



社群媒體分析

APEX LEGENDS™

Group 7

B104020019 黃婕妮

B104020023 蔡宜樺

M134020005 馮祐倫

M134020021 李翊曲

M134020030 戴廣琛

M134020046 宋旻家

M134610017 李逸華

Contents



-
- 1 專案動機**
 - 2 程式碼流程說明**
 - 2.1 資料處理**
 - 2.2 資料分析**
 - 3 遇到的困難和解決方式**

專案動機

這一年內遊戲環境的主要挑戰



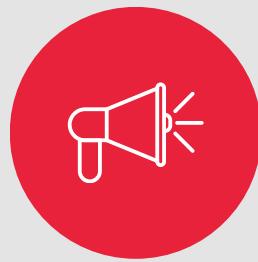
營運

EA做出了許多失望的改動與
錯誤的決策



社群

玩家社群中充斥著不滿與責罵



玩家數

遊戲人數為去年同期的一半

專案動機





專案目的

- ▶ 對於大型線上遊戲來說，健康的遊戲環境至關重要
- ▶ 分析Apex玩家社群對遊戲的看法，希望能提供遊戲公司一些啟示

資料來源



資料匯入

```
[ ] from google.colab import drive  
drive.mount('/content/drive')  
  
[ ] ## 讀取從steam上面爬下來的評論資料  
df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/data/english.csv')  
df
```

資料清理

```
#清理資料並統一格式
def data_cleaned(origin):
    ##將日期改為20xx-xx-xx
    origin["Date"] = origin["Date"].str.replace(r"張貼於: (\d{4}) 年 (\d{1,2}) 月 (\d{1,2}) 日", r"\1-\2-\3", regex=True)
    origin["Date"] = origin["Date"].str.replace(r"張貼於: (\d{1,2}) 月 (\d{1,2}) 日", "2025-\1-\2", regex=True)

    ##去除評論中包含日期的部分
    origin["Content"] = origin["Content"].str.replace(r"張貼於: (\d{4}) 年 (\d{1,2}) 月 (\d{1,2}) 日\n", "", regex=True)
    origin["Content"] = origin["Content"].str.replace(r"張貼於: (\d{1,2}) 月 (\d{1,2}) 日\n", "", regex=True)

    ##去掉isHelpful為空值的資料，用isUseful作為新的欄位，代表有多少人覺得這篇評論有參考性
    origin = origin.dropna(subset=['isHelpful'])
    origin["isUseful"] = origin["isHelpful"].str.extract(r"(\d+) 個人認為這篇評論值得參考") [0].str.replace(",","").astype(int)
    origin = origin.drop(["isHelpful"], axis=1)

    ##僅留下英文的評論
    origin = origin[origin['Content'].str.match(r'^[A-Za-z0-9\s.,!?\'-]*$', na=False)] # 保留只包含英文字符（字母、數字、空格和標點）的行

    ##去除只有一個單字的資料
    origin = origin[origin['Content'].str.split().str.len()>1]

return origin
```

斷詞斷句

- 清理文本：在將換行符「\n」取代為「,」，並移除非字母、數字和空格字符。
- 斷詞：使用 NLTK 的 word_tokenize 進行斷詞，將 DataFrame 轉換為「一行對應一個詞」的格式。並轉換單詞為小寫。
- 詞幹提取：使用 Porter Stemmer 進行詞幹提取，將 word 欄位轉換為詞幹 (stem_token)。

```
[ ] import re
#新增['sentence']欄位，用'.'取代'\n'
df['sentence'] = df['Content'].str.replace(r'\n', '.', regex=True)
##保留字母、數字和空格
df["sentence"] = df["sentence"].apply(lambda x: re.sub(r'[^w\s]', '', x))
```

```
[ ] df
```

```
[ ] import nltk
nltk.download('punkt')
from nltk.corpus import stopwords
from nltk.corpus import wordnet
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer

##使用NLTK的斷詞函式word_tokenize進行斷詞，將DataFrame處理成一個row一個斷詞的結果
df = df.assign(token = df['sentence'].apply(nltk.word_tokenize)).explode('token')
##轉小寫
df = df.assign(word = df['token'].str.lower())

df
```

```
[ ] porter = PorterStemmer()

##將資料使用字根的方式表達
df['word'] = df['word'].astype(str)
df = df.assign(stem_token = df['word'].apply(porter.stem)).reset_index(drop=True)

df
```

