國立東華大學財務金融學系

文字探勘期末作業

應用情感分析於商品評論之研究－以 Amazon 電商為例

A Study on Using Sentiment Analysis to Evaluate Product Reviews —Take Amazon E-commerce as an example

學生：賴冠霖

系級：財金二

學號：410936051

指導教授：林金龍 博士

中華民國 111 年 6 月

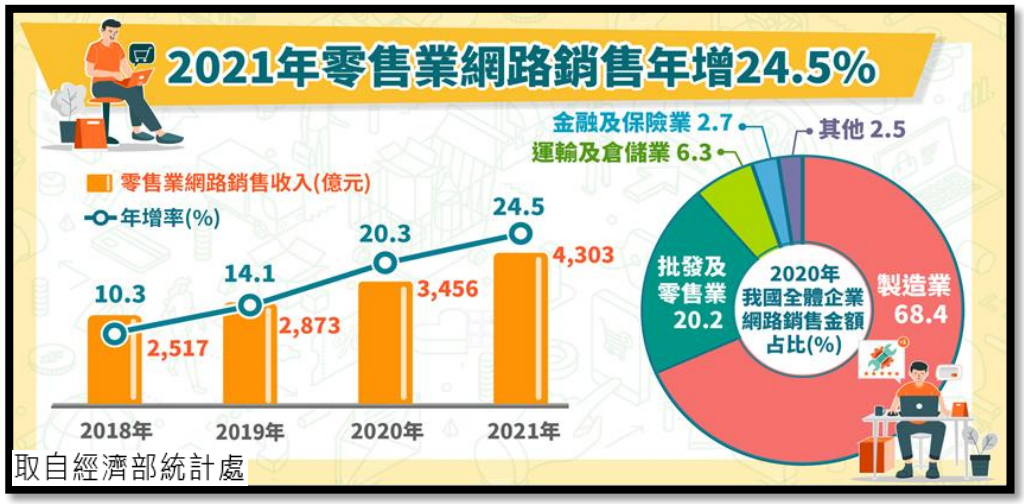
June, 2022

目錄

一、研究動機

隨著互聯網的快速發展，電子商務突飛猛進，網購成為人們消費必不可少的渠道。線上購物越來越流行，人們對於網上購物的需求變得越來越高，這讓蝦皮，Amazon 等電商平台得到了很大的發展機遇。而這種需求也推動了更多電商平台的發展，引發了激烈的競爭。在這種電商平台激烈競爭的背景下，除了提高商品質量，壓低商品價格外，了解消費者的心聲對於電商平台來說也越來越重要。大多數的電子商務網站都會為消費者提供相互交流的平台來發表其針對所購買商品的評論，因此其中非常重要的一種方式就是針對消費者所購買商品的評論數據進行內在信息的數據挖掘分析。

過去許多研究即指出情緒分析(Sentiment Analysis)有助於了解群眾的情感狀態、對特定事物的態度與觀點。這將有助於產品行銷、人機互動等應用價值。利用情緒分析方法來分析網絡商品評論的情感極性，是獲取顧客對該種類商品反饋的最直接方式，商家可以通過分析評論獲取顧客對所購商品的感受。這樣不僅可以為後續銷售計劃變更和產品改進及時作出決策；消費者也可以前瞻性地借鑒這些觀點作為購物參考依據。



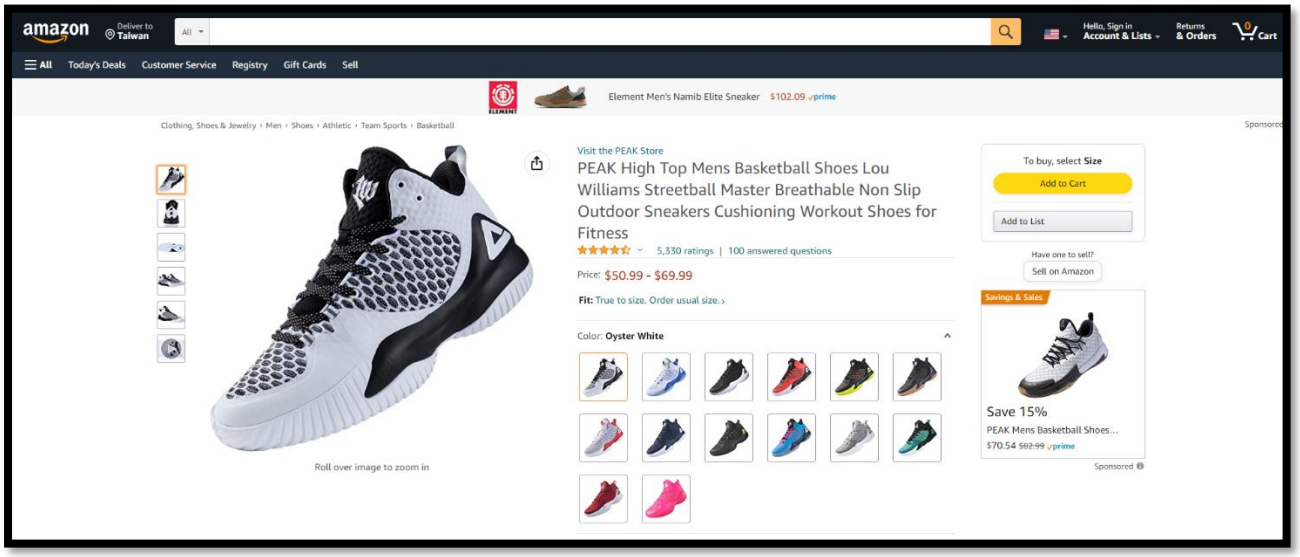
二、研究目的

此研究專案運用 Python 爬取網絡評論以進行文本分析，並透過主題建模演算法 LDA 以及 Python 情緒分析套件快速地分析出主題以及情感傾向。可以為消費者提供購物參考，也能幫助賣家完善商品質量，提高服務水平。

三、研究方法

本研究專案利用 R & Python 語言做文字探勘(Text mining)，針對籃球鞋商品的購買評論做資料收集、整理、並做文字處理以及情緒分析。情感傾向利用 R 做 LDA 主題分析以及 Python 的 TextBlob 套件來做情感分析。利用所得出的分析，分析消費者購買此商品的情緒、得出正負面評論的相應主題，並提出可行性建議。

- **分析商品：**



四、文獻探討

第一節 情緒分析

情緒分析(Sentiment Analysis)是用自然語言處理(Natural Language Processing, NLP)、資訊萃取與結構化和非結構化之資料探勘等技術來辨識和萃取文章中的主觀訊息(Kumar & Vadlamani, 2015)，也可以說情緒分析是一種跨學科的技術挑戰，用來確定文本中表達的情緒，並決定它們是否為正向或負向(或中立)。

Isidoros 與 loannis(2016)指出情緒分析的目的是為了瞭解人們透過不同媒介對某主題發表的意見極性，例如演說、臉部表情、手語或文字資料...等，而最常用來表達情緒的媒介是文字資料，主題來源可以是產品、服務、組織、個人、事件、時事議題及個體的屬性。在文本中，形容詞最常用來表達作者的意見極性，與個人主觀有很強的相關性。情緒分析是一個概念，涵蓋了許多任務，例如提取情

绪、情緒分類、主觀分類、意見總結或意見垃圾郵件檢測等(Jesus, Jose, Francisco & Enrique, 2015)。由此可知情緒分析已被應用在多個領域，它可幫助需求者更快速了解大眾對某特定事物或議題的意見情緒。

第二節 情緒分析方法

情緒分析主要是透過辨識文本中的特徵詞和意見詞的配對關係以獲取情緒極性，其中意見詞也有人稱之為情緒詞(Sentiment Keyword)。情緒分析處理方法可分為機器學習法和字典法。

