DATA EXPLORATION

Installing dependecies

In [1]:
```
!pip install autocorrect
!pip install wordcloud
```

```
Requirement already satisfied: autocorrect in c:\users\henry\anaconda3\envs\ecogeo_tool\lib\site-packages (2.
1.0)
Requirement already satisfied: wordcloud in c:\users\henry\anaconda3\envs\ecogeo_tool\lib\site-packages (1.8.
0)
Requirement already satisfied: numpy>=1.6.1 in c:\users\henry\anaconda3\envs\ecogeo_tool\lib\site-packages (f
rom wordcloud) (1.17.4)
Requirement already satisfied: matplotlib in c:\users\henry\anaconda3\envs\ecogeo_tool\lib\site-packages (fro
m wordcloud) (2.2.2)
Requirement already satisfied: pillow in c:\users\henry\anaconda3\envs\ecogeo_tool\lib\site-packages (from wo
rdcloud) (7.0.0)
Requirement already satisfied: cycler>=0.10 in c:\users\henry\anaconda3\envs\ecogeo_tool\lib\site-packages (f
rom matplotlib->wordcloud) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in c:\users\henry\anaconda3\envs\ecog
eo_tool\lib\site-packages (from matplotlib->wordcloud) (2.4.6)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\henry\anaconda3\envs\ecogeo_tool\lib\site-pac
kages (from matplotlib->wordcloud) (2.8.1)
Requirement already satisfied: pytz in c:\users\henry\anaconda3\envs\ecogeo_tool\lib\site-packages (from matp
lotlib->wordcloud) (2019.3)
Requirement already satisfied: six>=1.10 in c:\users\henry\anaconda3\envs\ecogeo_tool\lib\site-packages (from
matplotlib->wordcloud) (1.13.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\henry\anaconda3\envs\ecogeo_tool\lib\site-packag
es (from matplotlib->wordcloud) (1.1.0)
Requirement already satisfied: setuptools in c:\users\henry\anaconda3\envs\ecogeo_tool\lib\site-packages (fro
m kiwisolver>=1.0.1->matplotlib->wordcloud) (44.0.0.post20200106)
```

Loading dependecies

In [2]:
```
from data_exploring.data_exploring_functions import *
import plotly.express as px
import nltk
from wordcloud import WordCloud
import os
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\henry\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Loading and cleaning data

In [3]:
```
#File names and base path.
file_satisfaction = 'satisfaction_ratings'
file_response = 'NPS_responses'
base_datapath = os.path.join(os.getcwd(),'data_exploring','data')

#Loading the data using a function from data_exploring_functions. Spell correction, lower letters and elimina
tion of stopwrods is done in
#this step. It returns the loaded dataset with an extra column named "processed comment"
data_satisfaction = loading_data(file_satisfaction,base_datapath,'csv')
data_responses = loading_data(file_response,base_datapath,'csv')
```

Data visualization

In [4]:
```
data_satisfaction.head(3)
```

Out[4]:

	Unnamed: 0	Requester	User Id	Email	Ticket Id	Brand	Group	Assignee	Satisfaction	Co
0	0	Carlos Enrique brito	401415540751	carlosenrique1989n@hotmail.com	595380	OFFCORSS	Soporte OFFCORSS	CTS Transporte	good	
1	1	Lizeth Herrera	401737616791	lkhr328@gmail.com	600778	OFFCORSS	Soporte OFFCORSS	Devoluciones	good	
2	2	Carolina Sanchez	401569843672	caritoss1@hotmail.com	598181	OFFCORSS	Soporte OFFCORSS	CTS Transporte	bad	

ex

In [5]:
```
data_responses.head(3)
```

Out[5]:

	Unnamed: 0	Survey Date	Name	User Id	Email	Rating	Classification	Comment
0	0	2020-08-10	Johanna Vargas T	400842393092	vhannyt@gmail.com	7	passive	NaN
1	1	2020-08-10	Maria Carolina Parra Rincón	400932763371	mariacarolinaparrar@gmail.com	1	detractor	Atender a las reclamaciones a tiempo para evit...
2	2	2020-08-10	Luz Marina González Pulido	400646948972	14a793667beb4637bc67b25241ee1150@ct.vtex.com.br	0	detractor	No leen con atención, por favor dictar capacit...

Understanding repeated tickets Id

In [6]:
```
print('there are ' + str(len(data_satisfaction['Ticket Id'].unique())) + ' unique tickets')
print('there are ' + str(len(data_satisfaction)) + ' tickets')
duplicate_tickets = data_satisfaction.groupby('Ticket Id').size().sort_values(ascending=False).reset_index(na
me ='tickets count')
duplicate_example = data_satisfaction[data_satisfaction['Ticket Id'] == duplicate_tickets['Ticket Id'][3]]
print('---------------------------')
print('---------------------------')
print('Example of duplicate Tickets')
duplicate_example
```

```
there are 2657 unique tickets
there are 2774 tickets
---------------------------
---------------------------
Example of duplicate Tickets
```

Out[6]:

	Unnamed: 0	Requester	User Id	Email	Ticket Id	Brand	Group	Assignee	Satisfaction	Con
2254	2254	leidy huertas	400578886512	leidy.kari96@gmail.com	583147	OFFCORSS	Soporte OFFCORSS	CTS Transporte	good	BUENC QU RESPONDI UN TIEMI
2255	2255	leidy huertas	400578886512	leidy.kari96@gmail.com	583147	OFFCORSS	Soporte OFFCORSS	CTS Transporte	good	BUENC QU RESPONDI UN TIEMI
2256	2256	leidy huertas	400578886512	leidy.kari96@gmail.com	583147	OFFCORSS	Soporte OFFCORSS	CTS Transporte	good	

The repeated ticket Id could be for repeated comments, or for tracking the steps in a request. Therefore are eliminated the repeated tickets Ids with repeated comments.

In [7]:
```
data_satisfaction.drop_duplicates(subset=['Ticket Id', 'processed comment'], inplace = True)
```

Missing values analysis . It is found the percentage of missing comments, and they are replaced with empty values.

In [8]:
```
data_satisfaction['processed comment'].isna().sum()
percent_nan = 100*data_satisfaction['processed comment'].isna().sum() / data_satisfaction['processed comment'
].isna().count()
print('Percentage of empty comments for satisfaction data is ' , round(percent_nan,2),'%')
data_responses['processed comment'].isna().sum()
percent_nan = 100*data_responses['processed comment'].isna().sum() / data_responses['processed comment'].isna
().count()
print('Percentage of empty comments for responses data is ' , round(percent_nan,2),'%')
```

```
Percentage of empty comments for satisfaction data is  61.3 %
Percentage of empty comments for responses data is  48.47 %
```

In [9]:
```
data_satisfaction['processed comment'] = data_satisfaction['processed comment'].fillna('')
data_responses['processed comment'] = data_responses['processed comment'].fillna('')
```

Analysis of comments by groups

In [10]:
```
# Select comments by classification
satisfaction_good = data_satisfaction[(data_satisfaction['Satisfaction'] == 'good')]['processed comment']
satisfaction_bad = data_satisfaction[(data_satisfaction['Satisfaction'] == 'bad')]['processed comment']

responses_promoter = data_responses[(data_responses['Classification'] == 'promoter')]['processed comment']
responses_passive = data_responses[(data_responses['Classification'] == 'passive')]['processed comment']
responses_detractor = data_responses[(data_responses['Classification'] == 'detractor')]['processed comment']
```

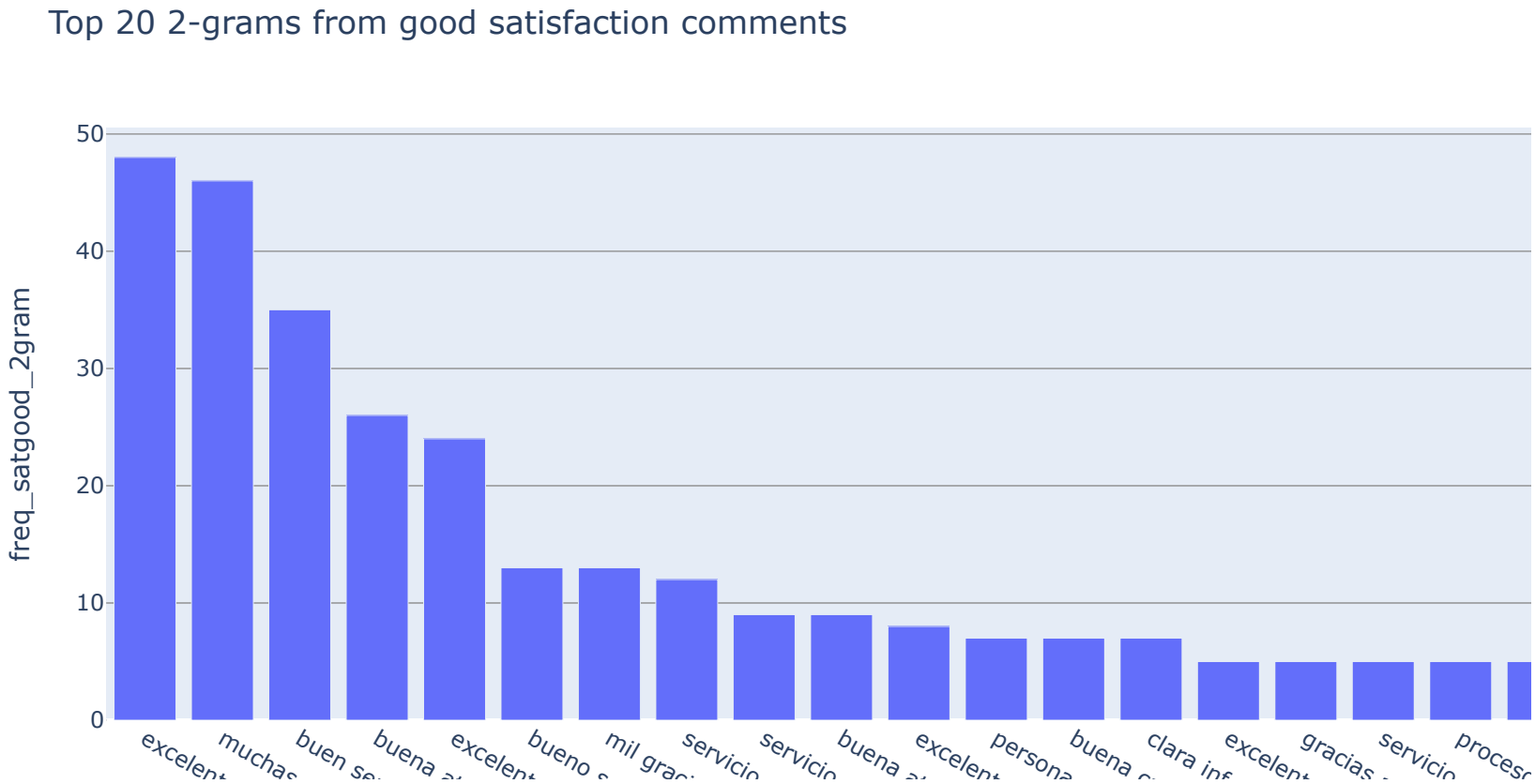
Data Satisfaction good-bad N-grams analysis

In [11]:
```
# Use the get_top_n_words function from data_exploring_functions (Natesh function) to get the most frequent n
-grams
# 1- gram for satisfaction_good
common_words = get_top_n_words(satisfaction_good, 20,1)
grams_df = pd.DataFrame(common_words, columns = ['word_satgood_1gram' , 'freq_satgood_1gram'])
fig =  px.bar(grams_df, x = 'word_satgood_1gram', y ='freq_satgood_1gram', title='Top 20 1-gram from good sat
isfaction comments',
             labels={'freq_satgood_1gram':'Frequency', 'word_satgood_1gram':'1-gram'})
fig.show()

