1091438 郁宸瑋

程式碼說明 & 如何編譯

1. 全域變數 (Global Variable)

```
14 // Global variable
15 map<string,int>words;
16 map<string,int>seq;
17 vector< vector<int> >fre_last; //the final ans of frequency
18 vector <string> id;
19 vector <double> cos_aver; //The files's average_cosine
20 int wt = 0; //wait or run
21 int totalwords = 0;
22 pid_t tid;
```

map<string, int>words、map<string, int>seq ：

words 用來存**詞頻**, 因 map 會自動排序(由 words[a], a 開頭排到 words[z], z 開頭)

因此多用了一個 map<string, int> seq 來調整順序, 決定文件中單詞出現的"**順序**", 再將順序存入 seq 中

vector<vector<int>>fre_last:

用來存放第一個文章(文件)到最後一個文章(文件)的**詞頻向量**

vector<string>id:

用來存放各個文章(文件)的**文件 ID**

vector<double>cos_aver:

用來存放各個文件(來源文件)與其他 N(全部文件數量)-1 個文件的餘弦相關係數的"**平均**"(Avg_cosine)

int wt = 0:

用來**控制輸出順序**

int totalwords = 0:

計算出當前已讀取過的文件的**全部詞彙**有多少個

pid_t tid:

存放子執行緒的 tid

2. void change(string &str, int size)

```
24 void change(string &str,int size){
25         for(int i = 0;i < size ; i++){
26                 if((str[i]<=57 && str[i]>=48) || (str[i]<=90 && str[i] >=65) || (str[i]<=122 && str[i] >= 97)){ //use ascii
27
28                 }
29                 else{
30                         str[i] = ' ';
31                 }
32         }
33         // Make everything except a-z,A-Z and 0-9 blank
34         int count = 0;
35         bool dele = false;
36         for(int i = 0;i < size ; i++){
37              if(str[i] == ' ' || i == size - 1){ // Because the last word don't have blank,need a extra judgement(i == size -1)
38                 int tmp = (i == size - 1) ? size : i;
39
40                 for(int j = count ; j < tmp; j++){
41                     if(str[j]<=57 && str[j]>=48){ // if the word have 0-9,ignore this word (dele = true)
42                         dele = true;
43                         break;
44                     }
45                 }
46                 if(dele){
47                         for(int j = count ; j < tmp; j++){
48                                 str[j] = ' ';
49                         }
50                         dele = false; //initialize
51                 }
52                 count = i + 1; // the start of the next word
53              }
54         }
55 }
```

此函式用來: (1) 將標點符號都先換成空白字元 (2)並將不是純字母組成的詞忽略不計

傳入參數 : str 為當前文件的內容, Ex: This is a book, size 為 str.size()

Bool dele = false :

用來判斷此詞**是否要忽略不計**, true 就忽略(此詞彙全部變成空白字元)

int count = 0:

存放當前詞彙的起始位置

首先, 25~32 行先將**除了**數字(0~9)、字母(A~Z、a~z), 通通變成空白字元, 我是**利用** ASCII 碼來進行判斷。

接下來, 34~54 行用來**將不是純數字組成的詞忽略**。一開始, (37、38 行)我檢查 str[i]是否為空白字元, 或是 i 是否等於 size - 1(若 str[i]為空白字元, 代表**前面有詞彙**, 要**檢查**此詞彙, 但最後一個詞彙後並無空格, 因此我用 i == size - 1 來**處理最後一個詞彙**), 再來我設一個 tmp(用來存此詞彙<要被檢查的>的"**後一個**"位置), 之後利用 40-45 行判斷(透過 ASCII 碼)此詞彙是否為"**純字母組成**"(**當前詞彙的組成為** str[count]~str[tmp-1]), 若不是, dele=true, 並在 46~51 行將此詞彙全部變成空白字元(忽略不計), 並在最後記下"**下一個詞彙的起始位置**"(count = i + 1)

3. void frequency(string &str,int size)

```
57 void frequency(string &str,int size){
58         for(int i=0;i<size;i++){
59                 string word_tmp;
60                 while(str[i] != ' ' && i != size){
61                         word_tmp+=str[i];
62                         i++;
63                 }
64                 if(words.find(word_tmp)==words.end()){ //Don't find the word_tmp,the first time this word has ever appear
65                         totalwords++;
66                         seq[word_tmp] = totalwords;
67                         words[word_tmp] = 1;
68                 }
69                 else{
70                         words[word_tmp]++;
71                 }
72         }
73         int tmp = 1;
74         vector<int>tmp_fre;
75         while(tmp<=totalwords){
76                 for(map<string,int>::iterator it1 = seq.begin();it1 != seq.end();++it1){
77                         if(it1->second==tmp){
78                                 tmp_fre.push_back(words[it1->first]);
79                                 words[it1->first] = 0; //initialize
80                         }
81                 }
82                 tmp++;
83         }
84         fre_last.push_back(tmp_fre);
85 }
```

