일에 대해서 계산에 필요한 시간이 급격히 증가한다. 일반적으로 10만 단어를 초과하는 입력파일에 대해서 성능이 크게 떨어지는데, 이것은 입력 파일을 적절한 크기로 쪼개어 검사를 수행함으로써 해결할 수 있을 것으로 보인다. 가장 좋은 성능을 보여주는 파일의 크기는 큰 파일들을 여러 크기로 쪼개어 실험을 진행할 계획이다. 또한 DeVASH에서 전처리를 위해 사용한 환경변수들은 모두 실험적으로 적정값을 얻었는데, 더욱 많은 실험을 통해 최적화된 값을 찾아내거나, 혹은 이를 이론적으로 증명할 수 있다면 전처리 성능을 더욱 높일 수 있을 것이다. 앞서 언급한대로 DeVASH는 전처리 과정을 위하여 GDIC를 사용하는데, GDIC의 생성 시간이 프로그램 전체 수행 시간의 30 ∼ 70% 정도로 상당 부분을 차지한다. 따라서 GDIC 생성 시간에 대한 정확한 측정과 최적화가 이루어진다면 전체 수행 시간을 크게 단축할 수 있을 것으로 생각된다. 또한 GDIC를 생성하는 모듈이 분리되어 있어 실행을 할 때 프로그램을 순차적으로 하나씩 수행해주어야 하고 전달인자도 각각 입력해 주어야 하는 불편함이 있다. GDIC 생성 모듈을 하나로 통합하고 검사 모듈과도 완전히 통합하여 DeVAC에서도 전달인자의 변경만으로 손쉽게 DeVASH를 수행할 수 있도록 수정보완해야 할 것으로 생각된다. 그리고 현재 DB 내부 검사 기능 중에서 서로 다른 DB 간 검사 기능이 없는데, 이를 추가하면 DeVASH의 활용도가 조금 더 높아질 것으로 기대된다.

7 결론

DeVAS에 전처리 모듈을 적용한 DeVASH에 대해 설명하였다. 전처리를 위해 사용한 방법은 불용어 처리, match count, match ratio 등을 동시에 적용하여 최대한 전처리 성능을 높이고자 하였다. 실험 결과 적절한 변수값을 사용하면 대용량 문서 집합에 대한 유사도 검사의 정확도를 80% 이상 유지하면서 동시에 전체 수행 시간을 20% 이하로 크게 단축시킬 수 있다. 이 부분에 대해서는 정확도를 더 높일 수 있는 변수 범위를 찾아낼 필요가 있다. DeVASH는 현재 GDIC의 생성과 유사도 검사를

따로 수행하게 되어 있으며, 각 모듈을 순차적으로 수행하여야 정확한 결과를 얻을 수 있는 상태이다. DeVASH의 전체 구조는 DeVAC에서 GDIC 생성 모듈과 DB 접근 모듈, 전처리 모듈이 추가되어 있으며 기능의 추가에 따라 전달인자가 변경되었다. 이로 인해 전체 수행 시간이 크게 단축되나 전처리를 위해 추가된 시간은 앞으로 개선을 통해 줄여나가야 할 부분이라 생각된다. 또한 DeVAS와 DeVASH 모두 10만 단어 이상의 큰 크기를 가진 파일에 대한 검사에 대해 약점을 보이는데 이 부분도 개선하여야 한다.

참고 문헌

1. A. Apostolico, "The myriad virtues of subword trees," *Combinatorial Algorithms on Words*, vol. 37, no. 3, pp. 85–96, 1985.
2. 류창건, 김형준, 조환규, "한글 말뭉치를 이용한 한글 표절 탐색 모델 개발," in *Proc. of the KIISE*, 2008, vol. 14, pp. 231–235.
3. J. L. Donaldson, A. Lancaster, and P.H. Sposato, "A plagiarism detection system," in *Proc. of the Twelfth SIGCSE Technical Symposium on Computer Science Education*, 1981, pp. 21–25.