# 2- gram for satisfaction_good
common_words = get_top_n_words(satisfaction_good, 20,2)
grams_df['word_satgood_2gram'] = [tuple_word[0] for tuple_word in common_words]
grams_df['freq_satgood_2gram'] = [tuple_word[1] for tuple_word in common_words]
fig =  px.bar(grams_df, x = 'word_satgood_2gram', y ='freq_satgood_2gram', title='Top 20 2-grams from good sa
tisfaction comments',
             labels={'word_satgood_2gram':'Frequency', 'word_satgood_2gram':'2-gram'})
fig.show()

# 3- gram for satisfaction_good
common_words = get_top_n_words(satisfaction_good, 20,3)
grams_df['word_satgood_3gram'] = [tuple_word[0] for tuple_word in common_words]
grams_df['freq_satgood_3gram'] = [tuple_word[1] for tuple_word in common_words]
fig =  px.bar(grams_df, x = 'word_satgood_3gram', y ='freq_satgood_3gram', title='Top 20 3-grams from good sa
tisfaction comments',
             labels={'word_satgood_3gram':'Frequency', 'word_satgood_3gram':'3-gram'})
fig.show()
```


Top 20 1-gram from good satisfaction comments
Frequency
180
160
140
120
100
80
60
40
20
0


Top 20 2-grams from good satisfaction comments
freq_satgood_2gram
50
40
30
20
10
0

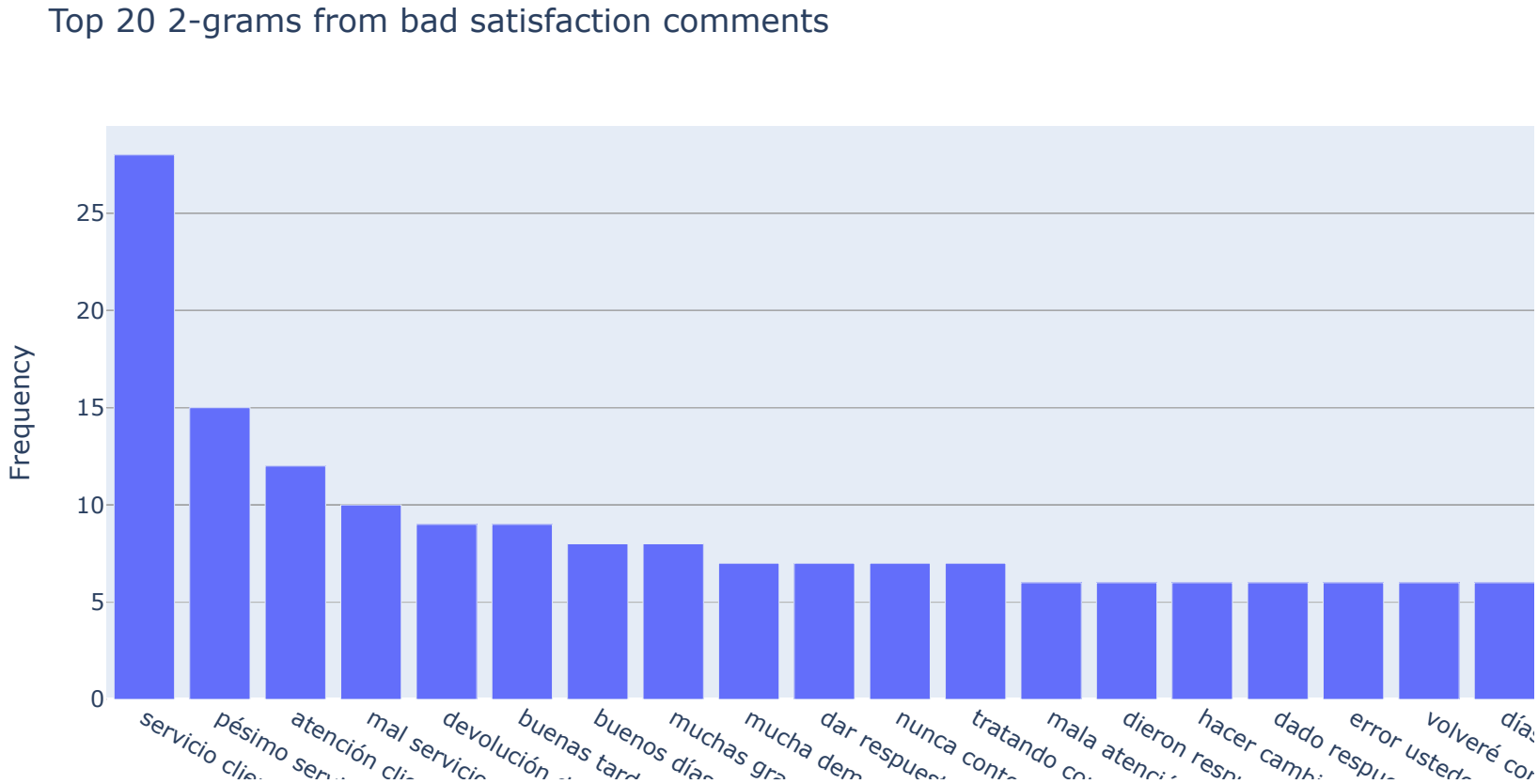

Top 20 3-grams from good satisfaction comments
freq_satgood_3gram
0
1
2
3
4
excelente servi
buen servicio cli
bueno excelente
atención mucha
buen servicio q
buena calidad p
bueno servicio
muchas gracias
gracias atención
excelente servi
excelente servi
servicio rápido
respuesta clara
línea servicio cli
respuesta rápid
quisiera saber
muchas gracias
bueno muchas
servicio p
bu

In [12]:
```
# Use the get_top_n_words function from data_exploring_functions (Natesh function) to get the most frequent n
-grams
# 1- gram for satisfaction_bad
common_words = get_top_n_words(satisfaction_bad, 20,1)
grams_df['word_satbad_1gram'] = [tuple_word[0] for tuple_word in common_words]
grams_df['freq_satbad_1gram'] = [tuple_word[1] for tuple_word in common_words]
fig =  px.bar(grams_df, x = 'word_satbad_1gram', y ='freq_satbad_1gram', title='Top 20 1-gram from bad satisf
action comments',
             labels={'freq_satbad_1gram':'Frequency', 'word_satbad_1gram':'1-gram'})
fig.show()

# 2- gram for satisfaction_bad
common_words = get_top_n_words(satisfaction_bad, 20,2)
grams_df['word_satbad_2gram'] = [tuple_word[0] for tuple_word in common_words]
grams_df['freq_satbad_2gram'] = [tuple_word[1] for tuple_word in common_words]
fig =  px.bar(grams_df, x = 'word_satbad_2gram', y ='freq_satbad_2gram', title='Top 20 2-grams from bad satis
faction comments',
             labels={'freq_satbad_2gram':'Frequency', 'word_satbad_2gram':'2-gram'})
fig.show()

