

PROJECT 02
Mối quan hệ của dữ liệu

1. Thông tin thành viên

STT	Thành viên 1	Thành viên 2
MSSV	20424008	20424013
Họ và tên	Dương Mạnh Cường	Phạm Nguyễn Mỹ Diễm
Email	20424008@student.hcmus.edu.vn	20424013@student.hcmus.edu.vn

2. Phân công và kế hoạch thực hiện

STT	Công việc	Thành viên thực hiện	Thời gian thực hiện (Deadline: 4 tuần)	Hoàn thành của thành viên (%)	Hoàn thành tổng thể (%)
1	Thống kê dữ liệu cơ bản như trung bình, độ lệch chuẩn, kỳ vọng, phương sai và một số công thức thuộc về thống kê dữ liệu.	Diễm	Tuần 1 - 2	100	100
2	Thảo luận và chọn các trường dữ liệu để thể hiện trực quan bằng các loại biểu đồ đã học.	Cường – Diễm	Tuần 1 - 2	100	100
3	Nhận xét, code/thuật toán để thực hiện trực quan các mối quan hệ giữa các trường dữ liệu.	Cường – Diễm	Tuần 1 - 2	100	100
4	Thể hiện quan hệ phải tích hợp dần dần nghĩa là từ đơn giản đến phức tạp, từ một trường đơn đến quan hệ giữa nhiều trường....	Cường	Tuần 2 – 4	100	100
5	Ngoài quan hệ độc lập, xem xét liệu trong dữ liệu có quan hệ nhân quả không (cause-effect). Chứng minh thông qua các phép trực quan dữ liệu.	Cường – Diễm	Tuần 2 – 4	100	100
6	Triển khai nhiều mối quan hệ nhất có thể và phủ được nhiều loại biểu đồ đã học. Tối thiểu 10 quan hệ với 6 dạng biểu đồ khác nhau.	Cường – Diễm	Tuần 2 – 4	100	100
7	Viết báo cáo	Cường – Diễm	Tuần 4	100	100

3. Báo cáo theo các mục ở phần 2

3.1 Thực hiện các thống kê dữ liệu cơ bản như trung bình, độ lệch chuẩn, kỳ vọng, phương sai và một số các công thức thuộc về thống kê dữ liệu.

- Đọc dữ liệu từ project 1.

```
1 reviews = pd.read_csv("./data/normalize_reviews.csv").fillna("")
```

```
1 reviews.head()
```

	raw_comment	label	normalize_comment	emoji
0	Giao hàng kh đúng cần phê bình hjjjjjhhd...	0	giao hàng không đúng cần phê bình	
1	Chất lượng sản phẩm tạm được. Giao...	0	chất lượng sản phẩm tạm được giao ...	
2	Ko có lắc tay như hình	0	không có lắc tay như hình	
3	Giao hàng lâu. Bảo có lắc tay mà k thâ...	0	giao hàng lâu bảo có lắc tay mà không ...	
4	Mình mua 2 cái, một dùng ok. Một cái k...	0	mua cái một dùng ok một cái không chạ... 😢	😢

- Bây giờ ta sẽ tạo ra hai feature có tên là:

- **length**: là chiều dài của feature **normalize_comment**.
- **no_words**: là số từ của feature **normalize_comment**.

```
1 reviews['length'] = reviews['normalize_comment'].str.len()
2 reviews['no_words'] = reviews['normalize_comment'].str.split(" ").agg([len])
```

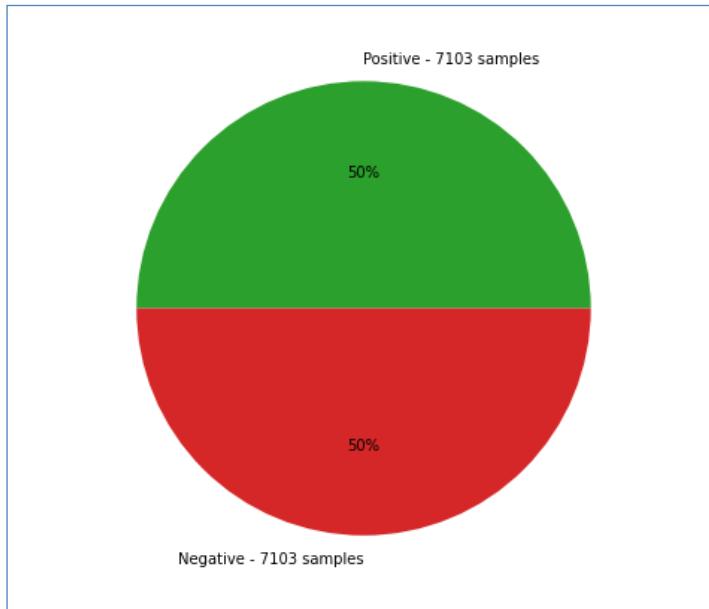
```
1 reviews.head()
```

	raw_comment	label	normalize_comment	emoji	length	no_words
0	Giao hàng kh đúng cần phê bình hjjjjjhhd...	0	giao hàng không đúng cần phê bình		40	7
1	Chất lượng sản phẩm tạm được. Giao...	0	chất lượng sản phẩm tạm được giao ...		79	13
2	Ko có lắc tay như hình	0	không có lắc tay như hình		31	6
3	Giao hàng lâu. Bảo có lắc tay mà k thâ...	0	giao hàng lâu bảo có lắc tay mà không ...		151	28
4	Mình mua 2 cái, một dùng ok. Một cái k...	0	mua cái một dùng ok một cái không chạ... 😢	😢	55	10

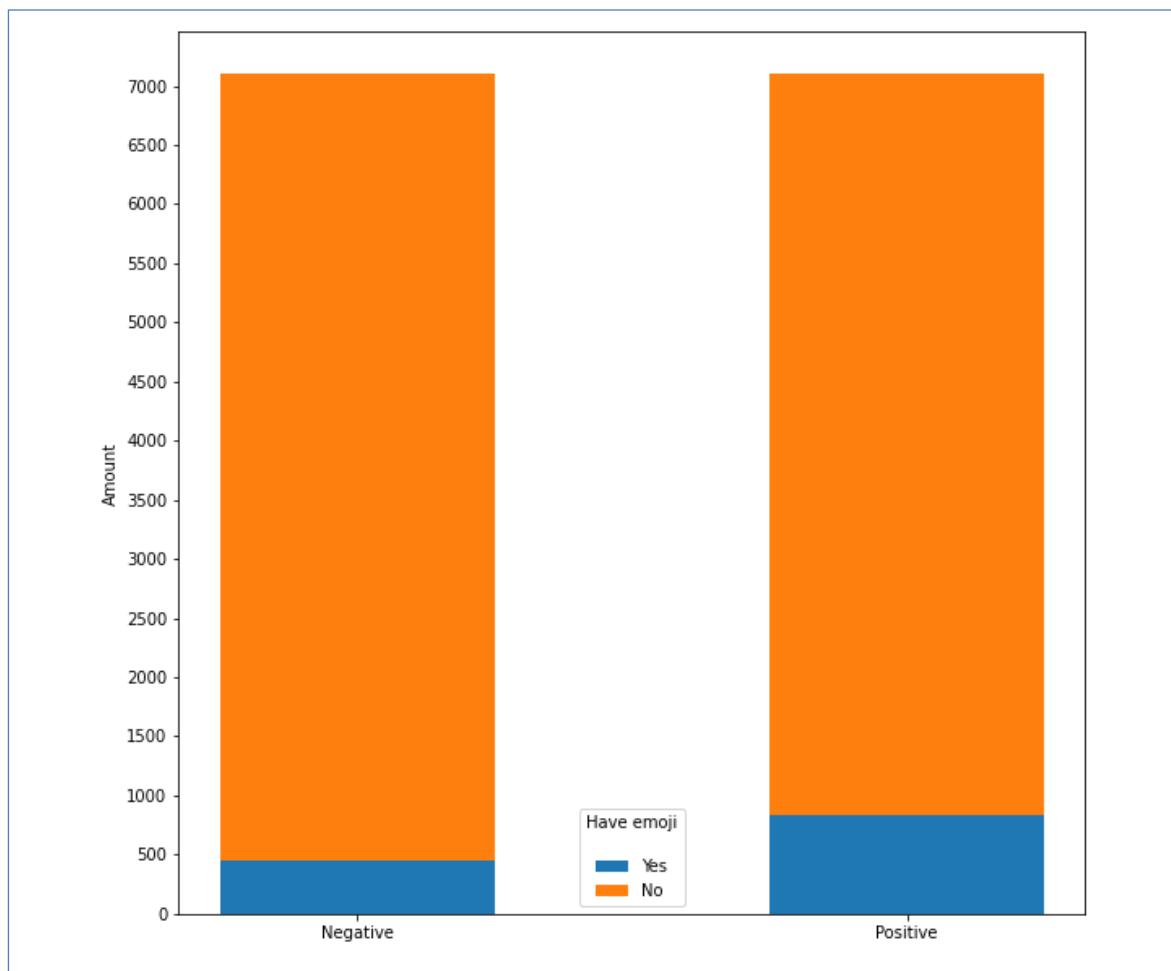
- Bây giờ chúng ta sẽ sắp xếp lại thứ tự của các feature trong reviews.

1	<code>reviews = reviews[['raw_comment', 'normalize_comment', 'length', 'no_words', 'emoji', 'label']]</code>
1	<code>reviews.head()</code>
<hr/>	
0	raw_comment normalize_comment length no_words emoji label
0	Giao hàng kh ^d ng c ^a n ph ^e binh h ^j jjjhhd... giao hàng kh ^d ng c ^a n ph ^e binh 40 7 0
1	Ch ^a t l ^u ng s ^a n p ^h am t ^a m đ ^u rc. Giao... ch ^a t l ^u ng s ^a n p ^h am t ^a m đ ^u rc giao ... 79 13 0
2	Ko có l ^a c t ^a y nh ^u h ⁱ nh kh ^o ng có l ^a c t ^a y nh ^u h ⁱ nh 31 6 0
3	Giao hàng l ^a u. B ^a o có l ^a c t ^a y m ^a k th ^a ... giao hàng l ^a u b ^a o có l ^a c t ^a y m ^a kh ^o ng ... 151 28 0
4	Minh mua 2 c ^a í, m ^o t d ^u ng ok. M ^o t c ^a í k... m ^u a c ^a í m ^o t d ^u ng ok m ^o t c ^a í kh ^o ng ch ^a ... 55 10 😊 0

- Mặc dù ở **project 1** chúng ta đã biết khá rõ ta đã làm mọi thử để ta có một dataset mà **cân bằng về số lượng observe ở các class**. Mặc dù vậy ta có thể trực quan hóa về số lượng các sample trong từng class của dataset này lên một biểu đồ nào đó. Vì ta chỉ có 2 class là **negative** và **positive** - nên pie chart sẽ là một biểu đồ lí tưởng để làm điều này.

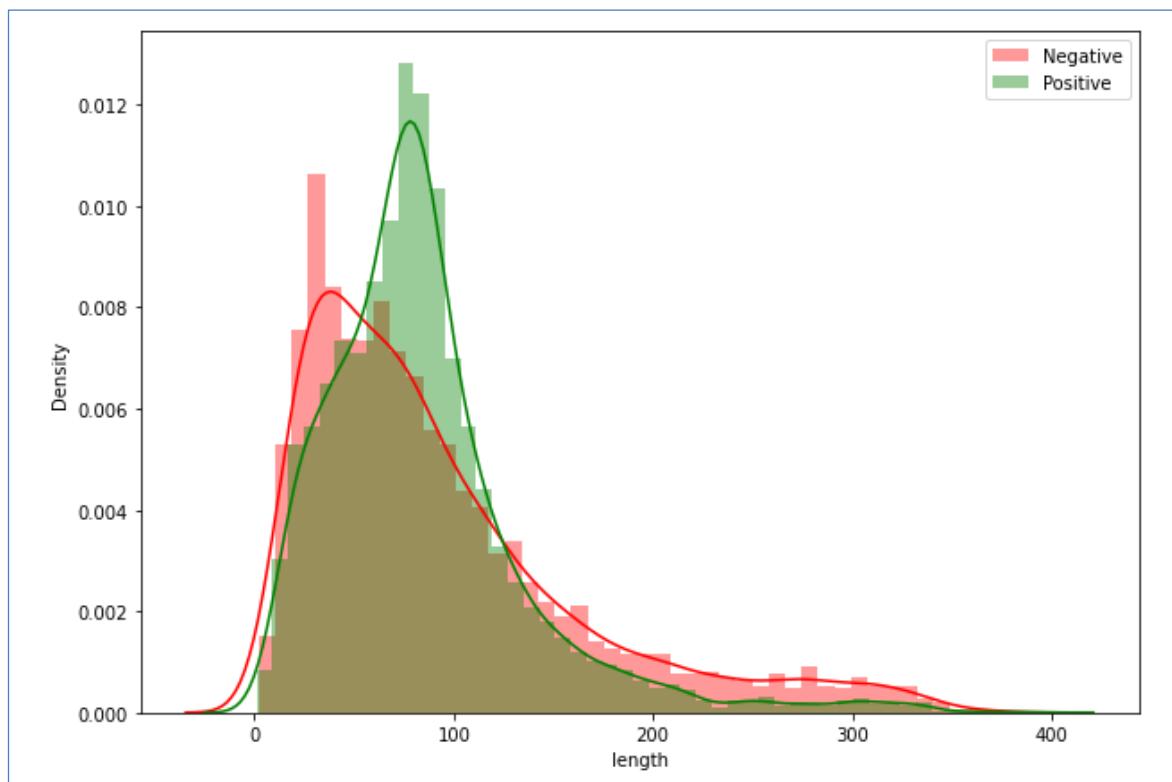


- **Nhận xét:** Vì dữ liệu của ta ở **project 1** ta đã làm nó có sự cân bằng giữa hai class, và với dataset này cũng chỉ có 2 class trong target variable nên ta chưa thấy được lợi ích của quá trình data visualization, nếu target variable của chúng ta có nhiều class hơn thì pie chart giúp ta thấy được class nào đang áp đảo các class nào hơn về mặt số lượng, tỉ lệ giữa các class - điều mà nếu chỉ nhìn bằng các con số ta đôi khi khó mường tượng và rất mơ hồ.
- Tiếp theo, chúng ta cũng cần đặt câu hỏi: "*Liệu giữa negative và positive class, khách hàng của class nào có xu hướng sử dụng comment trong phần bình luận của họ hơn?*".
- Vì ở đây, chúng ta đang **quan tâm về số lượng** nên bar chart là một biểu đồ phù hợp để ta trực quan hóa được điều này.



- **Nhận xét:**

- Có thể thấy rằng số khách hàng thuộc nhóm positive có xu hướng sử dụng comment nhiều hơn trong comment của họ so với khách hàng nhóm negative, tuy nhiên khác biệt này cũng không quá lớn.
- Ngoài ra, ta có thể thấy khách hàng trên shopee đa phần cũng sẽ không sử dụng emoji trong comment của mình cho lắm. Diễn hình với khách hàng nhóm **negative**, tỉ lệ khách hàng có dùng emoji trong comment của mình chỉ khoảng từ (5, 7) phần trăm. Ở nhóm **positive** là khoảng (10, 13) phần trăm.
- Tiếp theo, ta có thể xem phân phối của feature **length** để xem giữa class **negative** và **positive** có điểm gì khác nhau.



- **Nhận xét:**

- Các khách hàng thuộc nhóm negative thông thường có chiều dài comment nằm trong khoảng từ 25 đến 80.
 - Trong khi đó, các khách hàng nhóm positive thì thường có bình luận dài hơn so với nhóm negative, thường nằm trong khoảng từ 40 đến 130.
 - Một điều nữa là khách hàng nhóm positive không chỉ comment dài hơn mà còn có mật độ tập trung cao hơn.
 - Tuy nhiên vẫn có các comment mà có chiều dài chạm mốc hơn 300, liệu chúng có phải outlier do quá trình tiền xử lí ta làm không tốt không, bước dưới đây ta sẽ nhìn kĩ vào hơn các sample này.
- Ở bước này, ta sẽ xem các comment mà có chiều dài lớn hơn 300, xem chúng có phải là outlier hay noise không.

```

1 cmt_gt_300 = reviews.loc[reviews['length'] > 300].index[['normalize_comment']]
2
3 len(cmt_gt_300)

```

234

```

1 for cmt in cmt_gt_300[:10]:
2     print(cmt, "\n")

```

hang không như hình mẫu trong mẫu viền mặt đồng hồ màu vàng nhưng mua về màu bạc có vết sứt to dây đồng hồ mỏng manh giao hàng siêu siêu chậm đi kèm là màu trắng ở mặt đeo hiệu rất trắng rất mới kết hợp dây màu đen tuy nhiên rất hợp lí cả giá rẻ nên tạm chấp nhận

không trung thực nhìn thấy địa chỉ là nên mới đặt vì thực sự cần gấp đồ cũng gọi điện trực tiếp nhờ giục giao gửi đồ từ kho tb nhưng khẳng định là gửi từ kho thanh xuân nhờ cả bên bưu cục check mã đơn hàng tự lấy còn không thêm rep d ịch vụ kém

mua chiếc khác nhau chiếc ok rất ưng nhưng chiếc lăm khá thất vọng trước cũng đặt bên lần chiếc y bây giờ mua tiếp nh ưng màu dây khá đậm đen bóng xấu còn phần dây cài có mỗi dây cài trong dây cài bé quá không cài được luôn cố néh vô dứt luôn dây

ỗn có lẽ hợp giá có bùa yêu thề bảo hành mùi không hắc không ôn nhiều chỉ thừa cắp bắn ở chỗ chất vải mỏng quai đeo y ếu thề bảo hành nhưng có tác dụng như nào p xem vì không feedback không như mong muốn bùa yêu mà sản phẩm như ai yêu nỗi

chất lượng tớ còn buồn thực sự vì mua tặng nên thông tin người nhận là bạn tớ bạn nhận được tin kia nghĩ gọi từ chối là được nhưng bên ship gần như ép bạn nhận hàng không thích tự hoàn nhắn không được biết là ngu nhưng phiền quá nên b ạn nhận cho xong

đặt bờ nhạt nhưng hết nên dành phải chọn nau ở ngoài nhìn đậm hơn trên hình già không hài lòng báo hết màu thích nói có thể chờ nhưng nói phải giao hàng không huỷ đợi tuần sau hàng về đặt nhưng nếu đợi giá không tốt thấy không đ ược hỗ trợ tốt

