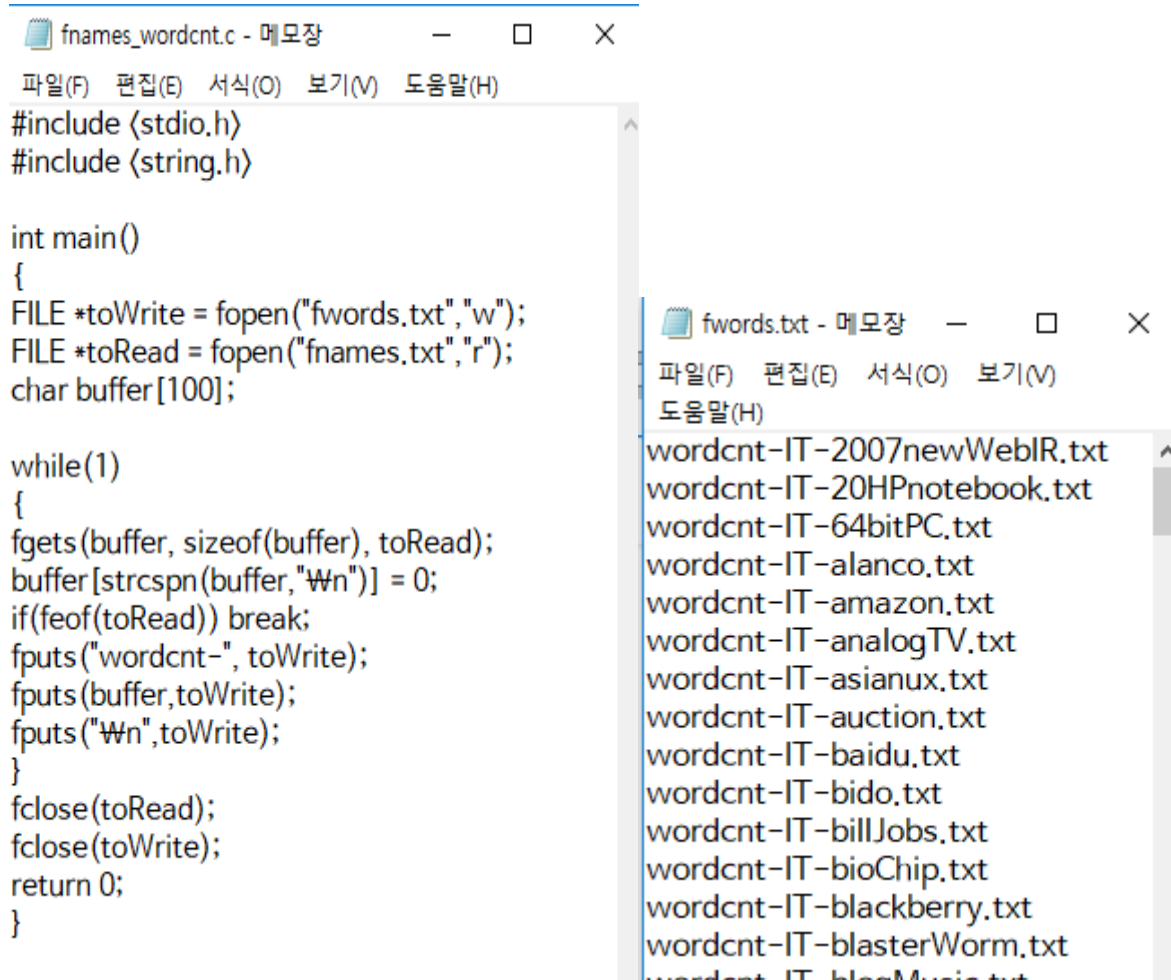


20163170 컴퓨터공학부 최은주 정보검색과 데이터마이닝 문서 유사도 계산 과제

이전 과정은 검색엔진 구현 1차 과제에서 사용하였던 파일들을 이어서 사용하였습니다.

