Báo cáo xử lý dữ liệu Python

Nguyễn Phan Hiển - 22022534

Bước 1: Thu thập dữ liệu

Ở đây ta sử dụng Facebook Scraper, sau đó cho data vào một file csv để tiện xử lý

```
%pip install facebook_scraper pandas numpy
from facebook_scraper import get_posts
import pandas as pd
import numpy as np
import os
import json
```

```
FANPAGE_LINK ="Steam"
FOLDER_PATH = "Data/"
COOKIE_PATH = "./mbasic.facebook.com_cookies.txt"

PAGES_NUMBER = 100 # Number of pages to crawl

post_list = []
for post in get_posts(FANPAGE_LINK,
                      options={"comments": True, "reactions": True, "allow_extra_requests": True},
                      extra_info=True, pages=PAGES_NUMBER, cookies=COOKIE_PATH):
    print(post)
    post_list.append(post)
```

Bước 2: Tiền xử lý dữ liệu

Dữ liệu khi mới crawl vẫn còn quá nhiều hàng và cột thừa, lặp cũng như dữ liệu bị thiếu do công cụ hoặc do yếu tố khác. Vì vậy ta cần thực hiện những bước làm sạch và tiền xử lý cơ bản

1. Drop những cột không dùng đi và những cột có quá nhiều giá trị rỗng

```python
#Drop những column không cần dùng và những column có quá nhiều giá trị NaN
df = df.dropna(thresh=10, axis='columns')
df.columns
```

✓ 0.0s

```
Index(['id', 'name', 'id_post', 'created_time', 'updated_time', 'message',
       'type', 'picture', 'source', 'properties', 'like', 'love', 'haha',
       'wow', 'sad', 'angry', 'comments', 'total_reactions', 'total_shares',
       'total_comments'],
      dtype='object')
```

```python
#Những cột này không có giá trị sử dụng khi phân tích dữ liệu
df = df.drop(columns=['id_post','source','updated_time','properties','id'])
```

✓ 0.0s

2. Fill những ô bị thiếu data một cách hợp lý

```python
#Có thể thấy ở cột picture những giá trị NaN có nghĩa là post không có ảnh
df['picture'] = df['picture'].fillna(" ")
```

✓ 0.0s

3. Loại bỏ hàng lặp

```python
#Loại bỏ những giá trị bị duplicate (nếu có)
df.drop_duplicates()
```

✓ 0.1s

4. Check và chuyển file đã làm sạch vào trong file csv khác

```
#Chuyển Data đã xử lý vào một file csv khác
df.to_csv('Data/Data_Final.csv', index=False)
```
✓ 1.1s

Data sau khi xử lý:

created_time	message	type	picture	like	love	haha	wow	sad	angry	comments	total_reactions	total_shares	total_comments
2023-11-18T04:13:09+0000	Một ngày làm việc hiệu quả của người lương 5 t...	photo	https://scontent.fhan3-5.fna.fbcdn.net/v/t39.3...	293	3	204	0	6	0	[{'created_time': '2023-11-18T04:13:24+0000', ...	506	3	19
2023-11-18T02:16:00+0000	Khách vote 1 ⭐ vì nhắn lúc 3h sáng không nhân ...	status		310	0	434	1	18	0	[{'created_time': '2023-11-18T02:17:04+0000', ...	763	9	47
2023-11-17T13:39:28+0000	"Em yên tâm, chỉ có chị với bé Linh biết chuyệ...	status		1023	3	1128	0	14	0	[{'created_time': '2023-11-17T13:41:25+0000', ...	2168	19	110
Người													

Bước 3: Phân tích dữ liệu và trực quan hóa

- Bài viết nào có số lượt tương tác lớn nhất

Trước tiên ta phải tạo một cột mới có tên là tương tác = tổng mọi reactions + share + comment

Sau đó qua việc sort_value theo cột tương tác và lấy giá trị đầu tiên, ta sẽ được post có lượt tương tác lớn nhất

Sau đó ta mô hình hóa những reaction theo số lượng của chúng

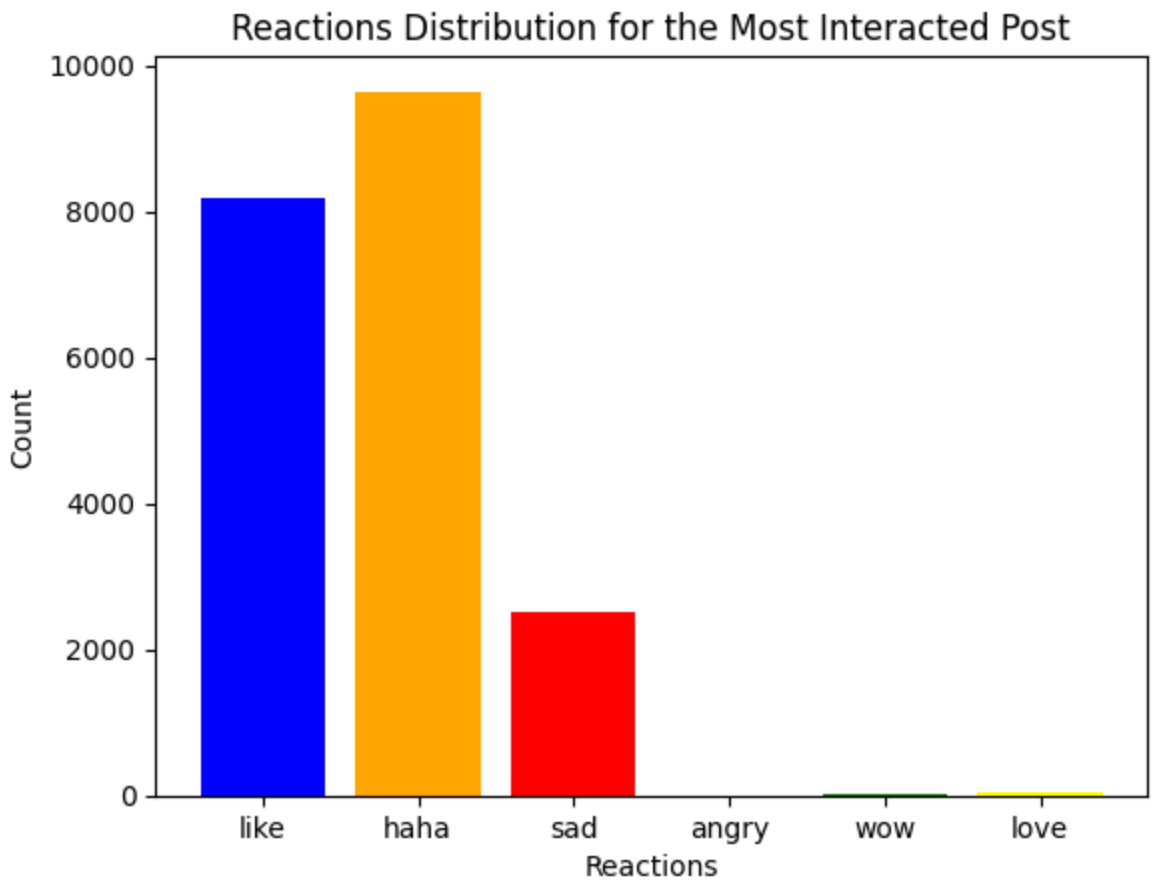
```python
#Bài viết có số lượt tương tác lớn nhất
df['tuong_tac'] = df['total_comments'] + df['total_reactions'] + df['total_shares']
df_sorted = df.sort_values(by='tuong_tac', ascending=False)
most_interacted_post = df_sorted.iloc[0]
most_interacted_post
```