此函式用來**計算詞頻(使用過 change 函式之後 <文件內僅剩空白字元和詞彙>)**

傳入參數 : str 為當前文件的內容,Ex: This is a book, size 為 str.size()

string word_tmp:

用來存放詞彙(Ex:This、is、a、book……)

vector<int>tmp_fre:

用來存當前文件的詞頻向量

58~72 行 :首先,我先將詞彙的各個字母組合再一起(60~63 行)存入 word_tmp,並在 64~71 行檢查此詞彙是否已經出現過,若出現過,將**此詞彙在當前文件出現的次數** + 1 (70 行)。否則,將此詞彙(第一次出現)放入 words 中,並將**此詞彙在當前文件出現的次數**設為 1、totalwords++(當前總詞彙數量),並將此詞彙出現的"順序"記錄下來(66 行)

73~84 行 :我用一個二維 vector(fre_last)來存放當前文件的詞頻向量。tmp 用來表示第 tmp 個"出現"的詞彙,若 it1 ->second = tmp(seq[**詞彙**] = it1->second ,用來找第 tmp 個出現的詞彙是誰),將他的詞頻(words[it->first]放入 tmp_fre 中,並讓 words[it1->first]歸 0<為了存下一個文件的詞頻>),之後 tmp++,找下一個出現的詞彙,最後將當前文件的詞頻向量存入 fre_last

4. int main(int argc, char *argv[]) & void *child(void *arg)

一開始:

```
130         double START,END;   //CPU Time
131         START = clock();
132         vector <string> inner;
133         string line;
134         ifstream myFile;
135         myFile.open(argv[1]);  //cin the file name
136         int count = 0; //The count of files
137         while(getline(myFile,line)){
138                 if(line!=""){
139                         id.push_back(line);
140                         getline(myFile,line);
141                         inner.push_back(line);
142                         count++;
143                 }
144                 else{
145                         break;
146                 }
147         }
148
```

計算 CPU Time

讀檔
argv[1]為您的檔名
(Ex:data.txt)

vector<string>inner:
用來存放文件內容(Ex:This is a book)

int count :
用來計算有幾個文件被讀取進來

137~146 行:
將讀入的文件內容(getline(myFile,line))分別存入 id(文件 ID)和 inner(文件內容)中

讀完檔案之後,開始整理文件的內容(inner):

```
149     for(int i = 0;i < count;i++){
150             string tmp1 = inner[i];       // To store the inner[i].size()
151             change(inner[i],tmp1.size());
152             frequency(inner[i],tmp1.size());
153     }
154     int fresize = fre_last.size();
155     for(int i = 0;i < fresize ; i++){
156             int fre_isize = fre_last[i].size();
157             for(int j = 0;j < fre_isize ; j++){
158                     if(j == fre_isize - 1  && j <totalwords){
159                             for(int k = fre_isize ;k < totalwords ; k++){
160                                     fre_last[i].push_back(0); //fill with 0
161                             }
162                     }
163             }
164     }
```

change(inner[i],tmp1.size()):
將 inner[i]的內容,標點符號換成空白字元,將不是純字母的字忽略

frequency(inner[i],tmp1.size()):
算出 inner[i]的詞頻向量

154~164 行:前面有說到,totalwords 為**當前**已讀取過的全部詞彙。經過 149~153 行之後,totalwords 即為"**所有文件的所有詞彙**"總數。要進行 158~162 行,是因為"第一個文件的 totalwords 不一定和最終的 totalwords 一樣",

假設: 0001 (這是文件 ID)

This is a pen => 此時的 totalwords 為 4 , 詞頻[1,1,1,1]

0002

This is a book => 此時的 totalwords 為 5 , 詞頻[1,1,1,1,1]

0001 少了一個 book 的詞頻(因為此時並未讀到 book),因此需要補 0,使 0001 的詞頻變為[1,1,1,1,0]

將各個文件的詞頻向量弄好之後,開始透過 thread 來計算:

```
166         pthread_t thread[count]; //How many Files,how many child thread
167         for(int i = 0 ; i < count ; i++){
168                 pthread_create(&thread[i],NULL,child,&i); // i = current file
169                 while(true){
170                         if(wt == 1){
171                                 cout<<"[Main thread]: create TID:"<<tid<<",DocID:"<<id[i]<<endl;
172                                 wt++;
173                                 break;
174                         }
175                 }
176                 pthread_join(thread[i],NULL);
177                 wt = 0; // give the next thread
178         }
```

```
87 void *child(void *arg){
88           double START,END;   //CPU Time
89           START = clock();
90           int n = *(int *)arg;
91           tid = gettid();
92           wt++;
```

pthread_t thread[count]:

count 為文件總數,有幾個文件就創幾個子執行緒

167~178 行:

建立新 thread(有 count 個) -> child(執行緒的入口函數名字),且帶著 i(當前是第 i 個文件)過去。169~175 行是為了**控制輸出順序(wt)**,當 wt = 1,代表 thread id 已經存入 tid 中,即 main thread 知道 tid 為多少,就可以輸出 171 行,wt++。接下來就等待子執行緒結束(176 行),並將 wt = 0(為了下一個子執行緒)

進入到子執行緒,計算每一份文件對其他 M-1 份文件的平均餘弦相似係數

```
87 void *child(void *arg){
88         double START,END;  //CPU Time
89         START = clock();
90         int n = *(int *)arg;
91         tid = gettid();
92         wt++;
93
94         while(true){
95                 if(wt == 2){
96                         cout<<"[TID="<<tid<<"] DocID:"<<id[n]<<"["<<fre_last[n][0];
97                         for(int i = 1;i <totalwords ; i++){
98                                 cout<<","<<fre_last[n][i];
99                         }
100                        cout<<"]"<<endl;
101                        wt++;
102                        break;
103                }
104        }
105
106        double aver = 0.0;
107        for(long unsigned int i = 0; i <id.size();i++){
108                if(i != (long unsigned int)n){
109                        int up = 0;
110                        double down_left = 0.0,down_right = 0.0,sim = 0.0;
111                        for(int j = 0 ; j < totalwords;j++){
112                                up = up + fre_last[i][j]*fre_last[n][j];
113                                down_left = down_left +fre_last[i][j]*fre_last[i][j]; //Vi
114                                down_right = down_right + fre_last[n][j]*fre_last[n][j]; //Vs
115                        }
116                        sim = up / (sqrt(down_left)*sqrt(down_right));
117                        cout<<"[TID="<<tid<<"] cosine("<<id[n]<<","<<id[i]<<")="<<fixed<<setprecision(4)<<sim<<endl;
118                        aver = aver + sim;
119                }
120        }
121        aver = aver / ((double)id.size()-1.0);
122        cout<<"[TID="<<tid<<"] Avg_cosine: "<<aver<<endl;
123        cos_aver.push_back(aver);
124        END = clock();  //CPU Time
125        cout<<"[TID="<<tid<<"] CPU time: "<<fixed<<setprecision(0)<< (END - START)<<"ms"<<endl;
126        return NULL;
127 }
```

計算並輸出 CPU 時間

用來存放 Average Cosine(123 行)

94~104 行: 當 171 行輸出後,wt = 2,即可輸出 96~100 行(如下圖)

```
[Main thread]: create TID:1946,DocID:0001
[TID=1946] DocID:0001[1,1,1,1,0,0,0]
```

接下來**開始計算並輸出(106~122 行)**第 n 個文件與其他 count-1 個文件的餘弦相似係數

double aver : 用來存放平均餘弦相似係數(118、121 行)

這裡為 up 存放的值(112 行)

$$Sim(V_s,V_x)=\cos(V_s,V_x)=\frac{V_s\cdot V_x}{|V_s|\times|V_x|}=\frac{\sum_{i=1}^{n} v_{s,i}\times v_{x,i}}{\sqrt{\sum_{i=1}^{n} v_{s,i}^2}\times\sqrt{\sum_{i=1}^{n} v_{x,i}^2}}$$

down_right(114 行)

down_left(113 行)

double sim : 用來存放餘弦相關係數 (116 行)

輸出結果:

```
[Main thread]: create TID:1946,DocID:0001
[TID=1946] DocID:0001[1,1,1,1,0,0,0]
[TID=1946] cosine(0001,0002)=0.7500
[TID=1946] cosine(0001,0003)=0.7906
[TID=1946] cosine(0001,0004)=0.5000
[TID=1946] Avg_cosine: 0.6802
[TID=1946] CPU time: 3433ms
[Main thread]: create TID:1947,DocID:0002
[TID=1947] DocID:0002[1,1,1,0,1,0,0]
[TID=1947] cosine(0002,0001)=0.7500
[TID=1947] cosine(0002,0003)=0.4743
[TID=1947] cosine(0002,0004)=0.2500
[TID=1947] Avg_cosine: 0.4914
[TID=1947] CPU time: 2242ms
[Main thread]: create TID:1948,DocID:0003
[TID=1948] DocID:0003[0,1,2,2,0,1,0]
[TID=1948] cosine(0003,0001)=0.7906
[TID=1948] cosine(0003,0002)=0.4743
[TID=1948] cosine(0003,0004)=0.6325
[TID=1948] Avg_cosine: 0.6325
[TID=1948] CPU time: 2303ms
[Main thread]: create TID:1949,DocID:0004
[TID=1949] DocID:0004[0,1,0,1,0,1,1]
[TID=1949] cosine(0004,0001)=0.5000
[TID=1949] cosine(0004,0002)=0.2500
[TID=1949] cosine(0004,0003)=0.6325
[TID=1949] Avg_cosine: 0.4608
[TID=1949] CPU time: 1009ms
```