停用字處理

```
# 下載停用詞資料
nltk.download('stopwords')

# 取得英文停用詞列表
stop_words = stopwords.words('english')

# 保留不是停用詞的行
df = df[~df['stem_token'].astype(str).str.lower().isin(stop_words)]

df
```

計算詞頻

- 使用 Counter 計算 stem_token（詞幹）欄位中每個詞出現的次數。
- 將詞頻計數結果轉換為 DataFrame，包含 stem_token（單字）和 frequency（詞頻），並按頻率排序。

```
# 以 df['word'] 欄位計算
word_counts = Counter(df['stem_token'].astype(str))

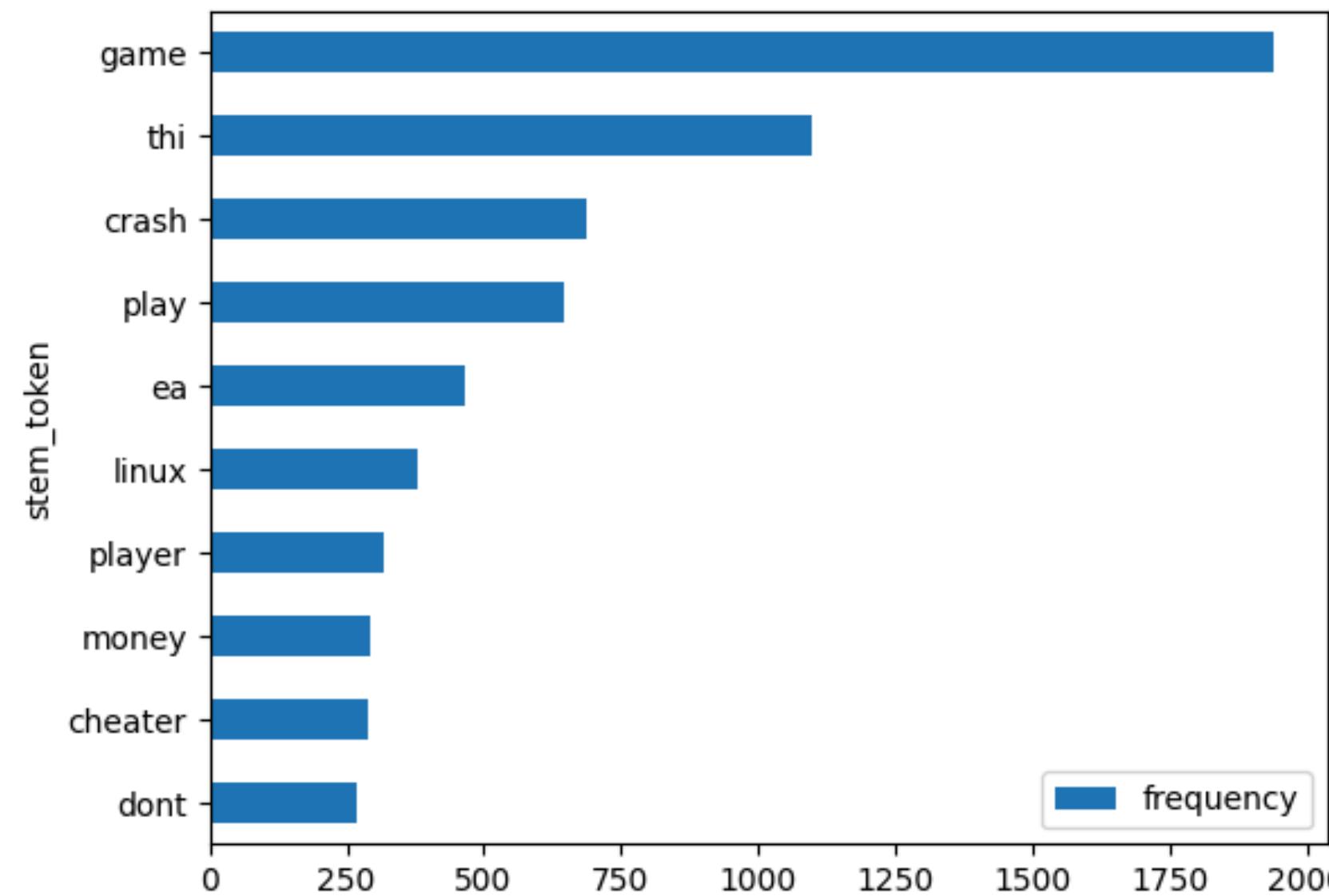
# 轉換為 DataFrame 方便查看
word_freq_df = pd.DataFrame(word_counts.items(), columns=['stem_token', 'frequency']).sort_values(by='frequency', ascending=False)

print(word_freq_df.head(10)) # 顯示前 10 個詞頻最高的單字

# 繪製詞頻為前10高的長條圖
word_freq_df.head(10).plot.barh(x = 'stem_token', y = 'frequency').invert_yaxis()
plt.show()
```