機器學習法可依不同的應用範圍分為監督式和非監督式兩大類。一般來說，從文本中提取複雜的特徵，找出哪些特徵是相關的，並且選擇分類演算法是機器學習法中的基本問題。監督式機器學習法需要提供大量已標記過的訓練資料讓機器學習，根據學習函數透過特徵向量或標籤來反覆學習，例如支援向量機(Support Vector Machine,SVM)。過去已有研究指出結合 SVM 演算法的情緒分析其精確率會顯著的提高(Emma, Xiaohui & Yong, 2013)。

非監督式學習法則不需大量人工標記的訓練資料，而是利用特定模式來分析文本，將高相似度的字詞分群群聚，群體彼此之間的相似度較低，同一群體內的相似度較高，以達到分群的效果，例如隱含狄利克雷分布(Latent Dirichlet Allocation, LDA)即為典型的非監督式學習法。LDA 是一種以 Topic Model 為基礎概念發展的模型，可在一系列的文本語料庫中產生離散數據集合概率模型，即 LDA 的原理為一篇文本是由多組字詞構成的集合，詞與詞之間沒有先後順序關係(徐筱雁，2014)。

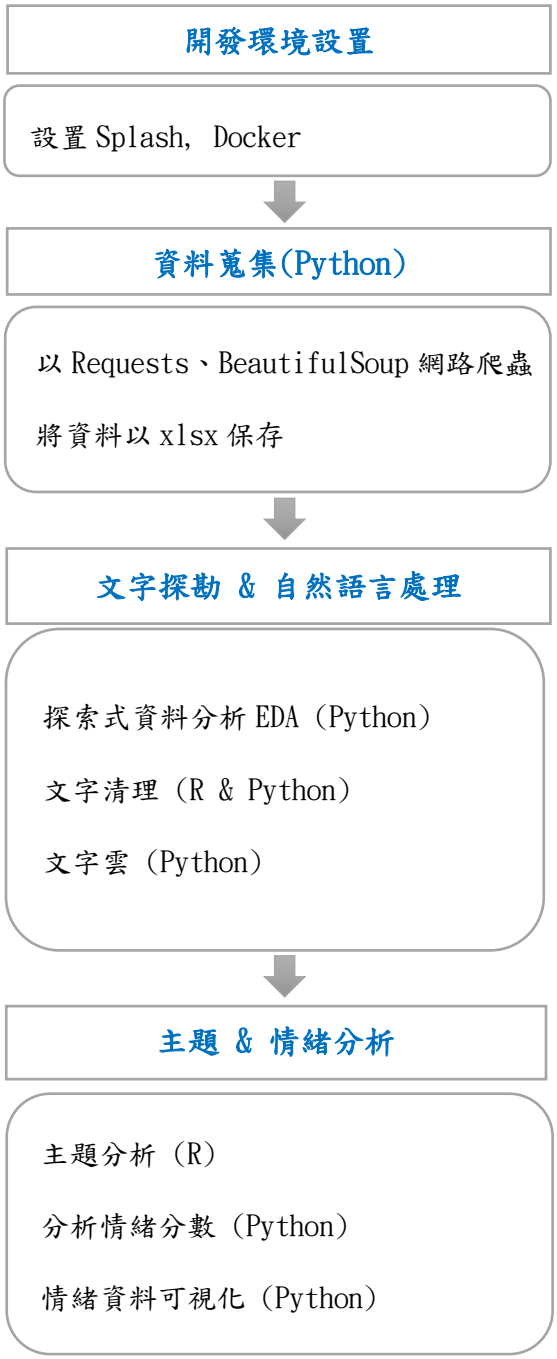
第三節 LDA 主題模型

隱含狄利克雷分布(LDA)，是一種主題模型，它可以將文本集合中每篇文本的主題按照機率分布的形式给出。同時它是一種非監督式學習演算法，在訓練時不需要人工標注的訓練集，只需要文本集以及指定主題的數量即可。此外 LDA 的另一個優點則是，對於每一個主題均可找出一些字詞來描述它。

LDA 是一種主題模型分析的方式，廣泛地被應用在機器學習和資訊檢索等

相關領域。近年來由於社群網路(Social network)的興起，許多學者將 LDA 應用在這些領域並擷取出欲研究主題進而去分析、探討；在此 LDA 扮演在大量文本中發掘潛在主題的工具。

五、研究與程式流程



六、程式與內容說明

➢ Step 1：爬蟲環境設置

網路上關於建置 Docker 跟 Splash 的教學文章非常多，因此在期末研究專案這邊就不再針對如何設置這兩個應用多做說明了。我們來談談為何這個研究專案會需要用到這兩個工具，以及這兩個工具有什麼用途。

Q: 為何會使用 Docker 跟 Splash ?

Splash: 大部分的 web 頁面都是動態的，是經過 Javascript 渲染的。所以無法直接用 Python 套件做網路爬蟲，我們需要透過 Splash 幫助我們獲得生成的動態頁面。

Docker: 幫助我們建置及運行 Splash。

➢ Step 2：資料蒐集

標的：Peak Basketball Shoes 之商品評論

步驟：

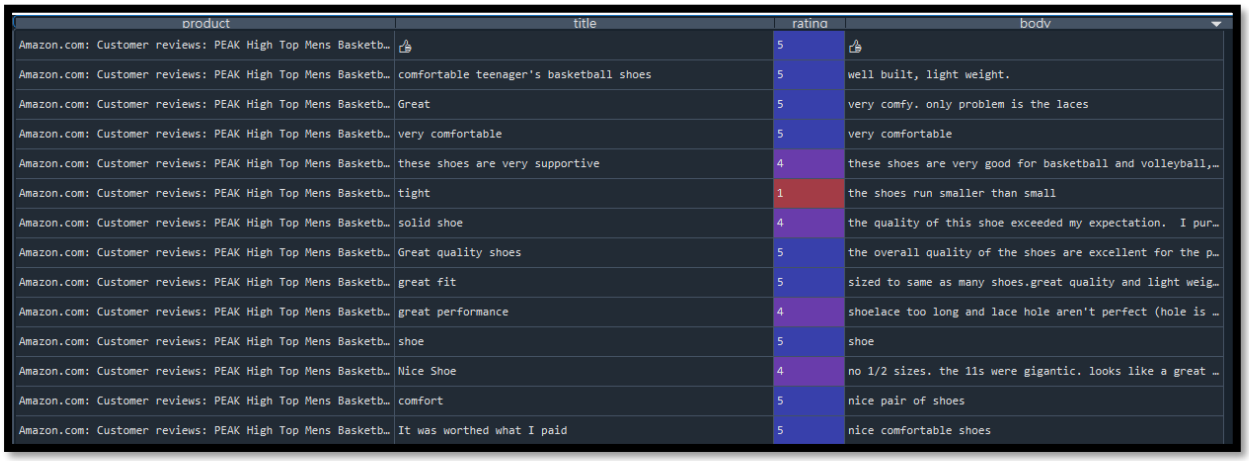
1. Requests 載入我們指定的網頁 html 結構
2. 進行 BeautifulSoup 的爬取(開始爬取網站之前一定要觀察該網頁的變化)
3. 將爬蟲結果保存成 xlsx 格式

```
# 網路爬蟲

import requests
from bs4 import BeautifulSoup
import pandas as pd
reviewlist = [] #存評論
# 爬網頁訊息function
def get_soup(url):
    r = requests.get('http://localhost:8050/render.html', params={'url'

: url, 'wait': 2}) #載入指定網頁的html 結構
    soup = BeautifulSoup(r.text, 'html.parser')
    return soup
```

```
#抓評論function
def get_reviews(soup):
    reviews = soup.find_all('div', {'data-hook': 'review'})
    try:
        for item in reviews:
            review = {
            'product': soup.title.text.replace('Amazon.co.uk:Customer reviews:
', '').strip(),
            'title': item.find('a', {'data-hook': 'review-title'}).text.strip
(),
            'rating':  float(item.find('i', {'data-hook': 'review-star-rating
'}).text.replace('out of 5 stars', '').strip()),
            'body': item.find('span', {'data-hook': 'review-body'}).text.strip
(),
            }
            reviewlist.append(review)
    except: #例外處理
        pass

# 自動翻頁抓取評論
for x in range(1,999):
    soup = get_soup(f'https://www.amazon.com/PEAK-Basketball-Streetball-Breatha
ble-Cushioning/product-reviews/B088LP6Y8J/ref=cm_cr_getr_d_paging_btm_prev_1?i
e=UTF8&reviewerType=all_reviews&pageNumber={x}')
    print(f'Getting page: {x}') #x:頁數，隨翻頁變動
    get_reviews(soup)
    print(len(reviewlist))
    if not soup.find('li', {'class': 'a-disabled a-last'}): #翻到評論最後一頁會停
止
        pass
    else:
        break

#將爬蟲程序進度印出來

## Getting page: 1 ##目前爬取到評論的一頁
## 10 #目前累積爬取10則評論
## Getting page: 2
## 20
```

```
## Getting page: 3
## 30

~略過~

## Getting page: 84 #共抓取了84 頁評論

## 802 #共抓取了802 條評論

#將爬蟲結果保存成 xlsx 格式

df = pd.DataFrame(reviewlist)
df.to_excel('scrape_text.xlsx', index=False)
print('Finish') #輸出檔案完成印Finish

## Finish
```