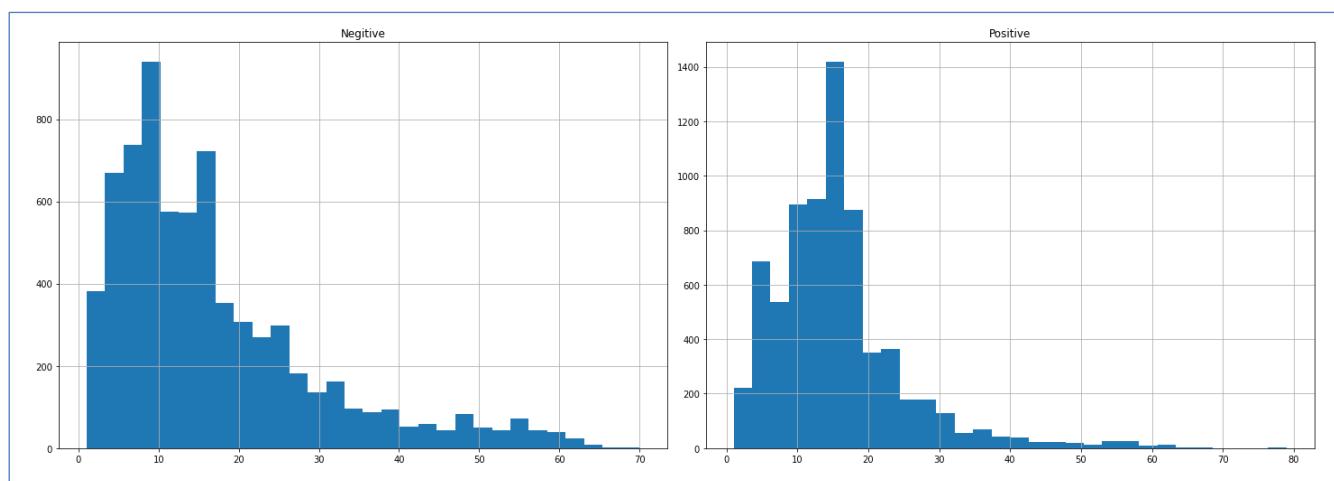
chưa bao giờ phải cho bán hàng cả cũng chưa bao giờ phải đợi đến ngày cho việc mua sản phẩm trên nhưng mang tiếng mà làm ăn dịch vụ cực kì kém chất lượng sản phẩm cũng bình thường tiền nào của ấy các bạn nhé đóng gói kém đi xa bong cá túi ngoài chán

quá kinh khủng túi ngoài có xác gián phân huỷ vừa mốc vừa thối bẩn cả nhà cả tay giặt xà phòng mà vẫn còn thối trong rách không kiểm tra hàng bên ngoài lẫn trong đừng giải thích sai sót vì trăm ngàn đơn được quyền sai sót bán hàng có lấy sót đồng tiền nào không

sản phẩm ok không tốt có xin thêm túi làm quà nói lu bu không được ok có thể cảm việc sản phẩm mới không có nhãn mác rất bức mua làm quà mà không nhãn mác không túi hăng fan hồi đọc mà không thèm trả lời luôn biết không nhãn mác là ch ắc chắn không mua sản phẩm

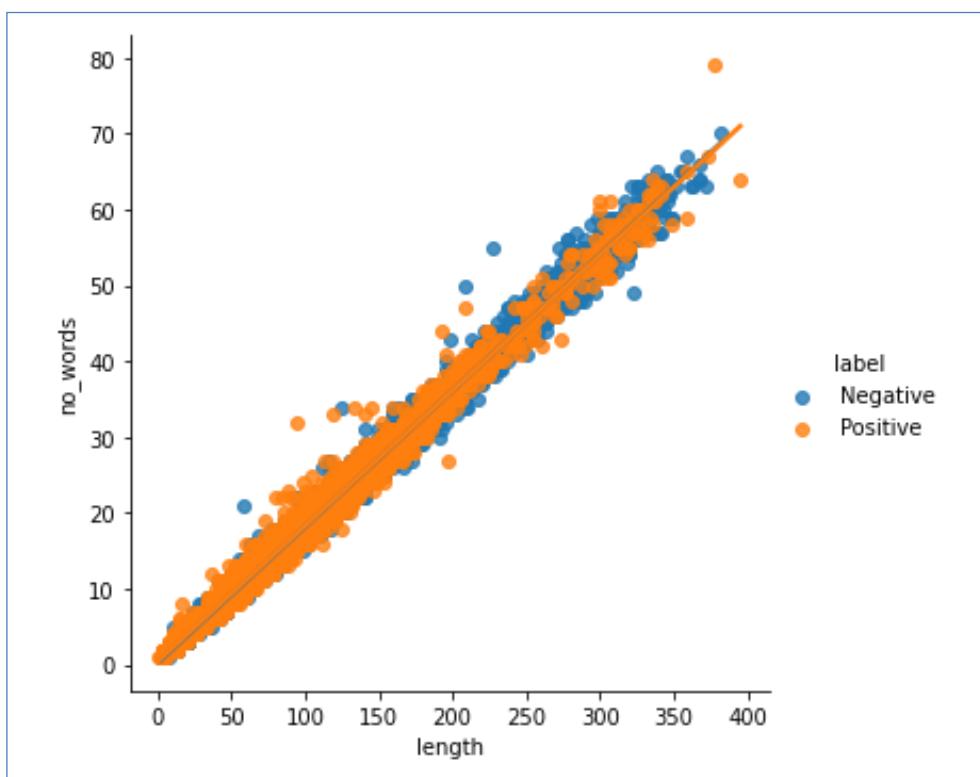
thấy mọi người khá thích áo vì vải dày dặn mát nhưng riêng cá nhân thấy mỏng có thể là lộ nội y mờ xí không như mong đợi lầm vì bỏ ra không mua cái áo mong muốn chất lượng tốt như giá tiền còn dầu là thêu khía thích cách đóng gói ok ch ất áo cân nhắc

- **Nhận xét:** Có lẽ là do ta suy nghĩ quá nhiều, dù có đến hơn 200 bình luận có chiều dài hơn 300 nhưng những bình luận này mang lại cho ta nhiều thông tin hơn là các bình luận thuộc nhóm có mật độ cao mà ta đã làm bước trước.
- Bước trước ta đã minh họa một distribution plot trên feature **length**, bước này ta có thể sử dụng histogram để biểu diễn frequency của feature **no_words** trong các comment, nghe có vẻ dư thừa vì số từ nhiều đồng nghĩa với việc câu sẽ dài ra. Nhưng với một dữ liệu mà ta chỉ mới tiếp xúc lần đầu, mọi thao tác xử lý trên nó đều có giá trị, biết đâu sẽ cho ta có được nhiều các nhìn hơn về dữ liệu hiện có.
- Tuy nhiên, ta cũng cần lưu ý, vì câu của ta có chiều dài đa phần nằm trong khoảng từ 25 đến 130, nên ta cần phải chọn một **bin value** phù hợp, ở đây ta sẽ thử với bin 30.



- **Nhận xét:**

- Nhóm khách hàng negative thường dùng khoảng từ 5 đến 15 từ trong một bình luận, frequency là 600.
- Nhóm khách hàng positive thường dùng khoảng 8 đến 18 từ trong một bình luận với frequency cao hơn là 800, đặc biệt họ thường dùng khoảng 15 từ để bình luận với frequency chạm mốc hơn 1400.
- Nhìn chung, các từ tiếng việt phổ biến thường trong khoảng 4 kí tự, vậy chiều dài trung bình của một bình luận nếu ta tính một cách ước chừng là $4 * 10 + 9 = 49$. Khi ta xem lại distribution plot mà ta đã vẽ cho feature length, ta thấy được hai đường KDE cắt nhau lần đầu cũng tại length có giá trị đâu đó nằm loanh quanh 50.
- Từ đây thì với tiếng việt, suy nghĩ của ta ban đầu là từ nhiều thì câu dài cũng có phần chính xác. Với các ngôn ngữ khác, điều này có thể không còn đúng nữa, trong tiếng việt từ dài nhất là **nghiêng** với 7 kí tự và ngắn nhất là **ba** với 2 kí tự, trong khi đó với các ngôn ngữ khác như tiếng anh có từ lên đến 45 kí tự là "**Pneumonoultramicroscopicsilicovolcanoconiosis**" chỉ dùng để chỉ một căn bệnh về phổi, ngoài ra ta cũng có thể tìm hiểu về độ dài trung bình của các từ trong các ngôn ngữ khác nhau ở đây (<http://www.ravi.io/language-word-lengths>).
- Để thêm phần khẳng định cho tuyên bố trên, chúng ta có thể biểu diễn mối quan hệ giữa **length** và **no_words** có thực sự có mối quan hệ tuyến tính hay không thông qua biểu đồ **reg plot**.



- **Nhận xét:**

- Biểu đồ này là một bằng chứng mạnh mẽ cho một positive correlation giữa **length** và **no_words** với 2 regression line nằm hầu như tiếp vào nhau.
- Và cũng như histogram plot trước đó, ta thấy có sự chồng chéo giữa các class trong feature **label**.

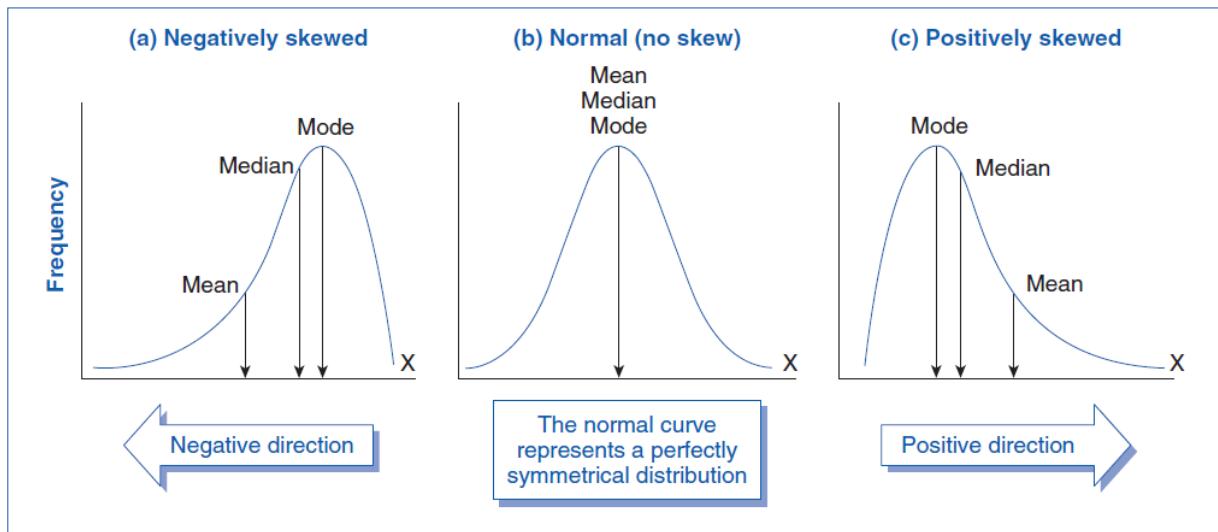
- Ở các bước trên, ta dùng phương pháp visualization phân tích dữ liệu theo từng class positive và negative riêng biệt. Nếu ta muốn xem toàn bộ bức tranh, không phân biệt negative hay positive, thì ta có thể áp dụng các phương pháp của thống kê mô tả trên hai feature là **length** và **no_words** bằng hàm **describe()**.

```
1 reviews[['length', 'no_words']].describe()
```

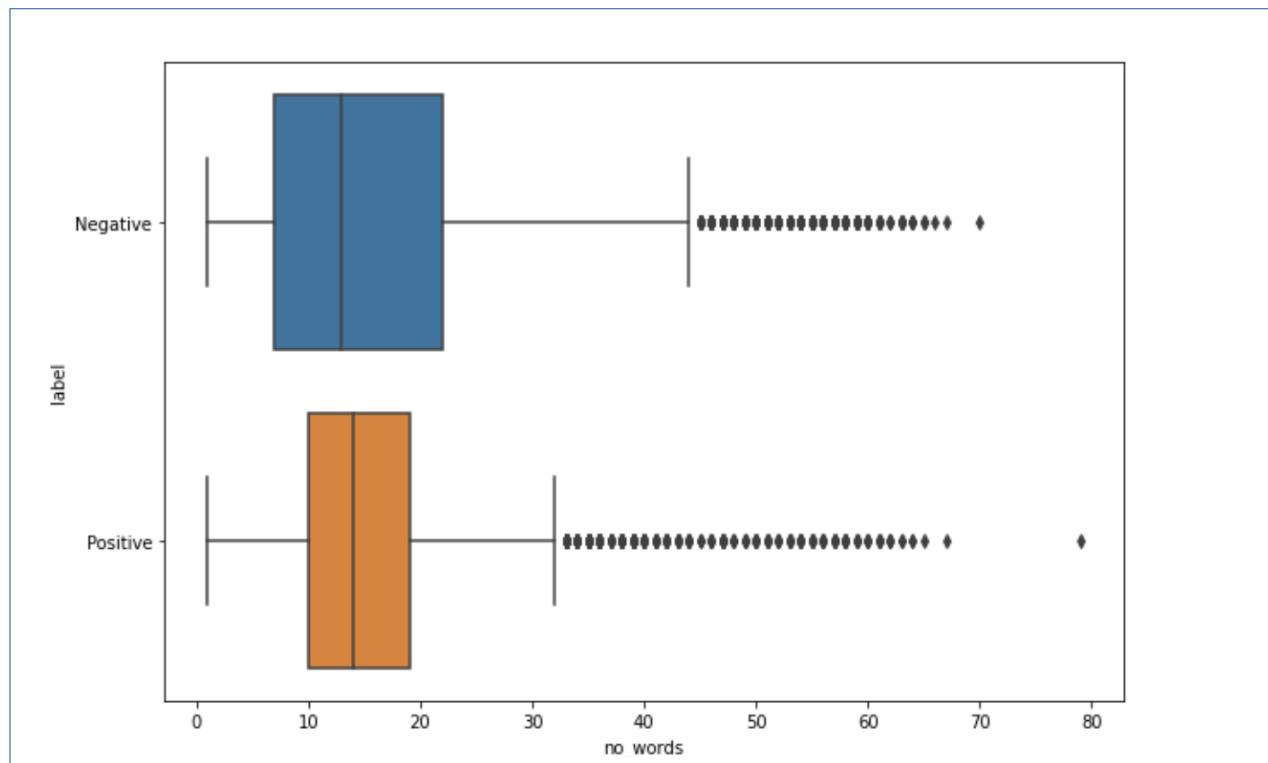
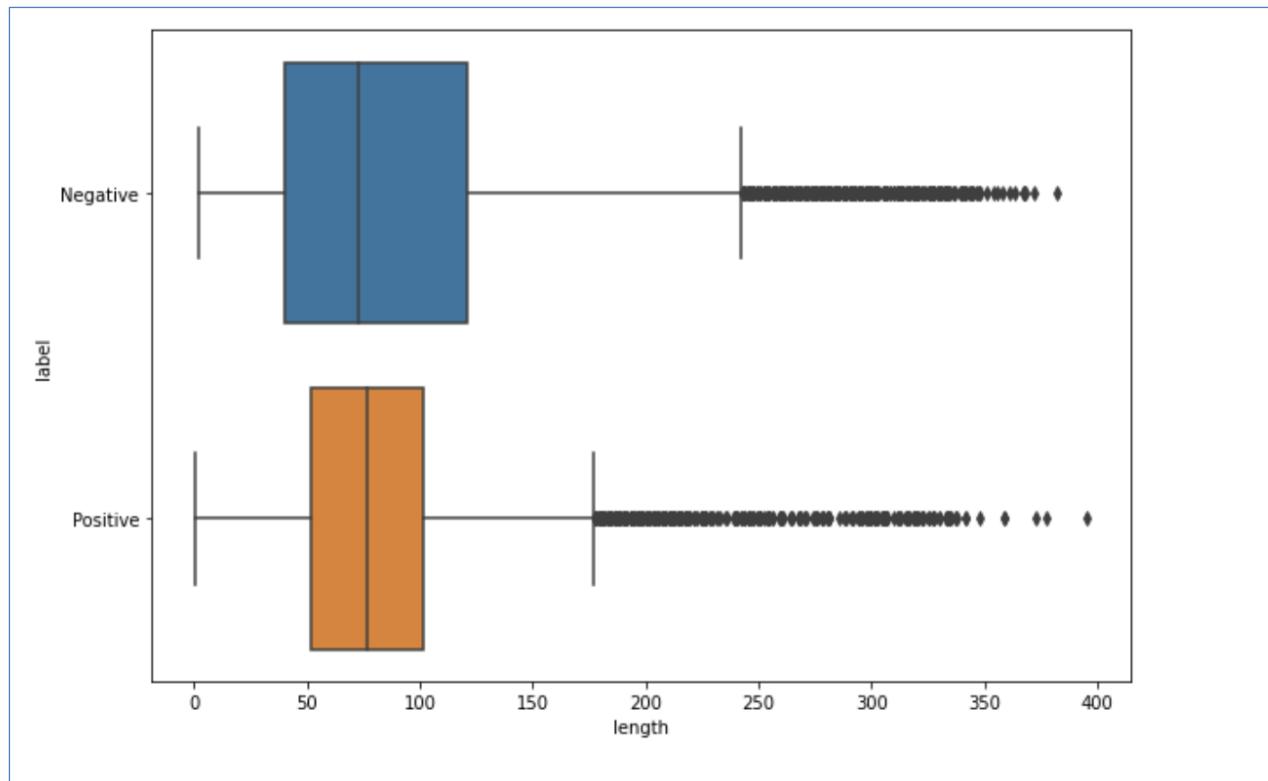
	length	no_words
count	14206.000000	14206.000000
mean	88.684922	16.200831
std	62.465665	11.249186
min	1.000000	1.000000
25%	46.000000	9.000000
50%	76.000000	14.000000
75%	110.000000	20.000000
max	395.000000	79.000000

- **Nhận xét:**

- o Chúng ta thấy rằng bình luận có **length** ngắn nhất là 1, cũng chưa biết từ tiếng việt nào có 1 kí tự nữa, dài nhất là 395, trung bình là khoảng 88. Mọi thứ đều hợp lý với biểu đồ distribution mà ta vẽ cho từng class riêng biệt.
- o Ở feature **no_words**, thì bình luận có nhiều từ nhất là 79 từ với trung bình là 16 từ cho một bình luận.
- o Ở đây, không cần trực quan hóa mà chỉ cần nhìn vào số liệu, ta rõ ràng thấy được cả **length** và **no_words** đều có phân phối lệch phải (**right-skewed**). Tuy nhiên, bài toán của ta là sentiment analysis, ta quan tâm hơn về mặt ngữ nghĩa từ ngôn ngữ - cấu trúc câu hơn là số liệu thống kê. Giả sử đây là bài toán hồi quy, thì ta có thể áp dụng **log-transformation** để giảm độ lệch phải của dữ liệu.