子執行緒均結束後,在 main thread 中輸出 Highest Average Cosine & 主執行緒 CPU Time

```
179     double max = -2; //cos's range is -1 ~ 1
180     int KeyID;
181     for(int i = 0 ; i < count ; i++){
182             if(max < cos_aver[i]){
183                     KeyID = i;
184                     max = cos_aver[i];
185             }
186     }
187     cout<<"[Main thread] KeyDocID:"<<id[KeyID]<<" Highest Average Cosine:"<<fixed<<setprecision(4)<<max<<endl;
188     END = clock();  //CPU Time
189     cout<<"[Main thread] CPU time: "<<fixed<<setprecision(0)<< (END - START)<<"ms"<<endl;
```

計算並輸出 CPU Time

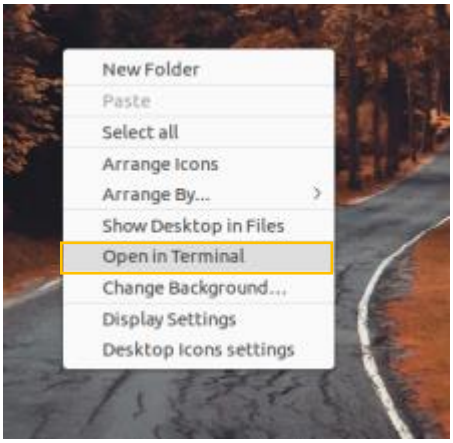
179~186:

找出 Highest Average Cosine,並存入 max 中

(順便將 Highest Average Cosine 的文件 id 記下<存入 KeyID 中>)

5. 如何編譯

(1)在 Ubuntu 桌面右鍵 - > 選擇 Open Terminal



(2)輸入 g++ /home/yui/Desktop/testread.cpp -o/home/yui/Desktop/testread.out -Wall (g++ /檔案路徑/檔案名稱.cpp -o/檔案路徑/檔案名稱.out -Wall),並按下 Enter

```
yui@yui-VirtualBox:~/Desktop$ g++ /home/yui/Desktop/testread.cpp -o/home/yui/Desktop/testread.out -Wall
```

(3)輸入 ./testread.out data.txt (./檔案名稱.out 輸入文字檔案名稱.txt)

```
yui@yui-VirtualBox:~/Desktop$ ./testread.out data.txt
```

(4)觀看結果

```
[Main thread]: create TID:1946,DocID:0001
[TID=1946] DocID:0001[1,1,1,1,0,0,0]
[TID=1946] cosine(0001,0002)=0.7500
[TID=1946] cosine(0001,0003)=0.7906
[TID=1946] cosine(0001,0004)=0.5000
[TID=1946] Avg_cosine: 0.6802
[TID=1946] CPU time: 3433ms
[Main thread]: create TID:1947,DocID:0002
[TID=1947] DocID:0002[1,1,1,0,1,0,0]
[TID=1947] cosine(0002,0001)=0.7500
[TID=1947] cosine(0002,0003)=0.4743
[TID=1947] cosine(0002,0004)=0.2500
[TID=1947] Avg_cosine: 0.4914
[TID=1947] CPU time: 2242ms
[Main thread]: create TID:1948,DocID:0003
[TID=1948] DocID:0003[0,1,2,2,0,1,0]
[TID=1948] cosine(0003,0001)=0.7906
[TID=1948] cosine(0003,0002)=0.4743
[TID=1948] cosine(0003,0004)=0.6325
[TID=1948] Avg_cosine: 0.6325
[TID=1948] CPU time: 2303ms
[Main thread]: create TID:1949,DocID:0004
[TID=1949] DocID:0004[0,1,0,1,0,1,1]
[TID=1949] cosine(0004,0001)=0.5000
[TID=1949] cosine(0004,0002)=0.2500
[TID=1949] cosine(0004,0003)=0.6325
[TID=1949] Avg_cosine: 0.4608
[TID=1949] CPU time: 1009ms
[Main thread] KeyDocID:0001 Highest Average Cosine:0.6802
[Main thread] CPU time: 28241ms
```