爬取之評論資料(前幾筆資料)：



product	title	rating	body
Amazon.com: Customer reviews: PEAK High Top Mens Basketb…	👍	5	👍
Amazon.com: Customer reviews: PEAK High Top Mens Basketb…	comfortable teenager's basketball shoes	5	well built, light weight.
Amazon.com: Customer reviews: PEAK High Top Mens Basketb…	Great	5	very comfy. only problem is the laces
Amazon.com: Customer reviews: PEAK High Top Mens Basketb…	very comfortable	5	very comfortable
Amazon.com: Customer reviews: PEAK High Top Mens Basketb…	these shoes are very supportive	4	these shoes are very good for basketball and volleyball,…
Amazon.com: Customer reviews: PEAK High Top Mens Basketb…	tight	1	the shoes run smaller than small
Amazon.com: Customer reviews: PEAK High Top Mens Basketb…	solid shoe	4	the quality of this shoe exceeded my expectation. I pur…
Amazon.com: Customer reviews: PEAK High Top Mens Basketb…	Great quality shoes	5	the overall quality of the shoes are excellent for the p…
Amazon.com: Customer reviews: PEAK High Top Mens Basketb…	great fit	5	sized to same as many shoes.great quality and light weig…
Amazon.com: Customer reviews: PEAK High Top Mens Basketb…	great performance	4	shoelace too long and lace hole aren't perfect (hole is …
Amazon.com: Customer reviews: PEAK High Top Mens Basketb…	shoe	5	shoe
Amazon.com: Customer reviews: PEAK High Top Mens Basketb…	Nice Shoe	4	no 1/2 sizes. the 11s were gigantic. looks like a great …
Amazon.com: Customer reviews: PEAK High Top Mens Basketb…	comfort	5	nice pair of shoes
Amazon.com: Customer reviews: PEAK High Top Mens Basketb…	It was worthed what I paid	5	nice comfortable shoes

➢ Step 3：文字探勘分析&自然語言處裡

1 讀取資料

```
#讀取資料

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib as plt
import matplotlib.pyplot as plt
import matplotlib.font_manager
plt.rcParams['font.sans-serif'] = ['Taipei Sans TC Beta']
pd.options.mode.chained_assignment = None  # default='warn'
```

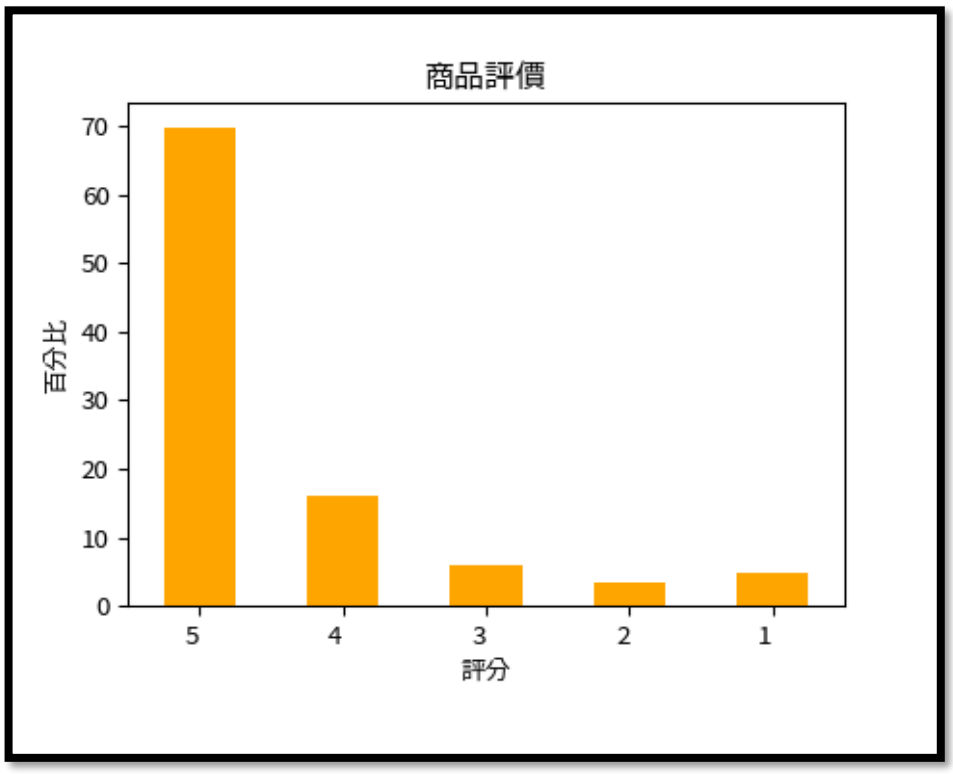
```
df = pd.read_excel("scrape_text.xlsx")
#df = df.dropna()
```

2 探索式資料分析 EDA

```
# EDA

print("Shape of the dataset:")

## Shape of the dataset:

print(df.shape)

## (802, 4)

print("Column names:")

## Column names:

print(df.columns)

## Index(['product', 'title', 'rating', 'body'], dtype='object')

print("Datatype of each column:")

## Datatype of each column:

print(df.dtypes)

## product    object
## title      object
## rating      int64
## body       object
## dtype: object

# ANALYSIS OF RATING
#distribution of ratings across dataset

rating_pct = df['rating'].value_counts()/len(df) * 100
rating_pct = round(rating_pct,2)
rating_pct = rating_pct.sort_index(ascending = False)
print(rating_pct)
## 5    69.83
## 4    16.08
## 3     5.99
```

```
## 2     3.37
## 1     4.74
## Name: rating, dtype: float64
```

```
#plot

fig,ax = plt.subplots()
rating_pct.plot.bar(color ="orange")
plt.title("商品評價")
plt.xlabel("評分")
plt.ylabel("百分比")
fig.autofmt_xdate(rotation=0)
plt.show()
```