- Tuy nhiên, để mọi thứ rõ ràng, ta nên có một thứ gì đó để trực quan hóa mọi giá trị của thông kê mô tả. Và có một loại biểu đồ mà có thể thực hiện được điều này là boxplot, nó giúp ta dễ dàng so sánh giữa các số liệu thống kê trên các trường dữ liệu.



- Nhận xét:

- Nếu ta chỉ nhìn vào các giá trị được đưa ra bởi hàm **describe()** thì có lẽ ta đã bị đánh lừa. Rõ ràng ở đây, ta thấy rằng phân phối của hai feature **length** và **no_words** trên nhóm khách hàng positive rất gần phân phối chuẩn (nếu như không tính phần outlier được boxplot thể hiện) vì giá trị mean của boxplot nằm giữa hộp vuông.
 - Còn với nhóm khách hàng nhóm negative, nó bị lệch phải nhưng không quá nhiều. Nếu đây là các feature mà ta quan tâm trong lúc model training, ta có thể áp dụng log-normalization lên chúng trước sau đó training sau.
- Gần đây, chúng em được một bạn trên Twitter quảng cáo về một package là **pandas-profiling** dùng để thống kê mô tả và trực quan hóa dữ liệu nhanh trên pandas dataframe, cài đặt và dùng như sau:
- pip install [pandas-profiling](#)
 - pip install [ipywidgets](#)

```

1 import pandas_profiling as pp
2
3 profile = pp.ProfileReport(reviews)
4 profile
5
6 Summarize dataset: 100%|██████████| 23/23 [00:02<00:00,  8.99it/s, Completed]
7 Generate report structure: 100%|██████████| 1/1 [00:01<00:00,  1.64s/it]
8 Render HTML: 100%|██████████| 1/1 [00:00<00:00,  3.52it/s]

```

Pandas Profiling Report Overview Variables Interactions Correlations Missing values Sample

Overview

Overview Alerts 16 Reproduction

Dataset statistics		Variable types	
Number of variables	6	Categorical	4
Number of observations	14206	Numeric	2
Missing cells	0		
Missing cells (%)	0.0%		
Duplicate rows	0		
Duplicate rows (%)	0.0%		
Total size in memory	666.0 KiB		
Average record size in memory	48.0 B		



Alerts

raw_comment has a high cardinality: 14206 distinct values	High cardinality
normalize_comment has a high cardinality: 14206 distinct values	High cardinality
emoji has a high cardinality: 310 distinct values	High cardinality
length is highly correlated with no_words	High correlation
no_words is highly correlated with length	High correlation
length is highly correlated with no_words	High correlation
no_words is highly correlated with length	High correlation
length is highly correlated with no_words	High correlation
no_words is highly correlated with length	High correlation
length is highly correlated with no_words	High correlation
no_words is highly correlated with length	High correlation
raw_comment is uniformly distributed	Uniform
normalize_comment is uniformly distributed	Uniform
label is uniformly distributed	Uniform
raw_comment has unique values	Unique
normalize_comment has unique values	Unique

Pandas Profiling Report

Overview Variables Interactions Correlations Missing values Sample

Variables

raw_comment

Categorical

[HIGH CARDINALITY](#)
[UNIFORM](#)
[UNIQUE](#)

Distinct	14206
Distinct (%)	100.0%
Missing	0
Missing (%)	0.0%
Memory size	111.1 KiB

chất vải rất ok hình đều là thêu,... 1
Hàng oke hàng oke. Đóng hói t... 1
Chất lượng sản phẩm tốt, giao ... 1
Đồ đẹp bé mắc vua hài lòng 1
Sản phẩm chất lượng tốt mua ... 1
Other values (14201) 14201

[Toggle details](#)

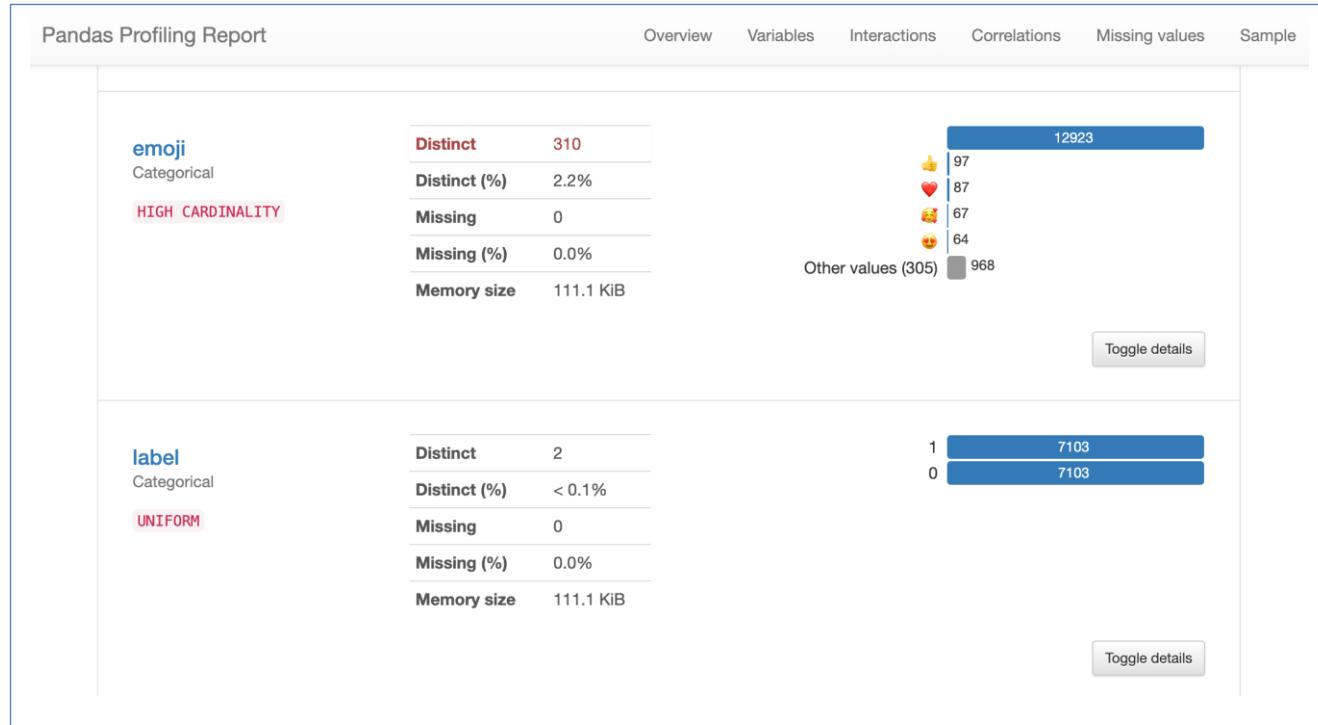
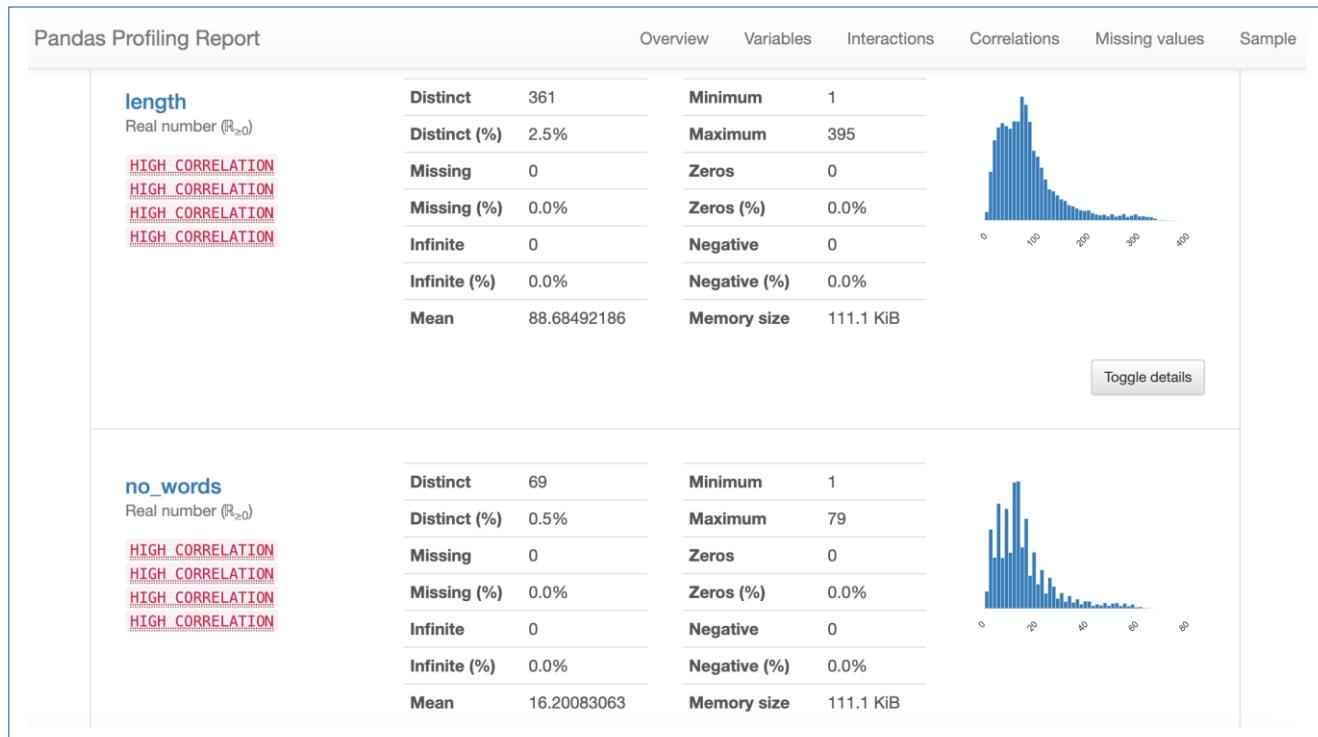
normalize_comment

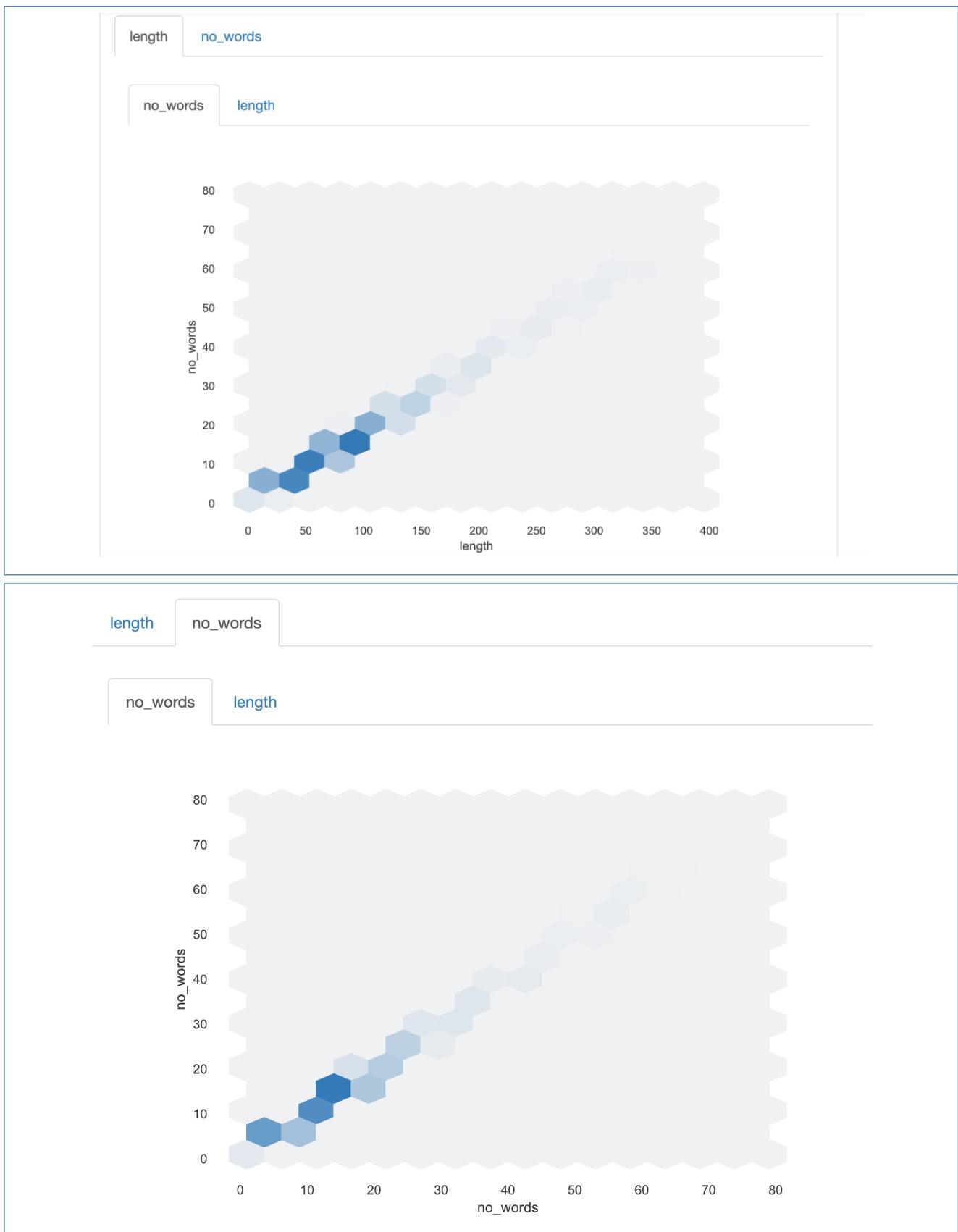
Categorical

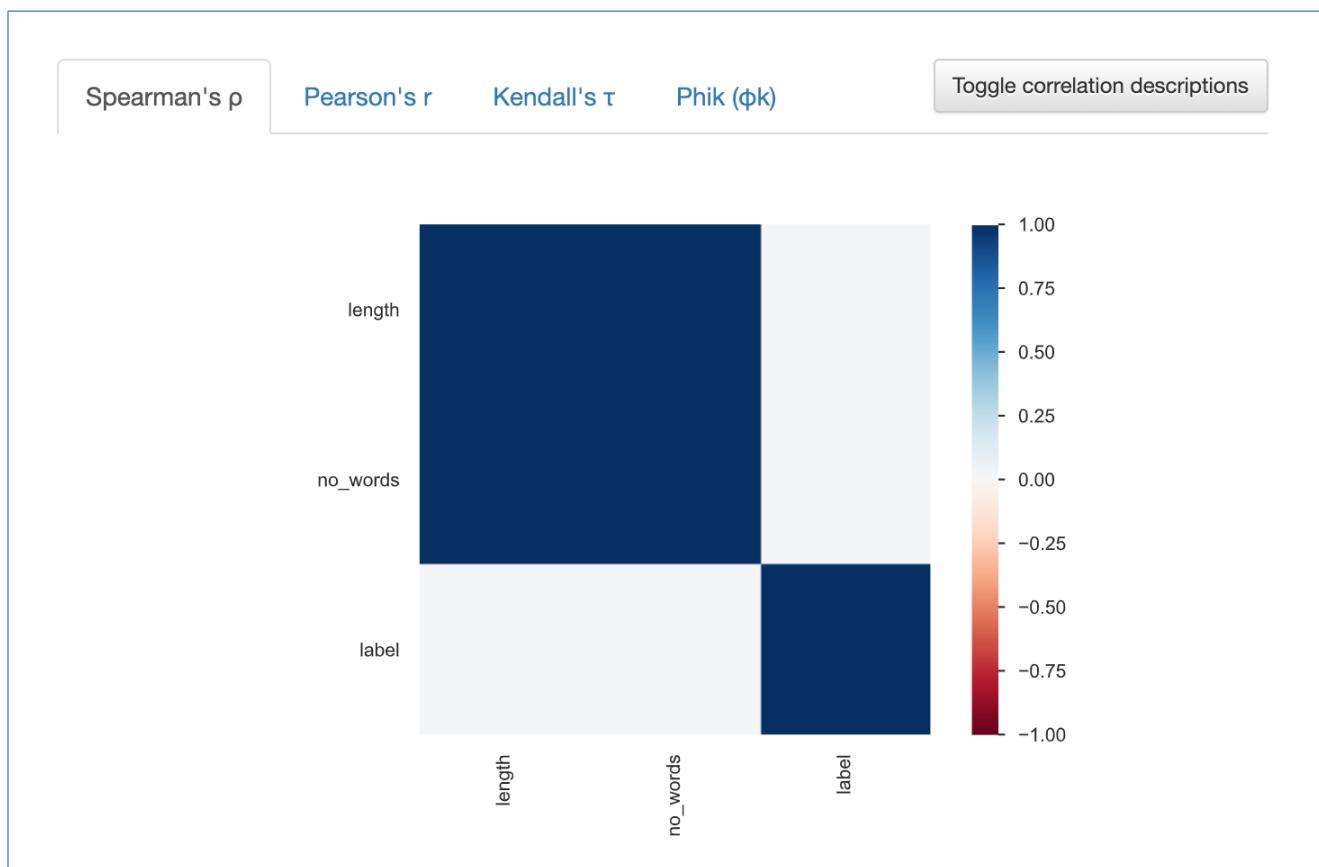
[HIGH CARDINALITY](#)
[UNIFORM](#)
[UNIQUE](#)

Distinct	14206
Distinct (%)	100.0%
Missing	0
Missing (%)	0.0%
Memory size	111.1 KiB

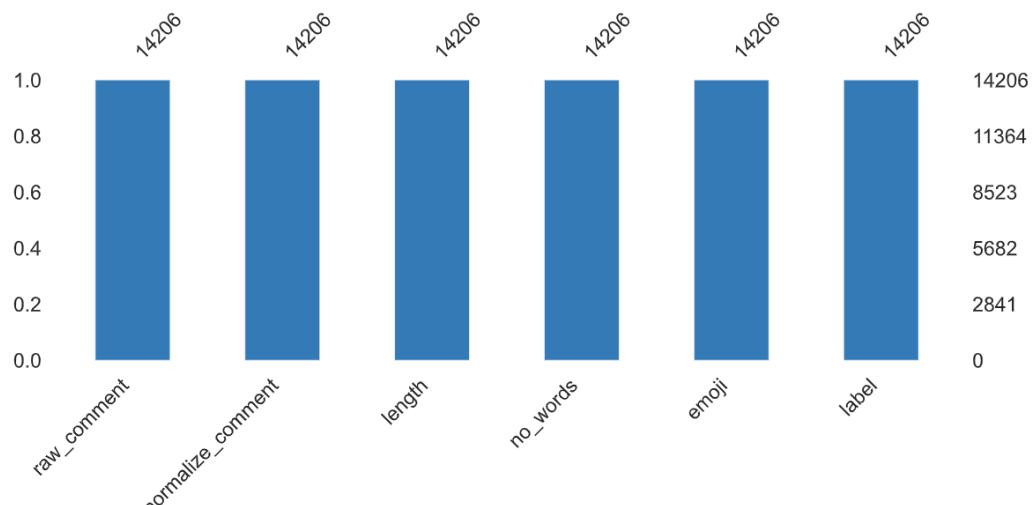
bất ngờ nhận được hàng quần ... 1
mông chưa mặc rách 1
hang vietnam không phải quan... 1
tất đẹp chồng không vừa nên đ... 1
chất đẹp hơn cả tưởng a ơi già... 1
Other values (14201) 14201







Missing values

[Count](#)
[Matrix](#)


A simple visualization of nullity by column.