✓ 0.0s

```
name                                              Biết thế éo đi làm
id_post                              101084031778928_858034806328828
created_time                                   2023-08-29T01:50:26+0000
message             Có một vạn hành tinh nhưng tôi lại sinh ra ở h...
type                                                            photo
picture             https://scontent.fhan3-3.fna.fbcdn.net/v/t39.3...
like                                                             8185
love                                                               35
haha                                                             9642
wow                                                                14
sad                                                              2501
angry                                                               2
comments            [{'created_time': '2023-08-29T01:51:05+0000', ...
total_reactions                                                 20379
total_shares                                                     1058
total_comments                                                   1079
tuong_tac                                                       22516
Name: 200, dtype: object
```

```python
reactions = ['like', 'haha', 'sad','angry','wow', 'love']
counts = [most_interacted_post[reaction] for reaction in reactions]

# Plotting
plt.bar(reactions, counts, color=['blue', 'orange', 'red', 'black', 'green', 'yellow'])
plt.title('Reactions Distribution for the Most Interacted Post')
plt.xlabel('Reactions')
plt.ylabel('Count')
plt.show()
```

✓ 0.0s

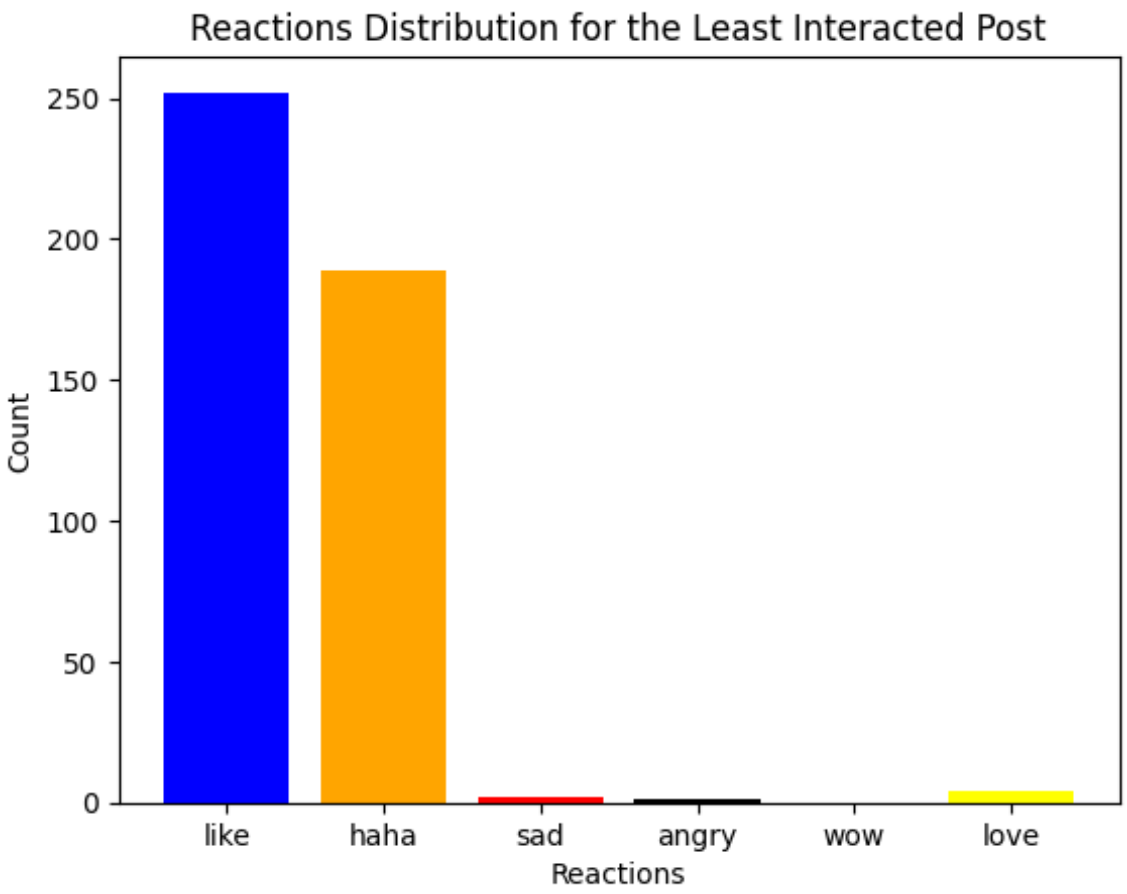


Như có thể thấy trong đồ thị trên, “haha” đã được sử dụng nhiều nhất, kế theo đó là “like” và “sad” trong khi những reaction như “angry”, “wow” và “love” hiếm khi được sử dụng. Từ đó ta có thể suy ra rằng đây là một bài đăng mang nhiều yếu tố hài hước. Những bài viết mang nhiều yếu tố hài hước có xu hướng thu hút lượng khán giả lớn hơn là những bài viết mang yếu tố không hài hước.
Ta có thể chứng minh điều này thông qua bài viết ít được tương tác nhất

```python
df_sorted = df.sort_values(by='tuong_tac', ascending=True)
least_interacted_post = df_sorted.iloc[0]
least_interacted_post
```