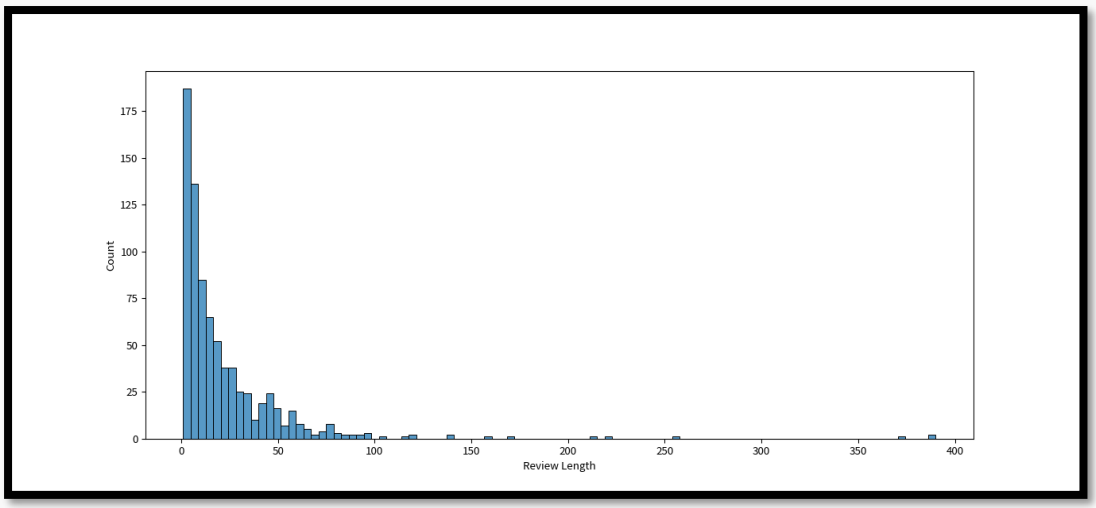



```
# Rating Summary

print(df['rating'].describe(include='all'))
```

```
## count    802.000000
## mean       4.428928
## std        1.066133
## min        1.000000
```

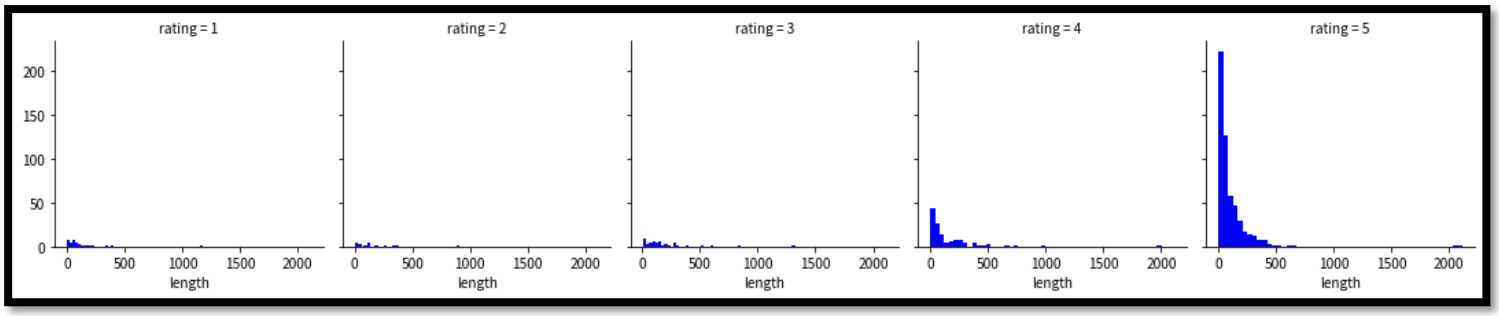
```
## 25%        4.000000
## 50%        5.000000
## 75%        5.000000
## max        5.000000
## Name: rating, dtype: float64
```

```
# ANALYSIS OF REVIEWS
#Words per review
df = df.dropna()
plt.figure(figsize = (14,6))
WordsPerReview = df['body'].apply(lambda x: len(x.split(' ')))

sns.histplot(WordsPerReview,bins = 100)
plt.xlabel('Review Length')
plt.show()
```



圖示說明：Words per review

```
# Creating a new column in the dataset for the number of words in the re
view

df['length'] = df['body'].apply(len)
 # Comparing text length to stars
graph = sns.FacetGrid(data=df,col='rating')
graph.map(plt.hist,'length',bins=50,color='blue')
```

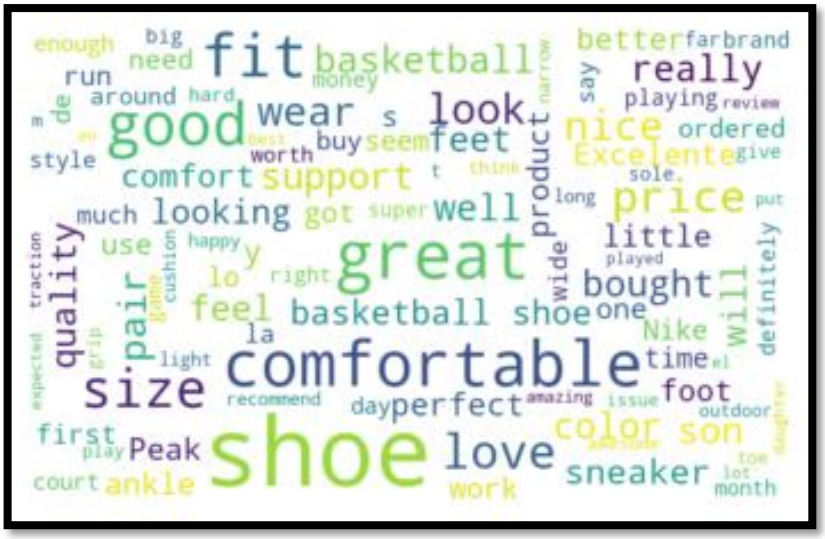


圖示說明：此圖可以幫助我們了解評論文章篇幅長度跟評級之間的數量分布。我們可以發現給出五星評級的顧客最多且其寫的評論篇幅也最長。

3 文字雲(文字清理前)

使用 wordcloud 這個套件，並搭配 matplotlib 來畫圖。文字雲幫助我們快速的了解評論裡的重要關鍵字。

```
# 文字雲(文字清理前)

from wordcloud import WordCloud
txt = ' '.join(review for review in df.body)

wordcloud = WordCloud(
            background_color = 'white',
            max_font_size = 100,
            max_words = 100,
            width = 800,
            height = 500
            ).generate(txt)

plt.imshow(wordcloud,interpolation = 'bilinear')
plt.axis('off')

plt.show()
```



4-1 文字清理(Python)

這裡主要使用 string 和 nltk 進行文字處理，為後續我們運用 TextBlob 套件做情緒分析需要的文本資料做清理。

文字資料清理步驟：

1. 將句子斷詞
2. 刪去重複且不重要的字
3. 移除標點符號
4. 移除一些不是數字或英文的字
5. 將所有單字小寫
6. 將詞意相近的字歸類為相同的字
7. 回傳結果

```
# 文字清理(python)

from nltk.corpus import stopwords
from nltk import wordpunct_tokenize
from nltk.stem import WordNetLemmatizer
import string
import nltk
nltk.download('stopwords')
```

```
nltk.download('wordnet')

#string 和 nltk 進行文字處理

def clean_review(review_text, *arg):
     allwords = list()
     for sentence in review_text:
         punct_token = wordpunct_tokenize(sentence)

         #remove stopwords
         stops = set(stopwords.words("english"))
         stops.update({'br'})
         punct_token = [word for word in punct_token if word not in
         stops]
         #remove string.punctuation
         punct_token = [word for word in punct_token if word not in
         string.punctuation]
         #remove word that is not alphabat or number
         punct_token = [word for word in punct_token if
         word.isalnum()==True]
         #lower words
         punct_token = [word.lower() for word in punct_token ]
         #stem words
         punct_token = [WordNetLemmatizer().lemmatize(word) for word in
         punct_token]
         allwords.append(' '.join(punct_token))
     return allwords
 clean_tokens = clean_review(df['body'])
 df_clean = pd.DataFrame(clean_tokens)
 df_clean = df_clean.rename(columns={0:'body'})
```

4-2 文字清理（R）

這裡進行文字處理的概念大致與 Python 相同，因此我們這邊就不再多做說明了。這步驟是為了將後續我們運用 LDA 演算法所需要的文本資料做清理。

1 read in our data

```
rm(list=ls())
library(tidyverse)
library(tidytext)
library(topicmodels) # LDA topic modelling
library(tm)
library(SnowballC)
library(readxl)
library(dplyr)
library(tm)
library(NLP)

# read in our data
dataset <- read_excel("scrape_text.xlsx")
dataset <-  na.omit(dataset)
dataset = rename(dataset,Text=body)

#head(dataset,100)
#dim(dataset)
```