First rows

raw_comment
normalize_comment

0 Giao hàng kh <e1>ng</e1> đ <e1>ng</e1> c <e1>n</e1> n ph <e1>e</e1> b <e1>inh</e1> h <e1>j</e1> <u>jjjjj</u> hhdjdjdjdjdjjfjjfnno	giao hàng k <e1>h</e1>
1 Chất lượng sản phẩm tạm được. Giao sai 1 món. Đóng gói chắc chắn	chất lượng :
2 Ko có lắc tay như hình	không có lắc
3 Giao hàng lâu. Bảo có lắc tay mà k thấy giao. Dây da thì bị bong tróc. Dù biết là tiền nào của đấy nhưng cảm thấy không hài lòng.	giao hàng lâ
4 Minh mua 2 cái, một dùng ok. Một cái không chạy được 😢	mua cái một
5 Đường may ko cẩn thận, đường ly lê ra phải thêm 1 đường ngang để tránh bung chỉ. Áo mới mà các đường ly đã bung chỉ rồi, mặc vài lần chắc sẽ bung hết	đường may
6 Áo y hình, chất vải cũng đẹp. Form lên ôm vừa vắn dáng. Nhưng mà ko hiểu sao mình mặc thử vô ngứa khắp người luôn. Ko biết lí do tại sao lại bị nữa.	áo y hình ch
7 Giao hàng nhanh\nmỗi tội áo hơi nhau nhau nhau nhau hàng giật mình luôn\nĐề nghị lần sau đóng gói đẹp hơn	giao hàng n
8 Màu già hơn hình mẫu mặc	màu già hơ
9 vải nhăn, đường may k được đẹp, haizzz , thấy hơi bị tiếc tiền khi mua	vải nhăn đư

Last rows

raw_comment

14196	Ôn, bình thường. Không tệ cũng k quá tốt. Chuẩn bị hàng nhanh, đóng gói tốt
14197	Hàng vải rất đẹp rất đáng mua thời gian giao hành chuẩn bị rất nhanh rất hài lòng
14198	Quần đẹp, Tất đẹp. Hộp sang dã man giá cả hợp lý. Đặt hôm trước hôm sau nhận được luôn
14199	Áo thì là vải mịn đẹp. Quần thì là quần kiểu vải mỏng hơn. Nhưng nói chung giá này được bộ là đẹp rồi nha
14200	Mua về cho chồng mà để con em nó mặc thử trước 😊. Hàng y hình, phù hợp với giá tiền, ship cực nhanh, mua hôm trc hôm sau có luôn
14201	san pham ok giao nhanh sđt ứng hộ lần sau xxxxxxxxxxxxxxxxxxxxxxxxx
14202	Chất vải đẹp đúng như shop giới thiệu....ung lắm sđt ứng hộ shop nữa
14203	Chat luong san pham tuyet voi dong goi san pham rat dep Chat luong san pham tuyet voi dong goi san pham rat dep Chat luong san pham tuyet voi dong goi san pham rat dep
14204	Sản phẩm giống hình, dễ thương, mình không rõ kích thước nên chọn loại hơi to cho cặp tóc nửa đầu
14205	Nhin giống hình nhưng Chất bóp ko mịn, ko biết thọ dc 12 tháng như shop quảng cáo ko. Thời gian TL rồi nhận xét thêm.

- **Nhận xét:**

- **Pandas-profiling** cung cấp cho ta tất cả các số liệu về thống kê mô tả cơ bản mà một dataset bất kì nào cũng cần, dĩ nhiên nó còn làm tốt hơn chúng ta nhiều.
- Bước tiếp theo, ta sẽ tìm hiểu về mối quan hệ giữa biến phân loại **label** với lần lượt hai biến liên tục là **length** và **no_words**.
- Tiếp theo, ta sẽ kiểm chứng mối quan hệ giữa biến phân loại **label** và biến liên tục **length** và **no_words** với **null hypothesis** là giữa các class trong label chịu tác động từ **length** hoặc **no_words** là như sau. Ta sẽ áp dụng **ANOVA one-way** để kiểm tra điều này.

```
1 Detective.onewayANOVA(reviews, 'length')
```

	df	sum_sq	mean_sq	F	PR(>F)
label	1.0	2.628255e+05	262825.483317	67.673463	2.095801e-16
Residual	14204.0	5.516451e+07	3883.730374	NaN	NaN

- **Nhận xét:**

- Vì giá trị **p-value** ($PR(>F)$) nhỏ hơn 5% tức ta phải bác bỏ **null hypothesis**.
- Tuy nhiên, ta chỉ biết là **label** chịu tác động không đều giữa các class bởi **length**, nhưng ta không biết được class nào sẽ chịu tác động lớn hơn hoặc kém hơn so với các class còn lại, bước tiếp theo ta sẽ áp dụng **post-hoc testing** để kiểm chứng điều này.
- Ta sẽ áp dụng **t-test** để kiểm chứng điều này, tuy nhiên do số class trong **label** chỉ có hai, làm điều này là hơi thừa thãi, và các sơ đồ boxplot ở phía trên cũng đã cho ta thấy được giữa các class của **label** là có sự khác biệt đáng kể về giá trị trung bình, tức có class chịu tác động lớn hơn bởi **length**, người ta thường áp dụng **post-hoc testing** khi có từ 3 class trong categorical variable trở lên, nhưng ta cứ áp dụng thử xem mọi thứ phía trước ta làm cho đến bây giờ bao gồm trực quan hóa, thống kê mô tả có đúng trên **hypothesis testing** không.

```
1 Detective.onewayANOVA(reviews, 'length', True)
```

	coef	std err	t	P> t	Conf. Int. Low	Conf. Int. Upp.	pvalue-hs	reject-hs
Positive-Negative	-8.602562	1.045728	-8.226388	2.095801e-16	-10.652326	-6.552799	2.095801e-16	True

- **Nhận xét:**

- Rõ ràng, ở trường **reject-hs** với giá trị **True** tức là có sự khác biệt về giá trị trung bình đáng kể bởi **length** lên các class trong **label**.
- Tiếp theo, ta cũng có thể áp dụng tương tự cho cặp feature **label** và **no_words** mặc dù ta đã biết trước là cũng cho ra kết luận tương tự như cặp **label** và **length** thôi. Vì ở bước trực quan ta biết được rằng càng nhiều từ thì bình luận càng dài.

```
1 Detective.onewayANOVA(reviews, 'no_words')
```

	df	sum_sq	mean_sq	F	PR(>F)
label	1.0	7.382661e+03	7382.660918	58.577054	2.080427e-14
Residual	14204.0	1.790177e+06	126.033326	Nan	Nan

```
1 Detective.onewayANOVA(reviews, 'no_words', True)
```

	coef	std err	t	P> t	Conf. Int. Low	Conf. Int. Upp.	pvalue-hs	reject-hs
Positive-Negative	-1.441785	0.188381	-7.653565	2.080427e-14	-1.811036	-1.072534	2.080427e-14	True

- **Nhận xét:** Như nhận xét ở bên trên, mọi kết luận của cặp **label** và **no_words** đều tương tự như cặp **label** và **length**.

- **Kết luận:**

- Mặc dù những gì ta làm trên các feature **length** và **no_words** mang đến cho ta có cái nhìn đa chiều và nhiều điểm thú vị trong dataset, nhưng thực chất chúng chỉ dùng lại ở mức tìm hiểu, ta khó có thể (thậm chí là không) sử dụng chúng trong việc training model, vì bài toán của chúng ta là sentiment analysis, nó phu thuộc vào ngữ nghĩa của từ và cấu trúc ngữ pháp.
- Dù rằng ta nói là khó có thể sử dụng hai feature là **length** và **no_words** vào quá trình training model, nhưng những bước ta làm phía trên cũng góp phần giúp ta đánh giá được dữ liệu của chúng ta đã sạch chua, còn outlier hoặc noise sample không, điển hình là có bình luận chỉ dài 1 kí tự và trong tiếng việt ta cũng chưa biết có từ nào là 1 kí tự cả - rất có thể đây là từ viết tắt mà ta chưa biết, ta có thể bổ sung nó vào bước **data pre-processing**.
- **Công việc tiếp theo:** Ta đã làm những việc cơ bản ban đầu cần có. Phần sau ta sẽ tập trung vào việc tìm hiểu chi tiết hơn về các bình luận bằng cách trực quan các từ lên, đếm tần suất xuất hiện của từ, xem các từ nào giao nhau giữa nhóm positive và negative, tìm hiểu xem có cần bổ sung thêm cho stopword hay tập các từ viết tắt hay không,....

3.2 Sử dụng nhận xét, code/thuật toán để thể hiện trực quan các mối quan hệ giữa các trường dữ liệu

- Đọc dữ liệu từ project 1 lên và sắp xếp lại dữ liệu.

```

1 reviews = pd.read_csv("./data/normalize_reviews.csv").fillna("")
2 reviews = reviews[['raw_comment', 'normalize_comment', 'emoji', 'label']]
3
4 reviews.head()

```

	raw_comment	normalize_comment	emoji	label
0	Giao hàng kh đúng cắn phê bình haaaaahhd...	giao hàng không đúng cắn phê bình	0	
1	Chất lượng sản phẩm tạm được. Giao...	chất lượng sản phẩm tạm được giao ...	0	
2	Ko có lắc tay như hình	không có lắc tay như hình	0	
3	Giao hàng lâu. Bảo có lắc tay mà k thâ...	giao hàng lâu bảo có lắc tay mà không ...	0	
4	Mình mua 2 cái, một dùng ok. Một cái k...	mua cái một dùng ok một cái không chạ...	😊	0

- Ở phần này, chúng ta sẽ tập trung đi sâu vào mặt khám phá các từ ngữ, bây giờ chúng ta cùng xem bức tranh tổng thể trước. Cụ thể là trong dataset của chúng ta đâu là các từ vựng phổ biến. Chúng ta thực hiện điều này bằng một Python's package là **WordCloud**:

pip install wordcloud

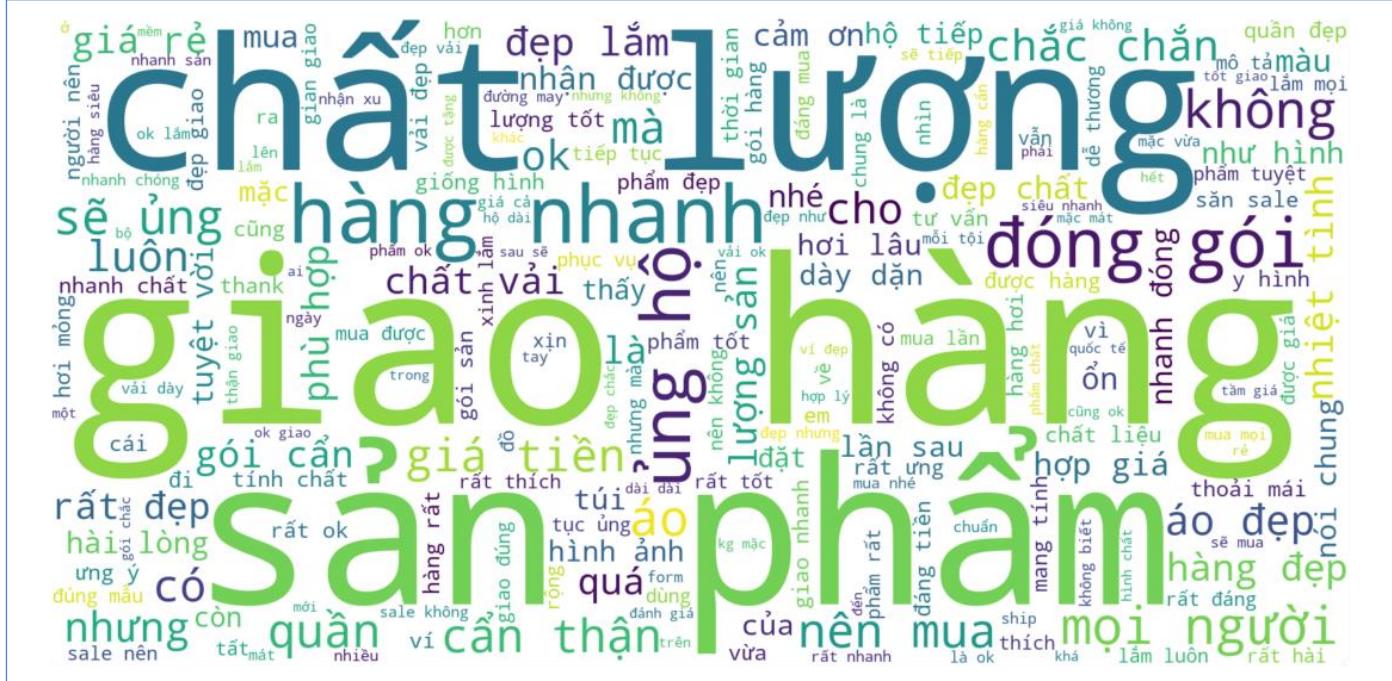


- **Nhận xét:**

- Nhìn chung, ở bước tiền xử lí dữ liệu ta đã xử lí khá tốt.
- WordCloud tiện lợi trong việc cho ta cái nhìn tổng quát trên dữ liệu dạng text, cho ta thấy được các từ nào chồng chéo lên nhau trên các class.

- Ngoài ra, nó giúp ta liệt kê được rõ ràng các từ ghép quan trọng, các từ ghép sẽ có tần số xuất hiện gần nhau nên sẽ có font-size giống nhau - điển hình ở đây ta thấy được các từ như: sản phẩm, giao hàng, chất lượng,... Tuy nhiên chúng ta thấy các từ này không đem lại nhiều giá trị cho lầm, vì đây là trang bán hàng, comment nào cũng sẽ có những từ này, tức chúng xuất hiện nhiều nhưng không quan trọng, cái ta quan tâm là các từ thể hiện cảm xúc của khách hàng. Ta có thể xem xét bỏ sang chúng vào stop word sau này.
- Tiếp theo, ta sẽ xem xét các từ nào hay xuất hiện trên từng specific class theo feature **label**.

- Xem theo label nhóm positive.



- Xem theo label nhóm negative.



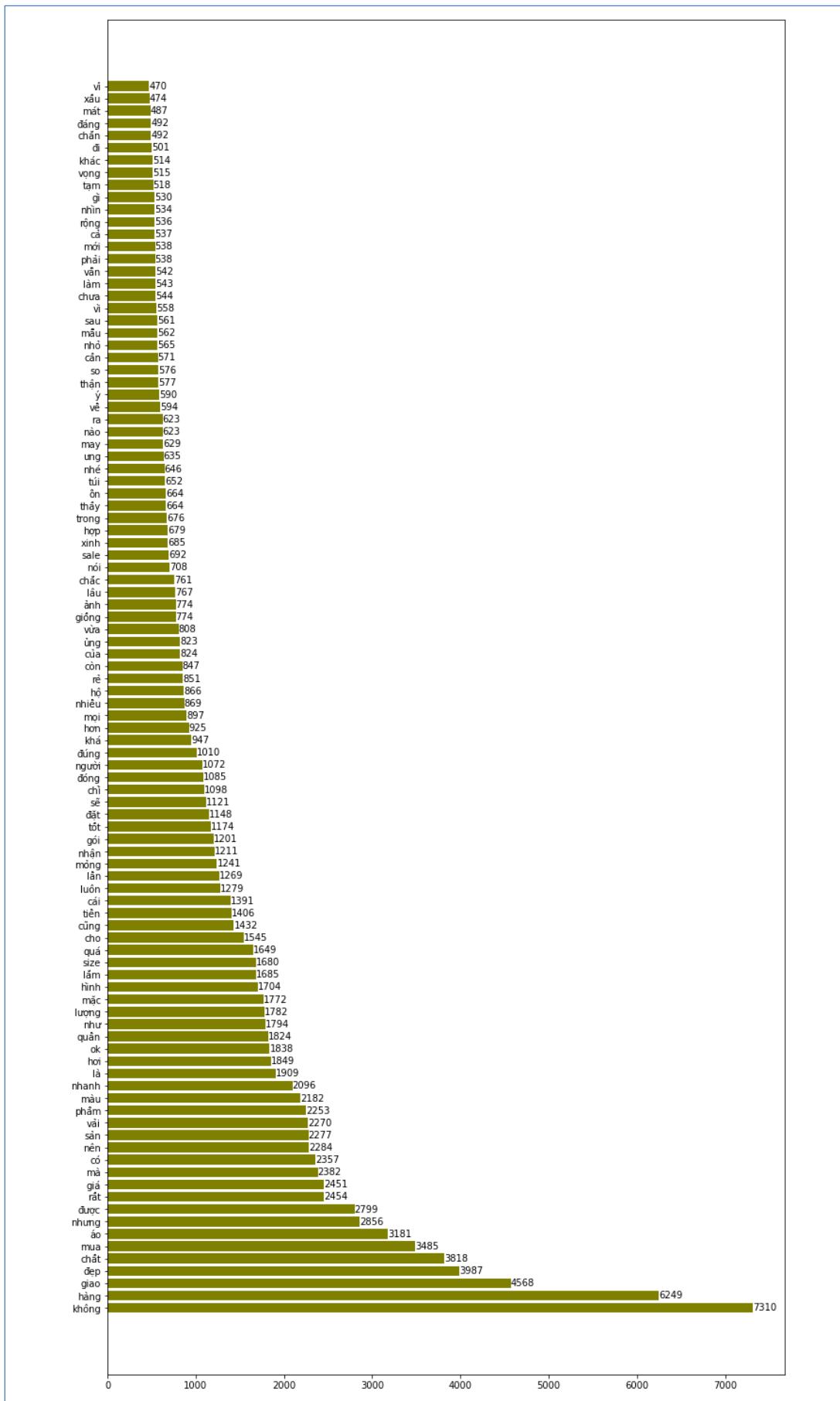
- **Nhận xét:**

- Các cụm từ như: "**sản phẩm**", "**giao hàng**", "**chất lượng**" xuất hiện trên cả hai wordcloud của hai class. Nếu chúng ta sử dụng các model machine learning đơn giản, thì các từ này có thể gây ra sự bối rối cho model trong việc quyết định đâu là class tốt nhất. Giải pháp là chúng ta có thể xem xét và đưa chúng vào stopwords.
 - WordCloud cung cấp cho ta cái nhìn nhanh và bức tranh tổng thể của text-data, và đây cũng là các hạn chế khi nó chỉ đơn giản zoom in từ **A** và **XONG**, ta không có được một cái nhìn rõ ràng là từ **A** so với từ **B** nhiều hơn hay ít hơn bao nhiêu lần.
- Để thấy được chi tiết sự khác nhau giữa về tần số mà các từ trong từng class xuất hiện. Một trong những cách khi gặp vấn đề này đôi với chúng ta thì ta sẽ áp dụng lần lượt các phương pháp là **Bag of words** rồi sau đó là **TF-IDF**, sau đó ta chia thành các đoạn [m, n] và thống kê frequency xem các từ nào mà nằm trong các đoạn này.

