VSM english essay grader with statics analysis

廖彥綸 (b05902001) 黃詠恩 (b07612008) 林軒誠 (b08209033) 梁仁謙 (b08203031)

I. 簡介

針對英文寫作能力的評量以往需要由英語專業人士評量。但人數遠遠不足，且受到時間及空間的限制。本文將英文寫作能力分為四級，包括英文寫作新手 (約為學測英文作文 10 級分以下)、中等英文寫作能力 (約為學測英文作文 11 級分至 16 級)、高級英文寫作能力 (約為學測英文作文 17 級以上) 及母語為英文人士的作品。並對於四個類別進行統計上的分析。最後提出語言模型對任一篇文章的內容進行評分。如果能限縮資料至單一主題的文章，評分結果將更加準確。

II. 資料取得

文章取至於 github 上文章評分的資料庫 (Autograde)[1]、學測英文寫作優秀範文 [3]、母語人士作文 [2]、中國地區小學生作文、以及組員的寫作練習 (經過專業教師評分)。一共 312 篇文章。但上述網站儲存資料的方式不同，文章需要經過前處理才能使用。

III. 原始碼運作說明與結果

A. 使用文字平均長度

* 創造每篇文章的 bag of words

```
bagofbeginner <- str_split(beginner$text[c
    (1:94,96:100)], " ")
```

* 計算平均文字長度，使用 magic_for() 建立為 dataframe

```
magic_for(print, silent = T)

for (i in 1:99){
  avg_word_length <- mean(nchar(bagofbeginner
      [[i]]))
  print(avg_word_length)
}

(beginnerwl <- magic_result_as_dataframe())
```

* 畫散佈圖，分割不同英語程度

```
ggplot(whole) +
  geom_point(aes(x = number, y = avg_word_
      length), color = "#DB481F") +
  geom_smooth(aes(x = number, y = avg_word_
      length)) +
  geom_vline(aes(xintercept = 99)) +
  geom_vline(aes(xintercept = 139)) +
  geom_vline(aes(xintercept = 248)) +
  labs(title = "Average Word Length", x = "
      English-Level" , y = "avg.")
```

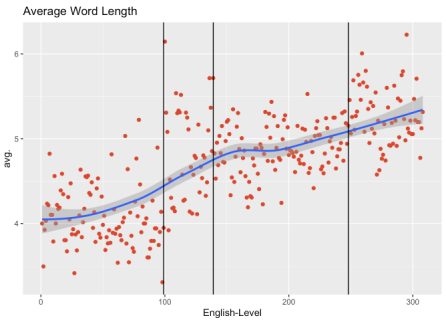

Fig. 1: 使用文字平均長度

B. 文字豐富度 (詞頻)

文字豐富度代表一篇文章中，相異的文字的數量與整篇文章中的文字總數量之比值。

將使用 paste 把分成數行的文章內容黏成一行後，將文章的標點符號清除 (使用 gsub) 以及大寫字母轉成小寫字母 (使用 tolower)。

```
post1_medium<-paste(post_medium,collapse = "
    ")
post2_medium<-gsub("[[:punct:]]|[0-9]", "",
    post1_medium)
post3_medium[i]<-tolower(post2_medium)
```

使用 tibble() 建立 dataframe 後，用 corpus() 轉換成 Corpus object

```
docs_df_medium <- tibble::tibble(id = seq_
    along(fps_medium), content = post3_medium)
quanteda_corpus_medium <- corpus(docs_df_
    medium,
                          docid_field = "id",
                          text_field = "
                              content")
```

用 tokens() 取出各篇文章的 token 後，使用 textstat_lexdiv() 作出詞頻表

```
qcorp_tokens_medium <- tokens(quanteda_corpus_
    medium, "fastestword"
df_medium <-textstat_lexdiv(qcorp_tokens_
    medium)
```

將作出的詞頻表依照數值分成 0.2 0.3、0.3 0.4、0.4 0.5、0.6 0.7、0.7 0.8、0.8 0.9、0.9 1 共七類

```
df_medium<-df_medium%>%mutate(TTR_percent=
    ifelse(TTR>=0.9, "0.9~1",
                     ifelse(TTR>=0.8&TTR<0.9 ,
                         "0.8~0.9",
                     ifelse(TTR>=0.7&TTR<0.8 ,
                         "0.7~0.8",
                     ifelse(TTR>=0.6&TTR<0.7 ,
                         "0.6~0.7",
                     ifelse(TTR>=0.5&TTR<0.6 ,
                         "0.5~0.6",
```

```
6                    ifelse(TTR>=0.4&TTR<0.5 ,
                         "0.4~0.5",
7                    ifelse(TTR>=0.3&TTR<0.4 ,
                         "0.3~0.4","0.2~0.3")))
                         )))))
```