✓ 0.0s

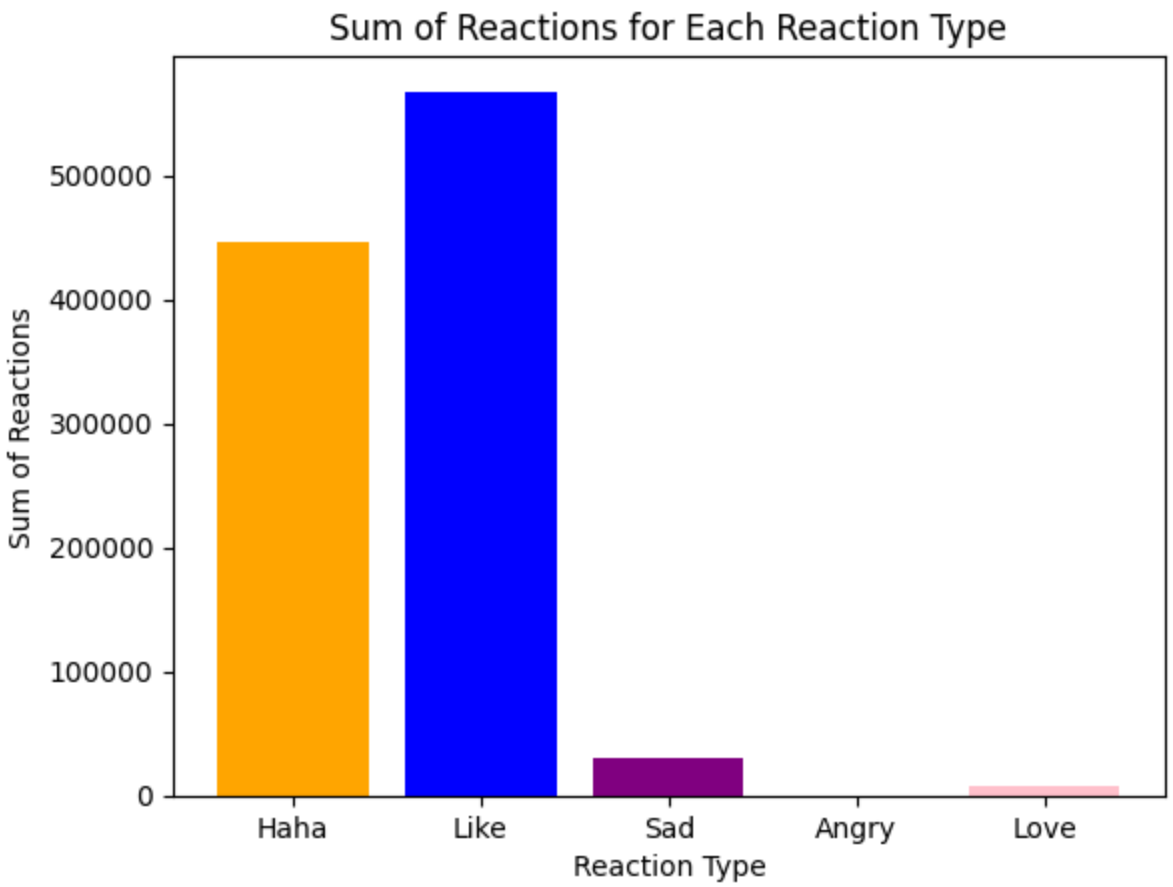
```
name                                                Biết thế éo đi làm
id_post                              1010840317778928_867326945399614
created_time                                  2023-09-11T14:04:46+0000
message             Khi sếp nói tôi đi làm suốt ngày cười #BTEDL #...
type                                                             video
picture             https://scontent.fhan4-1.fna.fbcdn.net/v/t15.5...
like                                                               252
love                                                                 4
haha                                                               189
wow                                                                  0
sad                                                                  2
angry                                                                1
comments            [{'created_time': '2023-09-11T14:16:57+0000', ...
total_reactions                                                    448
total_shares                                                         3
total_comments                                                      34
tuong_tac                                                          485
Name: 116, dtype: object
```



Ở mô hình thể hiện số lượng reactions của post ít tương tác nhất, ta có thể thấy rõ ràng một sự sụt giảm của tỷ lệ giữa số lượng “haha” và số lượng “like”. Thậm chí ở post ít được tương tác nhất, số lượng “haha” còn thấp hơn số lượt “like”.

Đây là sơ đồ tính tổng lượng reaction của page:

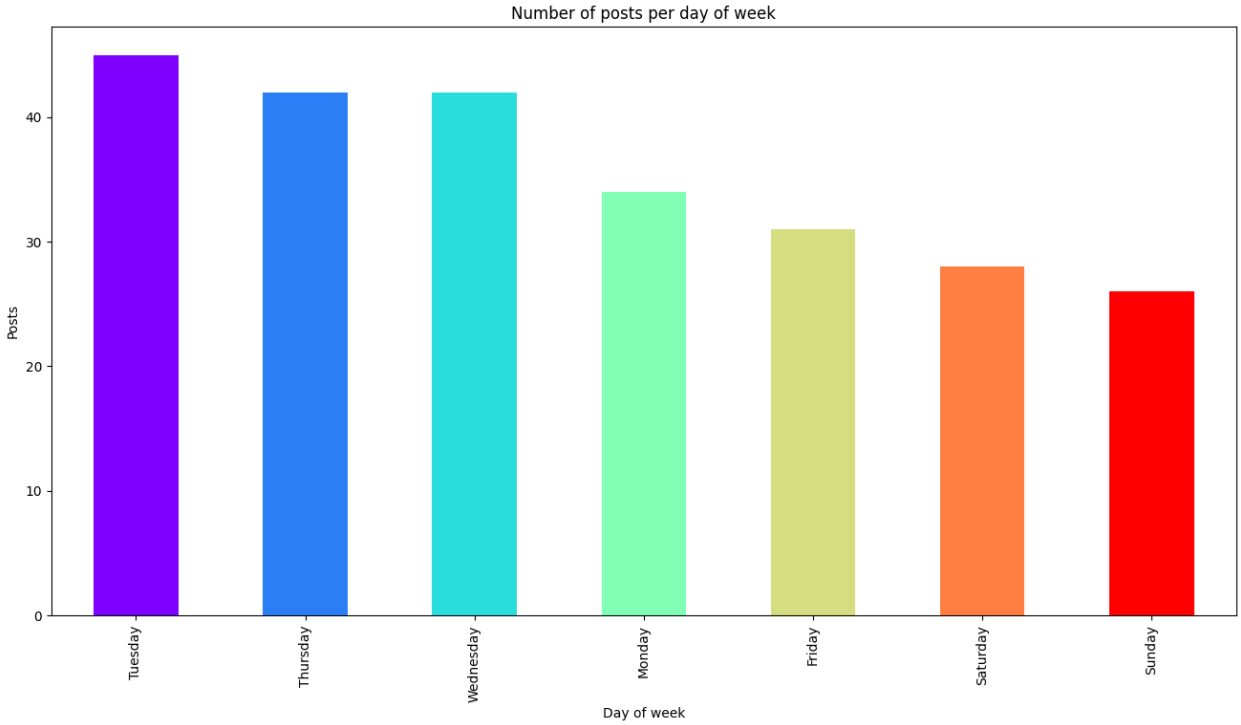
```
haha = df['haha'].sum()
like = df['like'].sum()
sad = df['sad'].sum()
angry = df['angry'].sum()
love = df['love'].sum()
reaction_types = ['Haha', 'Like', 'Sad', 'Angry', 'Love']
reaction_sums = [haha, like, sad, angry, love]

plt.bar(reaction_types, reaction_sums, color=['orange', 'blue', 'purple', 'red', 'pink'])
plt.title('Sum of Reactions for Each Reaction Type')
plt.xlabel('Reaction Type')
plt.ylabel('Sum of Reactions')
plt.show()
```



Ta thấy sự chênh lệch rất lớn giữa reaction “haha” và “like” so với các reaction còn lại. Qua đó, ta có thể thấy sự hài hước chính là yếu tố chính để thu hút người theo dõi của page này.