2 Corpus

```
#Corpus
release_corpus <- Corpus(VectorSource(dataset$Text))
#inspect(release_corpus)

#delete numbers and punctuation
library(stringr)
release_corpus <- tm_map(release_corpus, removeNumbers)
release_corpus <- tm_map(release_corpus, removePunctuation)

#remove stop words
release_corpus <- tm_map(release_corpus, removeWords, words=stopwords("e
n"))

#character to lower
release_corpus <- tm_map(release_corpus, content_transformer(tolower))

#stemming
```

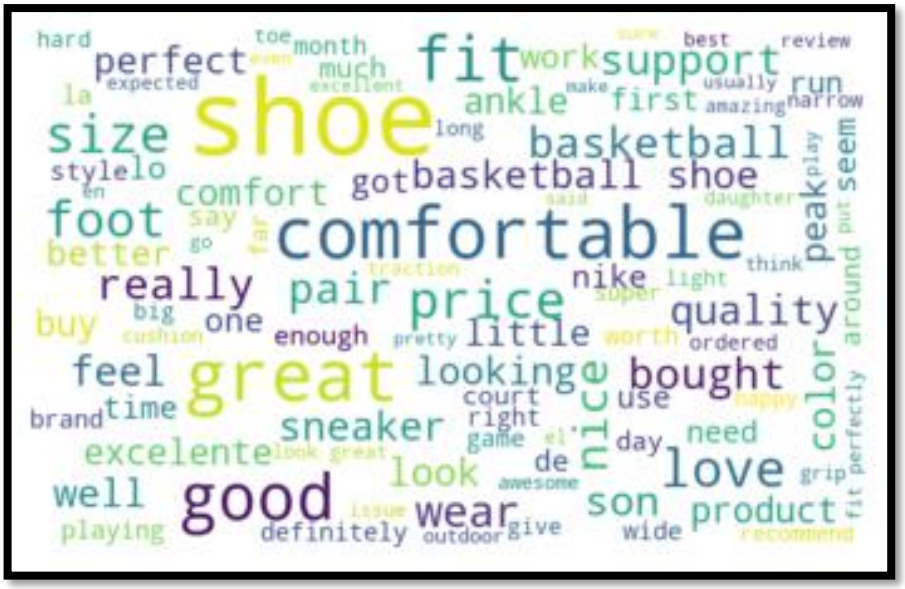
```
library(SnowballC)
release_corpus <- tm_map(release_corpus,stemDocument) # this function is
 to transform the plural words

#remove other unnecessary words
rmwords <- c("they","this","just","even","these","will","dont","much","a
mazon","amazoncom","give","first",
             "also","think","make","now","want","still","never","lot","go
t","thought","sure",
             "without","whenever","unlike","somehow","yes","tend","today
","the","have","day","not","but")
release_corpus <- tm_map(release_corpus, removeWords, words=rmwords)

tdm <- TermDocumentMatrix(release_corpus)
#tdm

#sparse terms
tdm <- removeSparseTerms(tdm,1-(10/length(release_corpus))) #term-docume
nt matrix
#tdm

inspect(tdm[1:20, 1:5])
```

```
     ## <<TermDocumentMatrix (terms: 20, documents: 5)>>
## Non-/sparse entries: 41/59
## Sparsity           : 59%
## Maximal term length: 7
## Weighting          : term frequency (tf)
## Sample             :
##         Docs
## Terms    1 2 3 4 5
##   arch   1 0 0 0 2
##   ball   1 1 0 0 1
##   brand  2 1 0 0 0
##   court  2 1 0 1 0
##   cushion 4 1 0 0 3
##   feel   1 3 0 0 0
##   fit    2 1 0 2 0
##   high   2 0 0 1 1
```

```
##   insol   1 2 0 0 0
##   ive     2 0 1 0 0

tdm1=(as.matrix(tdm))
#dim(tdm1)

freq_tdm <- rowSums(tdm1)
sort_tdm <- sort(freq_tdm, decreasing = TRUE) #frequency of words
#sort_tdm
summary(sort_tdm)

     ##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.00   13.25   19.00   34.86   33.75  512.00

     #inspect(release_corpus)
```

5 文字雲(Python 文字清理後資料)

```
# 文字雲(文字清理後)
from wordcloud import WordCloud

txt = ' '.join(review for review in df_clean.body)

wordcloud = WordCloud(
            background_color = 'white',
            max_font_size = 100,
            max_words = 100,
            width = 800,
            height = 500
            ).generate(txt)

plt.imshow(wordcloud,interpolation = 'bilinear')
plt.axis('off')

plt.show()
```



➢ Step 4-1：主題分析（R）

方法：使用 LDA 模型進行主題分析。

LDA（Latent Dirichlet Allocation）是一種文檔主題生成模型。所謂生成模型，就是說，我們認爲:

- 一篇文章以一定概率選擇了某個主題。
- 這個主題以一定概率選擇了某個詞語得到。
- 每一篇文檔代表了一些主題所構成的一個概率分佈，而每一個主題又代表了很多單詞所構成的一個概率分佈。