1) weight계산을 위한 TF 받아오기



```
fnames_wordcnt.c - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
#include <stdio.h>
#include <string.h>

int main()
{
FILE *toWrite = fopen("fwords.txt", "w");
FILE *toRead = fopen("fnames.txt", "r");
char buffer[100];

while(1)
{
fgets(buffer, sizeof(buffer), toRead);
buffer[strcspn(buffer, "\n")] = 0;
if(feof(toRead)) break;
fputs("wordcnt-", toWrite);
fputs(buffer, toWrite);
fputs("\n", toWrite);
}
fclose(toRead);
fclose(toWrite);
return 0;
}

fwords.txt - 메모장
파일(F) 편집(E) 서식(O) 보기(V)
도움말(H)
wordcnt-IT-2007newWebIR.txt
wordcnt-IT-20HPnotebook.txt
wordcnt-IT-64bitPC.txt
wordcnt-IT-alanco.txt
wordcnt-IT-amazon.txt
wordcnt-IT-analogTV.txt
wordcnt-IT-asianux.txt
wordcnt-IT-auction.txt
wordcnt-IT-baidu.txt
wordcnt-IT-bido.txt
wordcnt-IT-billJobs.txt
wordcnt-IT-bioChip.txt
wordcnt-IT-blackberry.txt
wordcnt-IT-blasterWorm.txt
wordcnt-IT-blockMusic.txt
```

위 코드로 배치파일을 만들어 Fnames.txt와 같은 형식의 fwords.txt 생성

2) weight계산을 위한 DF 받아오기

| 파일(F) | 편집(E) | 서식(O) | 보기(V) | 도움말(H) |
|---------|---------|----------|--------|--------------------------|
| 2 | % (| 1 ++ | 1 // | 1 0 1 0.05% 1 0.07 1 |
| 0.07 | 센트 | 1 | 0.10 | 1 0.10미크론 1 0.11 1 0.11포 |
| 인트 | 10.12 | 1 | 0.12 | 미크론급 1 0.13 1 0.13미크론 1 |
| 0.14 | 1 | 0.14% | 1 | 0.14센트 1 0.2% 1 0.3% |
| 1 | 0.63 | 1 | 0.63 | 주 1 0.96 1 0.96달러 17 |
| 00 | 1 0000 | 1 | 000N | 1 00년 91 01 1 010 |
| 3 | 011 | 1016 | 2017 | 1018 1019 107 02 1 026 |
| 103 | 03 | 1 030311 | 1 | 031 103월 90 04 78 05 |
| 59 | 06 | 1 060 | 80 | 07 1 070 87 08 85 09 |
| 302 | 1 | 7 1% | 131.0 | 2 1.0버전 2 1.1 3 1.2 1 |
| 1.22% | 2 | 1.25 | 1 | 1.2메가와트 1 1.3 1 1.33 1 |
| 1.3배 | 1 | 1.4 | 1 1.40 | 1 1.43% 1 1.47 1 |
| 1.4M | 6 | 1.5 | 2 1.5% | 1 1.51 1 1.51% 1 |
| 1.51달러대 | 1 | 1.52 | 1 | 1.52달러 1 1.53 1 1.5M |
| 1 | 1.5TB | 1 | 1.5TB급 | 1 1.5kg 1 1.5배 3 |
| 1.6 | 1 1.65% | 1 | 1.6달러 | 5 1.7 1 1.7% 1 1.7통 |
| 1 | 1.8 | 1 1.8% | 1 | 1.8 2 1.8% 1 1.83% 1 |

623개의 파일에서 unique한 색인어 추출 후 cmd에서 copy명령어를 통해 all.txt로
합친 후 wordcount를 통해 DF 계산한 파일