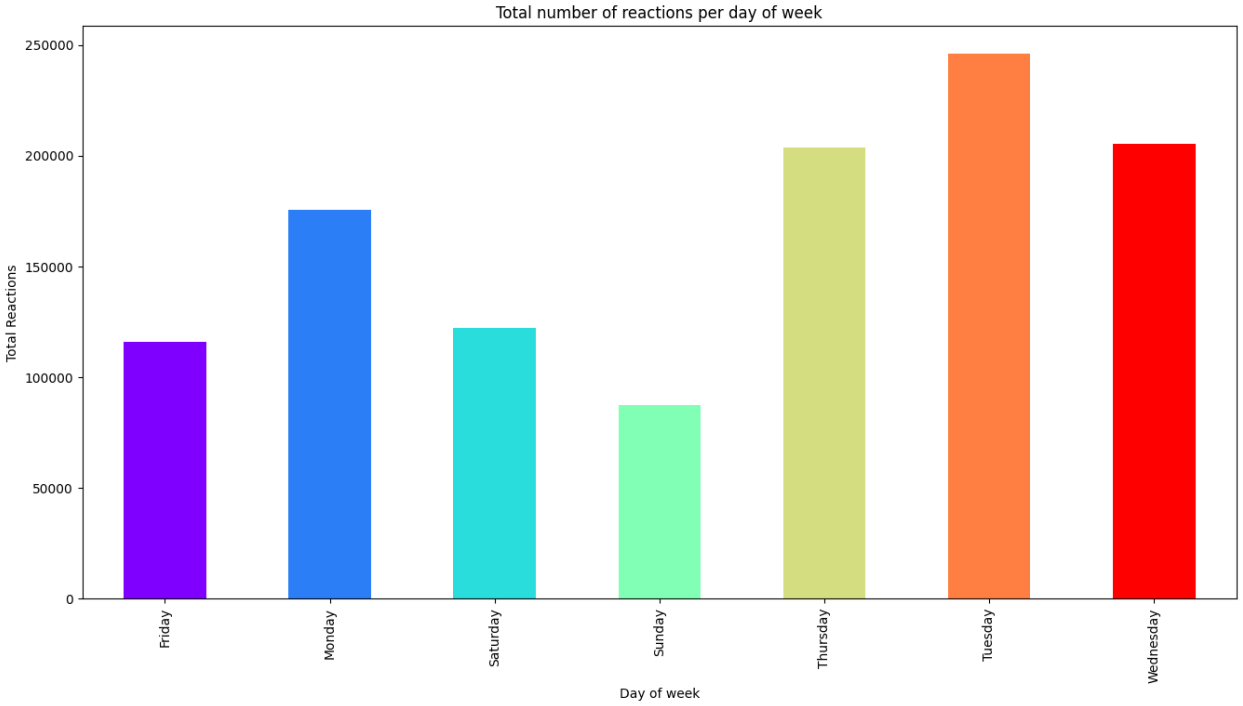
- Thời gian mà page thường xuyên đăng bài

```
df['created_time'] = pd.to_datetime(df['created_time'])
week_day = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
df['week_day'] = df['created_time'].apply(lambda x: week_day[x.weekday()])

colors = cm.rainbow(np.linspace(0, 1, len(df['week_day'].unique())))
df['week_day'].value_counts().plot(kind='bar', figsize=(16,8), color=colors)
plt.title('Number of posts per day of week')
plt.xlabel('Day of week')
plt.ylabel('Posts')
```



Ta biến đổi cột created_time thành dạng date_time, sau đó tạo cột week_days trong để df để có thể phân loại các post vào. Từ đó ta có thể có sơ đồ trên.
Nhìn vào sơ đồ này, ngày thứ tư là ngày có nhiều bài viết nhất với hơn 40 bài còn chủ nhật là ngày có ít bài viết nhất trong tuần. Hơn nữa, số lượng bài viết vào trong những hôm giữa tuần cao hơn hẳn số lượng bài viết vào những ngày đầu tuần và cuối tuần

Điều này phần nào cũng ảnh hưởng đến lượng tương tác của người theo dõi page trong các ngày trong tuần, được chứng minh qua biểu đồ sau:



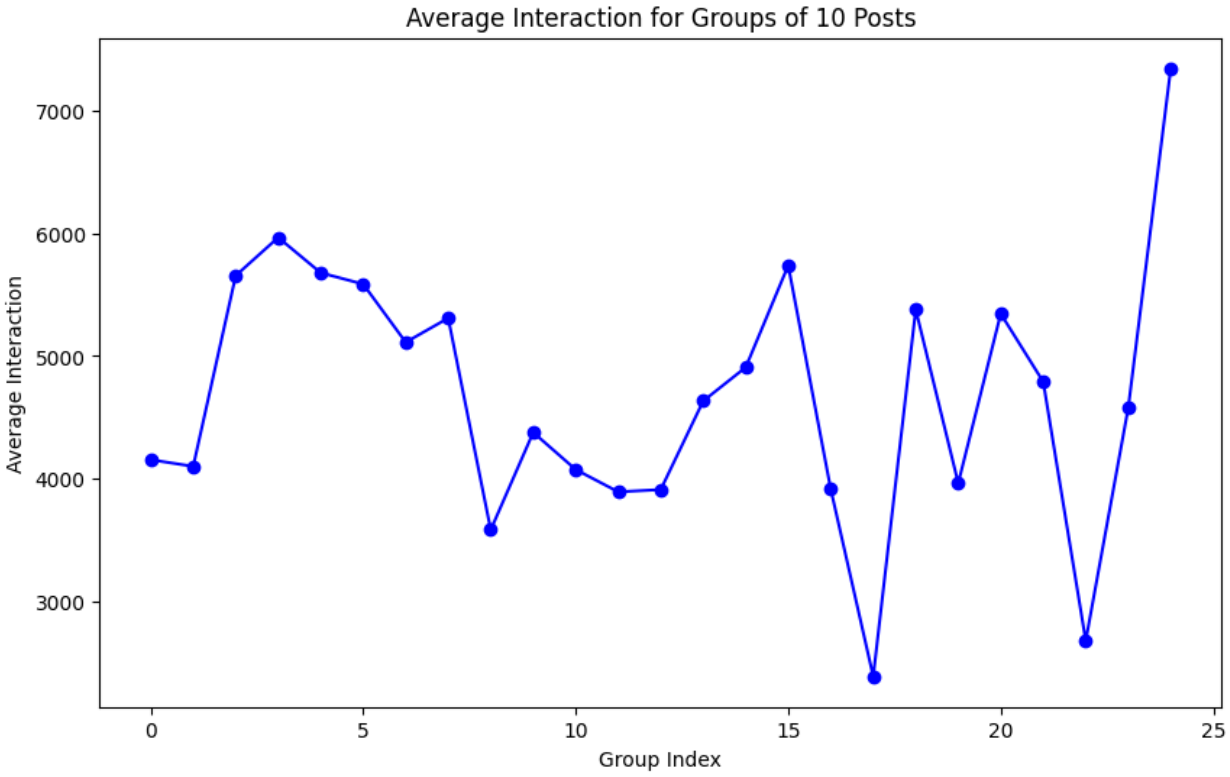

```
grouped_data = df.groupby('week_day')['tuong_tac'].sum()

# Plotting
colors = cm.rainbow(np.linspace(0, 1, len(grouped_data)))
grouped_data.plot(kind='bar', color=colors, figsize=(16, 8))
plt.title('Total number of reactions per day of week')
plt.xlabel('Day of week')
plt.ylabel('Total Reactions')
plt.show()
```