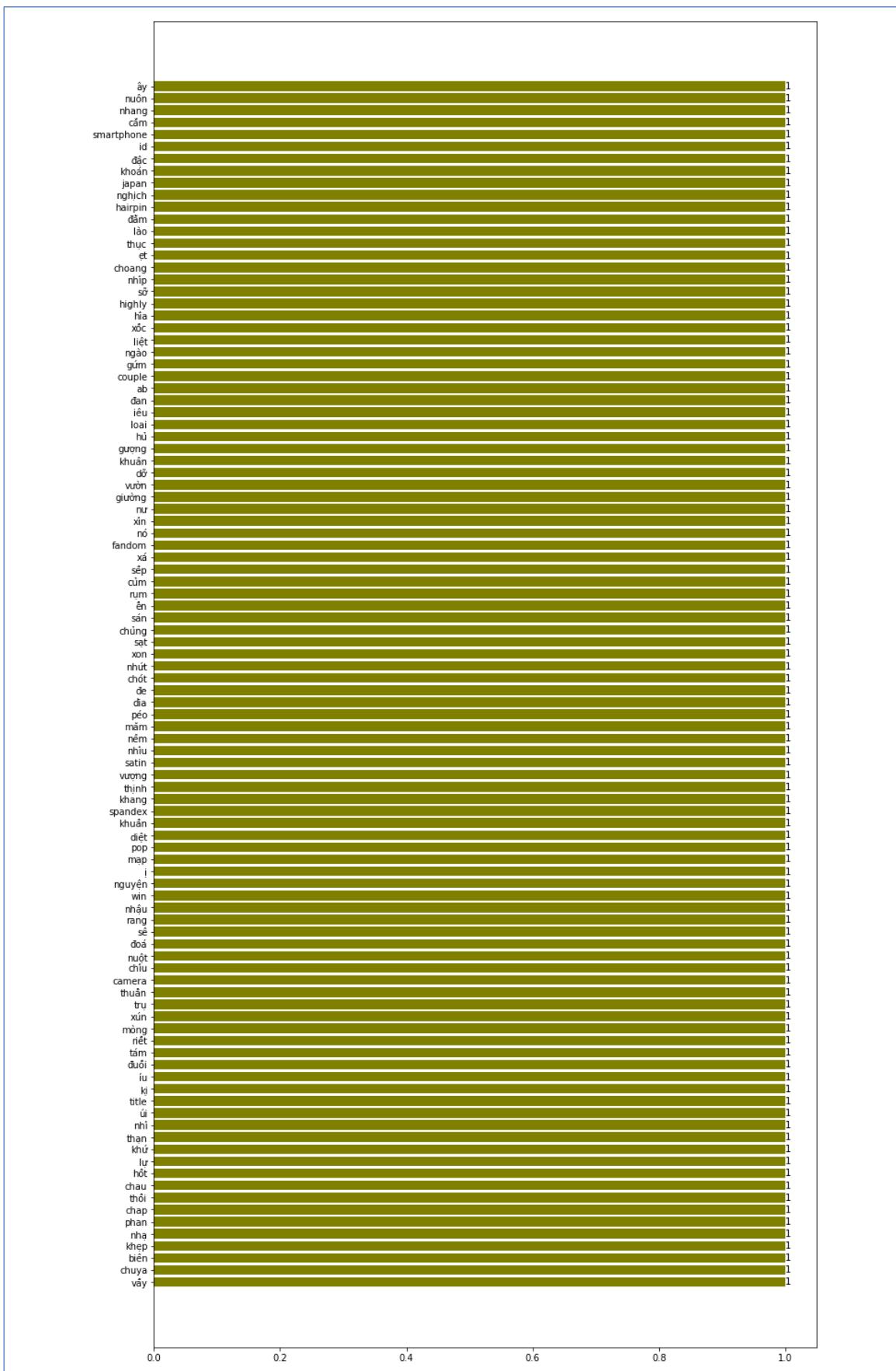
```
1 word_freq = Detective.createBagOfWordsFrequency(reviews['normalize_comment'])
2 print(len(word_freq))
```

3867

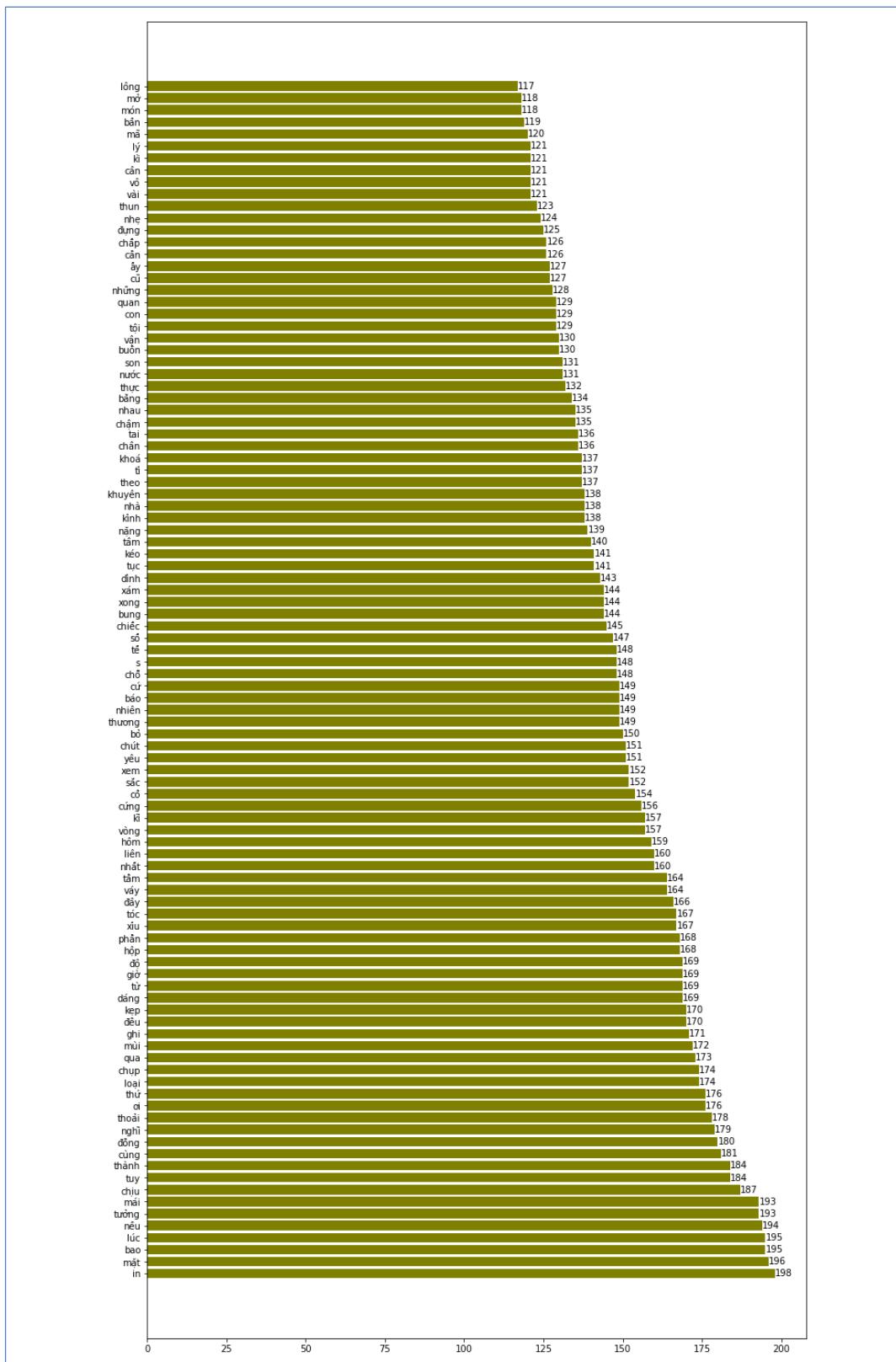
- **Nhận xét:** Có tổng cộng hơn 3800 unique word xuất hiện trong dataset của chúng ta.

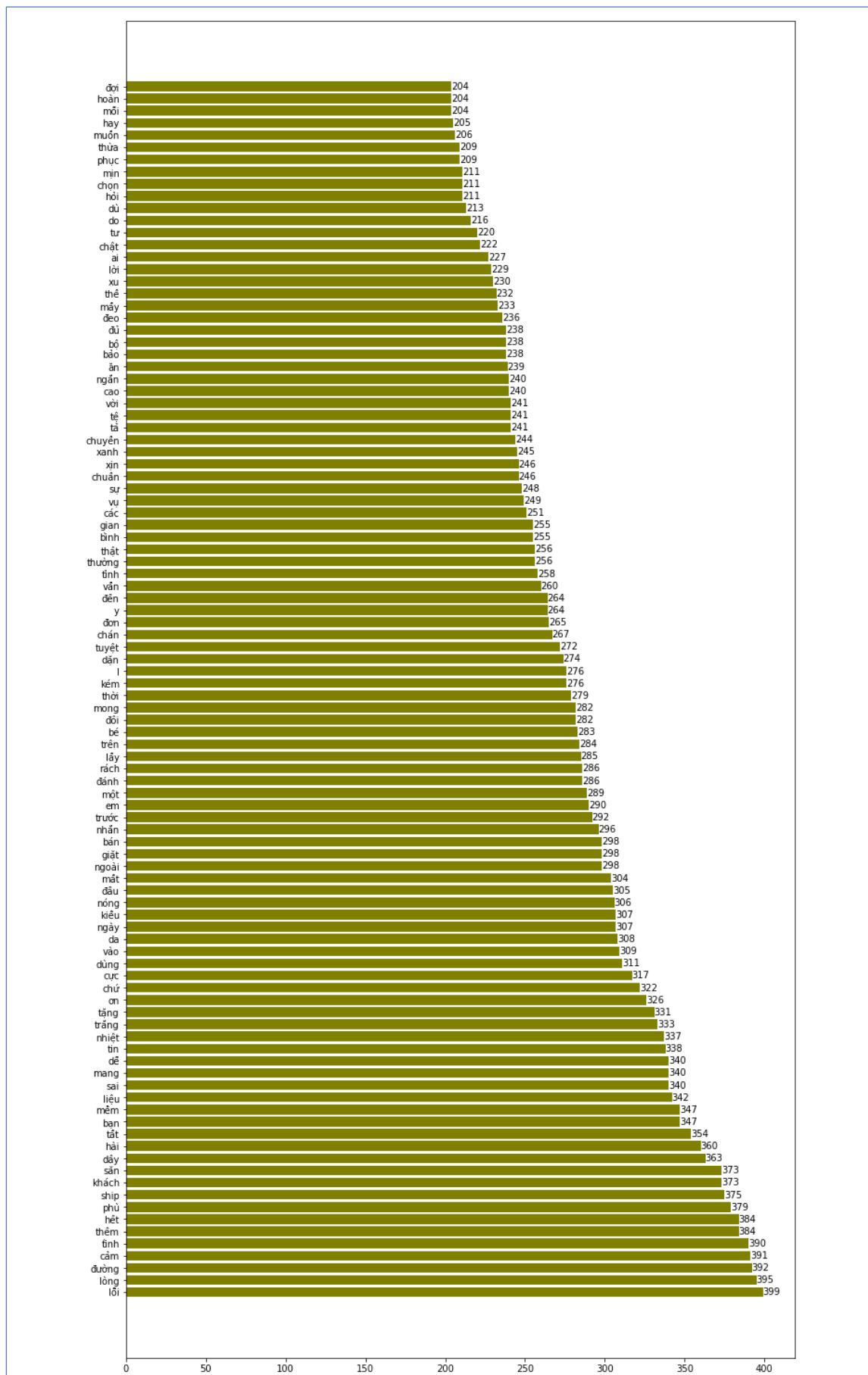


- **Nhận xét:** Trên đây là 100 từ có tần số xuất hiện nhiều nhất.



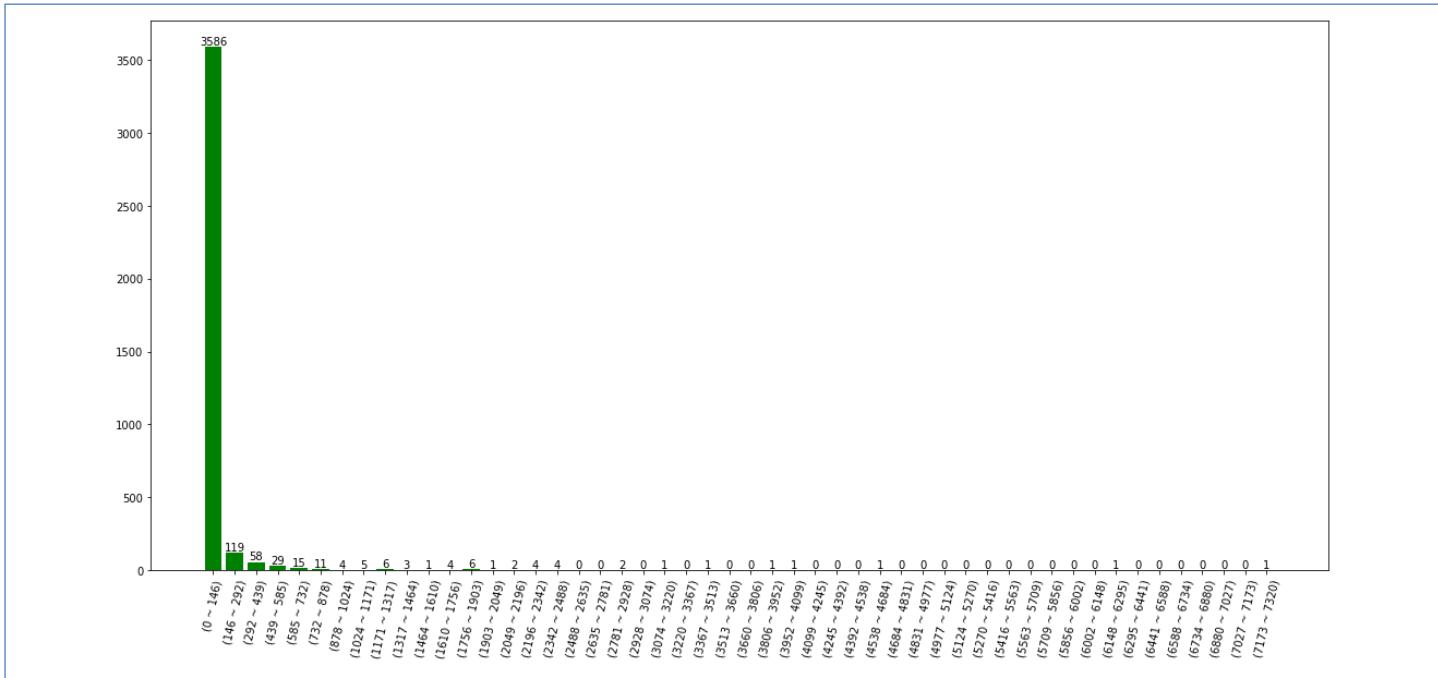
- **Nhận xét:** Đây là 100 từ có tần số xuất hiện ít nhất trong dataset. Chúng không đem lại tri thức cho model nhiều, chúng ta có thể cân nhắc trong việc xem xét và loại bỏ chúng khỏi các sentence để giảm tải input vector giúp model build nhanh hơn và chính xác hơn sau này.
- Ta cũng nên kiểm tra xem các từ có tần số xuất hiện sao cho $[m, n]$ nào đó.