3) weight 계산 및 배열에 저장

forweight.cpp - 메모장

— □

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

//20163170 컴퓨터공학부 최은주 정보검색과 데이터마이닝 과제

```
#include <iostream>
#include <fstream>
#include <cstring>
#include <math.h>
using namespace std;

int main() {
    ofstream outfile("output.txt"); //출력결과 보기위한 파일
    ifstream DFfile("allDF.txt"); // <DF, term> 형태의 파일
    int TID = 1; int DID = 1; //TID, DID 인덱스 지정용 변수
    int no_Term = 34000; int no_Doc = 623; // 단어의 개수(여분을 위해 실제 단어의 수보다 크게함)와 문서의 개수
    double **arr = new double*[no_Term];
    for(int i = 0; i < no_Term; ++i) {
        arr[i] = new double[no_Doc];
        memset(arr[i], 0, sizeof(double)*no_Doc);
    } // weight를 저장하기 위한 2차원 배열을 동적할당으로 생성

    while(!DFfile.eof()) { //3만여개의 모든 term을 체크함
        double DF; string DFterm; DFfile >> DF >> DFterm; // term 한 개와 그 term의 DF를 받아옴
        outfile << "TID : " << TID << ", TERM : " << DFterm << endl; TID++; //출력파일에 입력 후 TID 인덱스 증가

        ifstream fnames("fwords.txt"); //wordcnt-ITnews---.txt 형태의 623개 파일들의 이름이 적혀있는 파일(fnaems.txt처럼)
        DID = 1;
        //623개의 파일들을 돌때마다 DID를 증가시키기 때문에 DFfile에서 다음 term을 받을때 다시 1번째 문서부터 나타내기 위하여 초기화
        while(!fnames.eof()) { //623개의 모든 파일을 돌기위한 반복문
            string filename; fnames >> filename; //string타입으로 파일의 이름을 받아 그 파일을 열기 위함
            ifstream TFfile(filename.c_str()); //<TF, term>형태의 파일
            while(!TFfile.eof()) { //파일 하나를 계속 돌면서
                double TF; string TFterm; TFfile >> TF >> TFterm; //그 파일 속의 term을 받아오며 TF도 저장해둠
                if(TFterm == DFterm) { //만약 DFterm과 같다면 TFfile속에 그 term이 있다는 것이므로
                    double IDF = log10(623/DF); //log를 취한 IDF를 만들어
                    arr[DID][TID] = TF*IDF; //2차원 배열에 weight 저장
                    outfile << "DID : " << DID << ", WEIGHT : " << TF*IDF << endl; //결과를 보기 위해 출력파일에도 입력함
                    break; //이미 단어를 찾았다면 그 뒤는 불필요하므로 빠른 실행을 위해 중간에 빠져나가게 함
                }
            }
            DID++; //한 TF파일을 돌았으므로 DID를 증가(최대 623)
        }
        outfile << endl; //보기 좋게 하기 위해 출력파일에서 한 단어가 끝날때마다 개행을 함
    }

    DFfile.close();
    outfile.close();
    return 0;
}
```

위 코드를 통해 2차원 배열 arr[문서번호][단어번호]에 $weight(=TF * \log(623/DF))$ 를 할당하며

결과를 보기 위해 output.txt라는 파일 사용

(코드에 대한 내용은 주석을 통해 표시하였습니다.)

output.txt - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

TID : 52 , TERM : 1.0
DID : 7 , WEIGHT : 8.40272
DID : 62 , WEIGHT : 1.68054
DID : 167 , WEIGHT : 1.68054
DID : 175 , WEIGHT : 3.36109
DID : 204 , WEIGHT : 1.68054
DID : 239 , WEIGHT : 1.68054
DID : 249 , WEIGHT : 3.36109
DID : 385 , WEIGHT : 1.68054
DID : 450 , WEIGHT : 1.68054
DID : 476 , WEIGHT : 1.68054
DID : 514 , WEIGHT : 1.68054
DID : 552 , WEIGHT : 1.68054
DID : 579 , WEIGHT : 1.68054

