

執行環境：Dev C++ 5.11

使用語言：C++

執行方式：

1. 以下路徑請使用者自行更改

int main()

- Stopword txt 文件位置："C:\\Users\\Mark\\Desktop\\大五\\資訊檢索\\Stopwords.txt"
 - 包含 http://ir.dcs.gla.ac.uk/resources/linguistic_utils/stop_words 與自行添加的 stopword
- Documents 資料夾位置："C:\\Users\\Mark\\Desktop\\大五\\資訊檢索\\IRTM12"
- 計算 TFIDF 中繼資料夾位置(需先創造)："C:\\Users\\Mark\\Desktop\\大五\\資訊檢索\\IRTM_OUT\\"
- 字典輸出位置："C:\\Users\\Mark\\Desktop\\dictionary.txt"

double cosine(string docX, string docY)

- 各文章 TFIDF 輸出位置(需先創造)："C:\\Users\\Mark\\Desktop\\大五\\資訊檢索\\IRTM_TFIDF\\"

void getTFIDF(string doc, map<string, Arr5> dict, double N)

- 計算 TFIDF 中繼資料夾位置(需先創造)："C:\\Users\\Mark\\Desktop\\大五\\資訊檢索\\IRTM_OUT\\"
- 各文章 TFIDF 輸出位置(需先創造)："C:\\Users\\Mark\\Desktop\\大五\\資訊檢索\\IRTM_TFIDF\\"

2. 用 Dev C++ 5.11 打開 HW02.cpp，Compile & Run

C:\Users\Mark\Desktop\HW2test.exe

IP Address	Time
10.0.0.1	0.0347828
10.0.0.2	0.0779519
10.0.0.3	0.0209741
10.0.0.4	0.0458607
10.0.0.5	0.0828478
10.0.0.6	0.0721534
10.0.0.7	0.0514705
10.0.0.8	0.206601
10.0.0.9	0.061115
10.0.0.10	0.0503422
10.0.0.11	0.0971425
10.0.0.12	0.0587794
10.0.0.13	0.0201586
10.0.0.14	0.0567295
10.0.0.15	0.0896077
10.0.0.16	0.118428
10.0.0.17	0.113659
10.0.0.18	0.127305
10.0.0.19	0.0322487
10.0.0.20	0.074721
10.0.0.21	0.0785828
10.0.0.22	0.0370768
10.0.0.23	0.047391
10.0.0.24	0.0435061
10.0.0.25	0.00735177
10.0.0.26	0.0230998
10.0.0.27	0.0610642
10.0.0.28	0.0190774
10.0.0.29	0.0509072
10.0.0.30	0.0337903
10.0.0.31	0.0332616
10.0.0.32	0.044645
10.0.0.33	0.126154
10.0.0.34	0.0811888
10.0.0.35	0.0592138
10.0.0.36	0.0483277
10.0.0.37	0.0385518
10.0.0.38	0.1095
10.0.0.39	0.204957

Process exited after 7 seconds with return value 0
請按任意鍵繼續

Cosine Similarity of document 1 and document 2 = 0.204957

IRTM_OUT 資料夾內 1.txt (中繼用資料)