Ta có thể thấy lượng tương tác vào thứ 5, thứ 4 và thứ 3 vượt trội hơn hẳn so với các ngày còn lại. Mô típ này hoàn toàn trùng khớp với số lượng bài đăng vào các ngày trong tuần ở trên. Tuy nhiên, số lượng bài viết không phải là yếu tố quyết định lớn nhất, thể hiện qua việc mặc dù thứ 6 có số post lớn hơn thứ 7 nhưng lượng reactions lại nhỏ hơn. Ngoài ra, số lượng post vào thứ 4 gập đôi chủ nhật nhưng lượng tương tác lại gấp 2,5 lần.

- Sự phát triển của page

Sau khi sort lại Dataframe theo trình tự thời gian, ta nhóm 10 post lại với nhau và tính mean của lượng tương tác của chúng

```
df = df.sort_values(by='created_time')

grouped_data = df.groupby(df.index // 10)['tuong_tac'].mean()

plt.figure(figsize=(10, 6))
plt.plot(grouped_data.index, grouped_data, marker='o', linestyle='-', color='b')
plt.title('Average Interaction for Groups of 10 Posts')
plt.xlabel('Group Index')
plt.ylabel('Average Interaction')
plt.show()
```



Có thể thấy mặc dù page ngày càng có những bài đăng thu hút người theo dõi hơn, độ ổn định trong mức độ tương tác lại giảm đi. Trong khi ở nhóm từ 0 đến 10, lượng tương tác thấp nhất là khoảng 4000 và cao nhất là 6000 thì từ nhóm 10 đến 25 tương tác thấp nhất là khoảng 1000 còn cao nhất là 8000.

- Các từ khóa thường xuyên xuất hiện

```
full_post_text = str(df['message'])

wordcloud = WordCloud(stopwords=STOPWORDS,
                      background_color='white',
                      max_words=300,
                      width=2000, height=1200
                      ).generate(full_post_text)

plt.figure(figsize=(40,20))
plt.clf()
plt.imshow(wordcloud)
plt.axis('off')

plt.show()
```

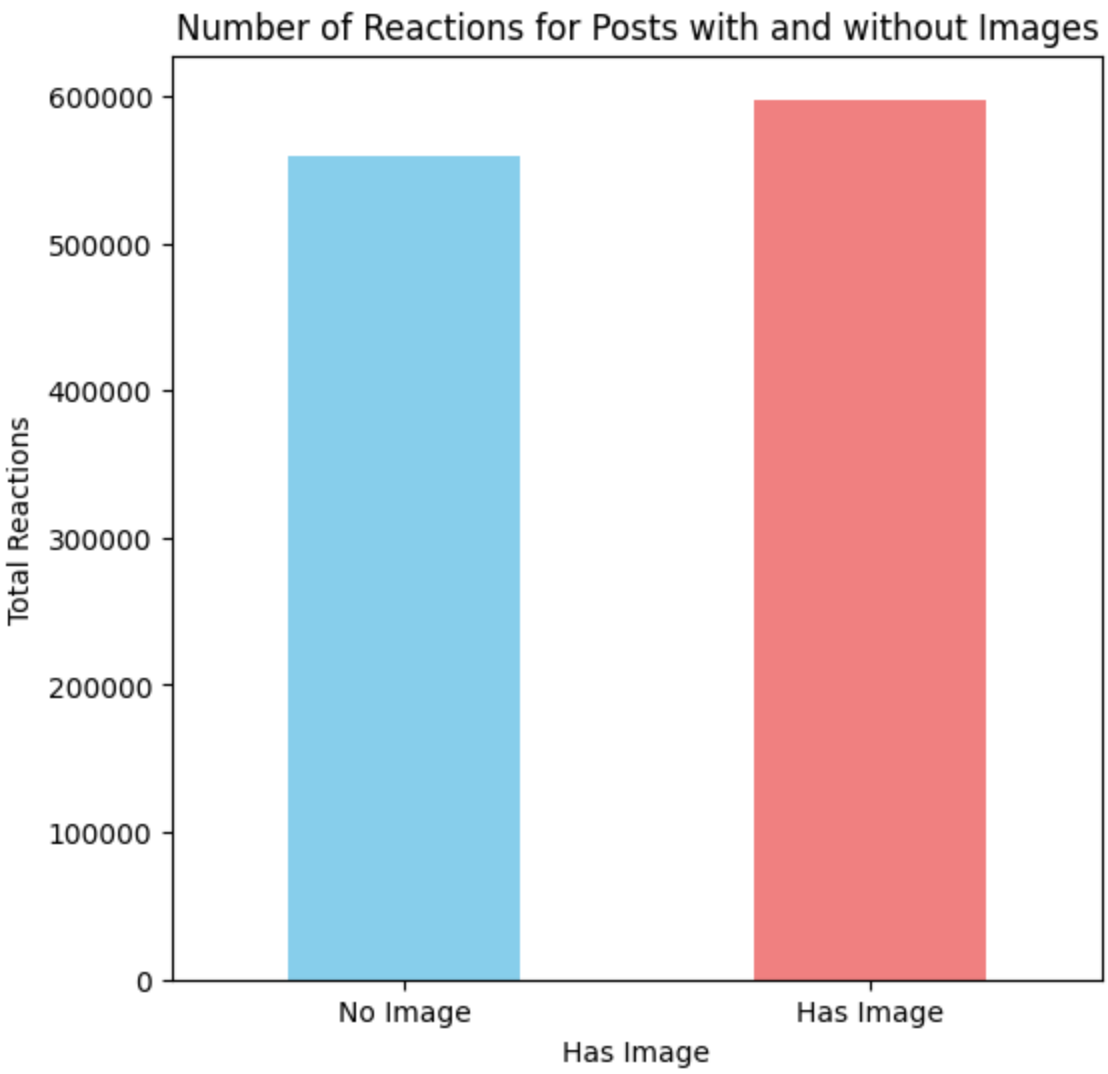
Những từ khóa phổ biến nhất bao gồm: “người”, “nghiệp”, “BTEDL”, “được”, “sếp”, “đi”, “bao”. Nhìn về tổng thể, ta có thể việc sử dụng những từ lóng rất thông dụng, cùng với

đó là những từ liên quan đến chủ đề việc làm. Ví dụ: “đồng nghiệp”, “sếp”, Lương”, Agency”, … và những từ lóng như “Ảo”, “Zô”, “ké”,….

