

執行環境：Dev C++ 5.11

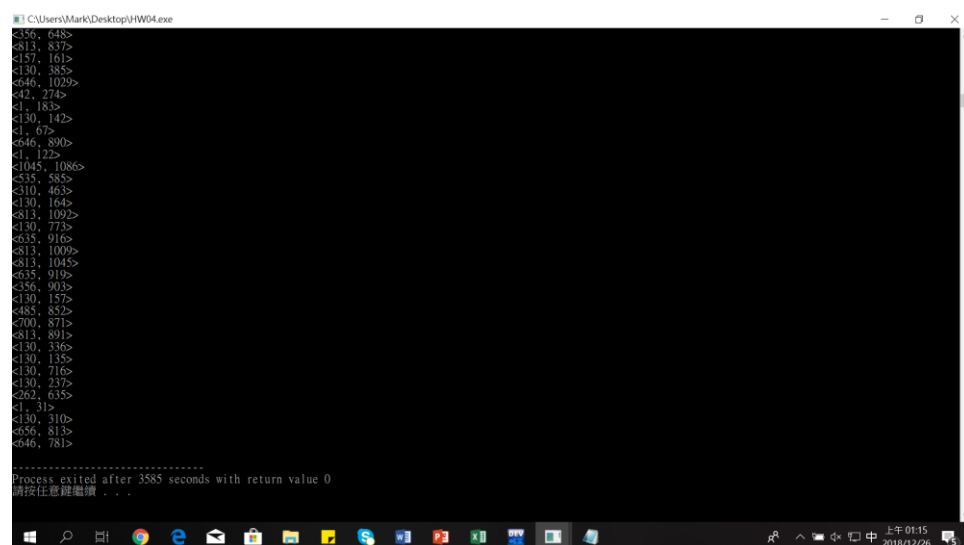
使用語言：C++

執行方式：

1. 以下路徑請使用者自行更改

void HAC()

- TF-idf txt 文件放置位置："C:\\Users\\Mark\\Desktop\\大五\\資訊檢索\\IRTM_TFIDF"
 - 需取名為 Doc 文件號碼.txt。例如：Doc998.txt
 - 每次 clustering 結果輸出位置："C:\\Users\\Mark\\Desktop\\大五\\資訊檢索\\IRTM_Clustering "
 - 最終 clustering 結果輸出位置："C:\\Users\\Mark\\Desktop\\大五\\資訊檢索\\clustering_answer"
2. 用 Dev C++ 5.11 打開 HW04.cpp，Compile & Run (int main 裡面的 HAC()參數要自行更改成要分幾群。例如：HAC(8)就會得到分 8 群的結果)



