HW4

前言：

由於老師上課時時常提及高雄，加上本人平生以來從未去過高雄，因此這次作業心血來潮想用 ptt 的高雄板來玩玩看，期望透過 ptt 對高雄有所了解，並看看會有什麼有趣的結果。

處理方式：

盡量讓符號的處理和文字的處理分開，

1. 特殊符號→盡量用 gsub 全部取代成空字串
2. 斷詞錯誤→字典
3. 冗詞→Stop_Tw

而由於十頁的文章數可能真的偏多，一篇篇文章檢查可能相對較沒效率，因此選擇看最前面兩頁，然後假設後面文章的特殊符號、斷詞錯誤、冗詞彙大致相同，做完這些前處理(去除 stop)最後再用十頁的文章 tf-idf 等分析。

雖然作業有要求要做 tfidf，但我們網路聲量的比較仍以 tf 當基準。

程式碼截圖：

```
setwd("C:\\Users\\Steven\\Desktop\\陽交109下\\巨量資料分析\\課程\\單元10：文字探勘\\範例程式與資料")
getwd()

# 載入ptt爬蟲與斷詞相關套件
library(XML)
library(RCurl)
library(tm)
library(jiebaR)

# 寫個迴圈，可以一次抓多頁所有文章的連結（每次抓兩頁修改，共十頁）
link <- NULL
for( i in 4998:5007){
  url <- paste0("https://www.ptt.cc/bbs/Kaohsiung/index", i, ".html")
  html <- htmlParse(getURL(url))
  url.list <- xpathSApply(html, "//div[@class='title']/a[@href]", xmlAttrs)
  link <- c(link, paste('https://www.ptt.cc', url.list, sep=''))
}
length(link)
```

```r
# 寫個迴圈，抓出所有儲存在link中的文章，處理並儲存
article <- NULL
for(i in 1:length(link)){
  Sys.sleep(runif(1,1,3)) # 休息一下，以免被誤會成攻擊程式
  html <- htmlParse(getURL(link[i]))
  doc <- xpathSApply(html, "//div[@id='main-content']", xmlValue)
  doc <- removePunctuation(doc)
  doc <- gsub("[A-Za-z0-9]", "", doc)
  doc <- gsub(" ", "", doc)
  doc <- gsub("\n", "", doc)
  doc <- gsub("[.]|[?]|[!]|[:]|[：]|[。]|[，]|[→]|[※]|[♡]", "", doc)
  doc <- gsub("[～]|[「]|[」]|[！]|[、]|[（]|[）]|[：]|[？]|[…]", "", doc)
  doc <- gsub("[【]|[】]|[／]|[《]|[》]", "", doc)
  article[i] <- doc
}
length(article)

# 刪掉一些Stop Words後斷句，然後將斷句後的結果予以儲存
seg <- worker(stop_word="Stop_Tw.txt")
corpus <- NULL
for(i in 1:length(article)){
  corpus[[i]] <- segment(article[i], seg)
}
length(corpus)
corpus[[170]]
edit_dict() #別忘了要重新載入新詞庫

# 產生詞袋（Bag of Words）
union <- NULL
for(i in 1: length(article)){
  union <- c(union, corpus[[i]])
}


# 去掉重複的
bag <- unique(union)
length(union)
length(bag)

# 文件集中有多份文件，寫個迴圈來執行
tf <- matrix(0, nrow=length(article), ncol=length(bag), dimnames=list(NULL,bag))
new.tf1 <- tf
new.tf2 <- tf
for(i in 1:length(corpus)){
  matchID <- match(corpus[[i]], bag)
  len <- length(matchID)
  for(j in 1:len){
    new.tf1[i, matchID[j]] <- 1
    new.tf2[i, matchID[j]] <- (new.tf2[i, matchID[j]] + 1)
  }
}

# 計算每個token在文件集中出現的次數
new.tf1[1:10, 1:10]
tf_count <- colSums(new.tf1)
tf_count <- sort(tf_count, decreasing = TRUE)
tf_count[1:5]
# idf
idf <- 1 + log(nrow(new.tf1)/colSums(new.tf1))
```

```
# 完成TFIDF表格
tfidf <- new.tf2
for(word in names(idf)){
  tfidf[,word] <- tfidf[,word] * idf[word]
}
tfidf[1:10, 1:10]

# 算相關性
cor_term <- cor(new.tf1)
cor_guardian <- cor_term["高雄",]
cor_guardian <- sort(cor_guardian, decreasing = TRUE)
cor_guardian[1:5]
cor_guardian <- cor_term["口罩",]
cor_guardian <- sort(cor_guardian, decreasing = TRUE)
cor_guardian[1:5]
cor_guardian <- cor_term["防疫",]
cor_guardian <- sort(cor_guardian, decreasing = TRUE)
cor_guardian[1:5]
cor_guardian <- cor_term["疫情",]
cor_guardian <- sort(cor_guardian, decreasing = TRUE)
cor_guardian[1:5]
cor_guardian <- cor_term["戴",]
cor_guardian <- sort(cor_guardian, decreasing = TRUE)
cor_guardian[1:5]
```