計算詞頻

- 輸出出現次數最高的前 10 個單字。
- 繪製前 10 個詞幹的詞頻長條圖，並將 Y 軸反轉，使詞頻最高的詞排在最上方。



自訂停用詞

- 將 stem_token 欄位的詞彙合併成一個大文本，並透過 WordCloud 產生文字雲來視覺化詞頻。
- 自訂停用詞，並擴充列表，以排除無意義的高頻詞。
- 只保留不在停用詞列表內的詞彙，重新計算詞頻

```
▶ # 以 df['word'] 欄位繪製
text = ' '.join(df['stem_token'].dropna()) # 合併所有單字

# 產生文字雲
wordcloud = WordCloud(width=800, height=400, background_color='white', stopwords=stop_words, colormap='viridis').generate(text)

# 顯示文字雲
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off") # 移除座標軸
plt.show()

[ ] # 手動加入停用字
newStops = ['thi', 'game', 'play', 'playing', 'played', 'even', 'also', 'one', 'still', 'could', 'player', 'much', 'support']
stop_words.extend(newStops)

[ ] new_df = df[~df['stem_token'].isin(stop_words)]
df.head(15)

[ ] new_freq_df = pd.DataFrame(new_df['stem_token'].value_counts())
new_freq_df = new_freq_df.reset_index()
new_freq_df.columns = ['word', 'freq']
new_freq_df.head(15)

[ ] # 第一次的詞頻圖
word_freq_df.head(15)
```