TID : 53 , TERM : 1.0버전
DID : 7 , WEIGHT : 4.98692
DID : 239 , WEIGHT : 2.49346

TID : 54 , TERM : 1.1
DID : 9 , WEIGHT : 2.49346
DID : 588 , WEIGHT : 7.48037

TID : 55 , TERM : 1.2
DID : 191 , WEIGHT : 2.31737
DID : 290 , WEIGHT : 4.63473
DID : 400 , WEIGHT : 2.31737

TID : 56 , TERM : 1.22%
DID : 566 , WEIGHT : 2.79449

4) 위에서 계산 weight를 이용해 문서들간의 유사도 계산

```
35     }
36 }
37 DID++; //한 TF파일을 들었으므로 DID를 증가(최대 623)
38 }
39 outfile << endl; //보기 좋게 하기 위해 출력파일에서 한 단어가 끝날때마다 개행을 함
40 }
41
42 ofstream simDoc("output2.txt"); //유사도 계산을 한 결과를 보기 위한 출력파일
43 double **SIM = new double*[no_Doc];
44 for(int i = 0; i < no_Doc; ++i) {
45     SIM[i] = new double[no_Doc];
46     memset(SIM[i], 0, sizeof(double)*no_Doc);
47 } //유사도 값을 넣기 위한 2차원 동적배열
48
49 for(int i = 0; i < 623; i++){
50     for(int j = 0; j < 623; j++){
51         double topsum = 0; //두 벡터의 내적을 위한 변수
52         double bottom1 = 0; //Di 벡터의 제곱을 위한 변수
53         double bottom2 = 0; //Dj 벡터의 제곱을 위한 변수
54         for(int k = 0; k < 32448; k++){ //단어의 개수만큼
55             topsum += arr[i][k] * arr[j][k]; //sum(Xik * Xjk)
56             bottom1 += pow(arr[i][k], 2); //sum(Xik의 제곱)
57             bottom2 += pow(arr[j][k], 2); //sum(Xjk의 제곱)
58         }
59         double simcos = topsum / sqrt(bottom1 * bottom2); //코사인 유사도 값 공식
60         SIM[i][j] = simcos; //배열에 유사도 값을 넣음
61         simDoc << simcos << " "; //유사도 값 출력
62     }
63     simDoc << endl; //보기 편하게 하기 위한 개행
64 }
65
66 for(int i = 0; i < no_Term; i++){
67     delete [] arr[i];
68 }
69 delete [] arr;
70 for(int i = 0; i < no_Doc; i++){
71     delete [] SIM[i];
72 }
73 delete [] SIM; //2차원 동적배열들의 메모리 해제
74
75 DFfile.close();
76 outfile.close();
77 simDoc.close();
78 return 0;
79 }
```