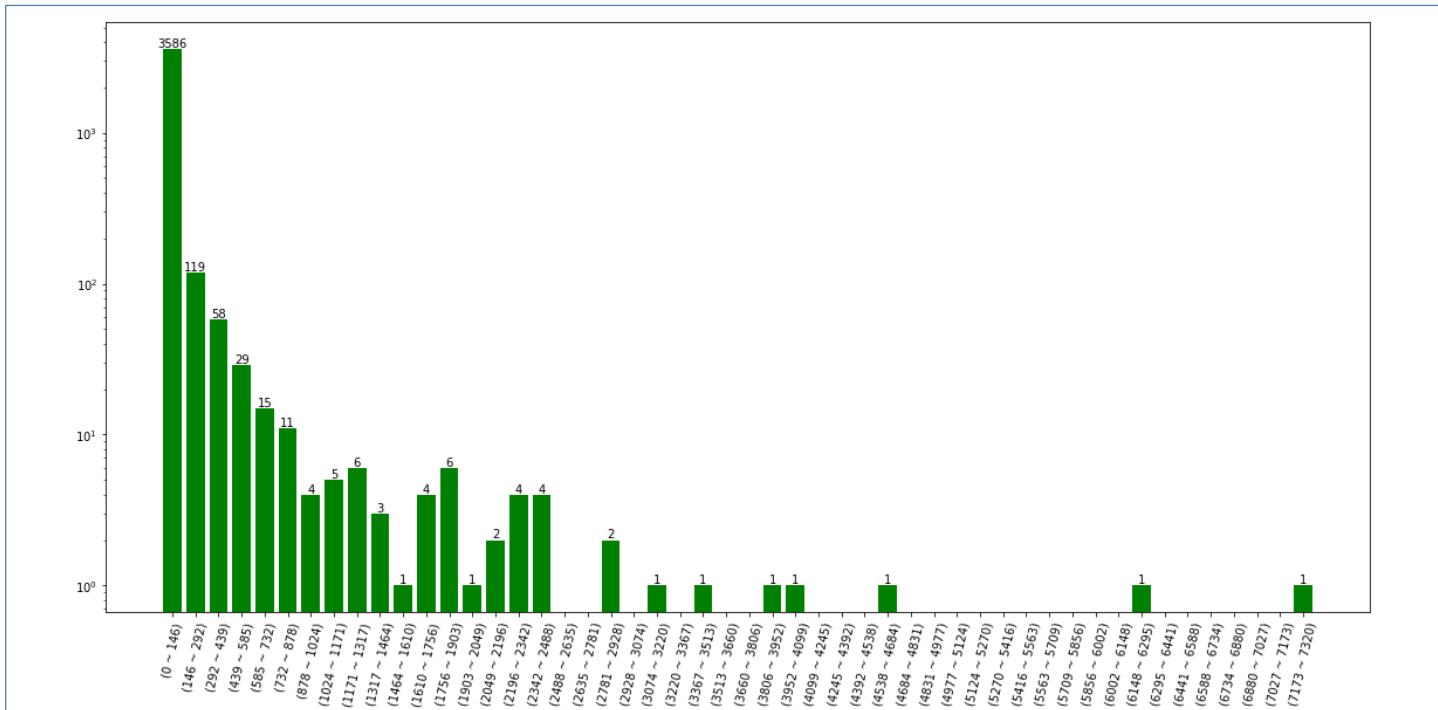


- **Nhận xét:**

- Phía trên ta đã trực quan hóa các từ có tần số xuất hiện trong các khoảng khác nhau.
- Ta hoàn toàn có thể trực quan toàn bộ chúng lên một barplot và nhóm chúng lại thành các nhóm frequent range để xem có bao nhiêu từ có tần số như thế trong từng nhóm.
- Tiếp theo đây, ta sẽ biểu diễn tần số xuất hiện của các từ theo các nhóm frequent range.



- **Nhận xét:** Nhìn vào biểu đồ **barplot** này, ta khó thấy được cách mà các từ được phân bổ như thế nào do cột đầu tiên đã chiếm đa số (trong trường hợp này ta dễ dàng thấy được do ta đã thêm label vào cho từng cột), ta có một giải pháp là sử dụng log-scale.



- **Nhận xét:**

- Sau khi sử dụng hàm log để scale trên trục y, ta dễ dàng thấy hơn là đa phần các từ có tần số xuất hiện trong khoảng từ [0, 146] là chủ yếu - có đến hơn 3500 từ nằm trong khoảng này, đây chính là một minh chứng cho một câu nói vui là "**tuy tôi yêu nhưng anh em tôi đồng**".
- Ngược lại, tuy có những từ hầu như trong comment nào cũng xuất hiện nhưng chúng chỉ có một mình, về sau nếu ta có áp dụng N-Grams thì chúng là những thành phần lẻ loi không ai đi cùng.
- Ta cũng có thể liệt kê ra các từ mà nằm trên cả hai class negative và positive, việc này giúp ta phát hiện ra các từ nào có nguy cơ chồng chéo lên nhau và gây ra cho model sự bối rối trong việc đưa ra quyết định.
- Chiến lược của ta sẽ như sau:
 - Trước tiên ta sẽ tạo ra một dictionary mà key là word, value là một mảng 2 chiều với phần tử đầu tiên là số lần từ đó xuất hiện ở class negative và phần tử thứ hai là số lần từ đó xuất hiện ở class positive.
 - Tiếp theo, ta duyệt lại dictionary này, chỉ lấy các từ mà có toàn bộ các phần tử trong mảng value đều khác 0, tức từ này xuất hiện ở cả hai class.
 - Cuối cùng ta sắp xếp lại chúng và trực quan hóa.
- Và phần này, chúng ta cũng cần một package nữa là **plotly**:

`pip install plotly`

- Kiểm tra xem negative class và positive class có unique words không?

```

2 print(f"Amount of negative words: {len(neg_words)}")
3 print(f"Amount of positive words: {len(pos_words)}")

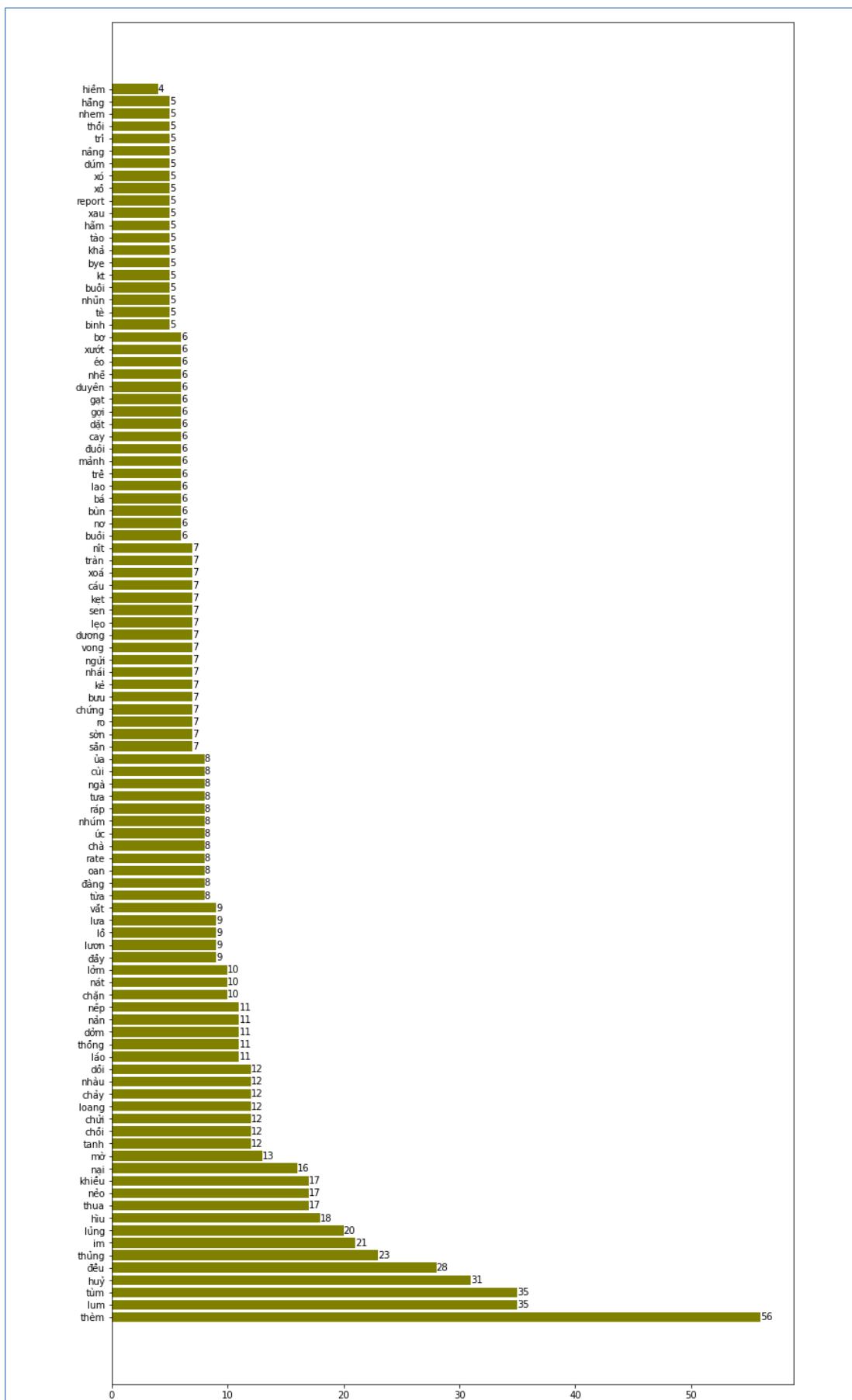
```

```

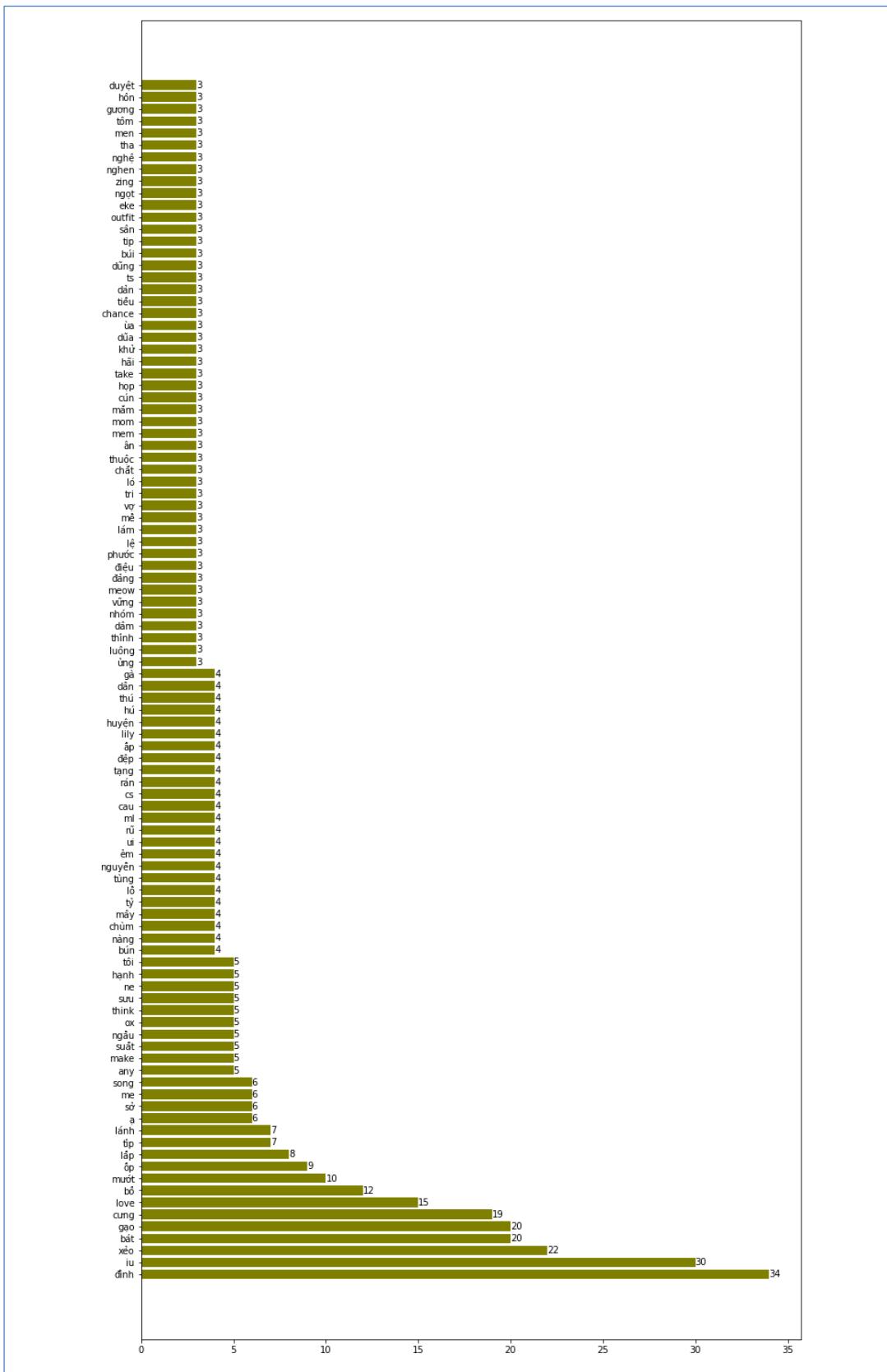
Amount of negative words: 1046
Amount of positive words: 789

```

- **Nhận xét:** Vậy cả hai class negative và positive đều có các unique word, ta sẽ thử trực quan hóa chúng bằng bar plot.
- Negative words.



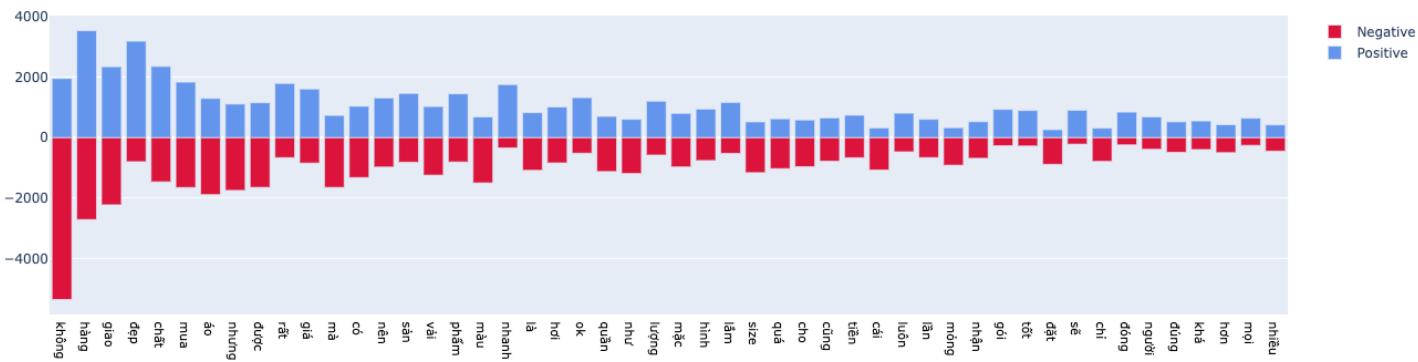
- Positive words.



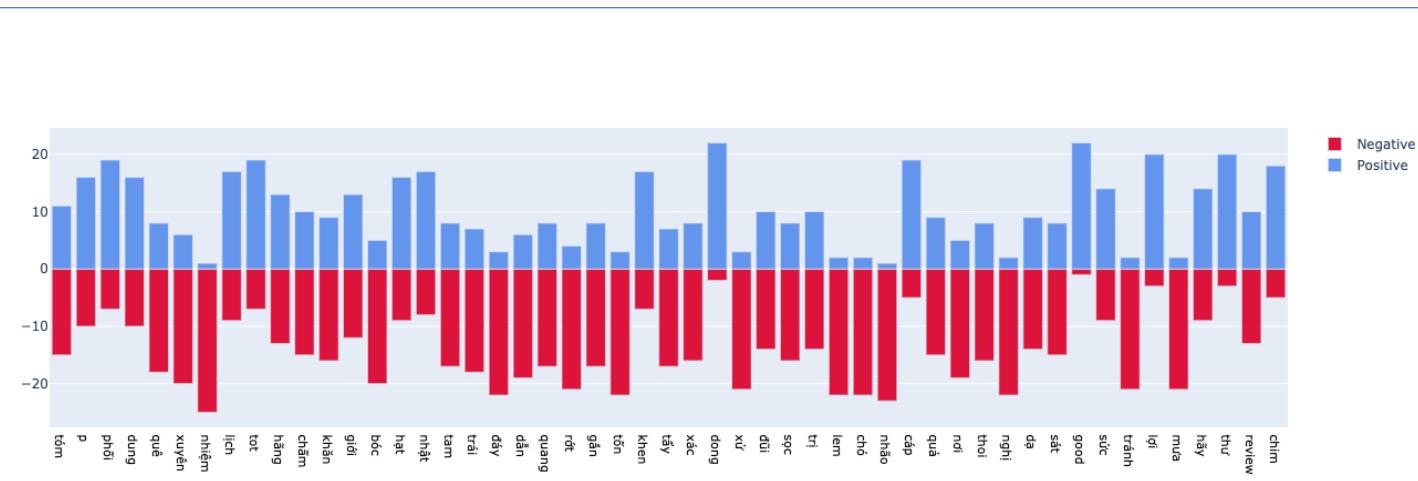
- **Nhận xét:** Các unique words này thể hiện đúng tính chất của một comment tương ứng với class đó.

- Bây giờ ta sẽ xem xét 50 từ đầu tiên chòng chéo trên cả hai class.

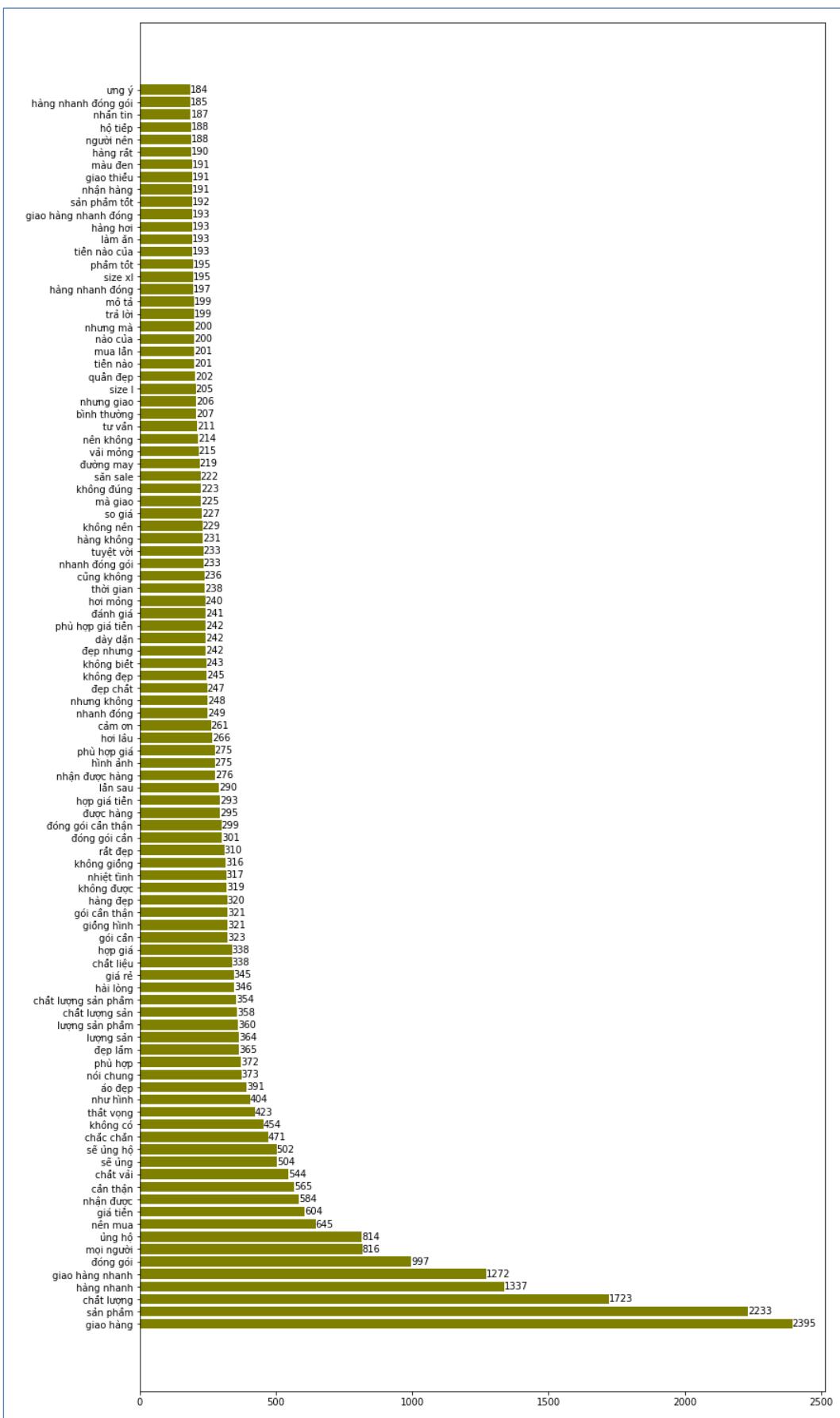
```
1 print(f"Amount of overlap words: {len(inter_words)}")  
Amount of overlap words: 2032
```



- 50 từ chồng chéo thuộc đoạn [800,850] trên cả hai class.