結果：

以出現次數為主的 Term Frequency（TF）：

（由於十分大，因此選擇只取 10*10 的大小）

```
> new.tf1[1:10, 1:10]
      防疫 規範 餐飲 疫情 爆發 外食 外帶 配戴 口罩 高雄
 [1,]    1    1    1    1    1    1    1    1    1    1
 [2,]    0    0    0    0    0    0    0    0    0    0
 [3,]    0    0    0    0    0    0    0    0    1    1
 [4,]    1    0    1    1    0    0    0    1    1    1
 [5,]    0    0    0    1    0    0    0    0    0    0
 [6,]    1    0    0    1    1    0    1    0    1    1
 [7,]    1    0    0    1    1    0    0    0    1    1
 [8,]    1    0    0    1    0    0    1    0    0    1
 [9,]    0    0    0    0    0    0    0    0    0    1
[10,]    1    0    0    1    1    0    0    0    1    1
```

TFIDF：

（由於十分大，因此選擇只取 10*10 的大小）

```
> tfidf[1:10, 1:10]
          防疫      規範      餐飲      疫情      爆發     外食      外帶      配戴      口罩      高雄
 [1,] 4.975385 8.688078 15.754295 6.725805 5.679923 12.2716  9.736280 11.21371  8.235810  8.259800
 [2,] 0.000000 0.000000  0.000000 0.000000 0.000000  0.0000  0.000000  0.00000  0.000000  0.000000
 [3,] 0.000000 0.000000  0.000000 0.000000 0.000000  0.0000  0.000000  0.00000  1.647162  1.179971
 [4,] 6.633846 0.000000  3.938574 3.362902 0.000000  0.0000  0.000000 11.21371 26.354593  7.079828
 [5,] 0.000000 0.000000  0.000000 1.681451 0.000000  0.0000  0.000000  0.00000  0.000000  0.000000
 [6,] 3.316923 0.000000  0.000000 3.362902 2.839962  0.0000 16.227133  0.00000  8.235810  3.539914
 [7,] 6.633846 0.000000  0.000000 8.407256 2.839962  0.0000  0.000000  0.00000  4.941486  3.539914
 [8,] 4.975385 0.000000  0.000000 1.681451 0.000000  0.0000  6.490853  0.00000  0.000000  1.179971
 [9,] 0.000000 0.000000  0.000000 0.000000 0.000000  0.0000  0.000000  0.00000  0.000000  1.179971
[10,] 9.950770 0.000000  0.000000 5.044353 2.839962  0.0000  0.000000  0.00000  8.235810 12.979685
```

網路聲量最大的 5 個關鍵字：

分別為高雄、口罩、防疫、疫情、戴

```
> tf_count[1:5]
高雄 口罩 防疫 疫情   戴
 142   89   88   86   81
```

找出與這 5 個關鍵字最有關的字詞(選前五個)：

與高雄最相關:

```
> cor_guardian[1:5]
     高雄      口罩      這樣      戴      違者
1.0000000 0.3384523 0.2978235 0.2966127 0.2866350
```

與口罩最相關:

```
> cor_guardian[1:5]
     口罩      戴      群聚      活動      不戴
1.0000000 0.8865307 0.5074095 0.4843830 0.4416153
```

與防疫最相關:

```
> cor_guardian[1:5]
     防疫      疫情      自己      確診      措施
1.0000000 0.4822715 0.4727614 0.4717999 0.4468505
```

與疫情最相關:

```
> cor_guardian[1:5]
     疫情      防疫      這樣      不會      自己
1.0000000 0.4822715 0.4467026 0.4405515 0.4204683
```

與戴最相關:

```
> cor_guardian[1:5]
      戴      口罩      群聚      活動      檢舉
1.0000000 0.8865307 0.5277058 0.4899595 0.4793763
```

討論：

讓我沒想到的是，幾乎熱門搜尋都跟疫情有關，但想想也不奇怪，因為這一兩個禮拜台灣疫情升溫，大家出於對病毒的警戒心，因此討論相關議題是非常正常的。

觀察結果可以發現，關鍵字高雄、口罩、防疫、疫情、戴以及他們的相關字元都是跟疫情沾上邊的，比方說群聚、確診、活動、檢舉，猜測應該是有網友抱怨有許多人都未遵守防疫規範因此想要去檢舉之類的話題。

但稍微可惜的是，原本是想對高雄有著更深入的了解，但因為疫情爆發使得整個關鍵字都被影響了，希望等疫情趨緩時可以把此程式碼再跑一次，相信到時候就能夠真正理解到網友最常討論高雄的什麼，並在未來有機會去觀摩觀摩。