文字雲比較

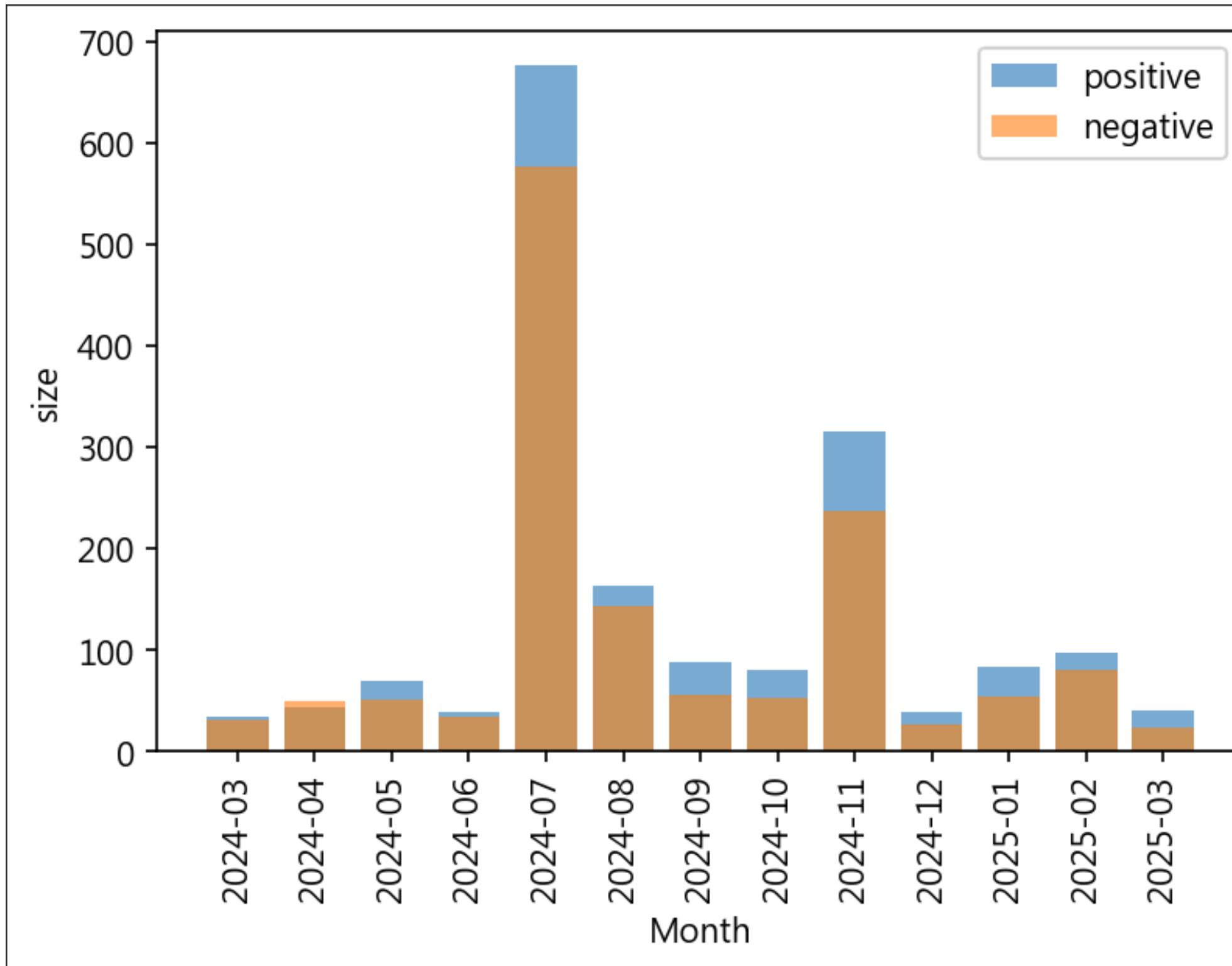
去除停用字前



去除停用字後



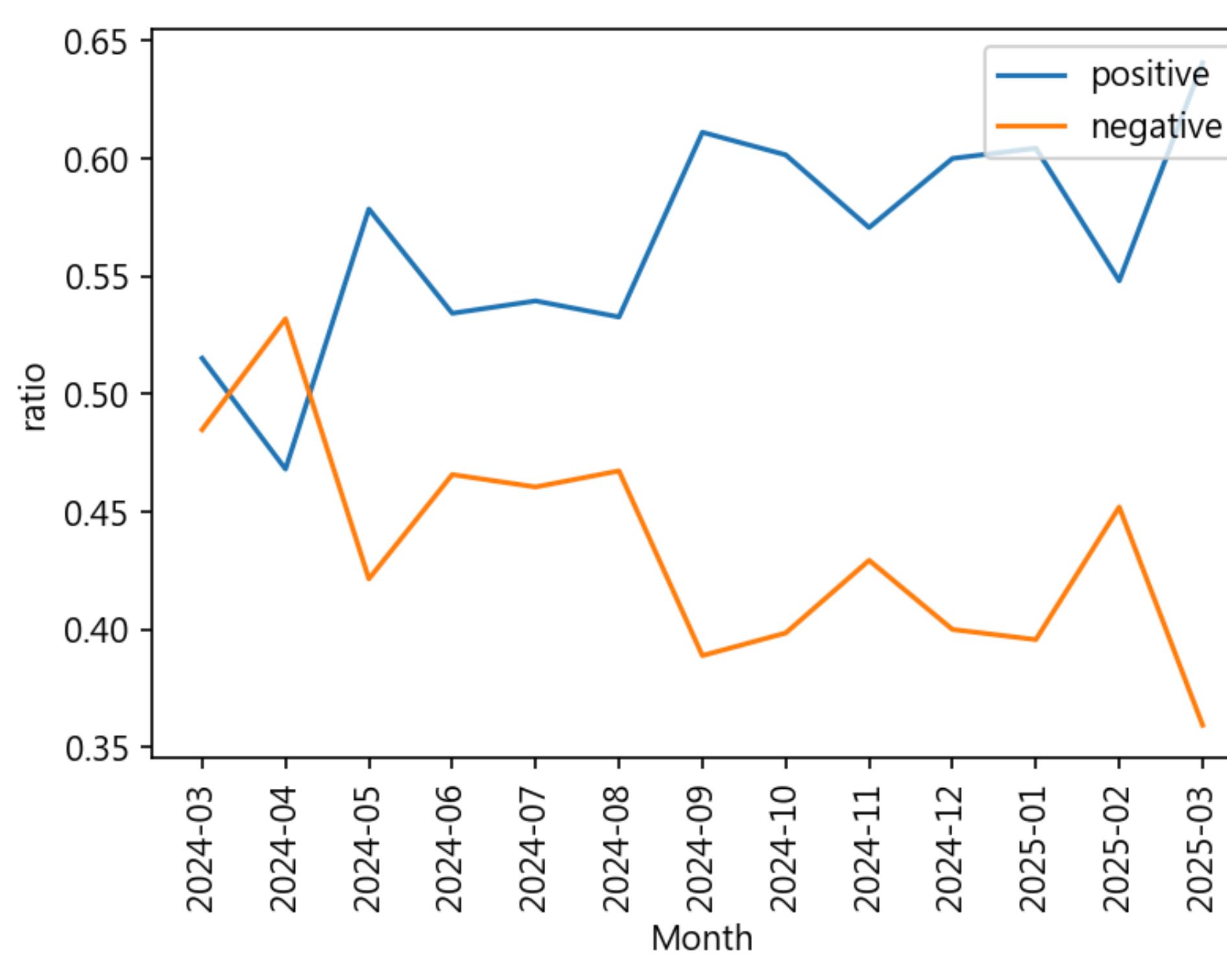
各月份正負情緒字數量



- 在2024-04月份時負面情緒字數量是高於正面情緒的
- 在2024-07月及11月份時正負面情緒字數量遠高於其他月份，討論度極高

	Month	negative	positive	sentiment_value
0	2024-03	32.0	34.0	2.0
1	2024-04	50.0	44.0	-6.0
2	2024-05	51.0	70.0	19.0
3	2024-06	34.0	39.0	5.0
4	2024-07	577.0	676.0	99.0