# 3- gram for satisfaction_bad
common_words = get_top_n_words(satisfaction_bad, 20,3)
grams_df['word_satbad_3gram'] = [tuple_word[0] for tuple_word in common_words]
grams_df['freq_satbad_3gram'] = [tuple_word[1] for tuple_word in common_words]
fig =  px.bar(grams_df, x = 'word_satbad_3gram', y ='freq_satbad_3gram', title='Top 20 3-grams from bad satis
faction comments',
             labels={'freq_satbad_3gram':'Frequency', 'word_satbad_3gram':'3-gram'})
fig.show()
```


Top 20 1-gram from bad satisfaction comments
Frequency
120
100
80
60
40
20
0

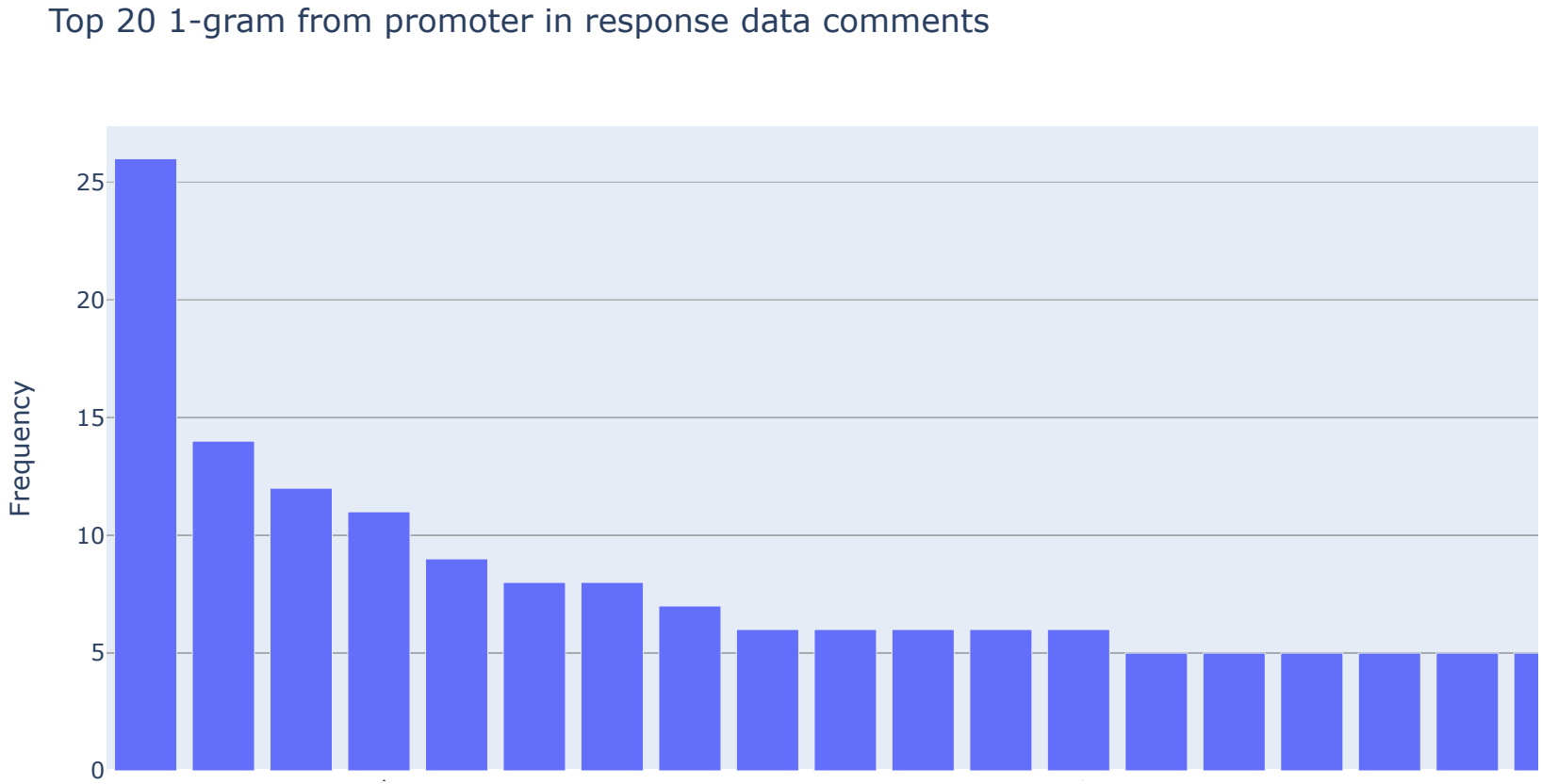

Top 20 2-grams from bad satisfaction comments
Frequency
0
5
10
15
20
25
servicio clie
pésimo ser
atención cli
mal servicio
devolución
buenas tard
buenos día
muchas gra
mucha dem
dar respues
nunca cont
tratando co
mala atenci
dieron resp
hacer cambi
dado respu
error usted
volveré co
día

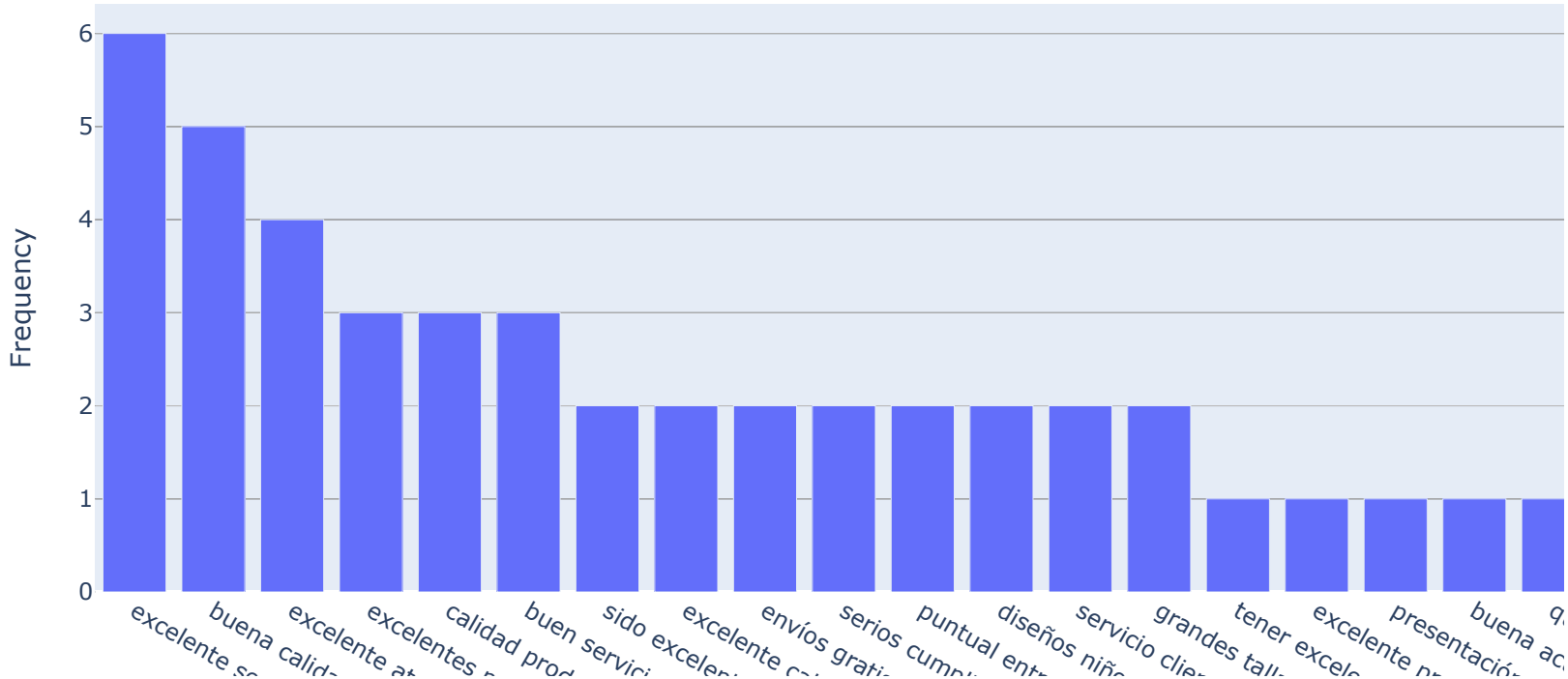
Data Response promoter-passive-detractor N-grams analysis

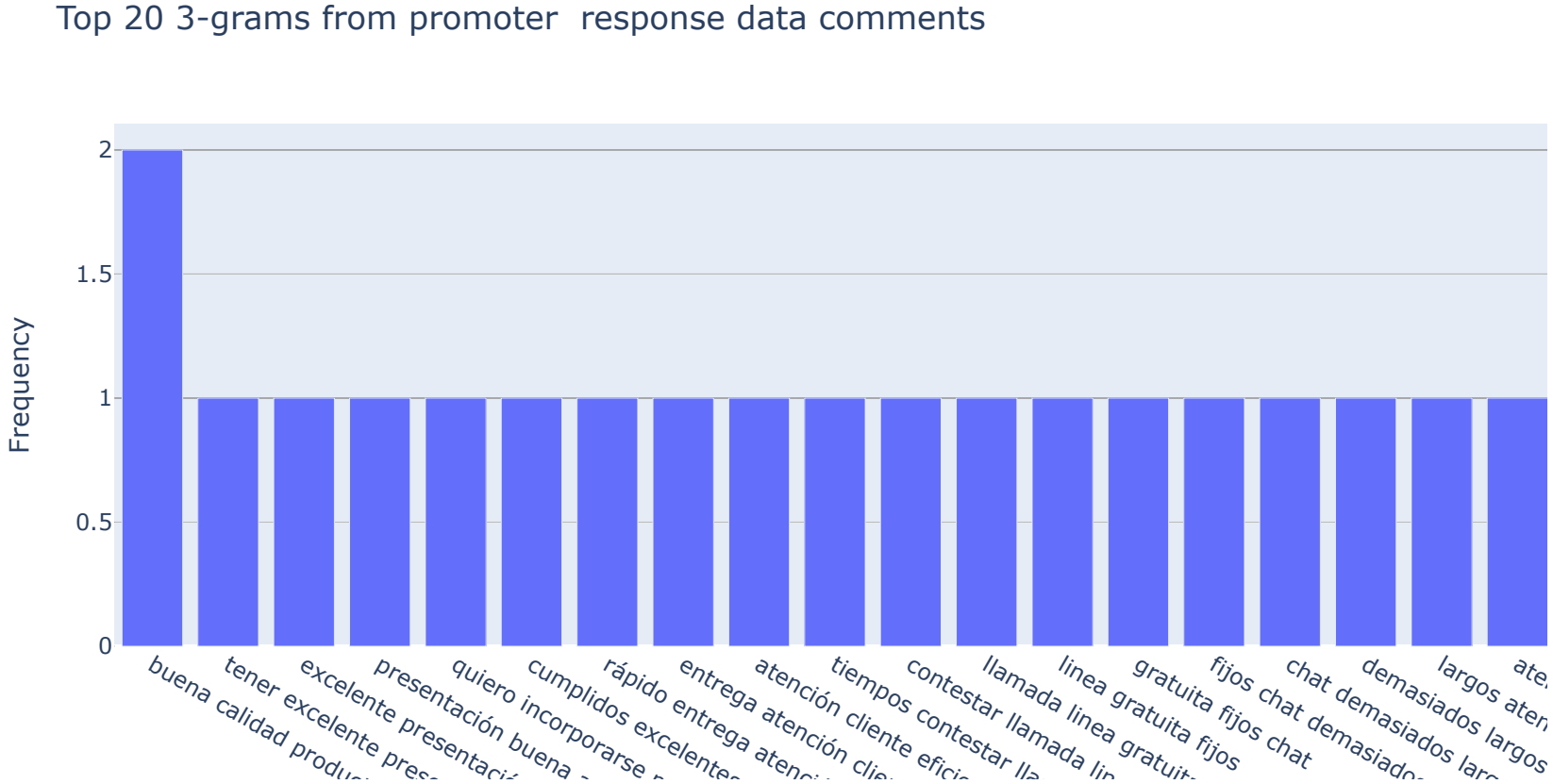
In [13]:
```
# Use the get_top_n_words function from data_exploring_functions (Natesh function) to get the most frequent n
-grams
# 1- gram for response promoter
common_words = get_top_n_words(responses_promoter, 20,1)
grams_df['word_respromoter_1gram'] = [tuple_word[0] for tuple_word in common_words]
grams_df['freq_respromoter_1gram'] = [tuple_word[1] for tuple_word in common_words]
fig =  px.bar(grams_df, x = 'word_respromoter_1gram', y ='freq_respromoter_1gram', title='Top 20 1-gram from
 promoter in response data comments',
             labels={'freq_respromoter_1gram':'Frequency', 'word_respromoter_1gram':'1-gram'})
fig.show()

# 2- gram for response promoter
common_words = get_top_n_words(responses_promoter, 20,2)
grams_df['word_respromoter_2gram'] = [tuple_word[0] for tuple_word in common_words]
grams_df['freq_respromoter_2gram'] = [tuple_word[1] for tuple_word in common_words]
fig =  px.bar(grams_df, x = 'word_respromoter_2gram', y ='freq_respromoter_2gram', title='Top 20 2-grams from
promoter  response data comments',
             labels={'freq_respromoter_2gram':'Frequency', 'word_respromoter_2gram':'2-gram'})
fig.show()