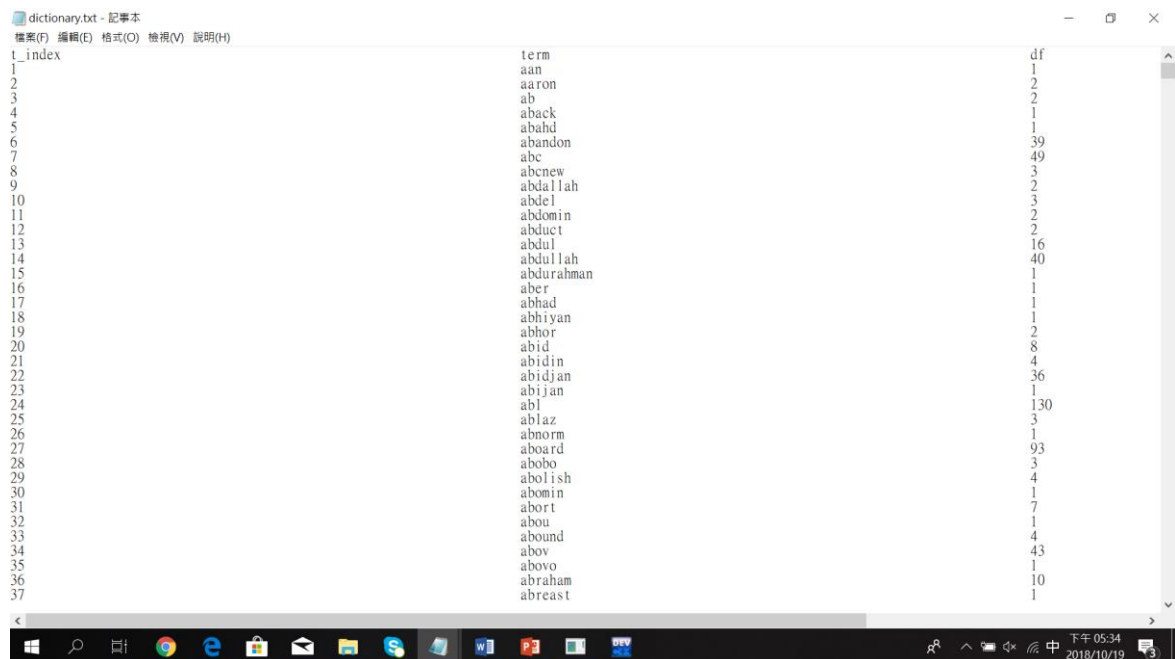
[illegible]

IRTM_TFIDF 資料夾內 Doc1.txt



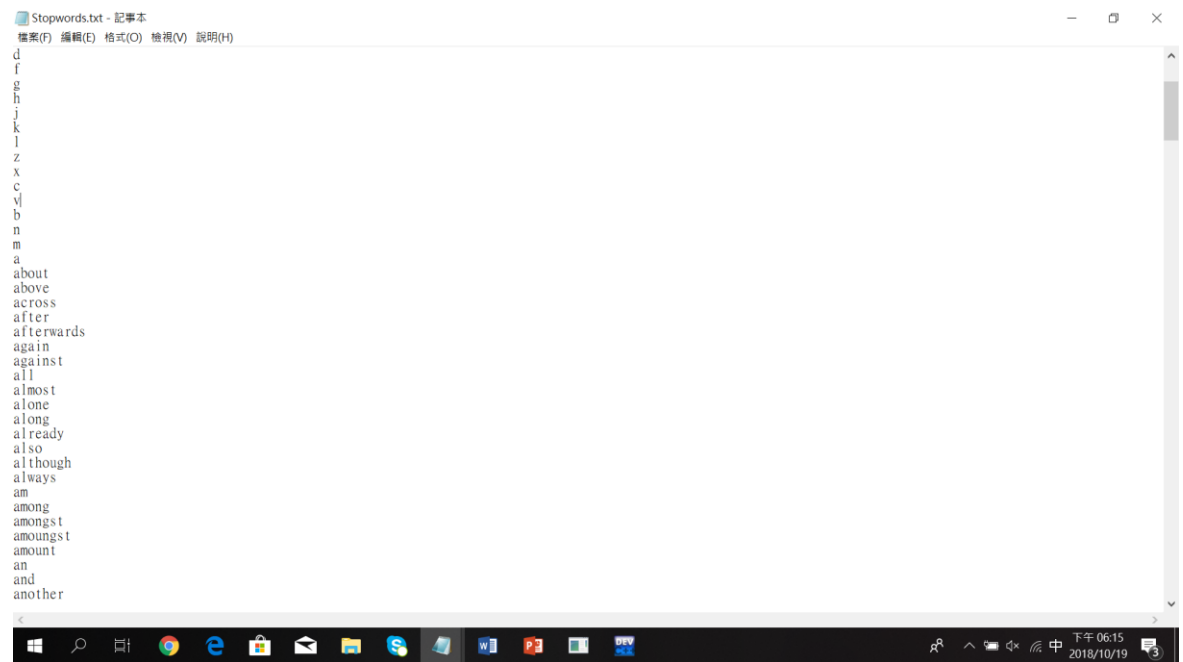
The size of document terms is: 118	
66	0.0535531
207	0.0409545
838	0.0779332
927	0.0487098
943	0.294209
997	0.054565
1073	0.13798
1567	0.045941
1676	0.0498768
1831	0.0709943
1897	0.0354446
1923	0.0868554
1997	0.0984108
1999	0.0697632
2102	0.0819918
2806	0.0972112
2913	0.10718
2994	0.0606259
3015	0.039124
3079	0.0768868
3200	0.0665791
3258	0.0308674
3260	0.0801698
3353	0.0739911
3416	0.0617068
3483	0.0702012
3922	0.0536176
4048	0.0813693
4180	0.0767283
4215	0.0972112
4423	0.0643279
4476	0.00920595
4529	0.0465299
4640	0.0628374
4645	0.0795913
4818	0.0399082
4837	0.0884422