- **Nhận xét:** Một điều đáng mừng là các overlap word này đa phần nghiêng hัก về một bên, điều này giúp cho model có khả năng tránh bị bối rối khi phải đưa ra quyết định khi gặp các từ này.
 - Ở phần gần cuối này, chúng ta sẽ tập trung vào việc tìm kiếm các từ ghép khả dĩ đang tồn tại trong dataset. Hay còn gọi là Ngrams.
 - Chúng ta có thể xem 100 từ ghép phổ biến và tần số xuất hiện của chúng, chúng ta cũng có thể xem 100 từ ghép ít phổ biến nhất nhưng điều này cá nhân ta thấy không cần thiết vì ta bảo đảm chúng toàn là có tần số là 1 thôi.



- **Nhận xét:**

- Ở đây, chúng ta thử tìm các từ ghép phổ biến với n-gram [2, 4].
 - Nhìn chung kết quả rất hợp lý, ta có thể thay các khoảng trắng trong các từ ghép này bằng underscore (_) và xem nó là một từ, điểm lợi là giúp cho model của ta thông tin hơn đồng thời cũng giảm tải đi dimension input đầu vào cũng như tránh sparse matrix quá lớn.
- Cho đến hiện tại, chúng ta đã cùng nhau phá tan tành dataset này, điều mà chúng ta muốn làm tiếp là sẽ biểu diễn các từ lên một không gian 2 chiều hoặc thậm chí là 3 chiều.
 - Điều này theo ta là rất hữu ích, vì giống như các unsupervised model, nhiều khi chúng ta cũng cần biết cách mà các observe phân phối như thế nào.
 - Nhưng trước tiên, ta cần một số thiết lập **ban đầu** như sau:

- Chúng ta cần download vài file cấu hình sẵn:

```
pip install fasttext
```

- Thực chất, hai gói mà ta vừa download trên là chứa tập các từ tiếng anh và tiếng việt đã được vectorization thành các vector 300 dimensions, dùng lo lắng về chất lượng của hai tập words embedding này bởi vì cha đẻ của nó là Facebook.
- Thời gian download hai gói này cực kì lâu, khoảng 4GB cho một gói - và về giải nén ra mỗi file sẽ khoảng 7.3GB, khuyến khích dùng nén download theo cách trên mà nên lên trang chủ của họ và download trực tiếp bằng Internet download manager hoặc Xtreme download manager sẽ nhanh hơn, khoảng 15 phút cho một gói - đây là trang chủ của họ: (<https://fasttext.cc/docs/en/crawl-vectors.html>)

```

1 import fasttext.util
2 import fasttext
3
4 fasttext.util.download_model('en', if_exists='ignore') # english
5 fasttext.util.download_model('vi', if_exists='ignore') # vietnamese
'cc.vi.300.bin'
```

- **Lưu ý:**

- Chỉ nên sài dòng này nếu máy tính của ta là từ 16GB RAM - Core I7 gen 7 dòng xung cao trở lên. Và nên dùng hệ điều hành Linux vì nó có RAM ảo hỗ trợ để máy chúng ta chỉ lag thôi chứ không đứng.

```
ft_vi = fasttext.load_model('cc.vi.300.bin')
```

```
ft_en = fasttext.load_model('cc.en.300.bin')
```

- Còn đây là chúng em đã giảm chiều từ một bộ data 7.2GB xuống còn 100 chiều, chúng ta có thể dùng dòng này, ta có thể giảm nhiều hơn nếu muốn. Và tốt nhất là ta cần tắt tất cả mọi chương trình đang chạy.

```
fasttext.util.reduce_model(ft_vi, 100)
```

```
ft_vi.save_model('cc.vi.100.bin')
```

```
fasttext.util.reduce_model(ft_en, 100)
```

```
ft_en.save_model('cc.en.100.bin')
```

- Bây giờ chúng ta sẽ load lại tập embed word 100 dims mà ta vừa tạo, nó sẽ ăn máy của chúng ta khoảng 3GB RAM.

```
1 ft_vi = fasttext.load_model('cc.vi.100.bin')
2 ft_en = fasttext.load_model('cc.en.100.bin')
```

Warning : `load_model` does not return WordVectorModel or SupervisedModel any more, but a `FastText` object which is very similar.

Warning : `load_model` does not return WordVectorModel or SupervisedModel any more, but a `FastText` object which is very similar.

- Dưới đây là vector 100 dims của từ "**xin chào**" và vài từ khác.
- Thực chất fasttext là một công cụ hỗ trợ cực kì tốt mà nhóm em biết đến nó thông qua một bài tweet trên Twitter. Hãy tin tưởng mà sử dụng nó, nó sẽ giúp cho cái bài toán NLP của chúng ta "**đỡ khổ**" hơn nhiều.

```
1 ft_vi.get_word_vector('xin chào')

array([-0.005858 ,  0.00591698,  0.05544396,  0.01845348,  0.03497462,
       -0.01731673, -0.0186325 , -0.05789406, -0.01456467,  0.05405328,
       0.0162955 ,  0.02371605, -0.02689386,  0.0168477 , -0.02608746,
      -0.00542725, -0.01952248,  0.02309453,  0.00077463, -0.01623438,
      -0.02708885, -0.02166499,  0.01338364, -0.00763521, -0.01389443,
      -0.00743966, -0.04979044, -0.02639596,  0.01744116, -0.02117856,
      0.00118052,  0.00367191, -0.00203112,  0.01383345,  0.01485165,
      -0.00397428,  0.02199381,  0.02537224, -0.00114648, -0.01440865,
      -0.01821862,  0.01354136,  0.00441786, -0.01989009,  0.00784927,
      -0.00614514,  0.01168528,  0.02575686,  0.01241738, -0.00843297,
      0.02774723,  0.00453759,  0.00838433,  0.01327751, -0.01783348,
      0.01831022, -0.01594335, -0.00435727,  0.00035578, -0.07555512,
      0.01310887, -0.04912862,  0.01182956,  0.03159093,  0.04497012,
      0.02143607,  0.00049193, -0.00196764,  0.00246742,  0.01363291,
     -0.02314509, -0.00219922, -0.02380401, -0.01026245, -0.01819856,
      0.0151787 ,  0.01194715, -0.00761893, -0.01721501,  0.01119621,
      0.02290164, -0.01710619, -0.02078362,  0.0228194 ,  0.00750548,
      0.00201767,  0.06676898, -0.00448752,  0.01720995,  0.00851261,
     -0.02720573,  0.01439547,  0.02121913, -0.01952045, -0.01194024,
     -0.01461318,  0.00467993, -0.00556966, -0.01434838, -0.0315196 ],  
dtype=float32)
```

```

1 ft_vi.get_word_vector("shipper")

array([-0.04482329,  0.1306968 ,  0.00602638,  0.08205491, -0.09409259,
       -0.00268238, -0.00441986, -0.14670041, -0.07580385,  0.11864945,
       -0.07212971, -0.04020993,  0.06071544, -0.05108281, -0.04871833,
      -0.13044995,  0.05104804, -0.0647935 ,  0.06504327,  0.02786569,
       0.02255378,  0.0658379 ,  0.00524131, -0.01714174,  0.02031365,
      0.04050732,  0.03624526, -0.00705281, -0.00607613, -0.04202025,
      0.02640752,  0.02390548,  0.03726061,  0.0165013 ,  0.08770727,
     -0.11049094,  0.03829083, -0.03813533, -0.06787217,  0.11119355,
      0.08849208,  0.02690171,  0.00534289, -0.06304946,  0.07952289,
      0.16372508,  0.02963781, -0.05313126, -0.05538091,  0.0243446 ,
      0.17476527,  0.05392487, -0.08586453, -0.03984641, -0.06173296,
      0.04835213, -0.04918601,  0.12133241,  0.01697406, -0.08753216,
     -0.05530317, -0.03141787, -0.07614072,  0.11230698, -0.01196967,
     -0.02775046, -0.0485909 , -0.08594808,  0.03448322,  0.07165931,
     -0.0421441 ,  0.02105435,  0.04268556,  0.05363652,  0.04589817,
      0.02782094,  0.02825668,  0.03267308, -0.0577312 ,  0.0668293 ,
      0.03514689,  0.05725601, -0.11705823, -0.0206674 ,  0.05662723,
     -0.05252227,  0.04314893, -0.02537445,  0.04091778, -0.01134288,
     -0.04383793,  0.03717593,  0.01454818,  0.01900238, -0.09251589,
     -0.04429725, -0.06188246, -0.08065203,  0.00760574, -0.0462208 ],
      dtype=float32)

```

```

1 ft_en.get_word_vector("shipper")

array([-2.99756192e-02, -5.92948869e-02, -1.14630638e-02, -4.70631421e-02,
       3.24248299e-02, -8.74208435e-02, -1.47009909e-01,  5.99884521e-03,
      3.20037752e-02,  2.02425458e-02,  2.08814330e-02, -5.86299365e-03,
     -2.06765789e-03, -4.06624787e-02, -2.24276762e-02, -1.59674391e-01,
     -7.36591890e-02, -4.33965698e-02, -1.39081981e-02,  1.87481493e-02,
      1.12330727e-01, -4.94616888e-02,  2.05947123e-02, -7.88073093e-02,
     -3.01143285e-02,  1.31405646e-03,  2.41592713e-02,  6.47805110e-02,
      1.57513142e-01,  4.18538265e-02,  1.05798662e-01, -7.28724003e-02,
      4.21688966e-02, -9.84481424e-02, -4.85281572e-02, -1.47368968e-01,
     -2.36060610e-03,  8.55448395e-02, -4.39639166e-02,  8.25370289e-03,
     -9.71954912e-02, -8.00786354e-03,  2.19022390e-02, -1.62983984e-01,
      6.80357963e-03,  8.00233632e-02,  6.63501471e-02, -3.30431424e-02,
      6.36976063e-02,  7.55386278e-02,  2.70912703e-02,  4.32227217e-02,
      2.33277492e-02, -3.52903306e-02, -5.81459468e-03,  4.24507372e-02,
     -5.04243672e-02,  5.94339799e-03,  1.48756085e-02, -3.91519666e-02,
     -3.18258777e-02, -9.08403366e-04, -3.27009484e-02, -3.72941494e-02,
     -2.72052698e-02,  6.37116134e-02, -2.60582920e-02, -4.85311672e-02,
     -5.07016145e-02, -8.58475119e-02, -1.22737423e-01, -2.04580538e-05,
      1.10536246e-02, -5.68021238e-02,  3.07537168e-02,  1.44481538e-02,
      1.33990750e-01,  1.13357507e-01,  3.47727421e-03,  2.81960070e-02,
     -1.29164639e-03, -5.12360483e-02,  1.29533574e-01,  4.64480789e-03,
     -8.16457346e-02, -1.50127932e-02,  6.17302349e-03,  5.20901009e-02,
     -1.25290742e-02,  4.37334478e-02,  5.52784614e-02,  5.58346286e-02,
     -3.61467116e-02, -2.41361298e-02,  7.62666315e-02,  2.38128286e-02,
      5.36810681e-02,  5.88007644e-02, -6.12015501e-02, -1.09539870e-02],
      dtype=float32)

```

- **Nhận xét:**

- Mặc dù là tập dữ liệu được dành riêng cho từng ngôn ngữ nhưng khi ta thử kiểm tra với một vài từ tiếng anh thì **ft_vi** vẫn cho ra kết quả tuy nhiên lại khác đi phần nhiều so với **ft_en**. Điều này là dễ hiểu vì đây là dữ liệu **thật** được Facebook gom về từ các dịch vụ của họ, và nói về khoảng quảng cáo bán hàng thì Facebook hiểu chúng ta thế nào thì chúng ta cũng quá rõ rồi.
- Và việc embedded vector của cùng một từ tiếng anh nhưng trên **ft_vi** và **ft_en** khác nhau cũng là chuyện dễ hiểu.

- Tiếp theo chúng ta sẽ vectorization các word có trong dataset của chúng ta bằng **ft_vi**.

```

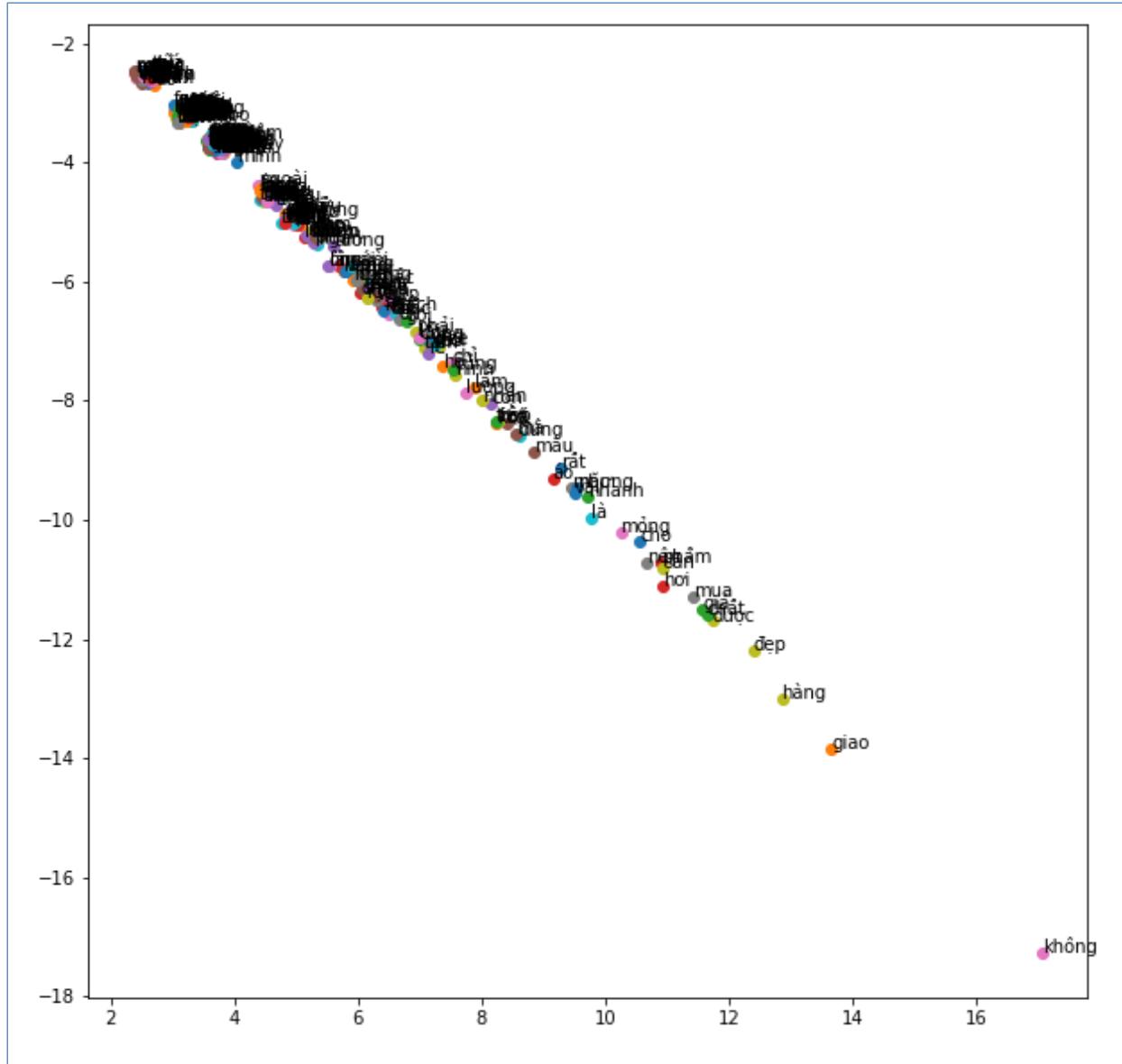
1 vec_words = Detective.ftVectorization(reviews, ft_vi)
2 vec_words

```

	word	freq	ft_vec	label
0	thèm	56	[0.12546667, 0.27063507, -0.3226054, 0.1076125...]	0
1	lum	35	[0.31741396, 0.60768604, -0.23061581, 0.342011...]	0
2	tùm	35	[0.34853685, 0.19606084, -0.22714649, 0.080849...]	0
3	huý	31	[0.24349461, 0.12490517, -0.20848149, -0.01994...]	0
4	dèu	28	[0.08494775, 0.045489177, -0.035257675, -0.022...]	0
...
3862	spam	2	[-0.06227298, 0.27240402, -0.012649586, 0.2511...]	2
3863	cấm	2	[0.036774382, 0.04028643, -0.031932976, 0.0316...]	2
3864	thường	2	[0.11008545, 0.026620911, -0.025946898, 0.0367...]	2
3865	hững	2	[0.04788875, -0.026248291, 0.025058057, 0.0359...]	2
3866	thăm	2	[0.13048768, -0.0068094754, -0.04752696, -0.02...]	2

- Tuy nhiên, chúng ta vẫn muốn xây dựng một model **word2vec** nhỏ nhở trên chính một phần nhỏ của dataset này, để ta có thể hiểu rõ hơn.
- Sẽ rất khó nếu ta tự build một **word2vec** model hoàn chỉnh giúp ta giải quyết toàn bộ dataset này vì nó yêu cầu nhiều về kiến thức đến phần cứng và tối ưu hóa.
- Vậy bằng cách nào mà ta tạo ra được chính các word vectors mà không dùng các resource bên ngoài. Nếu đã học qua học biết về Deep Neural Network, ta có 3 layer lần lượt là input layer, hidden layer và output layer. Và ta biết một DNN nó sẽ tự đào tạo các weight trong các hidden layer sao cho loss function thấp nhất có thể so với output layer. Và đây cũng chính là ý tưởng của các model như word2vec mà Glove, Gensim, Word2Vec của TensorFlow sử dụng và tùy biến. Ở đây chiến lược của ta sẽ như sau:
 - Trước tiên ta cần index hóa các unique word.
 - Tiếp theo, vì chúng em đang muốn so sánh với 100 dims word2vec model của **fasttext** nên chúng em sẽ cấu hình cho hidden layer của ta bao gồm 100 neurons.
 - Bước tạo ra output layer sẽ hơi phức tạp, trước tiên ta cần định nghĩa một **context window**, người ta định nghĩa nó là một số nguyên ≥ 2 , nó hoạt động như sau:
 - Giả sử ta có sentence: "**shop giao hàng siêu nhanh**", nó là một vector 5 chiều, vậy nếu đã biết về CNN, thì context window hoạt động y như vậy, ta sẽ trượt window này, khi tới một từ nhất định - ta gọi từ này là **focus word** và lấy 2 từ kế bên trái và 2 từ bên phải - các từ này được gọi là **context word**, và lúc này ta được các cặp 2.
 - Vậy giả sử với câu trên, focus word của ta là **hàng** thì 4 từ còn lại là các context word thì ta lần lượt đạt được các cặp từ như sau: (hang, shop), (hang, giao), (hang, sieu) và (hang, nhanh). Chúng ta tạm gọi đây là **word_list**.
 - Vậy bây giờ ta sẽ duyệt qua từng element trong **word_list**, với từng phần tử trong element ta lấy word encoding vector tương ứng. Lúc này các vector của phần tử đầu tiên trong từng element chính là input và của phần tử thứ hai chính là output.