# 3- gram for response promoter
common_words = get_top_n_words(responses_promoter, 20,3)
grams_df['word_respromoter_3gram'] = [tuple_word[0] for tuple_word in common_words]
grams_df['freq_respromoter_3gram'] = [tuple_word[1] for tuple_word in common_words]
fig =  px.bar(grams_df, x = 'word_respromoter_3gram', y ='freq_respromoter_3gram', title='Top 20 3-grams from
promoter  response data comments',
             labels={'freq_respromoter_3gram':'Frequency', 'word_respromoter_3gram':'3-gram'})
fig.show()
```


Top 20 1-gram from promoter in response data comments
Frequency
25
20
15
10
5
0

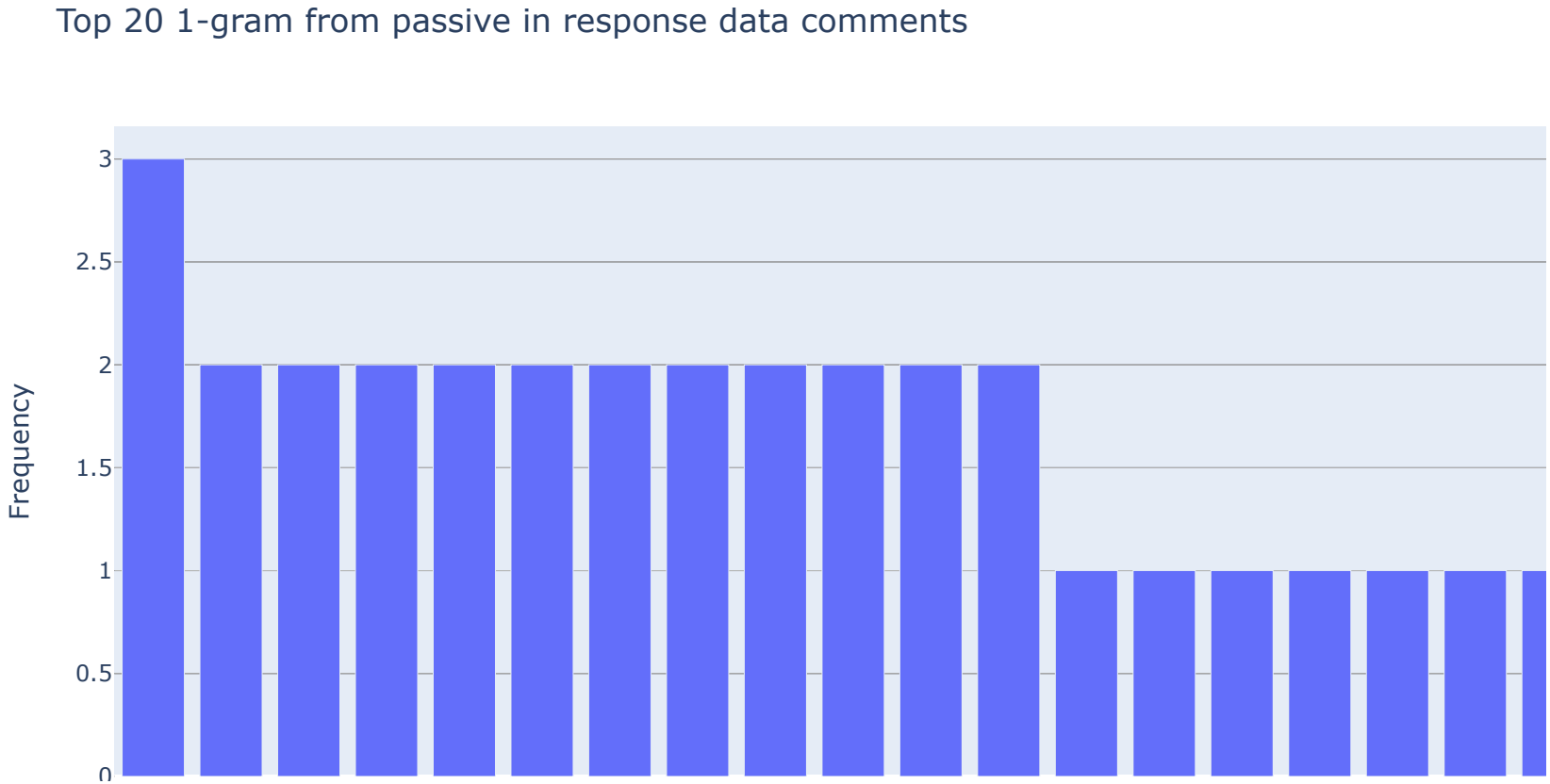

Top 20 2-grams from promoter response data comments
Frequency
0
1
2
3
4
5
6
excelente
buena calid
excelente
excelentes
calidad prod
buen servici
sido excele
excelente
envíos grati
serios cumpl
puntual ent
diseños niñ
servicio clie
grandes tall
tener excele
excelente
presentació
buena ac
q

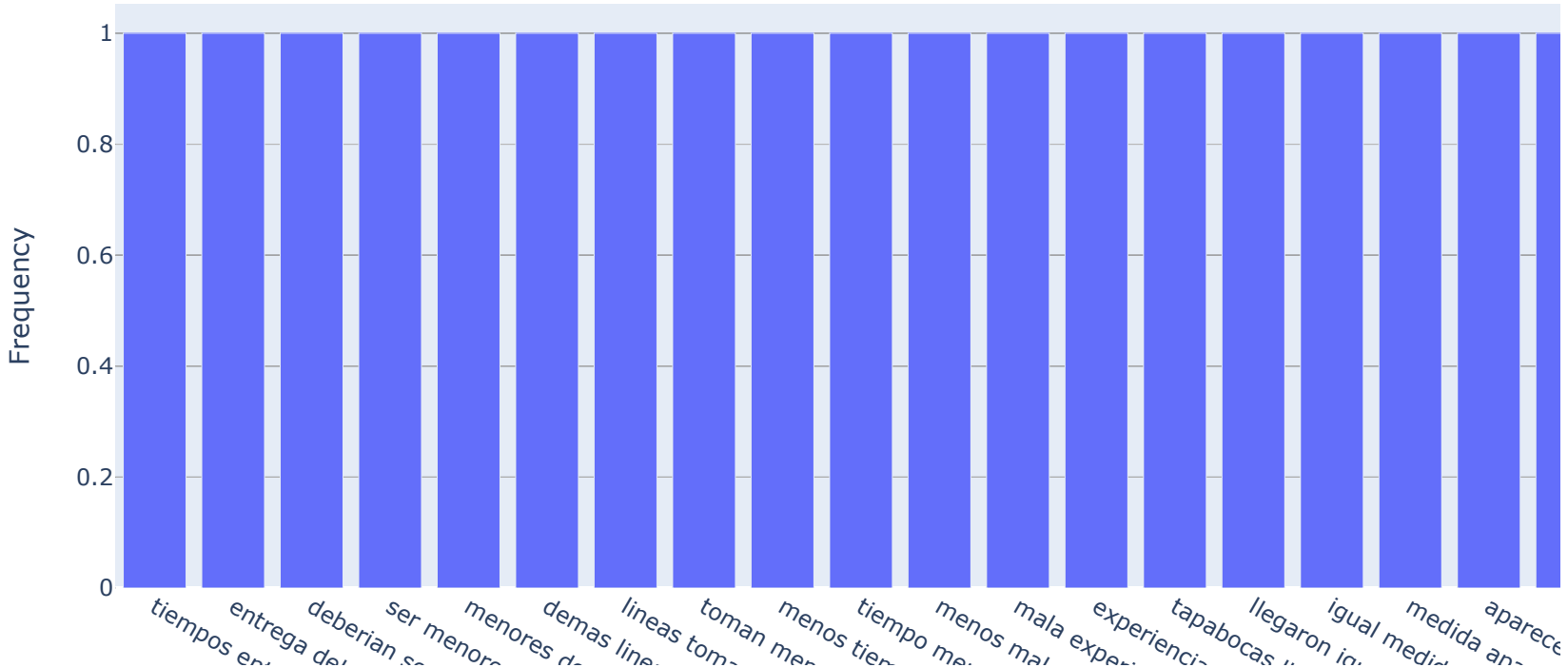

Top 20 3-grams from promoter response data comments
Frequency
2
1.5
1
0.5
0
buena calidad produ
tener excelente pres
excelente presentació
presentación buena
quiero incorporarse
cumplidos excelente
rápido entrega atenci
entrega atención clie
atención cliente efici
tiempos contestar lla
contestar llamada lin
llamada linea gratuit
linea gratuita fijos
gratuita fijos chat
fijos chat demasiado
chat demasiados lar
demasiados largos
largos aten
aten

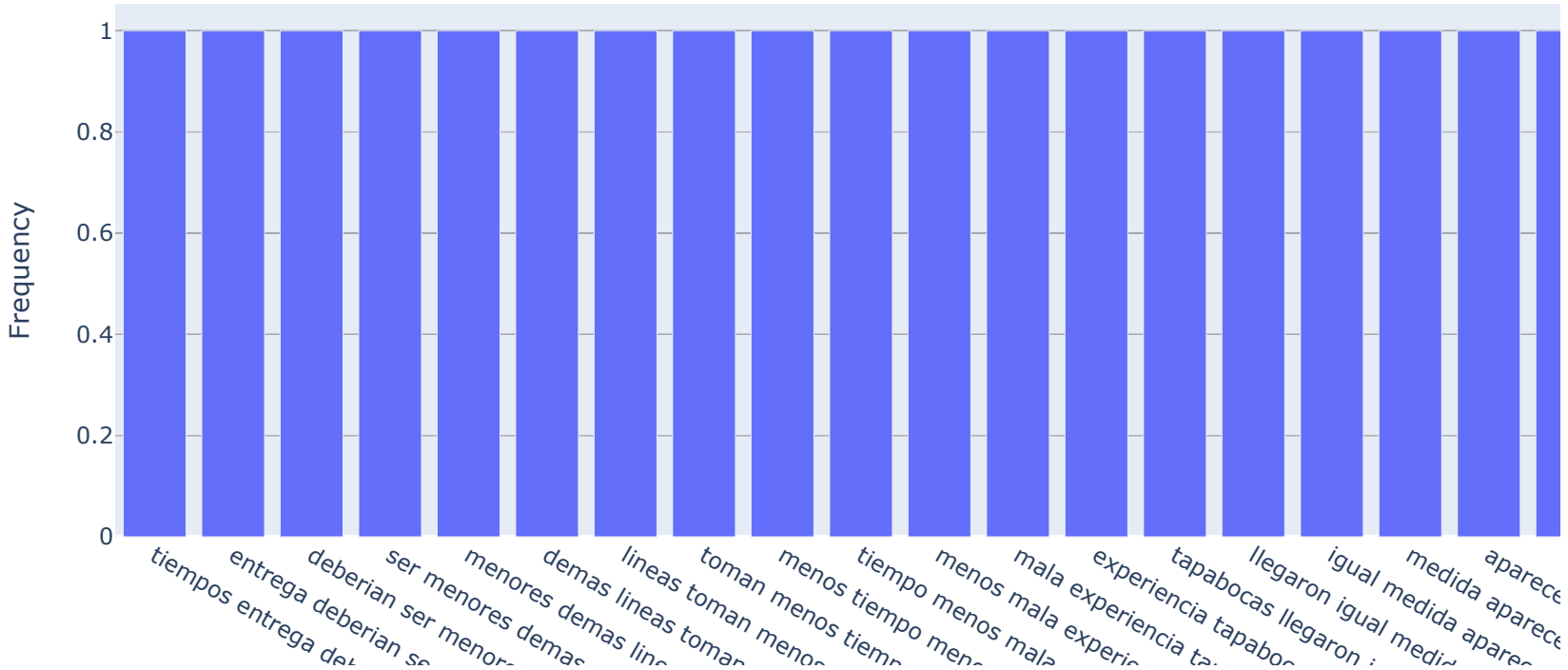
In [14]:
```
# Use the get_top_n_words function from data_exploring_functions (Natesh function) to get the most frequent n
-grams
# 1- gram for response passive
common_words = get_top_n_words(responses_passive, 20,1)
grams_df['word_respassive_1gram'] = [tuple_word[0] for tuple_word in common_words]
grams_df['freq_respassive_1gram'] = [tuple_word[1] for tuple_word in common_words]
fig =  px.bar(grams_df, x = 'word_respassive_1gram', y ='freq_respassive_1gram', title='Top 20 1-gram from pa
ssive in response data comments',
             labels={'freq_respassive_1gram':'Frequency', 'word_respassive_1gram':'1-gram'})
fig.show()

# 2- gram for response passive
common_words = get_top_n_words(responses_passive, 20,2)
grams_df['word_respassive_2gram'] = [tuple_word[0] for tuple_word in common_words]
grams_df['freq_respassive_2gram'] = [tuple_word[1] for tuple_word in common_words]
fig =  px.bar(grams_df, x = 'word_respassive_2gram', y ='freq_respassive_2gram', title='Top 20 2-grams from p
assive response data comments',
             labels={'freq_respassive_2gram':'Frequency', 'word_respassive_2gram':'2-gram'})
fig.show()