- Độ ảnh hưởng của ảnh đối với độ phổ biến của bài viết

Ta tạo một cột mới tên là has_image. Do trước đó khi làm sạch dữ liệu, tất cả mọi post không có ảnh thì cột picture sẽ là một string rỗng. Do đó ta có thể phân loại được post có ảnh và post không có ảnh.

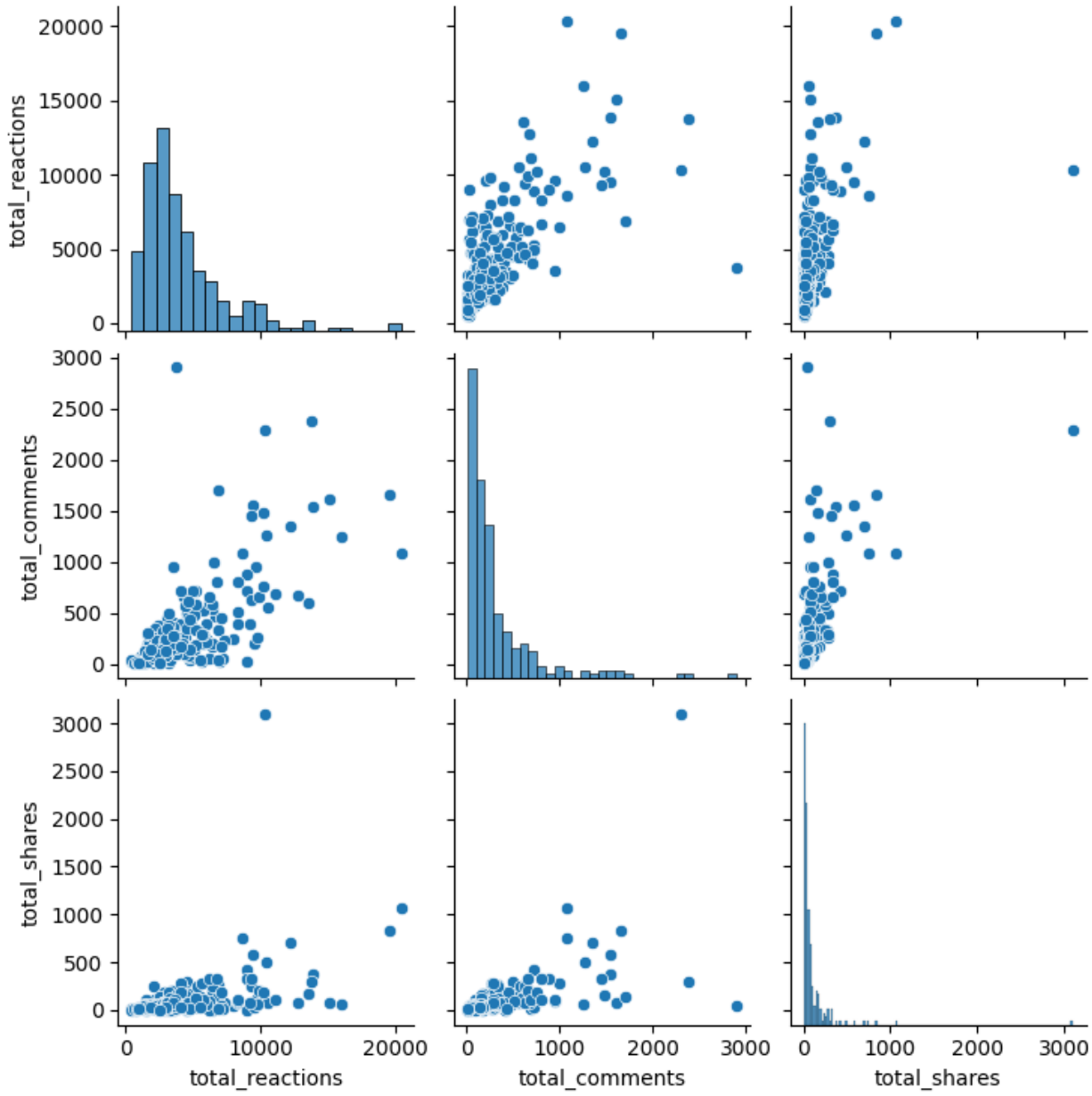
```
df['has_image'] = df['picture'].apply(lambda x: 1 if pd.notna(x) and x != ' ' else 0)

# Group by 'has_image' and calculate the sum of reactions for each group
grouped_data = df.groupby('has_image')['tuong_tac'].sum()

# Plotting
plt.figure(figsize=(6, 6))
grouped_data.plot(kind='bar', color=['skyblue', 'lightcoral'])
plt.title('Number of Reactions for Posts with and without Images')
plt.xlabel('Has Image')
plt.ylabel('Total Reactions')
plt.xticks([0, 1], ['No Image', 'Has Image'], rotation=0)
plt.show()
```