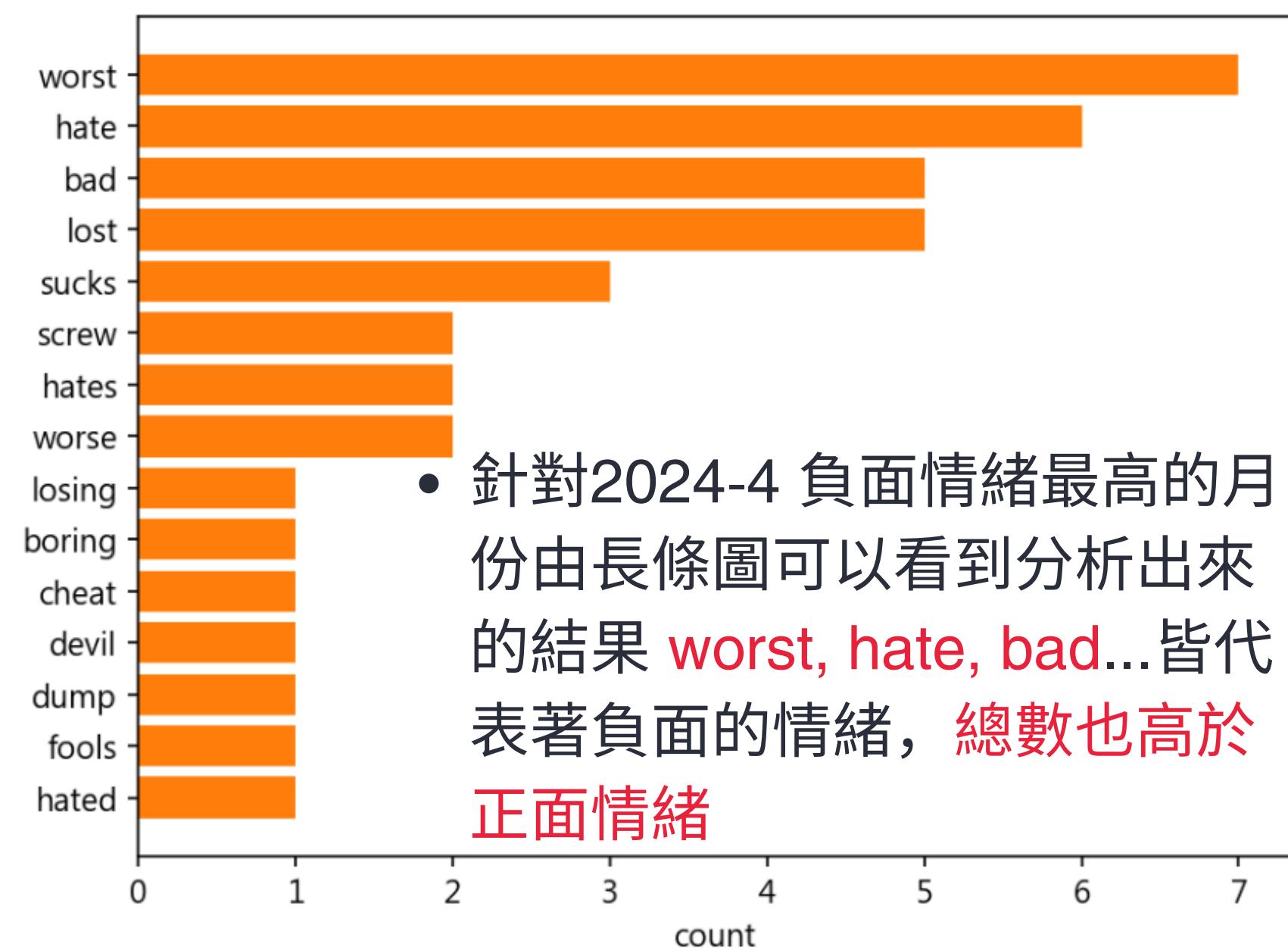