Dictionary.txt



t_index	term	df
1	aan	1
2	aaron	2
3	ab	2
4	aback	1
5	ababd	1
6	abandon	39
7	abc	49
8	abcnew	3
9	abdallah	2
10	abdel	3
11	abdomin	2
12	abduct	2
13	abdul	16
14	abdullah	40
15	abdurahman	1
16	aber	1
17	abhad	1
18	abhiyan	1
19	abhor	2
20	abid	8
21	abidin	4
22	abidjan	36
23	abijan	1
24	abl	130
25	ablaz	3
26	abnorm	1
27	aboard	93
28	abobo	3
29	abolish	4
30	abomin	1
31	abort	7
32	abou	1
33	abound	4
34	abov	43
35	abovo	1
36	abraham	10
37	abreast	1

Stopwords.txt



處理邏輯：

主程式

```
33 int main(){
34
35     string line, path;
36     map<string, Arr5> dict;
37     char delim[] = ".,\\n\"?();:~_#&$%1234567890}{/*&";
38     string stopwords[10000];
39     ifstream doc("C:\\Users\\Mark\\Desktop\\大五\\資訊檢索\\Stopwords.txt");
40     int count = 0, doc_num = 0;
41     while (getline(doc, line)){
42         stopwords[count] = line;
43         count++;
44     }
45
46     char *t;
47     int i=0, k=0, r=0;
48
49     char InputPath[65535] = "C:\\Users\\Mark\\Desktop\\大五\\資訊檢索\\IRTH12"; //若要讀取檔案的資料夾路徑到InputPath字串裡
50     char szDir[65535];
51     char dir[65535];
52     WIN32_FIND_DATA FileData;
53     HANDLE hList;
54     sprintf(szDir, "%s\\*", InputPath);
55
56
57     if ( (hList = FindFirstFile(szDir, &FileData))==INVALID_HANDLE_VALUE )
58         printf("No files be found.\n\n");
59     else {
60         while (1) {
61             if (!FindNextFile(hList, &FileData)) {
62                 if (GetLastError() == ERROR_NO_MORE_FILES)
63                     break;
64             }
65
66             //sprintf(dir, "%s\\%s", InputPath, FileData.cFileName);
67             string x = FileData.cFileName;
68             ifstream doc("C:\\Users\\Mark\\Desktop\\大五\\資訊檢索\\IRTH12\\%x");
69             cout<<endl<<<<endl; //print currently processing file
70             doc_num++;
71             cout<<doc_num<<endl;
72             k=0, r=0, i=0; //initialize
73             map<string, int> document; //map
74             map<string, int>::iterator mpi; //initialize
75             for(mpi = document.begin(); mpi != document.end(); mpi++){
76                 (*mpi).second = 0;
77             }
78             char target[15000][50];
79             memset(target, 0, sizeof(target));
80             string result[15000];
81             map<string, Arr5>::iterator mp; //initialize
82             for(mp = dict.begin(); mp != dict.end(); mp++){
83                 (*mp).second.num[3] = 0;
84             }
85             while (getline(doc, line)){
86                 transform(line.begin(), line.end(), line.begin(), ::tolower); //convert to lowercase
87                 t = strtok((char*)line.c_str(), delim); //parse with delim
88                 while (t != NULL){
89                     strcpy(target[i], t); //copy the string pointed to char array 'target'