- Model **word2vec** demo lấy ngẫu nhiên 50 comment từ dataset của chúng ta (**File:** 04.eda.ipynb).

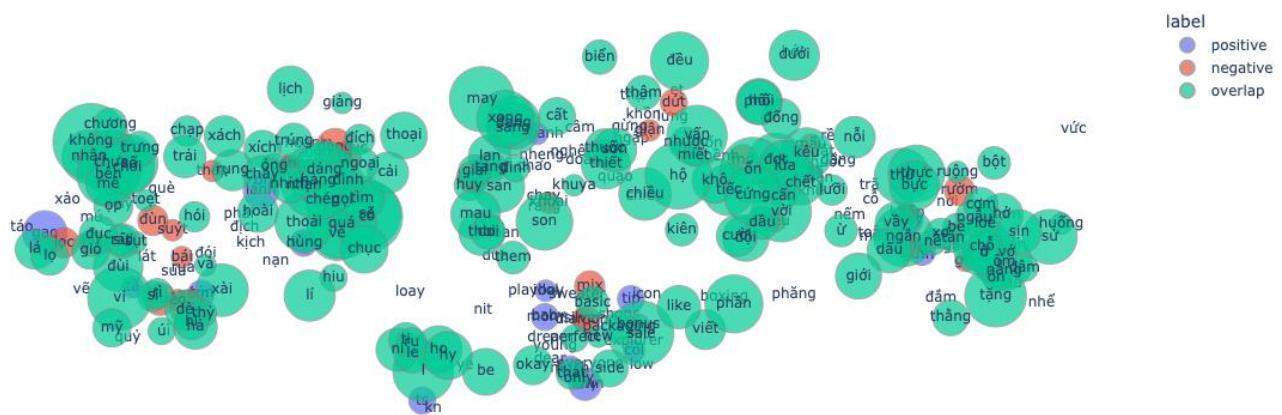


- Nhận xét:

- Nó hoạt động thật đấy nhưng cũng không tốt cho lắm, rõ ràng ta thấy được có vài từ nằm gần nhau tạo thành cụm - nếu suy ngẫm thì chúng cũng tạo nên những từ ghép có nghĩa. Nhưng ta cũng không thể quá mong đợi rằng nó tốt vì hidden layer của chúng ta chỉ có vỏn vẹn 2 neuron và nếu nhìn vào loss value ta thấy nó tăng dần đều.
 - Tuy nhiên ta cũng có thể khắc phục điều này bằng cách tìm ra một kiến trúc khác phù hợp cho mạng, ta có thể tăng số neuron lên vì 2 neuron là quá ít, sau đó ta có thể sử dụng PCA để chọn ra 2 thành phần quan trọng nhất để trực quan.
 - Đây chỉ là minh họa cho việc ta có thể tự build một **word2vec** nếu dataset của ta nhỏ. Nhưng tốt nhất ta nên sử dụng các gói công nghệ đã xây dựng sẵn.

- Nay giờ, ta sẽ trực quan hóa vectorization dựa vào fasttext lên một không gian 2 chiều. Chúng ta sẽ sử dụng công cụ PCA của object TSNE sklearn để giảm chiều **ft_vec** thành 2 component. Sau đó trực quan hóa chúng để xem với model word2vec của Facebook có tốt trên dataset của chúng ta không.

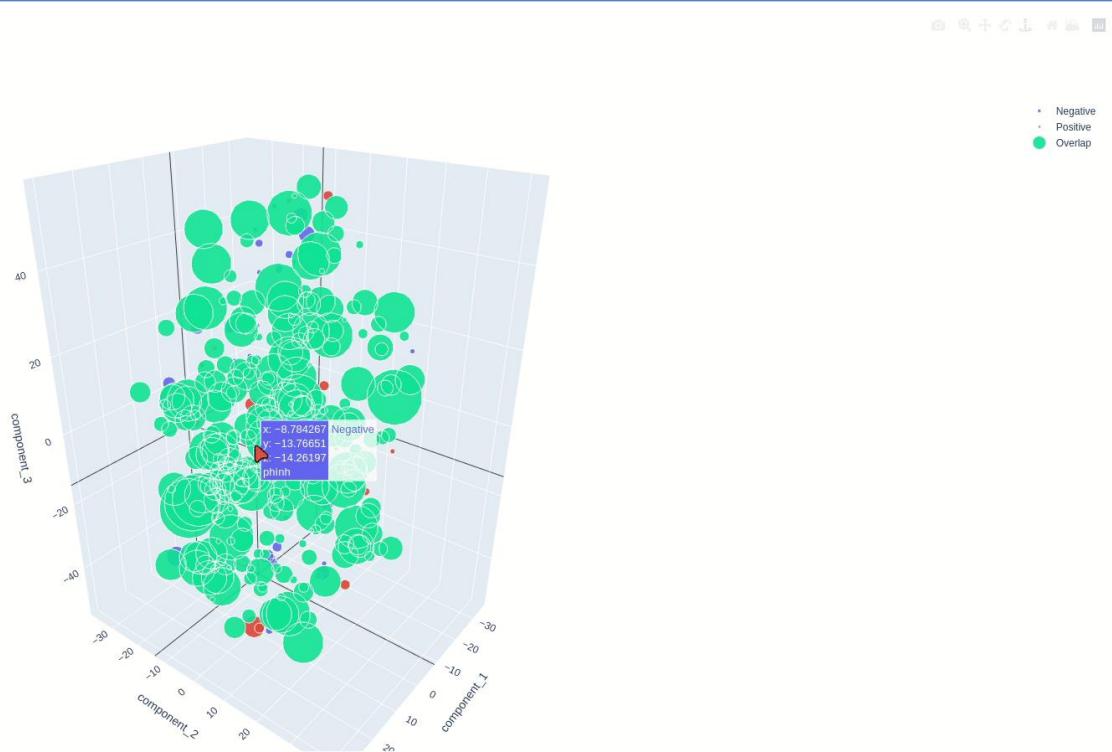
	word	freq	ft_vec	label	component_1	component_2
0	thèm	56	[0.12546667, 0.27063507, -0.3226054, 0.1076125...]	0	-41.806114	-7.578915
1	lum	35	[0.31741396, 0.60768604, -0.23061581, 0.342011...]	0	-13.998660	-20.602335
2	tùm	35	[0.34853685, 0.19606084, -0.22714649, 0.080849...]	0	-48.328167	-11.842845
3	huỷ	31	[0.24349461, 0.12490517, -0.20848149, -0.01994...]	0	-59.509418	-2.166198
4	đểu	28	[0.08494775, 0.045489177, -0.035257675, -0.022...]	0	60.595318	25.569269
...
3862	swatch	2	[0.08958084, 0.15783022, -0.00658212, 0.263154...]	2	13.500787	-17.500595
3863	shock	2	[-0.047569092, 0.16774213, -0.014833469, 0.254...]	2	14.231028	10.243698
3864	ối	2	[0.05223365, 0.014435465, 0.006510902, 0.02751...]	2	31.934618	10.317752
3865	nắng	2	[-0.0120164575, -0.0057069273, -0.0016485625, ...]	2	63.447075	-10.151298
3866	oe	2	[0.8302758, 1.1395676, -0.4882264, 0.68928546,...]	2	-13.129088	-29.835863

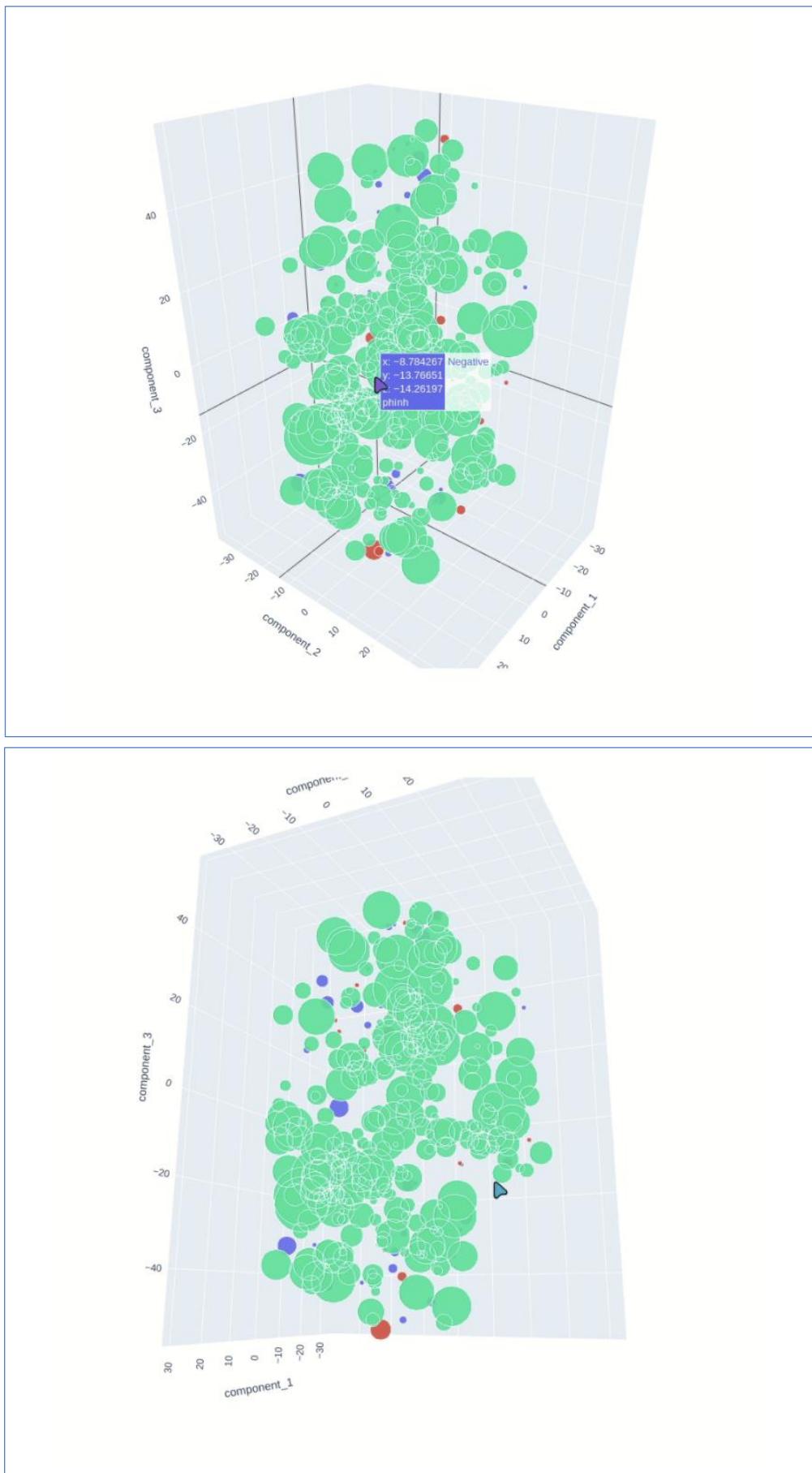


- **Nhận xét:**

- Nếu ta zoom-in vào thì ta sẽ có thể thấy có những từ về mặt ngữ nghĩa, liên quan với nhau thì chúng thật sự ở gần nhau.
 - Các từ phân bố đều và hợp lí hơn so với model word2vec mà ta tự build.
- Tuy nhiên, ta thấy biểu đồ 2D này còn chưa đủ, đôi khi chúng ta tham vọng hơn và ta muốn có một góc nhìn trên không gian nhiều chiều nhất mà con người có thể cảm nhận được là 3 chiều. Lúc này thay vì giữ lại 2 trong TSNE, ta sẽ giữ lại 3 component.

	word	freq		ft_vec	label	component_1	component_2	component_3
0	thèm	56	[0.12546667, 0.27063507, -0.3226054, 0.1076125...	0	14.217293	-10.800855	23.222706	
1	lum	35	[0.31741396, 0.60768604, -0.23061581, 0.342011...	0	17.756479	11.150264	6.434658	
2	tùm	35	[0.34853685, 0.19606084, -0.22714649, 0.080849...	0	12.160611	2.563452	31.675005	
3	huỷ	31	[0.24349461, 0.12490517, -0.20848149, -0.01994...	0	-1.496792	-14.013613	33.494926	
4	đều	28	[0.08494775, 0.045489177, -0.035257675, -0.022...	0	3.065969	29.492119	-31.330750	
...
3862	swatch	2	[0.08958084, 0.15783022, -0.00658212, 0.263154...	2	-7.675353	30.901701	-2.469785	
3863	shock	2	[-0.047569092, 0.16774213, -0.014833469, 0.254...	2	-14.763266	1.552083	-3.847000	
3864	ối	2	[0.05223365, 0.014435465, 0.006510902, 0.02751...	2	-18.706457	-1.169896	-16.921934	
3865	nắng	2	[-0.0120164575, -0.0057069273, -0.0016485625, ...	2	4.905975	-2.665275	-57.885429	
3866	oe	2	[0.8302758, 1.1395676, -0.4882264, 0.68928546,...	2	29.314922	15.599572	3.731873	







- **Nhận xét:** Với không gian 3 chiều, ta có một cái nhìn cực kì hay ho và thú vị với dataset, nhưng cũng phải nói nó không mang lại cho ta quá nhiều thông tin, rất có thể ở một không gian nào đó mà chúng sẽ phân cụm một cách tốt hơn, nhưng với giới hạn về mặt thị giác của con người thì chỉ được đến đây.

4. Đánh giá

STT	Tiêu chí	Hoàn thành (%)
1	Tiền xử lý dữ liệu.	100
2	Thống kê dữ liệu.	100
3	Chọn lựa, giải thích, trực quan các trường và các mối quan hệ giữa chúng.	100
4	Rút ra ý nghĩa hợp lý sau mỗi dữ liệu được trực quan.	100
5	Xem xét trên nhiều quan hệ, nhiều góc nhìn khác nhau.	100
6	Báo cáo trình bày bô cục và định dạng hợp lý, rõ ràng.	100

5. Cấu trúc mã nguồn

- + Folder “**modules**” chứa các user defined function, class.
- + Folder “**data**” chứa các data mà ta đã ghi ra, thực hiện và làm sạch dữ liệu từ project 1.
 - “**normalize_reviews.csv**”
 - “**complement_positive_reviews.csv**”
- + File “**02.eda.ipynb**”: là các phần **3.1** của báo cáo.
- + File “**03-04.eda.ipynb**”: chứa hai phần **3.2** của báo cáo.
- + File “**3D_Plot.html**”: biểu đồ 3D với 3 component.

6. Tài liệu tham khảo

- [T-Test]
(<https://www.investopedia.com/terms/t/t-test.asp>)
- [What Can You Say When Your P-Value is Greater Than 0.05?]
(<https://blog.minitab.com/en/understanding-statistics/what-can-you-say-when-your-p-value-is-greater-than-005>)
- [An introduction to the one-way ANOVA]
([https://www.scribbr.com/statistics/one-way-anova/#:~:text=In%20ANOVA%2C%20the%20null%20hypothesis,no%20difference%20among%20group%20means.&text=Significant%20differences%20among%20group%20means,\(the%20variance%20left%20over\)](https://www.scribbr.com/statistics/one-way-anova/#:~:text=In%20ANOVA%2C%20the%20null%20hypothesis,no%20difference%20among%20group%20means.&text=Significant%20differences%20among%20group%20means,(the%20variance%20left%20over)))
- [Four Ways to Conduct One-Way ANOVA with Python]
(<https://www.marsja.se/four-ways-to-conduct-one-way-anovas-using-python/>)
- [ONE-WAY ANOVA]



(<https://www.pythondatascience.org/anova-python/>)

- [Hands-On Deep learning algorithms with Python]
(<https://www.packtpub.com/product/hands-on-deep-learning-algorithms-with-python/9781789344158>)
- [Natural language processing with TensorFlow]
(<https://www.packtpub.com/product/natural-language-processing-with-tensorflow/9781788478311>)
- [Applied Natural Language Processing in the Enterprise]
(<https://www.amazon.com/Applied-Natural-Language-Processing-Enterprise/dp/149206257X>)
- [Natural Language Processing in Action]
(<https://www.amazon.com/Natural-Language-Processing-Action-Understanding/dp/1617294632>)
- [Python Data Science Handbook]
(<https://www.amazon.com/Python-Data-Science-Handbook-Essential/dp/1491912057>)
- [Deep Learning Quick Reference](<https://www.amazon.com/Deep-Learning-Quick-Reference-optimizing/dp/1788837991>)
- Các tài liệu chính thức trên trang chủ của Plotly, Matplotlib và Seaborn.

7. Mã nguồn

- Link Google Drive:

https://drive.google.com/drive/folders/1xE2fXXHllDHX3JertfFz_09x967QiUW3?usp=sharing