注意：LDA( )不會為每個主題分配任何標籤。它們只是主題 1,2,3 和 4...等等。

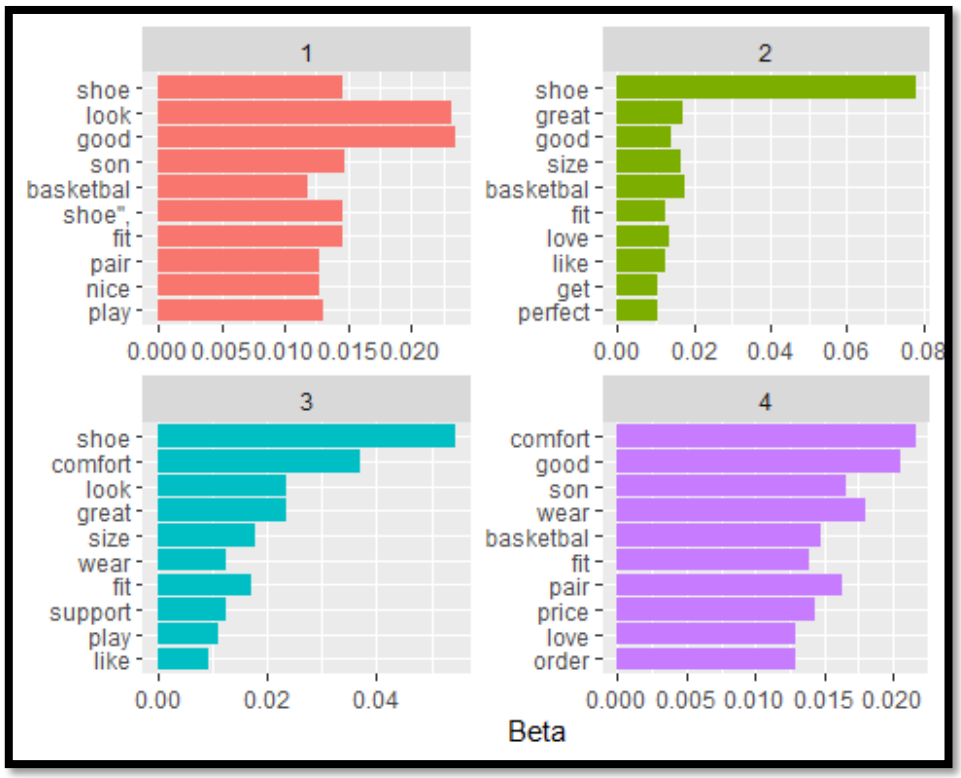
我們可以推斷這些與顧客評級或鞋子功能...有關，但它僅僅是我們的推論。

1 LDA model

```
#LDA model using package
library(topicmodels)
```

```r
# function to get & plot the most informative terms by a specificed numbe
r
# of topics, using LDA
top_terms_by_topic_LDA <- function(input_text, # columm from a dataframe
                                   plot = T, # return a plot? TRUE by defult
                                   number_of_topics = 4) # number of topics
 (4 by default)
{
  # create a corpus and document term matrix
  Corpus <- Corpus(VectorSource(input_text)) # make a corpus object
  DTM <- DocumentTermMatrix(Corpus) # get the count of words/document

  # remove any empty rows in our document term matrix
  unique_indexes <- unique(DTM$i) # get the index of each unique value
  DTM <- DTM[unique_indexes,]

  # preform LDA & get the words/topic in a tidy text format
  lda <- LDA(DTM, k = number_of_topics, control = list(seed = 1234))
  topics <- tidy(lda, matrix = "beta")

  # get the top ten terms for each topic
  top_terms <- topics  %>%
    group_by(topic) %>% # treat each topic as a different group
    top_n(10, beta) %>% # get the top 10 most informative words
    ungroup() %>%
    arrange(topic, -beta)

  # if the user asks for a plot (TRUE by default)
  if(plot == T){
    # plot the top ten terms for each topic in order
    top_terms %>% # take the top terms
      mutate(term = reorder(term, beta)) %>% # sort terms by beta value
      ggplot(aes(term, beta, fill = factor(topic))) + # plot beta by them
e
      geom_col(show.legend = FALSE) + # as a bar plot
      facet_wrap(~ topic, scales = "free") + # which each topic in a sepe
rate plot
```

```
      labs(x = NULL, y = "Beta") +
      coord_flip() # turn bars sideways
  }else{
    # if the user does not request a plot
    # return a list of sorted terms instead
    return(top_terms)
  }
}

# plot top ten terms in the Amazon product reviews by topic
top_terms_by_topic_LDA(release_corpus, number_of_topics = 4)
```



圖示說明：

目前問題-1：這邊我們不太能有效看出各主題的差異性。

解決方法：

1. 將 LDA 模型中的主題數參數做最佳化處理。
2. 更換分析標的，推測問題有可能是此標的的評論資料內容性質都太相近了。

目前問題-2：發現目前分類呈現結果無法讓我們有效了解負面評倫。

解決方法：把評論分開成正面評倫與負面評倫再做 LDA 主題分析。

正面評倫：評級 4 到 5 之評論。

負面評倫：評級 1 到 2 之評論。

因此，後面我們把評論分成正面與負面再分別做 LDA 主題分析。

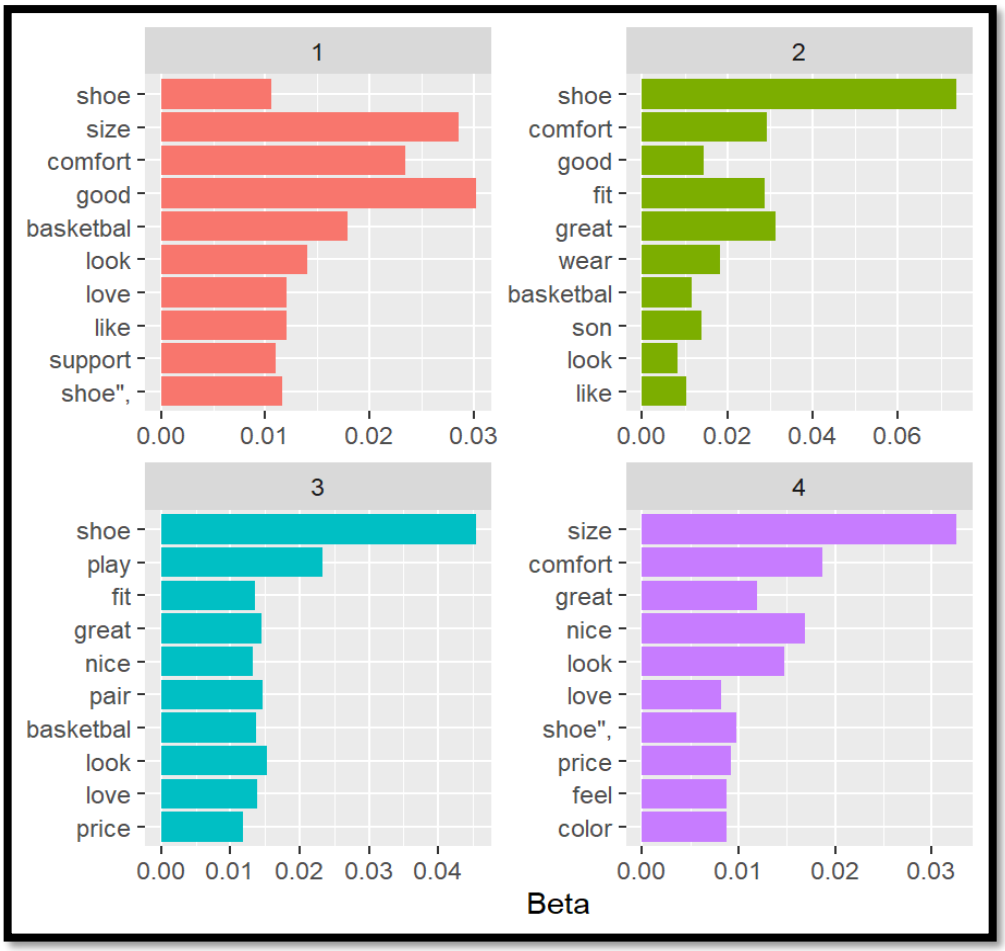
2 正面評倫

```
# 正面評倫
#Corpus

data_posi <- subset(dataset,rating==4 | rating==5) # 正面評倫
release_corpus_posi <- Corpus(VectorSource(data_posi$Text))
#inspect(release_corpus_posi)

#delete numbers and punctuation

release_corpus_posi <- tm_map(release_corpus_posi, removeNumbers)
release_corpus_posi <- tm_map(release_corpus_posi, removePunctuation)

#remove stop words
release_corpus_posi <- tm_map(release_corpus_posi, removeWords, words=st
opwords("en"))



#character to lower
release_corpus_posi <- tm_map(release_corpus_posi, content_transformer(t
olower))

#stemming

release_corpus_posi <- tm_map(release_corpus_posi,stemDocument)
# this function is to transform the plural words
```

```r
release_corpus_posi <- tm_map(release_corpus_posi, removeWords, words=rm
words)

tdm <- TermDocumentMatrix(release_corpus_posi)
#tdm

#sparse terms
tdm <- removeSparseTerms(tdm,1-(10/length(release_corpus_posi))) #term-d
ocument matrix
#tdm

#inspect(tdm[1:20, 1:5])
tdm1=(as.matrix(tdm))
#dim(tdm1)

freq_tdm <- rowSums(tdm1)
sort_tdm <- sort(freq_tdm, decreasing = TRUE) #frequency of words
#sort_tdm
summary(sort_tdm)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   11.00   14.00   21.00   36.88   36.00  409.00
```

```r
#inspect(release_corpus_posi)

top_terms_by_topic_LDA(release_corpus_posi, number_of_topics = 2)
```