使用 ggplot 做出不同詞頻的數量長條圖

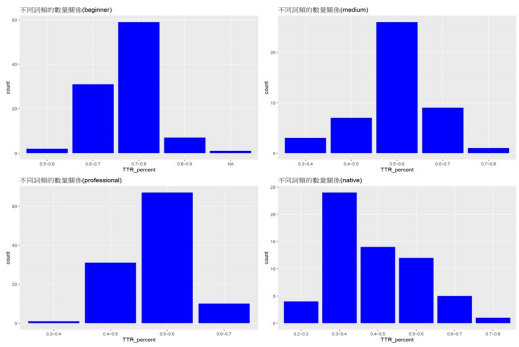

Fig. 2: 不同詞頻的數量關係圖

C. 去除功能詞之文字雲

使用 tokens_select() 將上述取出 token 後的資料去除功能詞，再使用 textplot_wordcloud 繪製成文字雲

```
1 toks_nostop_medium <- tokens_select(qcorp_
      tokens_medium, pattern = stopwords('en'),
      selection = 'remove')
2 document_term_matrix_medium <- dfm(toks_nostop
      _medium)
3 textplot_wordcloud(document_term_matrix_medium
      ,max_words = 25,min_size = 1, max_size =
      5,random_color = TRUE)
```

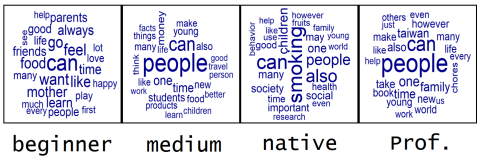

Fig. 3: 去除功能詞之文字雲

D. 常用介係詞和連接詞使用頻率

前處理:

使用 Python 寫前處理之主程式 (niceboat.py), 內附兩個副程式

建立絕對路徑, 讀入檔案

```
1 abspath_begin = ["" for x in range(len(
      files_begin))]
2 for i in np.arange(len(files_begin)):
3     abspath_begin[i] = filepath_begin + '\\' +
          files_begin[i]
4 for path_begin in abspath_begin:
5     txt = open(path_begin,encoding="utf-8")
```

副程式 clearnods.py 負責去除標點符號並統一格式

```
1 Punctuation_Marks =
      ('.',',',']','"',"'",'!',',',':','?','Â´
      ','["',',"']')
2 def clear(file):
3     for i in np.arange(len(Punctuation_Marks))
          :
4         file = file.replace(Punctuation_Marks[
              i],' ')
5         return file
```

副程式 freqcount.py 將切割為單字的文本讀入, 使用迴圈找到特定單字並記錄其出現次數, 回傳總出現次數

```
1 def counter(txt,array):
2     for i in np.arange(len(txt)):
3         for j in np.arange(len(conj_and_prep))
              :
4             if txt[i] == conj_and_prep[j]:
5                 array[j] = array[j] + 1
6             else:
7                 continue
8     return array
```

統一格式, 使用副程式整合, 結合各語言程度之頻率

```
1 for path_begin in abspath_begin:
2     txt = open(path_begin,encoding="utf-8")
3     str_file = txt.readlines()
4     str_file = str(str_file)
5     str_file = str_file.lower()
6     str_file = cr.clear(str_file)
7     str_file = str_file.split(' ')
8     freq_begin = count.counter(str_file,
          freq_begin)
9     freq = [0]*60
10    freq[0:14] = freq_begin
11    freq[15:29] = freq_medium
12    freq[30:44] = freq_professional
13    freq[45:59] = freq_native
```

輸出第一行 (單字), 第二行 (語言程度) 及第三行 (次數) 至 Excel(by column)

```
1 array = [conj_and_prep*4,
2          kind,
3          freq]
4 df = pd.DataFrame(array).T
5 df.to_excel(excel_writer =
6 r"C:\Users\n8748\Desktop\freq_str.xlsx")
```

數據處理和圖像輸出:

使用 R 語言, 讀入 Excel 之表格並處理行名稱和"0" 數據, 加入新欄位"percentage" 繪圖用

```
1 library(readxl)
2 library(ggplot2)
3 library(tidyverse)
4 freq_str <- read_excel("C:/Users/n8748/Desktop
      /freq_str.xlsx")
5 freq_str$...1 <- NULL
6 names(freq_str)[1] <- "string"
7 names(freq_str)[2] <- "type_of"
8 names(freq_str)[3] <- "times"
9 freq_str <- freq_str[-c(61, 62, 63, 64), ]
10 freq_str <- freq_str %>%
11   group_by(type_of) %>%
12   mutate(percentage = times/sum(times))
```

使用 ggplot2 繪圖,x 軸為詞,y 軸為出現次數, 以各詞作為填色, 以語言程度作為分別做出四個圖並對調 xy 軸, 然後輸出

```
1 pp <- ggplot(data = freq_str) +
2   geom_bar(mapping = aes(x = string,y =
        percentage,fill = string),stat = '
        identity') +
```

```
3   facet_wrap(vars(type_of)) +
4   labs(title = "String Frequency by Language
        Levels",x = 'String', y = 'Frequency') +
5   theme(plot.title = element_text(size = 22),
6         axis.title = element_text(size = 16))
              +
7   coord_flip()
8   ggsave("freq_str_by_type.png")
```