90                     cout<<i<<" "<<(target[i])<<endl;
91                     i++;
92                     t = strtok(NULL, delim); //t pointing to the next delimiter position
93                 }
94             }
95             /* 'target' is a char array, awaiting to undergo Porter's algorithm */
96             while(strlen(target[k]) != 0){
97
98                 step1(target[k]);
99                 step2(target[k]);
100                 step3(target[k]);
101                 step4(target[k]);
102                 step5(target[k]);
103                 step6(target[k]);
104
105                 int check_delete = 1;
106                 for(int j=0; j<sizeof(stopwords)/sizeof(stopwords[0]); j++){
107                     if (strcmp(target[k], stopwords[j].c_str())==0)
108                     {
109                         check_delete = 0;
110                         break;
111                     }
112                 }
113                 if(check_delete){
114                     result[r] = target[k];
115                     document[result[r]]++;
116                 }
117                 if(dict.count(result[r])){
118                     Arr5 temp;
119                     temp = dict[result[r]];
120                     temp.num[0]++;
121                     dict[result[r]] = temp;
122                     if (dict[result[r]].num[3] != 1){
123                         dict[result[r]].num[2] = dict[result[r]].num[2]+1;
124                         dict[result[r]].num[3] = 1;
125                     }
126                 }
127                 cout<<result[r]<<" "<<document[result[r]]<<" "<<dict[result[r]].num[2]<<endl;
128             }
129         }
130     }
```

```

130 |         else{
131 |             Arr5 temp1;
132 |             int array[] = {1,0,0,0,0};
133 |             std::copy(array, array+5, temp1.num);
134 |             dict[result[r]] = temp1;
135 |             if (dict[result[r]].num[3] != 1){
136 |                 dict[result[r]].num[2] = dict[result[r]].num[2]+1;
137 |                 dict[result[r]].num[3] = 1;
138 |             }
139 |             cout<<<<<< " <<result[r]<< " <<dict[result[r]].num[0]<< " <<dict[result[r]].num[2]<<endl;
140 |         }
141 |     }
142 |     r++;
143 | }
144 | k++;
145 | }
146 | }
147 | }
148 | cout<<endl<<endl;
149 |
150 | ofstream outfile ("C:\\Users\\VMark\\Desktop\\大五\\資訊檢索\\IRTH_OUT\\");
151 | outfile<<std::left<<setw(70)<<"term"<<setw(70)<<"tf"<<endl;
152 | map<string, int>::iterator it;
153 | for(it = document.begin(); it != document.end(); it++){
154 |     outfile<<std::left<<setw(70)<<(*it).first;
155 |     outfile<<std::left<<setw(70)<<(*it).second<<endl;
156 | }
157 | outfile.close();
158 | }
159 | }
160 | }
161 | }