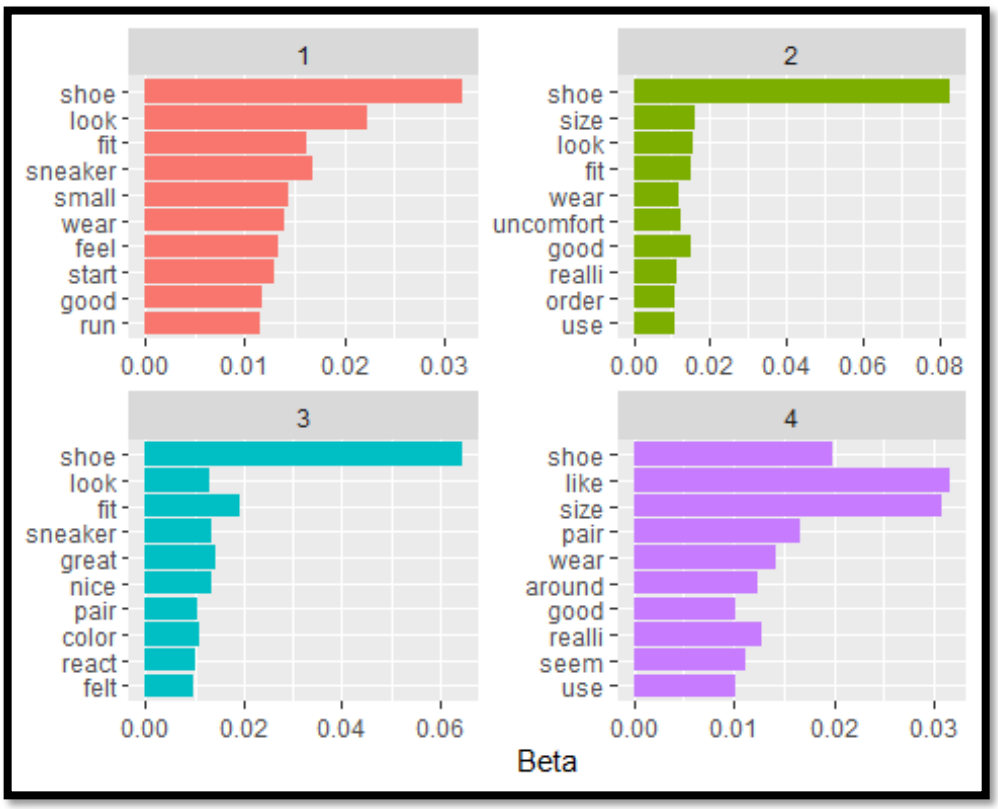
圖示說明：每個主題中最常見的詞。這種可視化讓我們了解從商品正面評論中提取的四個主題。話題 1 中最常見的詞語包括“comfort”、“size”，這表明它可能代表鞋子尺寸舒適度主題。話題 3 中最常見的包括“play”、“fit”，表示這個話題可能代表鞋子的實戰能力。關於每個主題中的單詞的一個重要觀察是，在這 4 個主題中，諸如“shoe”和“great”等一些詞語是常見的。表示自然語言中使用的話題可能存在一些重疊。此外，我們可以認定最大的區別是四個主題之間 β 差異最大的詞。

3 負面評論

```
# 負面評論
data_neg <- subset(dataset,rating==1 | rating==2) # 負面評論
#Corpus
release_corpus_neg <- Corpus(VectorSource(data_neg$Text))
#inspect(release_corpus_neg)

#delete numbers and punctuation
```

```
release_corpus_neg <- tm_map(release_corpus_neg, removeNumbers)
release_corpus_neg <- tm_map(release_corpus_neg, removePunctuation)

#remove stop words
release_corpus_neg <- tm_map(release_corpus_neg, removeWords, words=stop
words("en"))



#character to lower
release_corpus_neg <- tm_map(release_corpus_neg, content_transformer(tol
ower))

#stemming

release_corpus_neg <- tm_map(release_corpus_neg,stemDocument)

release_corpus_neg <- tm_map(release_corpus_neg, removeWords, words=rmwo
rds)

tdm <- TermDocumentMatrix(release_corpus_neg)
#tdm

#sparse terms
tdm <- removeSparseTerms(tdm,1-(10/length(release_corpus_neg))) #term-do
cument matrix
#tdm

#inspect(tdm[1:20, 1:5])
tdm1=(as.matrix(tdm))
#dim(tdm1)

freq_tdm <- rowSums(tdm1)
sort_tdm <- sort(freq_tdm, decreasing = TRUE) #frequency of words
#sort_tdm
summary(sort_tdm)
```

```
     ##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   11.00   11.50   12.00   23.33   29.50   47.00

     #inspect(release_corpus_neg)

top_terms_by_topic_LDA(release_corpus_neg, number_of_topics = 4)
```



圖示說明：此圖可幫助我們了解從負面評論中提取的四個主題。主題 1 中最常見的詞語包括“fit”、“small”，這表明它可能代表鞋子合腳程度、版型主題。主題 2 中最常見的包括“size”、“uncomfort”，表示這個主題可能是代表鞋子的尺寸大小、舒適度。我們可以發現在這四個主題中，諸如“shoe”和“fit”等一些詞語是常見的。綜上所述，我們這邊可以建議該鞋子製造商多注意及改善鞋子尺寸問題，例如是否有忽略了特定族群的尺寸設計(亞洲版型、歐洲版型...)。

➢ Step 4-2：情緒分析（Python）

運用 TextBlob 套件做情感分析的結果，會以元組(polarity, subjectivity)的方式進行返回。

情緒分類：Positive, Negative, Neutral

情緒分數解釋：

Subjectivity (主觀性): 是一個範圍為 [0.0 , 1.0] 的浮點數，其中 0.0 表示客觀，1.0 表示主觀的。

Polarity (情感極性): 是一個範圍為 [-1.0 , 1.0] 的浮點數，正數表示積極，負數表示消極，等於 0.0 表示中立。這裡的值幫助我們做情緒分類。

1 情緒分析

```
#情緒分析
from textblob import TextBlob
import re
def getSubjectivity(text):
    return TextBlob(text).sentiment.subjectivity
def getPolarity(text):
    return TextBlob(text).sentiment.polarity
# creat two columns
df_clean = pd.DataFrame(df_clean)
df_clean = df_clean.rename(columns={0:'body'})
df_clean['Subjectivity'] = df_clean['body'].apply(getSubjectivity)
df_clean['Polarity'] =df_clean['body'].apply(getPolarity)

#情緒分類function
def getAnalysis(score):
    if score <0:
        return 'Negative'
    elif score==0:
        return 'Neutral'
    else:
        return 'Positive'
df_clean['Analysis'] = df_clean['Polarity'].apply(getAnalysis)
df_clean.head()

##                                                 body  ...  Analysis
## 0  a lifelong nike jordan wearer i hesitant purch...  ...  Positive
```

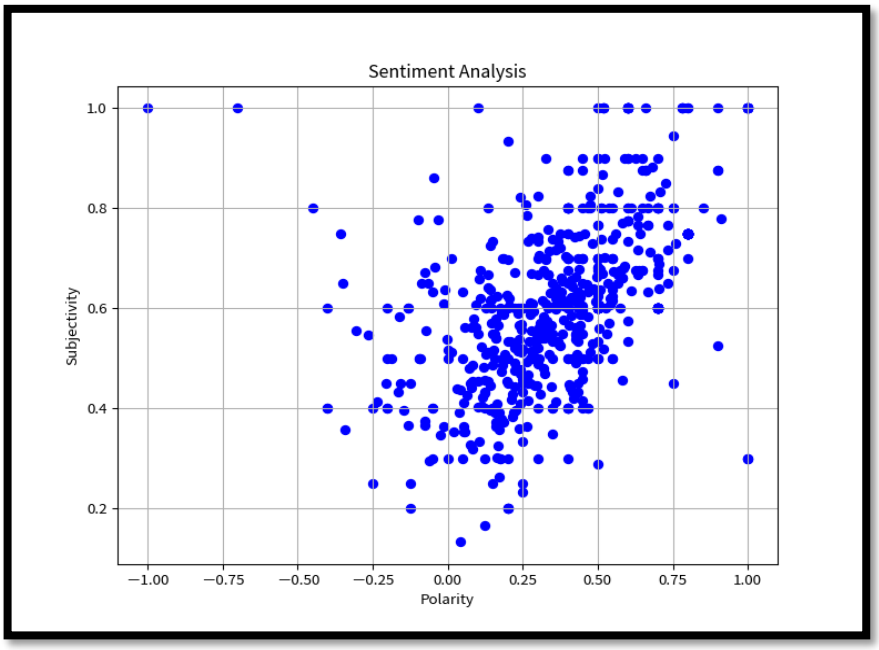
```
## 1  i like present review pro con overall summary ...  ...  Positive
## 2  excuse ashy leg pic lol overall really nice lo...  ...  Positive
## 3  i really like peak high top shoe they stylish ...  ...  Positive
## 4  first great looking basketball shoe if buying ...  ...  Positive
##
## [5 rows x 4 columns]
```

```
# list out all of the positive reviews(研究專案報告這裡就先不印出來，很占篇幅)

j = 1
sortedDF = df_clean.sort_values(by=['Polarity'])
for i in range(0,sortedDF.shape[0]):
    if(sortedDF['Analysis'][i] == 'Positive'):
       # print(str(j) + ')' + sortedDF['body'][i])
        #print()
        j = j+1

#Negative
j = 1
sortedDF = df_clean.sort_values(by = ['Polarity'],ascending = False)
for i in range(0,sortedDF.shape[0]):
    if(sortedDF['Analysis'][i] == 'Negative'):
        #print(str(j) + ')' + sortedDF['body'][i])
       # print()
        j = j+1
```