# 3- gram for response passive
common_words = get_top_n_words(responses_passive, 20,3)
grams_df['word_respassive_3gram'] = [tuple_word[0] for tuple_word in common_words]
grams_df['freq_respassive_3gram'] = [tuple_word[1] for tuple_word in common_words]
fig =  px.bar(grams_df, x = 'word_respassive_3gram', y ='freq_respassive_3gram', title='Top 20 3-grams from p
assive  response data comments',
             labels={'freq_respassive_3gram':'Frequency', 'word_respassive_3gram':'3-gram'})
fig.show()
```


Top 20 1-gram from passive in response data comments
Frequency
3
2.5
2
1.5
1
0.5
0

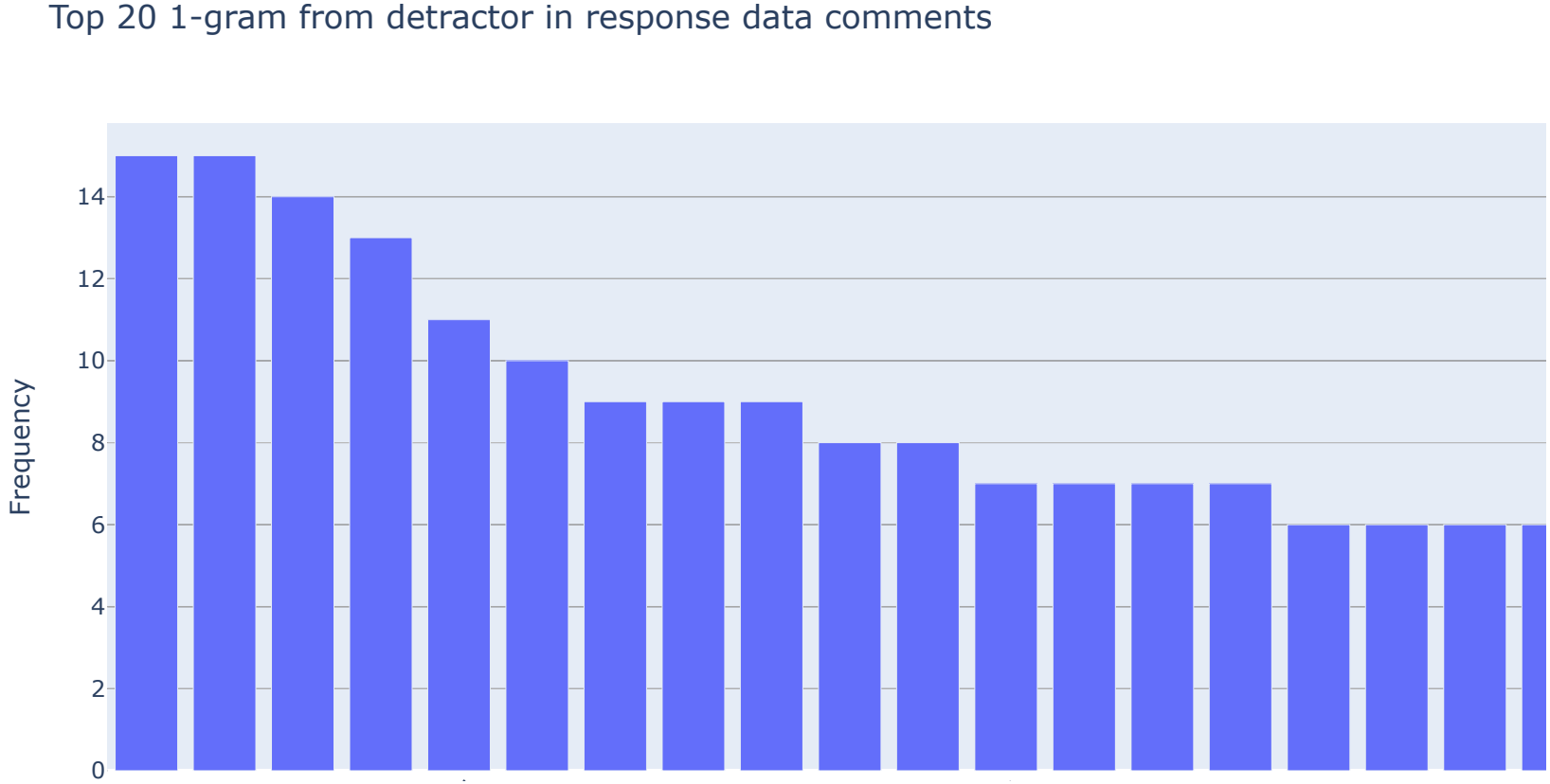

Top 20 2-grams from passive response data comments
Frequency
1
0.8
0.6
0.4
0.2
0

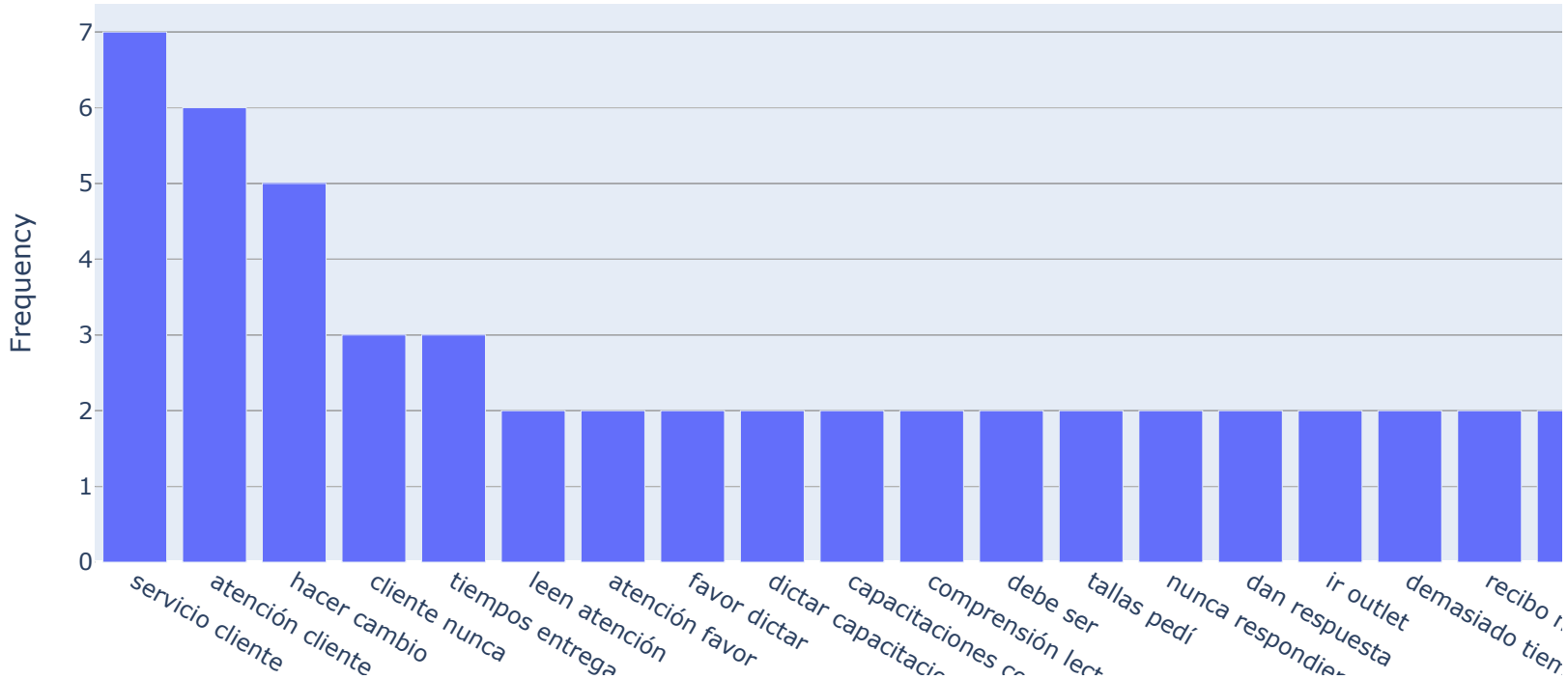

Top 20 3-grams from passive response data comments
Frequency
1
0.8
0.6
0.4
0.2
0
tiempos entrega deb
entrega deberian se
deberian ser menor
ser menores demas
menores demas line
demas lineas tomar
lineas toman menos
toman menos tiemp
menos tiempo mene
tiempo menos mala
menos mala experie
mala experiencia ta
experiencia tapaboc
tapabocas llegaron
llegaron igual medid
igual medida apare
medida aparece
aparece

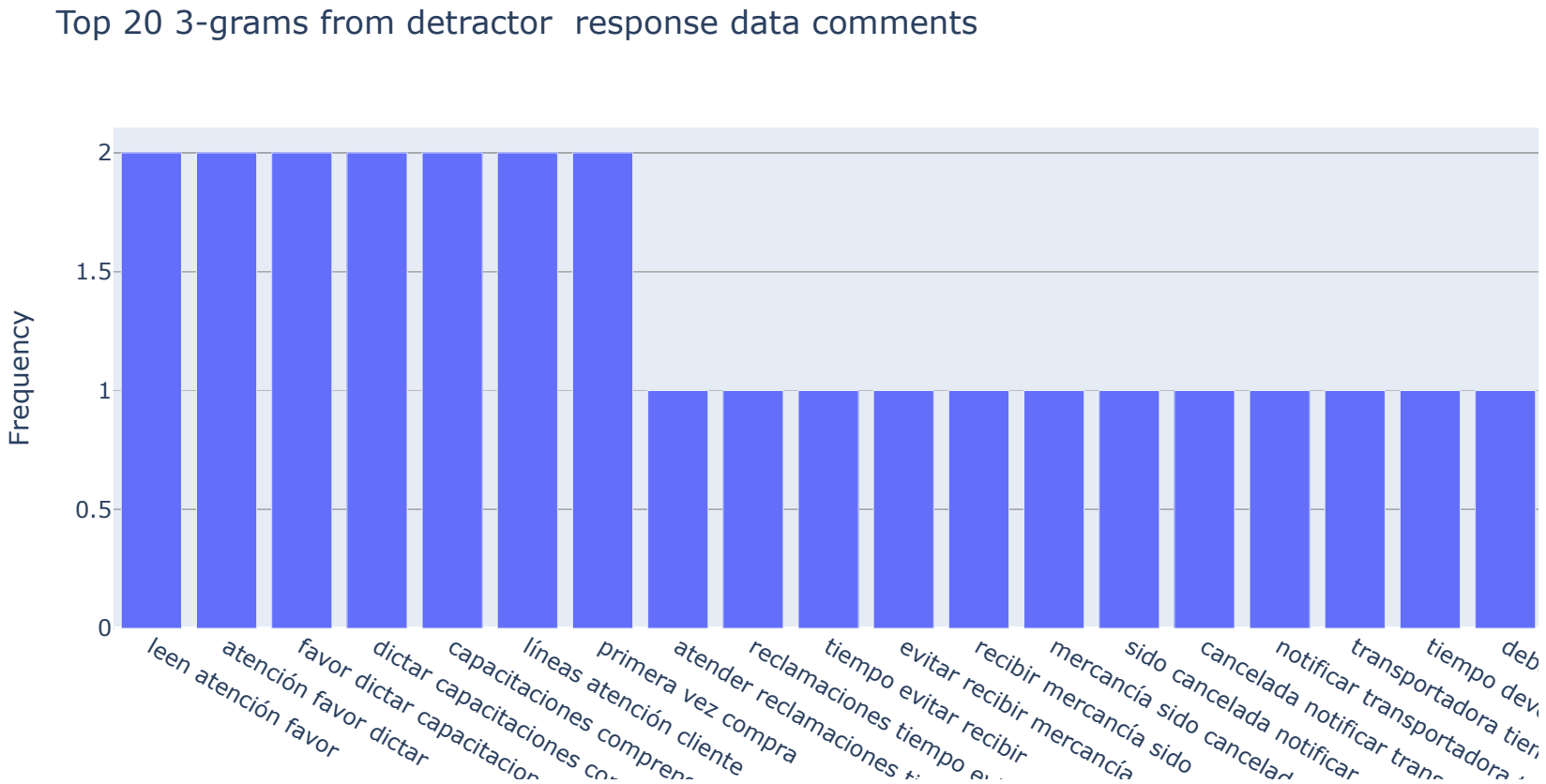
In [15]:
```
# Use the get_top_n_words function from data_exploring_functions (Natesh function) to get the most frequent n
-grams
# 1- gram for response detractor
common_words = get_top_n_words(responses_detractor, 20,1)
grams_df['word_resdetractor_1gram'] = [tuple_word[0] for tuple_word in common_words]
grams_df['freq_resdetractor_1gram'] = [tuple_word[1] for tuple_word in common_words]
fig =  px.bar(grams_df, x = 'word_resdetractor_1gram', y ='freq_resdetractor_1gram', title='Top 20 1-gram fro
m detractor in response data comments',
             labels={'freq_resdetractor_1gram':'Frequency', 'word_resdetractor_1gram':'1-gram'})
fig.show()