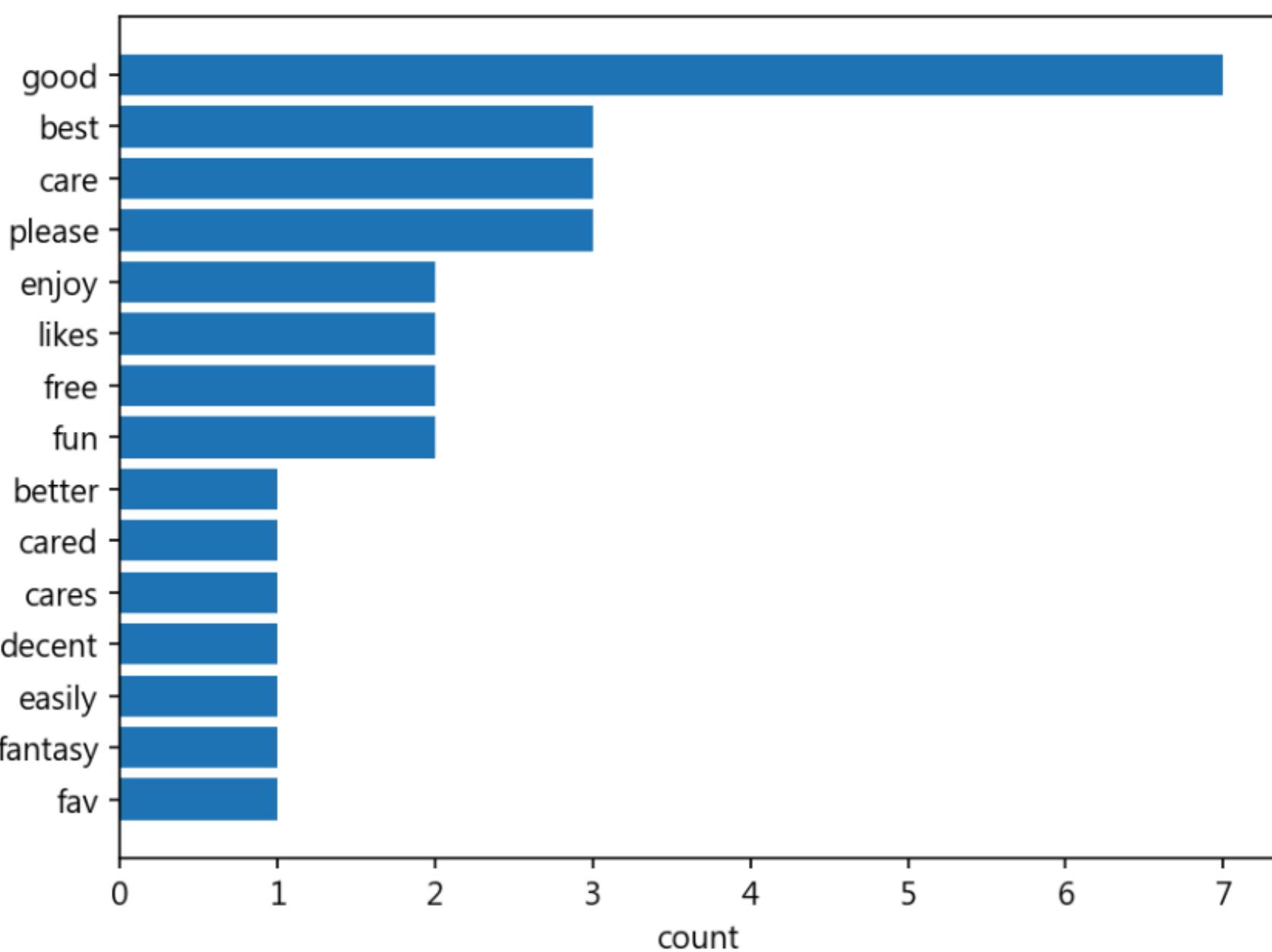
各月份正負情緒占比