```

```

162 |
163 | ofstream outfile ("C:\\Users\\VMark\\Desktop\\dictionary.txt");
164 | outfile<<std::left<<setw(70)<<"t_index"<<setw(70)<<"term"<<setw(70)<<"df"<<endl;
165 | map<string, Arr5>::iterator it;
166 | int v = 1, sum = 0;
167 |
168 | //dictionary export
169 | for(it = dict.begin(); it != dict.end(); it++){
170 |     (*it).second.num[1] = v;
171 |     cout<<std::left<<setw(70)<<(*it).second.num[1];
172 |     cout<<std::left<<setw(70)<<(*it).first;
173 |     cout<<std::left<<setw(70)<<(*it).second.num[2]<<endl;
174 |     outfile<<std::left<<setw(70)<<(*it).second.num[1];
175 |     outfile<<std::left<<setw(70)<<(*it).first;
176 |     outfile<<std::left<<setw(70)<<(*it).second.num[2]<<endl;
177 |     v++;
178 | }
179 | cout<<"total doc number: "<<doc_num-1; //total doc number
180 |
181 |
182 | char InputPath[65535] = "C:\\Users\\VMark\\Desktop\\大五\\資訊檢索\\IRTH_OUT"; //若要讀取檔案的資料夾路徑到InputPath字串裡
183 | char szDirA[65535];
184 | char dirA[65535];
185 | WIN32_FIND_DATA AFileData;
186 | HANDLE AhList;
187 | sprintf(szDirA, "%s\\*", InputPath);
188 | int fd;
189 | if ( (AhList = FindFirstFile(szDirA, &AFileData))==INVALID_HANDLE_VALUE )
190 |     printf("No files be found.\n\n");
191 | else {
192 |     while (1) {
193 |         if (!FindNextFile(AhList, &AFileData)) {

```

```

194 |             if (GetLastError() == ERROR_NO_MORE_FILES)
195 |                 break;
196 |         }
197 |         else{
198 |             f++;
199 |             string x = AFileData.cFileName;
200 |             getTFIDF(x, dict, doc_num-1);
201 |         }
202 |     }
203 | }
204 | string a = "1.txt", b = "2.txt";
205 | cosine(a, b);
206 | cout<<" " <<f-1;
207 | return 0;
208 | }

```

甲、windows.h 批次控制讀取資料夾內 txt 檔

乙、用 transform 函數把全部轉換成小寫

丙、用 strtok 函數斷詞，以 delim[]內的符號作為斷詞依據，存在 character

2D array 'target' 內

丁、進入 while 迴圈，讓每個單詞經過 porter' s algorithm 的處理

戊、與 stopwords 比對，若為 stopword 就不儲存

己、計算各文件 tf 與 dictionary 的 df

庚、利用己的資料計算各文件的 tf-idf vector

辛、計算文件 1 與文件 2 的 cosine similarity

資料結構：

1. 斷詞時使用二維 array
2. 轉存成 dictionary 做字典 df 與文件 tfidf 運算
 - dictionary 的 key 是 string，value 為一個包含大小=5 的陣列的 struct (該結構命名為 Arr5)
 - dict[result[r]].num[0] local variable，用來存該篇文章中此 term 出現的次數
 - dict[result[r]].num[1] t_index，照字母來排
 - dict[result[r]].num[2] df
 - dict[result[r]].num[3] 一個 flag，輔助計算 df
3. 最後用到 value 是 double 的 dictionary 來做各文章 tf-idf 運算

Library：

```
1  #include <iostream>
2  #include <fstream>
3  #include <string.h>
4  #include <vector>
5  #include <stdlib.h>
6  #include <stdio.h>
7  #include <algorithm>
8  #include <map>
9  #include <iomanip>
10 #include <windows.h>
11 #include <math.h>
```

函數介紹：

```
14 bool ends(string s, char target[]);
15 bool cons(int i, char target[]);
16 int m(char target[]);
17 bool vowelinstem(char target[]);
18 bool cvc(char target[]);
19 void step1(char target[]);
20 void step2(char target[]);
21 void step3(char target[]);
22 void step4(char target[]);
23 void step5(char target[]);
24 void step6(char target[]);
25 struct Arr5
26 {
27     double cosine(string docX, string docY);
28     void getTFIDF(string doc, map<string, Arr5> dict, double N);
29 }
30
31
```