# 2- gram for response detractor
common_words = get_top_n_words(responses_detractor, 20,2)
grams_df['word_resdetractor_2gram'] = [tuple_word[0] for tuple_word in common_words]
grams_df['freq_resdetractor_2gram'] = [tuple_word[1] for tuple_word in common_words]
fig =  px.bar(grams_df, x = 'word_resdetractor_2gram', y ='freq_resdetractor_2gram', title='Top 20 2-grams fr
om detractor response data comments',
             labels={'freq_resdetractor_2gram':'Frequency', 'word_resdetractor_2gram':'2-gram'})
fig.show()

# 3- gram for response detractor
common_words = get_top_n_words(responses_detractor, 20,3)
grams_df['word_resdetractor_3gram'] = [tuple_word[0] for tuple_word in common_words]
grams_df['freq_resdetractor_3gram'] = [tuple_word[1] for tuple_word in common_words]
fig =  px.bar(grams_df, x = 'word_resdetractor_3gram', y ='freq_resdetractor_3gram', title='Top 20 3-grams fr
om detractor  response data comments',
             labels={'freq_resdetractor_3gram':'Frequency', 'word_resdetractor_3gram':'3-gram'})
fig.show()
```


Top 20 1-gram from detractor in response data comments
Frequency
0
2
4
6
8
10
12
14

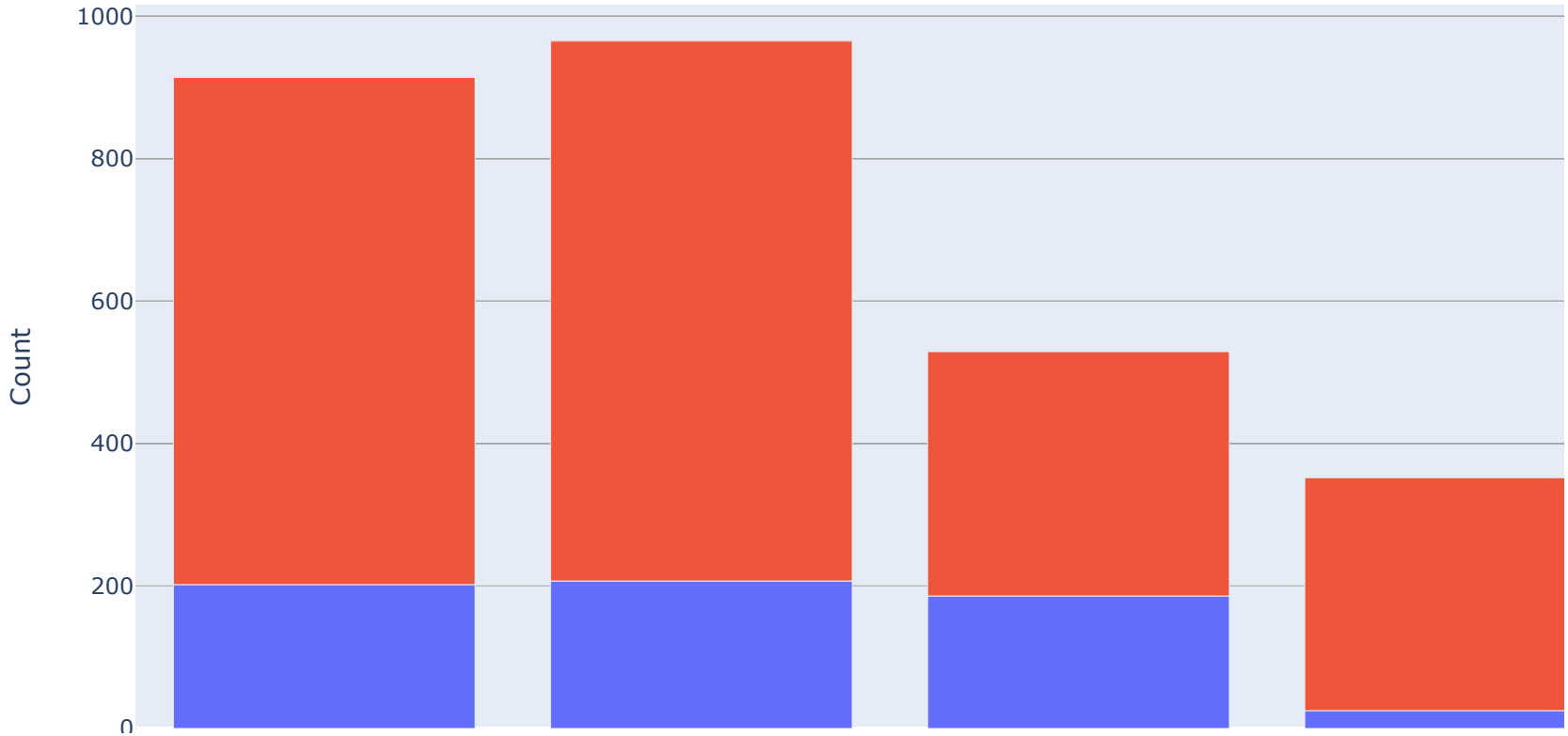

Top 20 2-grams from detractor response data comments
Frequency
0
1
2
3
4
5
6
7
servicio cliente
atención cliente
hacer cambio
cliente nunca
tiempos entrega
leen atención
atención favor
favor dictar
dictar capacitaciones
debe ser
tallas pedí
nunca respondieron
dan respuesta
ir outlet
demasiado tiempo



In [16]:
```
# Saving the n-grams dataframe.
grams_df.to_csv(os.path.join(base_datapath,'grams.csv'))
```

Analysis by organization groups

In [17]:
```
group_count = data_satisfaction.groupby(['Group','Satisfaction'])['Ticket Id'].count().reset_index(name='Count')
px.bar(group_count, x ='Group', y= 'Count', color = 'Satisfaction' )
```



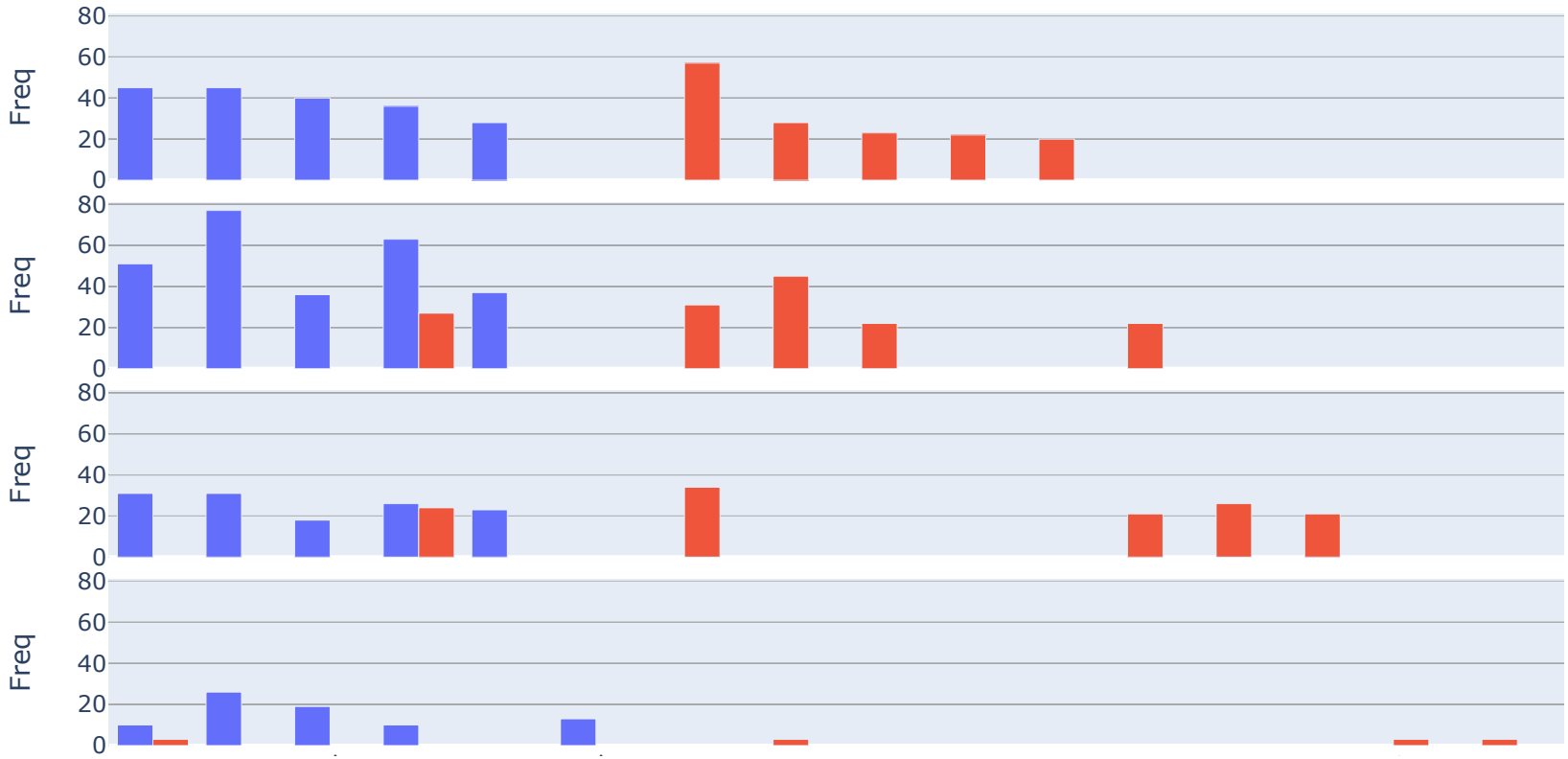
Finding the most frequently 1-grams by organization groups

In [18]:
```
uniq_groups = data_satisfaction.Group.unique()
n = 5
words_group = []
for group in uniq_groups:
    common_words = get_top_n_words(data_satisfaction[(data_satisfaction['Group'] == group) & (data_satisfacti
on['Satisfaction'] == 'good')]['processed comment'], n, 1)
    for word in common_words:
        words_group.append([word[0], word[1], group, 'good'])
    common_words = get_top_n_words(data_satisfaction[(data_satisfaction['Group'] == group) & (data_satisfacti
on['Satisfaction'] == 'bad')]['processed comment'], n, 1)
    for word in common_words:
        words_group.append([word[0], word[1], group, 'bad'])
words_1gram_group = pd.DataFrame(words_group, columns = ['1gram','Freq','Group','Satisfaction'])

conver_dict = {'Call Center': 'cc', 'Soporte OFFCORSS': 'sop', 'Tienda Virtual' : 'tvir', 'Venta Directa': 'v
dir'}
words_1gram_group['id'] = words_1gram_group['Group'].apply(lambda x: conver_dict[x])

fig = px.bar(words_1gram_group, x = '1gram', y = 'Freq', color = 'Satisfaction', facet_row='id', barmode='gro
up')
fig.show()