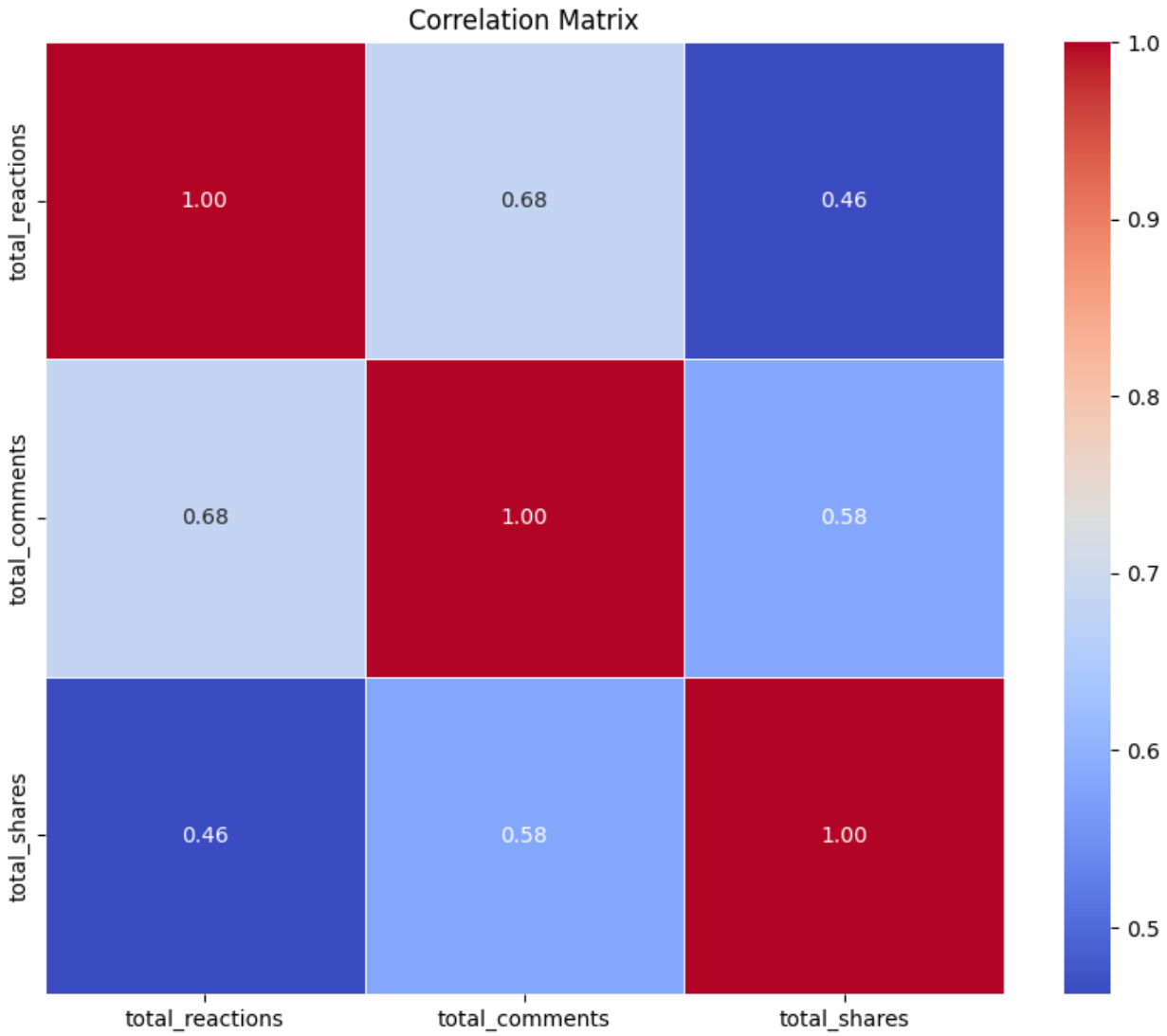
Có thể thấy, một bài viết có ảnh thường sẽ thu hút người theo dõi hơn. Tuy nhiên sự khác biết này là không nhiều.

- Sự tương quan giữa số lượng reactions, số lượng comments và số lượng shares

```python
#Sự tương quan giữa số lượng reactions và số lượng bình luận và share
columns_to_analyze = ['total_reactions', 'total_comments', 'total_shares']

# Subset the DataFrame with the selected columns
subset_df = df[columns_to_analyze]

# Calculate correlation matrix
correlation_matrix = subset_df.corr()

# Plot scatter plots for selected columns
sns.pairplot(subset_df)
plt.show()

# Plot correlation matrix as a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)
plt.title('Correlation Matrix')
plt.show()
```



Và đây là một biểu đồ nóng lạnh

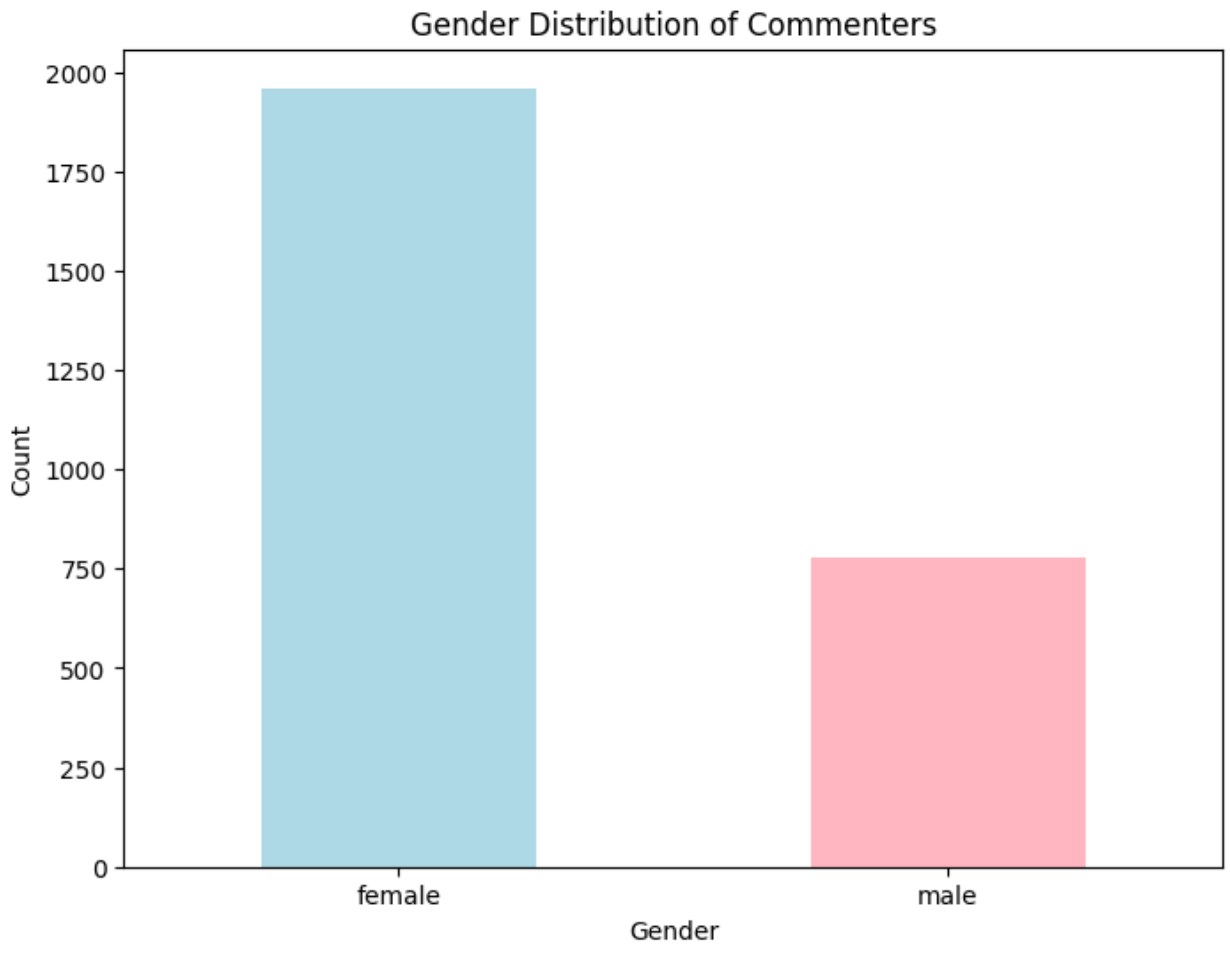


Từ hai biểu đồ này, ta có thể suy ra rằng, số lượng reactions và số lượng comment rất có tương quan với nhau trong khi số lượng reactions và số lượng shares thì không. Điều này có nghĩa rằng một người sau khi thả reaction rồi sẽ có 68% là viết comment trong khi chỉ có 46% là sẽ share. Việc có rất nhiều post có cả số lượng comment và reaction đều cao ủng hộ điều này.