程式邏輯

每次找兩個相似度最高的群合併，直到最後剩下 K 群，K=8, 13, 20。

實作時使用三層回圈，最外圈計算總共要合併幾次，其他兩圈用來倆倆比較相似度大小

函數介紹：

一、HAC：實作 HAC clustering

```
1056 void HAC(int cluster_num)
1057 {
1058     bool I[1096];
1059     int N[1096];
1060     list<pair<int, int>> A;
1061     std::vector<std::map<string, double>> vecOfMaps;
1062     double *C = new double[1096];
1063     double max = 0;
1064     int first = 0, second = 0;
1065     map<string, double> zero;
1066     vecOfMaps.push_back(zero);
1067     for(int i=1; i<=1095; i++)
1068     {
1069         int count = 0;
1070         ifstream docf("C:\\Users\\Mark\\Desktop\\大五\\資訊檢索\\IRTM_TFIDF\\Doc"+to_string(i)+".txt");
1071         map<string, double> dict1;
1072         string line;
1073         char *t;
1074         getline(docf, line);
1075         char delim[] = " ";
1076         while (getline(docf, line)){
1077             t = strtok ((char*)line.c_str(), delim); //parse with delim
1078             while (t != NULL){
1079                 string term(t);
1080                 s = strtok(NULL, delim); //t pointing to the next delimiter position
1081                 double ans = atof(s);
1082                 dict1[term] = ans;
1083                 t = strtok(NULL, delim);
1084             }
1085             count++;
1086         }
1087         vecOfMaps.push_back(dict1);
1088         N[i] = i;
1089     }
1090     /*map<string, double>::iterator it;
1091     for(it = vecOfMaps.at(5).begin(); it != vecOfMaps.at(5).end(); it++){
1092         cout<<("it).first<< " <<("it).second<<endl;
1093     }*/
1094     for(int i=1; i<=1095; i++)
1095     {
1096         C[i] = new double[1096];
1097         for(int j=1; j<=1095; j++)
1098         {
1099             string a = "Doc"+to_string(i)+".txt", b = "Doc"+to_string(j)+".txt";
1100             C[i][j] = cosine(a, b);
1101             I[i] = 1;
1102             //cout<<"Similarity of "<<a<< " <<b<< " is "<<C[i][j]<<endl;
1103         }
1104     }
1105     for(int k=1; k<=(1095-cluster_num); k++) //跑N-1次直到結束
1106     {
1107         max = 0, first = 0, second = 0;
1108         for(int i=1; i<=1095; i++)
1109         {
1110             for(int j=1; j<=1095; j++)
1111             {
1112                 if(C[i][j] > max && ((i != j) && (I[i]==1 && I[j]==1))){
1113                     first = i;
1114                     second = j;
1115                 }
1116             }
1117             max = C[i][j];
1118         }
1119         /*if((i != j) && (I[i]==1 && I[j]==1))
1120             cout<<("i"<< " "<<j<< " "<<C[i][j]<<endl;*/
1121     }
1122     dictMerge(first, second, N, vecOfMaps); //merge second to first
1123     //test
1124     ofstream outfile ("C:\\Users\\Mark\\Desktop\\大五\\資訊檢索\\IRTM_Clustering\\"+to_string(first)+"and"+to_string(second)+".txt");
1125     map<string, double>::iterator ittest;
1126     outfile<<"Total document : "<<N[first]<<endl;
1127     for(ittest = vecOfMaps.at(first).begin(); ittest != vecOfMaps.at(first).end(); ittest++){
1128         outfile<<std::left<<setw(70)<<("ittest).first;
1129         outfile<<std::left<<setw(70)<<("ittest).second<<endl;
1130     }
1131     A.push_back(make_pair(first, second));
1132     //cout<<"first: "<<first<< " second: "<<second<<endl;
1133     for(int g=1; g<=1095; g++)
1134     {
1135         C[first][g] = cosine(vecOfMaps.at(first), vecOfMaps.at(g));
1136         C[g][first] = cosine(vecOfMaps.at(g), vecOfMaps.at(first));
1137     }
1138     I[second] = 0;
1139 }
```

```

1149 int record_cluster[1096];
1150 for(int i=1; i<=1095; i++){
1151     record_cluster[i] = 1;
1152     bool exist_cluster[1096];
1153     for(int i=1; i<=1095; i++){
1154         exist_cluster[i] = 1;
1155     }
1156     ofstream outfile ("C:\\Users\\Mark\\Desktop\\大五\\資訊檢索\\clustering_answer\\" + to_string(cluster_num) + ".txt");
1157     cout<<"Merge List : "<<endl;
1158     for (list<pair<int, int>>::iterator ir = A.begin(); ir!=A.end(); ir++) {
1159         cout << "C"<<(*ir).first << ", " << (*ir).second<<"<<endl;
1160         if(record_cluster[(*ir).first] != 0 && record_cluster[(*ir).second] != 0){
1161             int temp = record_cluster[(*ir).second];
1162             for(int i=1; i<=1095; i++){
1163                 if(record_cluster[i] == temp)
1164                     record_cluster[i] = record_cluster[(*ir).first];
1165             }
1166             exist_cluster[temp] = 0;
1167         }
1168     }
1169     for(int i=1; i<=1095; i++){
1170         if(exist_cluster[i] == 1){
1171             for(int j=1; j<=1095; j++){
1172                 if(record_cluster[j] == i){
1173                     outfile<<j<<endl;
1174                 }
1175             }
1176             outfile<<endl;
1177         }
1178     }
1179 }

```

處理邏輯：三層 for 迴圈，最外層是看要合併幾次，裡面兩個 for 迴圈則是比較所有兩群的組合之 cosine similarity，擁有最大的 cosine similarity 的兩群合併 (dictMerge)。其中 C[i][j] 用來記錄文章 i 與文章 j 的 cosine similarity、I[k] 用來記錄第 k 篇文章是否已經被合併到別篇、list A 用來存每次合併是哪兩群。

二、dictMerge：合併兩群

```

1181 void dictMerge(int first1, int second1, int N[1096], std::vector<std::map<string, double>>& vecOfMaps)
1182 {
1183     map<string, double>::iterator it1;
1184     for(it1 = vecOfMaps.at(first1).begin(); it1 != vecOfMaps.at(first1).end(); it1++){
1185         (*it1).second = N[first1]*(*it1).second;
1186     }
1187     map<string, double>::iterator it;
1188     for(it = vecOfMaps.at(second1).begin(); it != vecOfMaps.at(second1).end(); it++){
1189         if(vecOfMaps.at(first1).find(*it).first == vecOfMaps.at(first1).end())
1190         {
1191             vecOfMaps.at(first1)[(*it).first] = N[second1]*(*it).second;
1192         }
1193         else
1194         {
1195             vecOfMaps.at(first1)[(*it).first] += N[second1]*(*it).second;
1196         }
1197     }
1198     map<string, double>::iterator it2;
1199     for(it2 = vecOfMaps.at(first1).begin(); it2 != vecOfMaps.at(first1).end(); it2++){
1200         (*it2).second = (*it2).second/(N[first1] + N[second1]);
1201     }
1202     N[first1] = N[first1] + N[second1];
1203 }
1204
1205

```

處理邏輯：傳入儲存 map 的 vector 作為參數，使用 centroid 法來求給定兩群距離，距離最短的兩群之 TF-idf 依照文章數目做加權平均 $((N1*tf-idf1+N2*tf-idf2)/(N1+N2))$ 。其中 N[k] 代表第 k 群內目前有多少 documents。

三、cosine：計算兩群相似度

```

692 double cosine(std::map<string, double> first1, std::map<string, double> second1)
693 {
694     double sum = 0;
695     map<string, double>::iterator it;
696     for(it = first1.begin(); it != first1.end(); it++){
697         if(second1.find(it->first) != second1.end()){
698             sum += (*it).second * (second1.find(it->first->second));
699         }
700     }
701     return sum;
702 }
703

```

處理邏輯：傳入兩個 map，對 TF-idf vector 內積。