2 資料視覺化

```
#plot the polarity and subjectivity

plt.figure(figsize=(8,6))
for i in range(0,df_clean.shape[0]):
    plt.scatter(df_clean['Polarity'][i],df_clean['Subjectivity'][i],col
or='blue')

plt.title('Sentiment Analysis')
plt.xlabel('Polarity')
```

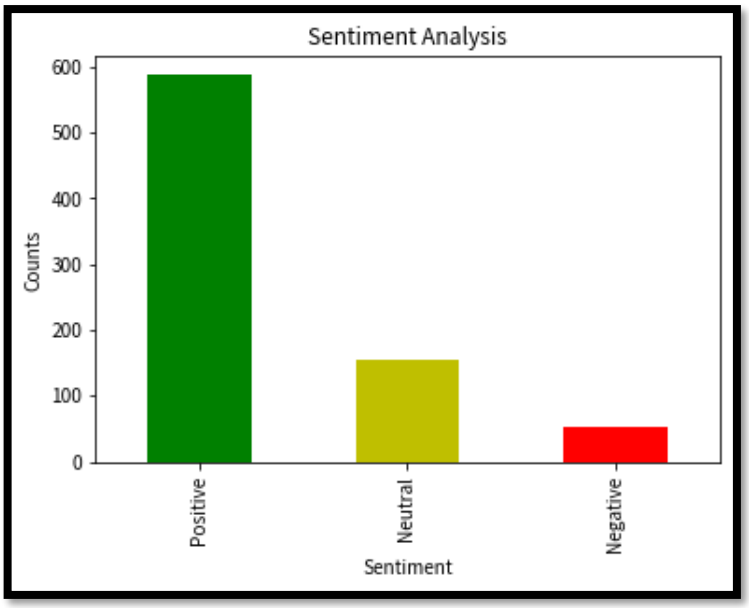
```
plt.ylabel('Subjectivity')
plt.grid()
plt.show()
```



圖示說明：此圖幫助我們了解商品評論的主觀性跟情感極性。我們從圖可觀察到評論分布集中在右半邊；也就是說 Polarity > 0，表示絕大多數評論的情緒為正面。

```
# Get the percentage of positive reviews

rPositive_reviews = df_clean[df_clean.Analysis =='Positive']
rPositive_reviews = rPositive_reviews['body']

round((rPositive_reviews.shape[0]/df_clean.shape[0])*100,1)

## 73.9

rPositive = round((rPositive_reviews.shape[0]/df_clean.shape[0])*100,1)

# Get the percentage of negative reviews
rNegative_reviews = df_clean[df_clean.Analysis =='Negative']
rNegative_reviews = rNegative_reviews['body']

round((rNegative_reviews.shape[0]/df_clean.shape[0])*100,1)

## 6.7
```

```
rNegative = round((rNegative_reviews.shape[0]/df_clean.shape[0])*100,1)

# Get the percentage of Neutral reviews
rNeutral_reviews = df_clean[df_clean.Analysis =='Neutral']
rNeutral_reviews = rNeutral_reviews['body']

round((rNeutral_reviews.shape[0]/df_clean.shape[0])*100,1)

## 19.4

rNeutral = round((rNeutral_reviews.shape[0]/df_clean.shape[0])*100,1)

#rPositive+rNegative+rNeutral

#Show the value counts
df_clean['Analysis'].value_counts()

## Positive    587
## Neutral     154
## Negative     53
## Name: Analysis, dtype: int64

# plot and visualize the counts

plt.title('Sentiment Analysis')
plt.xlabel('Sentiment')
plt.ylabel('Counts')
my_colors = list('gyr')
df_clean['Analysis'].value_counts().plot(kind='bar',color=my_colors)
plt.show()
```



圖示說明：可以發現商品評論大部分皆為正面情緒，分析結果很合理。因為此商品的五星評級為最多數，整體顧客的平均評級也非常高。

七、結論

- **研究專案成果**
  1. 在數據獲取這一塊，本研究專案運用 Python 使用爬蟲技術，提高了數據獲取的效率。
  2. 對商品評論進行文本清理。
  3. 使用 R 進行 LDA 主題建模，得出正負面評論的相應主題，並提出可行性建議。
  4. 運用 Python 對商品評論做了情感分析，並將結果資料可視化。
- **未來展望及改善建議**
  1. 在文字清洗部分可以做的更精細，例如：根據電商評論客制化停止詞詞庫。
  2. 可以套用更多不同的情緒分類模型及機器學習演算法，並比較各個模型的成效。
  3. 將 LDA 模型中的主題數參數做最佳化處理。
  4. 從 LDA 主題分析的結果發現我們不太能有效看出各主題的差異性，我認為有兩個解決方法。首先，我們可以將 LDA 模型中的主題數參數做最佳化處理。再來，我們可以更換分析標的，推測問題有可能是此標的的評論

資料內容性質都太相近。

5. 情感分析上增加更多情緒分類，例如：喜、怒、哀、樂。

八、參考文獻

1. 楊惠淳，2011 年，“以主客觀分析與相互資訊檢索探討情感分析之準確度-以電影評論為例”，國立臺北科技大學資訊與運籌管理研究所碩士論文。
2. 徐筱雁，2014 年，“情緒分析中屬性詞與情感詞的關係之探討-以牛肉麵食評為例”。國立聯合大學資訊管理學系研究所碩士論文。
3. 謝佩庭，2014 年，“基於使用者情緒關鍵字彙之臉書粉絲專頁評論分類與評分系統”，國立交通大學多媒體工程研究所碩士論文。
4. 李淑惠，2014 年，“運用文字探勘技術於口碑分析之研究”，東吳大學商學院資訊管理學系碩士論文。
5. 俞舒禔，2018 年，“應用情感分析於產品比較與品牌推薦系統-以美妝產平為利”，國立政治大學商學院統計學系碩士論文。
6. Kumar Ravi & Vadlamani Ravi (2015). A survey on opinion mining and sentiment analysis: Tasks, approachesand applications. Knowledge-Based Systems
7. Jesus Serrano-Guerrero, Jose A. Olivas, Francisco P. Romero & Enrique Herrera-Viedma (2015). Sentiment analysis: A review and comparative analysis of web services. Information Sciences.
8. Isidoros Perikos & Ioannis Hatzilygeroudis (2016). Recognizing emotions in text using ensemble of classifiers. Engineering Applications of Artificial Intelligence.
9. Jesus Serrano-Guerrero, Jose A. Olivas, Francisco P. Romero & Enrique Herrera-Viedma(2015). Sentiment analysis: A review and comparative analysis of web services. Information Sciences.
10. Emma Haddia, Xiaohui Liua & Yong Shi (2013). The Role of Text Pre-processing in Sentiment Analysis. Procedia Computer Science, Vol. 17, 26-32.