- Đối tượng giới tính chính của page

Bằng việc biến comment thành một list dictionary, ta có thể truy cập mọi dữ liệu bên trong các comment bao gồm cả giới tính của người comment. Hàm try để skip qua những row mà comment có kí tự đặc biệt

```python
#Giới tính của người tương tác
import json
def parse_json_safe(x):
    try:
        return json.loads(x) if pd.notna(x) else []
    except json.JSONDecodeError as e:
        print(f"Error parsing JSON: {e}")
        return []

# Apply the parse_json_safe function to the 'comments' column
df['comments'] = df['comments'].apply(parse_json_safe)

# Extract gender information
df['gender'] = df['comments'].apply(lambda comments: [comment.get('from', {}).get('gender', None) for comment in comments])

# Filter out None values
df['gender'] = df['gender'].apply(lambda genders: [gender for gender in genders if gender is not None])

# Explode the list of genders to individual rows
gender_counts = df['gender'].explode().value_counts()

# Plotting
plt.figure(figsize=(8, 6))
gender_counts.plot(kind='bar', color=['lightblue', 'lightpink'])
plt.title('Gender Distribution of Commenters')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.show()
```

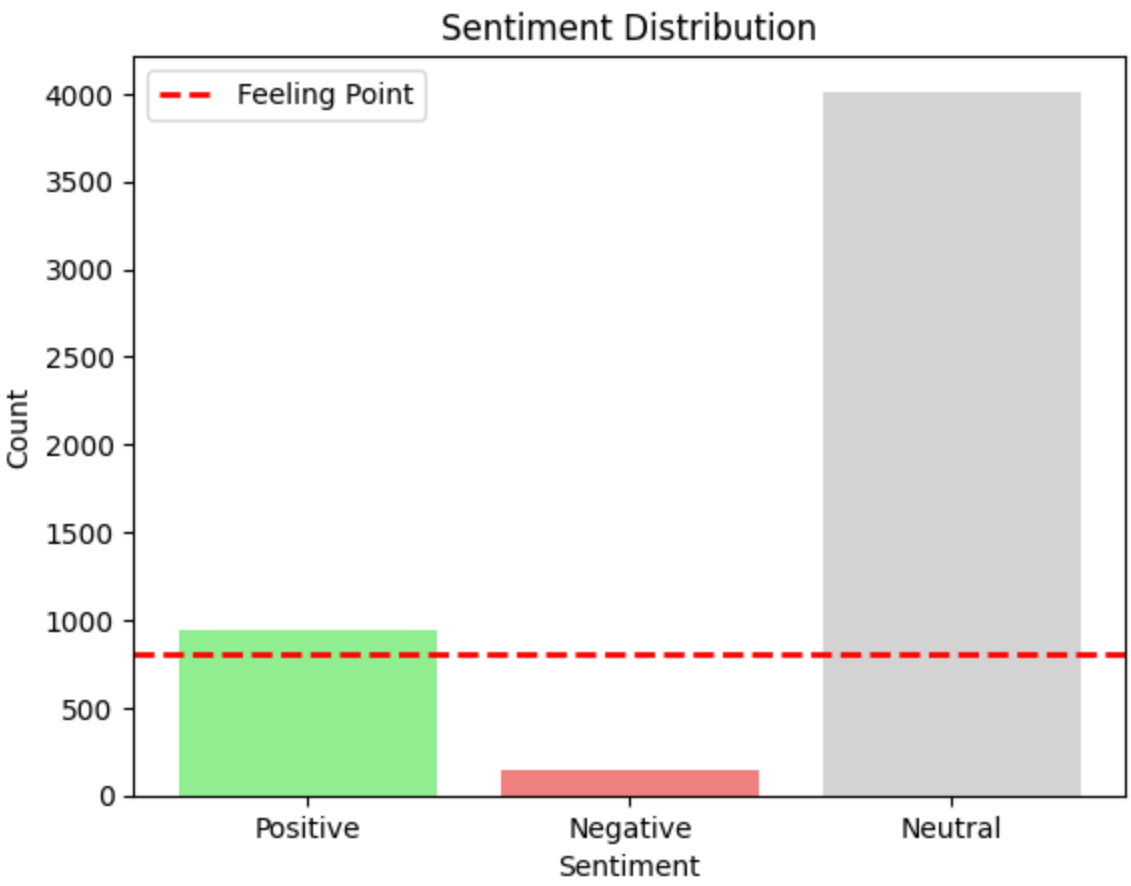


Từ biểu đồ trên có thể thấy nữ giới có vẻ là đối tượng được nhắm đến của page này với lượng người tương tác là nữ gần gấp 3 lần nam giới.

- Độ tích cực của comment

Ta sử dụng một thư viện có sẵn về NLP và Machine Learning để “đoán” trạng thái của comment: Tích cực, Tiêu cực hay Trung bình.

```
analyzer = SentimentIntensityAnalyzer()

# Predict sentiment for each message
sentiments = [analyzer.polarity_scores(message)['compound'] for message in all_messages]

# Classify sentiments as positive, negative, or neutral
classified_sentiments = ['Positive' if score >= 0.05 else 'Negative' if score <= -0.05 else 'Neutral' for score in sentiments]

# Display the results
negative = 0
positive = 0
neutral = 0
for message, sentiment in zip(all_messages, classified_sentiments):
    if sentiment == 'Negative':
        negative+=1
    if sentiment == 'Positive':
        positive+=1
    if sentiment == 'Neutral':
        neutral+=1
    print(f"Message: {message}, Sentiment: {sentiment}")

labels = ['Positive', 'Negative', 'Neutral']
counts = [positive, negative, neutral]
feeling_point = positive - negative
plt.bar(labels, counts, color=['lightgreen', 'lightcoral', 'lightgrey'])
plt.title('Sentiment Distribution')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.axhline(feeling_point, color='red', linestyle='dashed', linewidth=2, label='Feeling Point')

plt.legend()
plt.show()
```



Việc đa số comment đều là Neutral cũng có thể do thiếu sót của công cụ. Tuy nhiên, nhìn vào đồ thị này ta có thể thấy số lượng comment tích cực cao hơn hẳn số comment tiêu cực. Trong đó đường thẳng Feeling Point = Số tích cực - số tiêu cực. Điều này cho thấy sự hài lòng của đa số người theo dõi page.