words_1gram_group.to_csv(os.path.join(base_datapath,'1gram_organizationgroups.csv'))
```

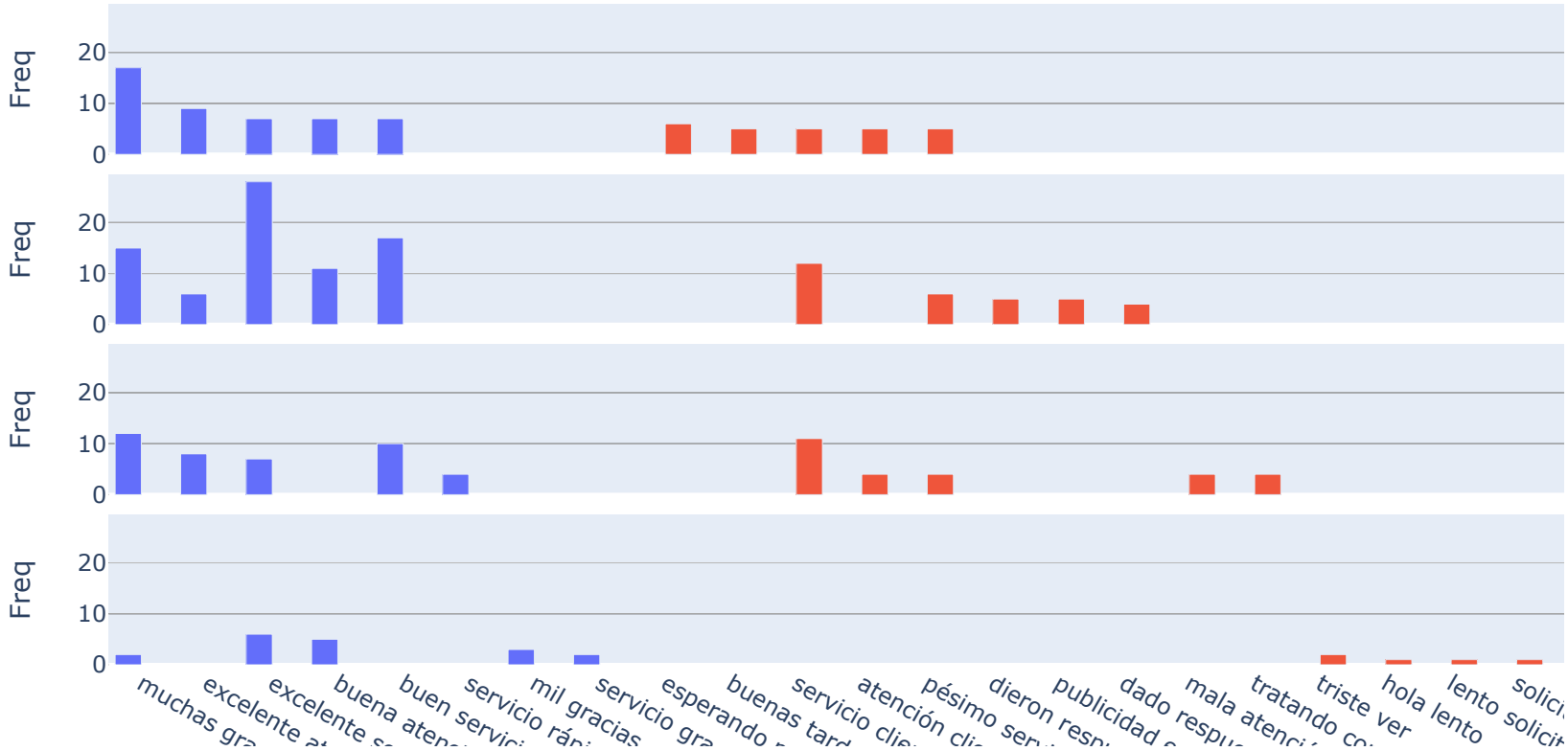


Finding the most frequently 2-grams by organization groups

In [19]:
```
uniq_groups = data_satisfaction.Group.unique()
n = 5
words_group = []
for group in uniq_groups:
    common_words = get_top_n_words(data_satisfaction[(data_satisfaction['Group'] == group) & (data_satisfacti
on['Satisfaction'] == 'good')]['processed comment'], n, 2)
    for word in common_words:
        words_group.append([word[0], word[1], group, 'good'])
    common_words = get_top_n_words(data_satisfaction[(data_satisfaction['Group'] == group) & (data_satisfacti
on['Satisfaction'] == 'bad')]['processed comment'], n, 2)
    for word in common_words:
        words_group.append([word[0], word[1], group, 'bad'])
words_2gram_group = pd.DataFrame(words_group, columns = ['2gram','Freq','Group','Satisfaction'])

conver_dict = {'Call Center': 'cc', 'Soporte OFFCORSS': 'sop', 'Tienda Virtual' : 'tvir', 'Venta Directa': 'v
dir'}
words_2gram_group['id'] = words_1gram_group['Group'].apply(lambda x: conver_dict[x])

fig = px.bar(words_2gram_group, x = '2gram', y = 'Freq', color = 'Satisfaction', facet_row='id', barmode='gro
up')
fig.show()
words_2gram_group.to_csv(os.path.join(base_datapath,'2gram_organizationgroups.csv'))
```



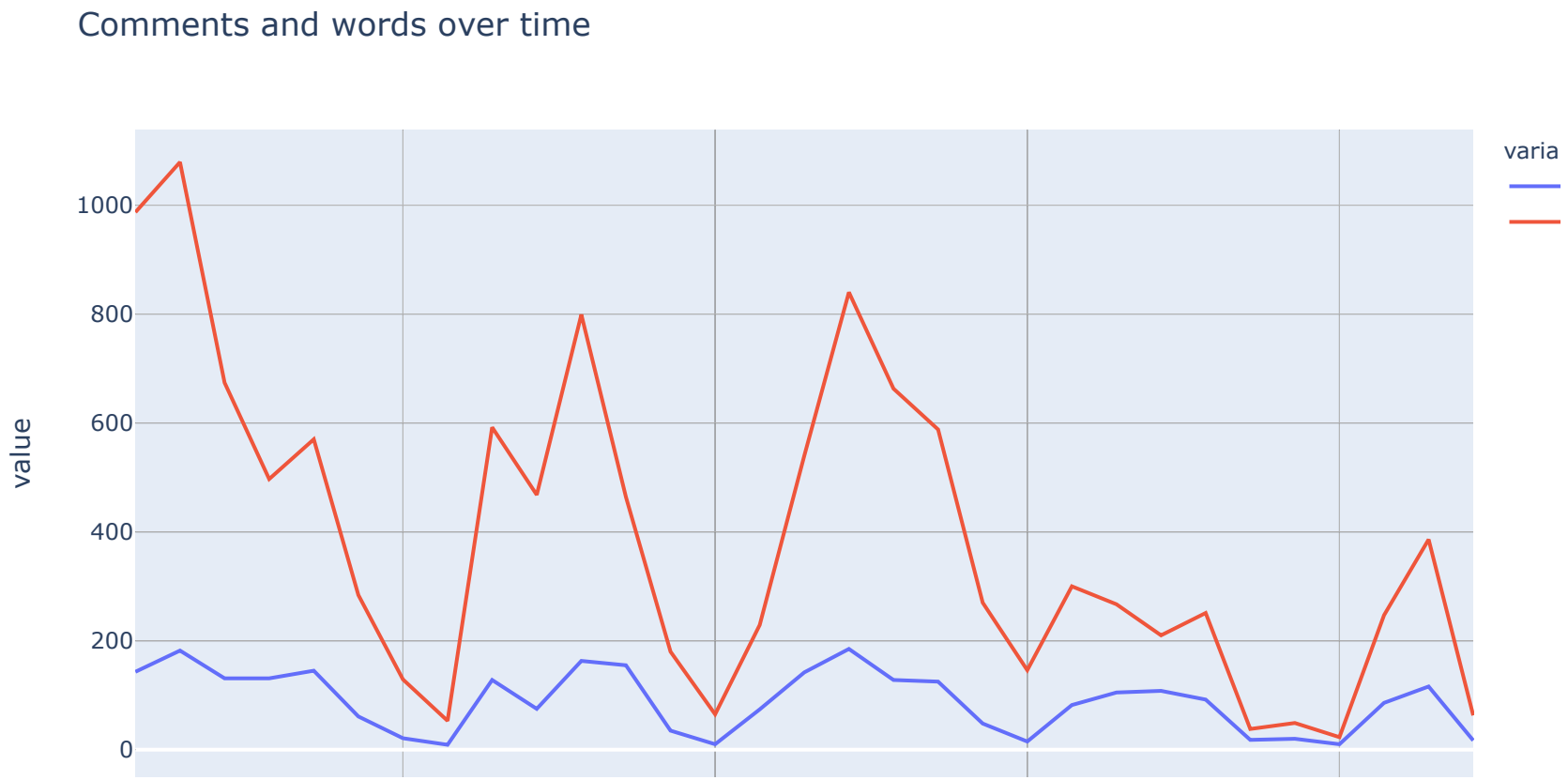
Time analysis

Now we plot the number of comments and words per day in the data satisfaction data.

In [20]:
```
data_satisfaction['datetime'] = pd.to_datetime(data_satisfaction['Survey Date'])
data_satisfaction['words'] = data_satisfaction['processed comment'].apply(lambda x : len(nltk.word_tokenize(x
)))
comments_day = data_satisfaction.groupby(by = data_satisfaction['datetime'].dt.to_period("D"))[['Ticket Id',
'words']].agg({'Ticket Id':'count', 'words':'sum'}).reset_index().rename(columns={'Ticket Id': 'Count of Comm
ents', 'words': 'Count of words'})
fig = px.line(comments_day, y = ['Count of Comments', 'Count of words'], x = comments_day['datetime'].dt.to_t
imestamp(), title = 'Comments and words over time')
fig.show()
```

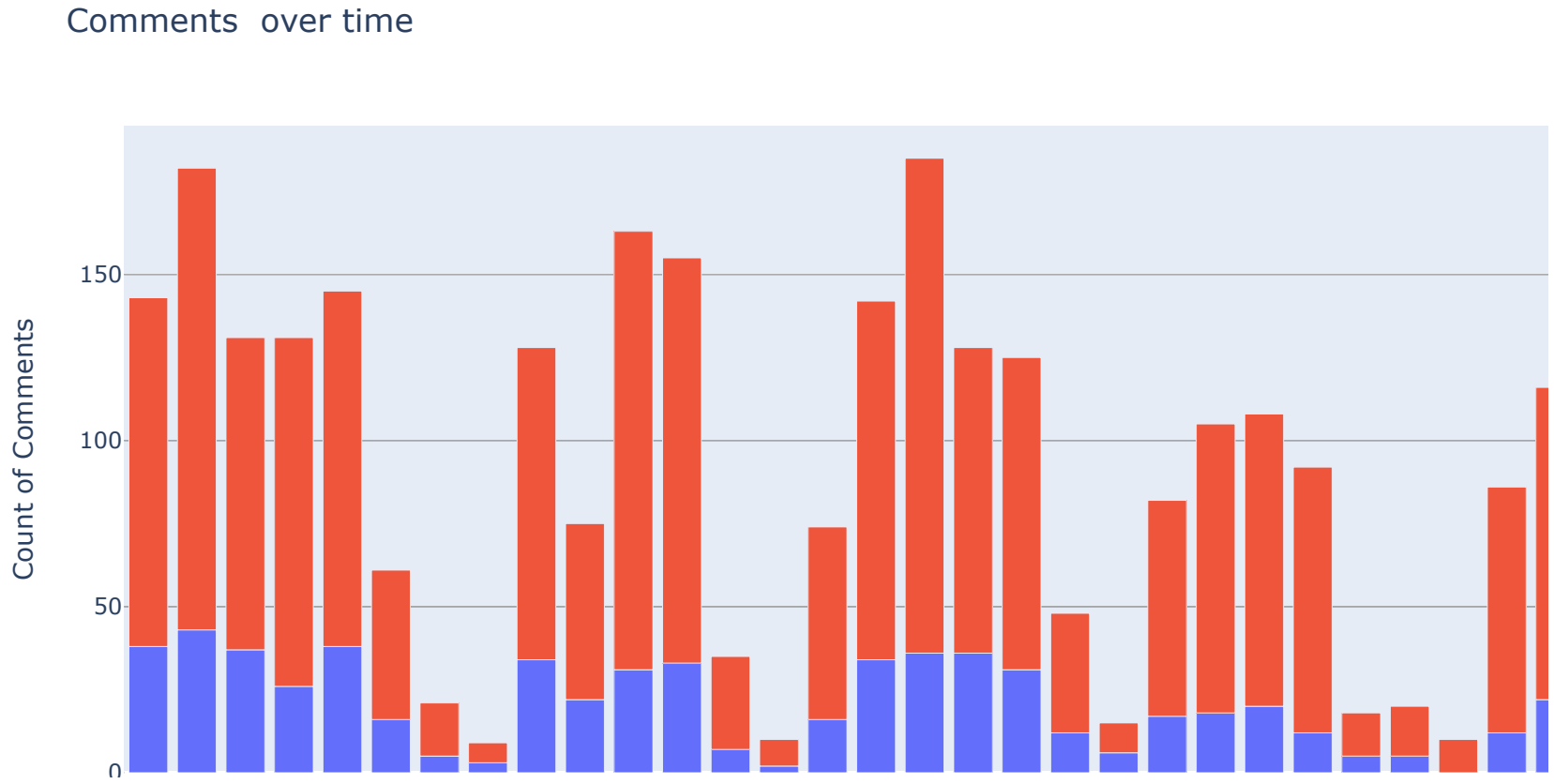
```
C:\Users\henry\Anaconda3\envs\ecogeo_tool\lib\site-packages\pandas\core\arrays\datetimes.py:1102: UserWarnin
g:

Converting to PeriodArray/Index representation will drop timezone information.
```

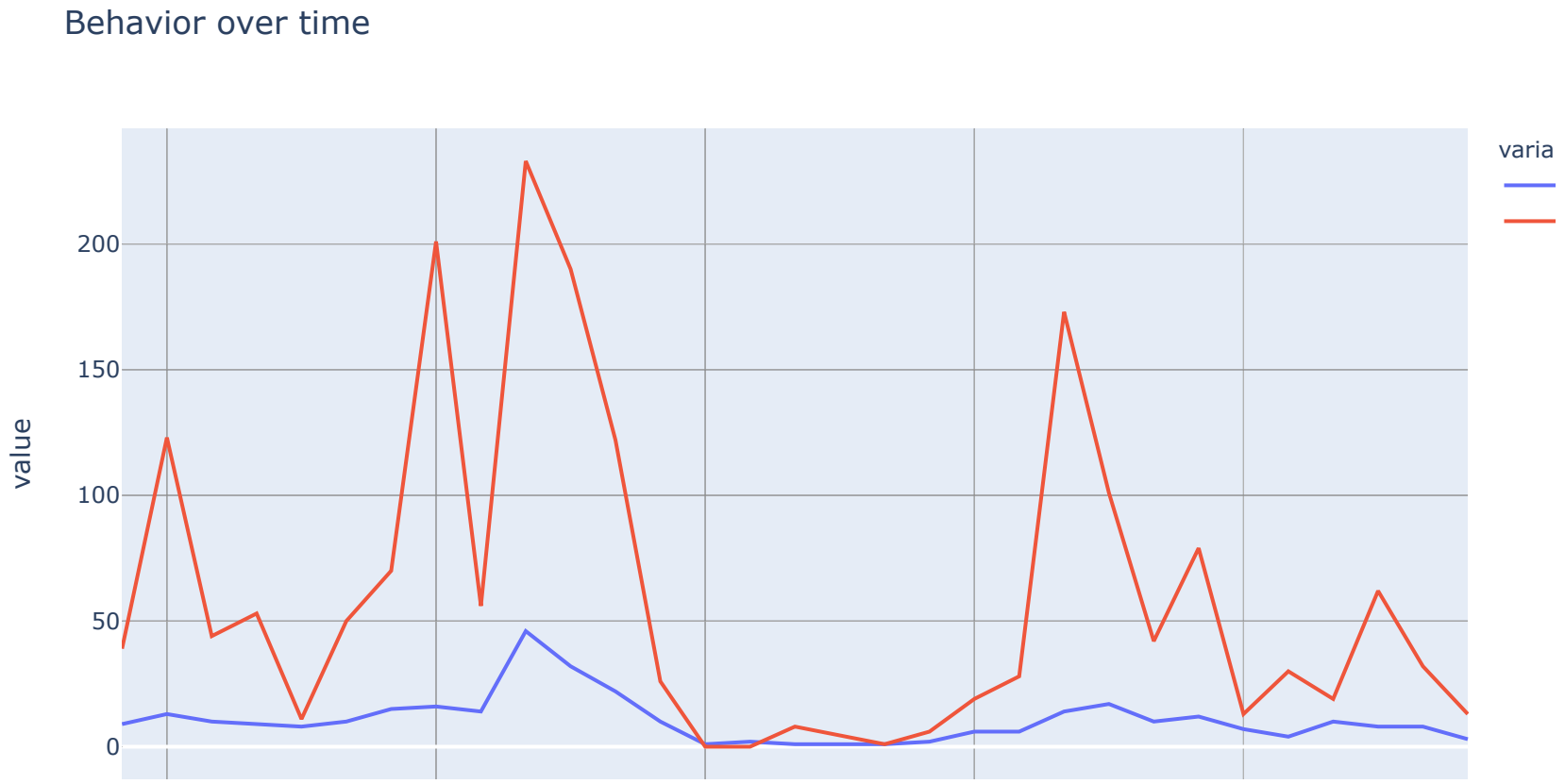


And we plot the good and bad comments per day

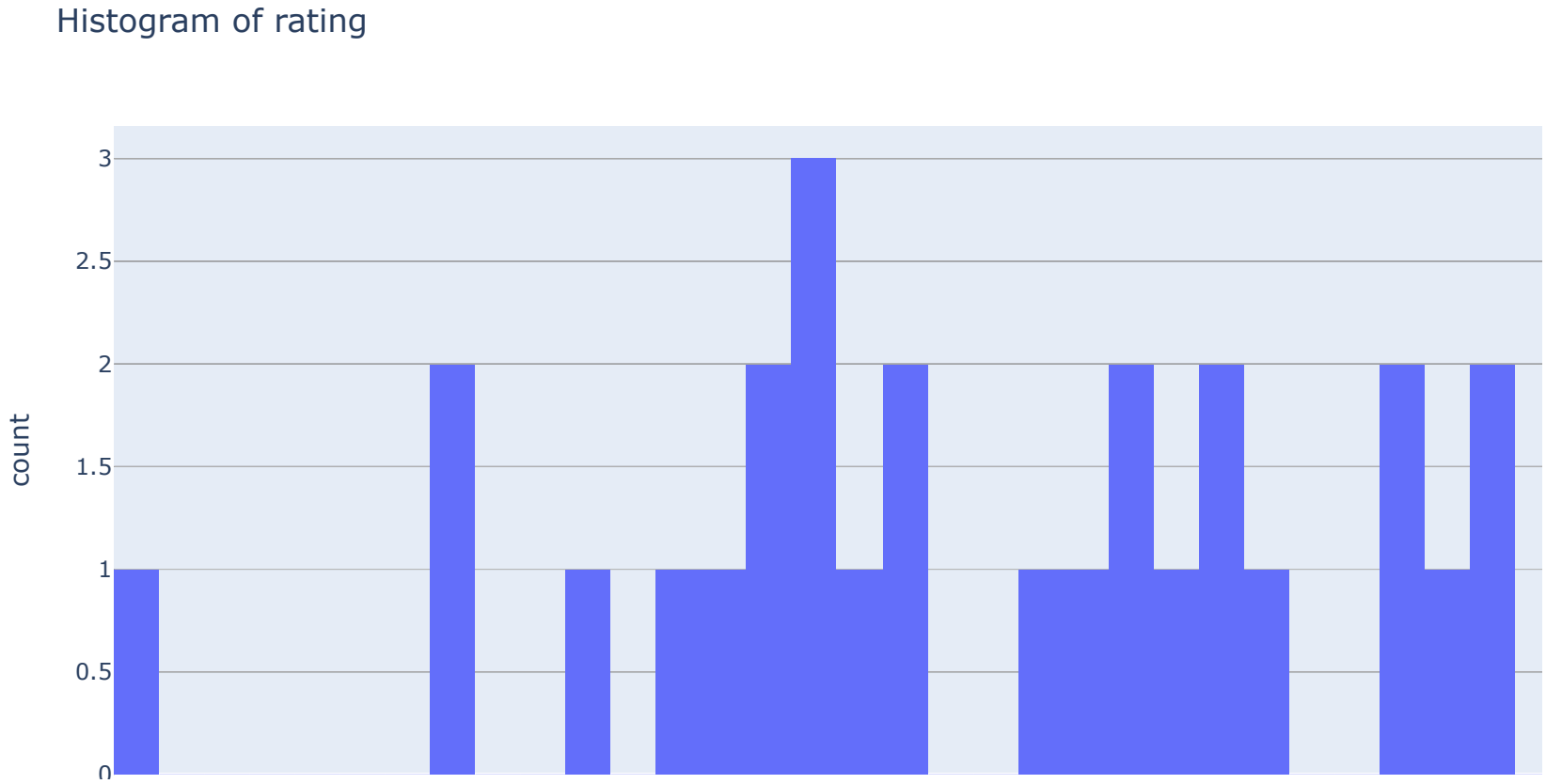
In [21]:
```
comments_day = data_satisfaction.groupby(by = [data_satisfaction['Satisfaction'],data_satisfaction['datetime'
].dt.to_period("D")])[['Ticket Id','words']].agg({'Ticket Id':'count', 'words':'sum'}).reset_index().rename(c
olumns={'Ticket Id': 'Count of Comments', 'words': 'Count of words'})
fig = px.bar(comments_day, y = 'Count of Comments', x = comments_day['datetime'].dt.to_timestamp(), title =
'Comments  over time', color = 'Satisfaction')
fig.show()
```



In [22]:
```
data_responses['datetime'] = pd.to_datetime(data_responses['Response Date'])
data_responses['words'] = data_responses['processed comment'].apply(lambda x : len(nltk.word_tokenize(x)))
comments_day = data_responses.groupby(by = data_responses['datetime'].dt.to_period("H"))[['Unnamed: 0','Rating','words']].agg({'Unnamed: 0':'count', 'Rating' : 'mean','words':'sum'}).reset_index().rename(columns={'Unnamed: 0': 'Count of Comments', 'words': 'Count of words'})
fig = px.line(comments_day, y = ['Count of Comments','Count of words'], x = comments_day['datetime'].dt.to_timestamp(), title = 'Behavior over time')
fig.show()
fig = px.line(comments_day, y = ['Rating'], x = comments_day['datetime'].dt.to_timestamp(), title = 'Rating over time')
fig.show()
fig = px.histogram(comments_day, x = ['Rating'], nbins = 50, title = 'Histogram of rating')
fig.show()
```


Behavior over time
value
200
150
100
50
0
varia


Rating over time
value
10
9
8
7
6
5
4


Histogram of rating
count
3
2.5
2
1.5
1
0.5
0