	Month	sentiments	size	ratio
3	2024-03	negative	32	0.484848
4	2024-03	positive	34	0.515152
8	2024-04	negative	50	0.531915
9	2024-04	positive	44	0.468085
14	2024-05	negative	51	0.421488

- 由折線圖可看出在2024-04月份時
負面情緒是高於正面情緒，負面為
0.53，正面為0.46

2024-4 月正負向情緒代表字



• 針對2024-4 負面情緒最高的月份由長條圖可以看到分析出來的結果 **worst, hate, bad...** 皆代表著負面的情緒，總數也高於正面情緒

2024-4月負向情緒代表字文字雲

serious suffering ugly tedious fools bad worse
screw losing cheat steal hated boring terrible sucks

hate piss hater hating haters

worst suck awful terrible

problem dump lowered

screw losing cheat steal hated boring terrible sucks

hate piss hater hating haters

worst suck awful terrible

problem dump lowered

利用coreNLP進行POS及NER分析

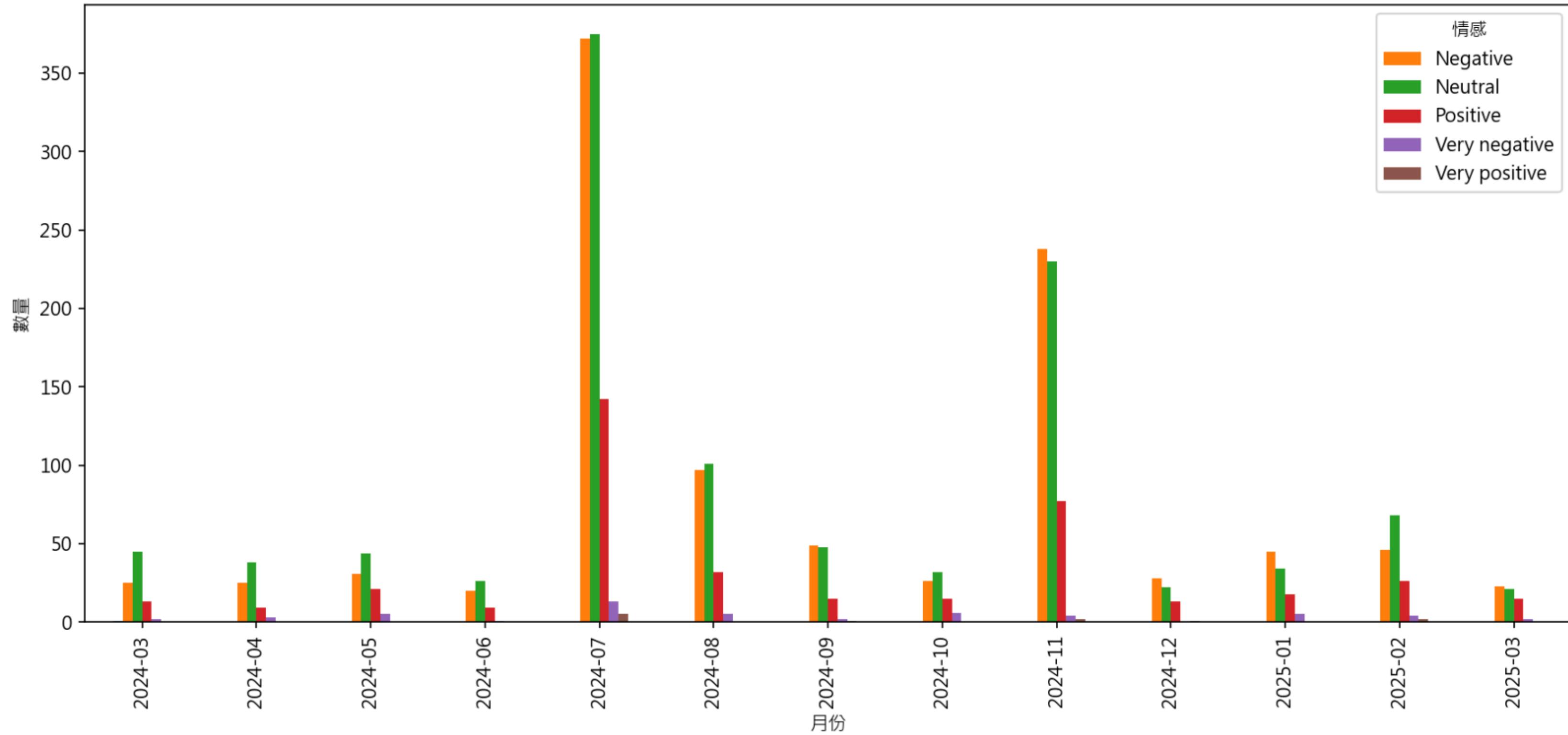
```
# 依序處理每個句子
for i, sent in enumerate(document.sentence):
    # 取得斷句結果：將該句內所有 token 的文字連接起來
    sentence_text = "".join([token.word for token in sent.token])
    print(f"Sentence {i+1} 斷句結果: {sentence_text}")

    # 輸出段詞詳細資訊，依據指定格式印出 token 的 word、lemma、pos 與 ner
    for t in sent.token:
        print("{:12s}\t{:12s}\t{:6s}\t{}".format(t.word, t.lemma, t.pos, t.ner))
    print("")
```

Sentence 1 斷句結果: 210dollarbattlepassesperseasonTitanfalldiedforthisslop			
		POS	NER
2	2	CD	MONEY
10	10	CD	MONEY
dollar	dollar	NN	MONEY
battle	battle	NN	0
passes	pass	VBZ	0
per	per	IN	0
season	season	NN	0
Titanfall	Titanfall	NNP	PERSON
died	die	VBD	0
for	for	IN	0
this	this	DT	0
slop	slop	NN	0

利用coreNLP情緒分析

各月份情感分佈



遇到的困難和解決方式

- 使用共編的colab平台時，當使用了coreNLP client去執行情緒分析，遇到資料量很大時（約2500筆），服務容易執行到一半就中斷。解決方法：換到硬體較好的本地端執行（或許批次處理也可以解決）
- 資料需要group_by 後再輸出圖表，選定的group 也很重要，否則圖表的線和直方圖會糊在一起

Thank You