參考資料：An algorithm for suffix stripping, M.F.Porter, 1980

<https://tartarus.org/martin/PorterStemmer/def.txt>

bool ends(string s, char target[]);

檢查 target 的結尾是否為第一個參數中的 string

bool cons(int i, char target[]);

檢查 target[i] 是否為子音，是的話回傳 true、是母音的話回傳 false。另外，y 的前面如果是子音，他就是母音

int m(char target[]);

用來檢查中間出現幾次 <V><C>，<V> 為連續的母音，<C> 為連續的子音

bool vowelinstem(char target[]);

檢查整個字內有沒有含子音

bool cvc(char target[]);

檢查結尾結構是否為 <子音><母音><不為 XYZ 的子音>

void step1(char target[]);

1(a)

結尾取代規則：

前	後
SS	SS
IES	I
SS	SS
S	

1(b)

結尾取代規則：

前	後
(m()>0) EED	EE
(contains a vowel) ED(a)
(contains a vowel) ING(b)

1(c) 若(a) or (b)成功

結尾取代規則：

前	後
AT	ATE
BL	BLE
IZ	IZE
(結尾重複且非 LSZ)	砍掉最後一個字
(m()=1 and cvc()==true)	E

void step2(char target[]);

結尾取代規則：

前	後
(含子音) Y	-> I

void step3(char target[]);

結尾取代規則：

前	後
(m>0) ATIONAL	-> ATE
(m>0) TIONAL	-> TION
(m>0) ENCI	-> ENCE
(m>0) ANCI	-> ANCE
(m>0) IZER	-> IZE
(m>0) ABLI	-> ABLE
(m>0) ALLI	-> AL
(m>0) ENTLI	-> ENT
(m>0) ELI	-> E
(m>0) OUSLI	-> OUS
(m>0) IZATION	-> IZE
(m>0) ATION	-> ATE
(m>0) ATOR	-> ATE
(m>0) ALISM	-> AL
(m>0) IVENESS	-> IVE
(m>0) FULNESS	-> FUL
(m>0) OUSNESS	-> OUS
(m>0) ALITI	-> AL
(m>0) IVITI	-> IVE
(m>0) BILITI	-> BLE

void step4(char target[]);

結尾取代規則：

前	後
(m>0) ICATE	-> IC
(m>0) ATIVE	->
(m>0) ALIZE	-> AL
(m>0) ICITI	-> IC
(m>0) ICAL	-> IC

(m>0) FUL ->

(m>0) NESS ->

void step5(char target[]);

結尾取代規則：

前	後
---	---

(m>1) AL ->

(m>1) ANCE ->

(m>1) ENCE ->

(m>1) ER ->

(m>1) IC ->

(m>1) ABLE ->

(m>1) IBLE ->

(m>1) ANT ->

(m>1) EMENT ->

(m>1) MENT ->

(m>1) ENT ->

(m>1) and (結尾為 S 或 T)) ION ->

(m>1) OU ->

(m>1) ISM ->

(m>1) ATE ->

(m>1) ITI ->

(m>1) OUS ->

(m>1) IVE ->

(m>1) IZE ->

void step6(char target[]);

結尾取代規則：

前	後
---	---

(m>1) E ->

(m=1 & cvc()==false) E ->

(m > 1 and 結尾重複且非 LL) -> 砍掉最後一個字

double cosine(string docX, string docY);

用來計算兩文件 tfidf vector 之 cosine similarity，參數為兩個文件的檔名

➤ Document 1 & Document 2 的 cosine similarity 結果請見 P.2

```
void getTFIDF(string doc, map<string, Arr5> dict, double N);
```

用來計算 doc 文件的 tfidf 並輸出，dict 為已經建立好的 dictionary，N 為文章數

*詳細實作內容請件附檔 HW02test.cpp