輸出圖 Fig.3

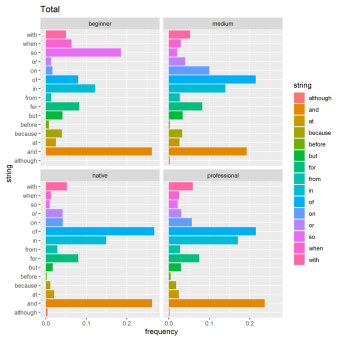

Fig. 4: 常用介系詞和連接詞之使用頻率

E. 這行放標題

F. *VSM* 模型

VSM(vector space model) 是衡量兩篇文章相似度的一種模型。可以透過將文章投影成向量的形式，之後比較文章中向量內積的最大值以找出關聯性高的文章。

這個模型前處理的部分由於只注重單詞，移出所有標點符號。以下列虛擬碼可以達到這目的。並做了適當的文字簡化 (統一成小寫)。

```
1 def clean_str(context):
2     replace_str = .,%()\;:?!"'
3     context = context.replace(replace_str, '
          ') #取代成空格
4     return context.lower()
```

最基本的 VSM 模型是使用 tf、idf 兩個變量控制。tf 表示某一詞在一篇文章中的重要性，通常愈常出現表示愈重要。idf 表示某一詞在所有文章中的重要性，通常一詞在愈多篇文章出現表示愈不重要。詳細公式如下：tf 對文章加種詞的總數後用總長度做標準化；idf 該詞為出現在所有文章中比例的倒數後取對數。

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

$$idf_i = \log \frac{|D|}{|\{j : t_i in d_j\}|}$$

程式碼及分別取四個寫作等級和測試文章的 tf-idf 值後輸出結果。

```
1 def tf-idf(context, degree):
2     for word in context:
3         score[word] += idf * idf
4     return score[word].sort()
```

為了避免除以 0 的情況，在 tf 及 idf 項皆會加上一個極小的值。另外為了避免單詞的頻率過於主導結果，實作時加上 bi-gram，評分則以 bi-gram 符合的情況得到較高的加權分數。

G. *essay eye* 文眼分析

按照上述方式前處理過後的資料可以同等用於找出每篇文章最重要的幾個字，又稱為關鍵字或文眼。

對使用單字的評分和 tf-idf 的衡量方式相同但有針對我們資料中未收錄的單字分數做加重。進行文演分析時前處理將 idf 分數最低的前 50 名剔除之後再做比較，在資料量少的情況下這種方式可以過濾掉常用字，更精確地分析出文眼。若取得的資料可以達到十萬筆以上，idf 對於慣用字的平衡效果會更加顯著。過濾的步驟即可刪除。

```
1 if word in total_dict:
2     score[word] += tf * idf * 1 *idf
3 else:
4     a * math.log(essay_num ,10) * math.log(
          essay_num, 10) / total_len
5 # a > 1
```

這部分是為了看出不同的使用者喜觀使用的重要字詞所以不經過過濾，僅用 idf 的方式平衡出現次數的加權。如圖：寫作新手可能因為常以日記等方式寫作，I 出現的次數最高。程度中間的寫手具備基本的文法知識，所以他們不會忘記在每個名詞前放上"the"，但也使文章略為枯燥。高級寫手或母語人士使用單一種字詞的習慣趨緩，也可以解釋為他們擅長靈活用不同字詞表達同一想法。

Fig. 5: 不同程度的英文寫手習慣的用詞圖

IV. 討論與貢獻

本文對不同程度的英語寫作人士的習慣做詳細分析並建構出一個預測模型，可以達到以下功用。一、可讓英語學習者藉由客觀統計資料學習，精進寫作能力。二、英語寫作評量系統開方所有人使用，在沒有專業師資在旁時依舊能夠對文章評分。三、但評分系統較適合對特定組題進行分析，如果能和大考中心或補習班合作能到所有分數分布的文章，評分可信度能大大提升。四、用詞長度、用字習慣等特徵皆納入衡量，以不同面向評量文章。

V. 附錄

A. 組員分工

1. 廖彥綸：資料前處理、VSM 模型、essay eye 文眼分析、寫手慣用詞文字雲、Readme 說明文件。
2. 梁仁謙：收集文本、文字豐富度分析、去除功能詞之文字雲、標點符號數量分析
3. 林軒誠：收集文本、資料前處理、介係詞和連接詞之使用頻率
4. 黃詠恩：收集文本、資料前處理、使用文字平均長度、印海報

Reference

[1] ChiragSoni95. Autograder. *https://github.com/ChiragSoni95/Autograder*.
[2] native people. article of native people. *https://www.ukessays.com/*.
[3] G. students. College entraance exam. *https://drive.google.com/drive/ folders/ 0BxEC0dgV3WQQNHBUZzREQTZUQ3M? fbclid=IwAR2oWP*$_1$*7pbs4vuKLCCzHefcNARNE4iNR2x6iDnwAuOmiITZx4ULBvhuHMc*.