위 코드 아래에 내용을 추가하여 문서 간 유사도 계산

결과를 보기 위해 output2.txt 파일 사용

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

0 0 0,036996 0 0 0,0626317 0 0 1,77711 0,133265 0 0 0,0501998 0 0,0580696 0 0 0 0 0,118625 0,0355679 0 0
0,0364014 0 0,0373484 0,0352936 0,134861 0,036533 0 0 0 0 0 0 0,0390469 0 0,0663101 0,0390469 0 0
0,0763482 0 0,0755136 0,0270874 0 0,069868 0 0 0,0957913 0 0 0,0355009 0 0,023962 0,0560673 0 0 0
0,032002 0 0,0640598 0 0,1149 0 0 0 0 0,0319071 0 0 0,0390469 0,0350337 0,13345 0,0929002 0 0 0,0373106
0,134696 0,0973814 0 0,072624 0,0369206 0 0,0724846 0,0763482 0,036533 0 0
0 0 0,216931 0 0 0,318961 0 0 0,133265 1,11705 0 0 0,231262 0 0,283625 0 0 0,321443 0 0 0,585642 0,200038 0
0 0,209639 0 0,221446 0,197022 0,637595 0,698829 0 0,472796 0 0 0 0 0 0,245596 0 0,351132 0,245596 0 0
0,465306 0 0,453747 0,128281 0 0,386337 0,434114 0 0 0,493947 0,494633 0 0 0,199295 0 0 0,108826 0,2694 0 0
0 0 0,165264 0 0 0,331008 0 0,558296 0 0 0 0 0,164449 0,494633 0 0,245596 0,194226 0,710023 0,469885
0,494633 0 0,220954 0,6365 0,444344 0 0,417153 0,215985 0 0,415507 0,465306 0,698829 0 0
0 0,118622 0 0,0622482 0 0 0,0149673 0 0 0,606041 0,122025 0,141934 0 0,121603 0 0 0,0593873 0,143478
0,258596 0,0633867 0,399245 0 0 0 0,249495 0 0 0 0 0,0613761 0 0 0 0 0,177975 0 0,158696 0 0 0,50952
0,1628 0 0 0,131898 0 0,106388 0,191293 0 0 0 0,297941 0 0 0 0,497627 0 0,310534 0,220416 0,17083 0 0 0
0,299545 0,177703 0 0,225131 0,108219 0 0,0955535 0 0,138935 0 0,0979187 0 0 0 0 0,083324 0 0 0,600033 0
0,0530923 0 0,495235 0 0,118178 0,209455 0,53793 0 0 0
0 0,117674 0 0 0,06211 0 0 0,0149654 0 0 0,122025 0,616509 0,141752 0,175613 0,121346 0 0,41329 0,0592672
0,143055 0,256145 0,0632408 0 0 0 0 0,247292 0 0 0 0 0,0612436 0 0 0 0 0,177171 0 0,156447 0 0 0 0 0
0,130598 0 0,105703 0,190294 0 0 0,296995 0 0,539732 0 0,511967 0 0,308638 0,22017 0,170512 0 0 0
0,298584 0,176902 0 0,224406 0,107498 0 0,0950558 0 0 0,0973834 0 0 0 0 0,0829934 0 0 0,503919
0,0530064 0 0,0999996 0 0,597385 0,208146 0,108508 0 0 0
0 0,141137 0,0624281 0 0,118695 0 0,110287 0,0431014 0 0,0501998 0,231262 0,141934 0,141752 2,14579 0
0,33902 0,248604 0,249713 0,328176 0,250205 0,285856 0,330907 0,0606839 0 0 0,061709 0,703446 0,347275
0,0603423 0,244682 0,061869 0 0 0,118014 0 0 0,694146 0 0 0,266462 0,0648332 0,146306 0,761376 0,0648332 0
0 0,127648 0 0,269976 0,0491081 0,390528 0,390819 0 0,247585 0 0,674576 0 0 0,257211 0,197724 0 0,41335
0,545612 0,443845 0 0 0 0,563589 0,926587 0 0,493114 0,138952 0,630142 1,05367 0 0,159008 0 0,811498
0,0559454 0 0 0,0648332 0,980325 1,01167 0,163996 0 0 0,0628046 0,903432 0,179653 0,136988 0,1232
0,203391 0,276195 0,262222 0,127648 0,061869 0 0
0 0 0,419002 0,77407 0 0 0 0 0 0 0,175613 0 2,3816 0 0 0,170979 0 0 0 0 0,409277 0 0,376615 0 0 0 0,407356 0
0 0 0,359712 0 0 0 0,77407 0,399073 0 0 0 0 0 0 0 0 0 0 0,420665 0 0,17085 0 0 0 0 0,222141 0 0 0
0 0,338564 0,383112 0 0 0 0 0,298292 0 0,764296 0 0,408253 0 0 0 0 0,385035 0,175979 0 0 0,408473 0 0,1749
0 0 0 0,365466 0
0 0,120482 0,0799818 0 0,0929912 0 0,491216 0,0292717 0 0,0580696 0,283625 0,121603 0,121346 0,33902 0
1,62572 0 0 0,0905328 0,2003 0,246039 0,342995 0,0764086 0 0 0,0784859 0 0,324871 0,07573 0,280692