Análisis de ubicación sin filtrar

May 21, 2020

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm

tweets = pd.read_csv('csv/train.csv', encoding='latin-1')
tweets.head()
```

[1]:
```
   id keyword location                                              text  \
0   1     NaN      NaN  Our Deeds are the Reason of this #earthquake M…
1   4     NaN      NaN             Forest fire near La Ronge Sask. Canada
2   5     NaN      NaN  All residents asked to 'shelter in place' are …
3   6     NaN      NaN  13,000 people receive #wildfires evacuation or…
4   7     NaN      NaN  Just got sent this photo from Ruby #Alaska as …

   target
0       1
1       1
2       1
3       1
4       1
```

Lleno los valores nulos con 'Not Especified' esto denotara que la keyword o la locacion no fueron especificadas en el tweet

[2]:
```
tweets = tweets.fillna(value= 'Not Especified')
```

[3]:
```
tweets['cuantity'] = 1
tweets.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7613 entries, 0 to 7612
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   id        7613 non-null   int64
 1   keyword   7613 non-null   object
 2   location  7613 non-null   object
```

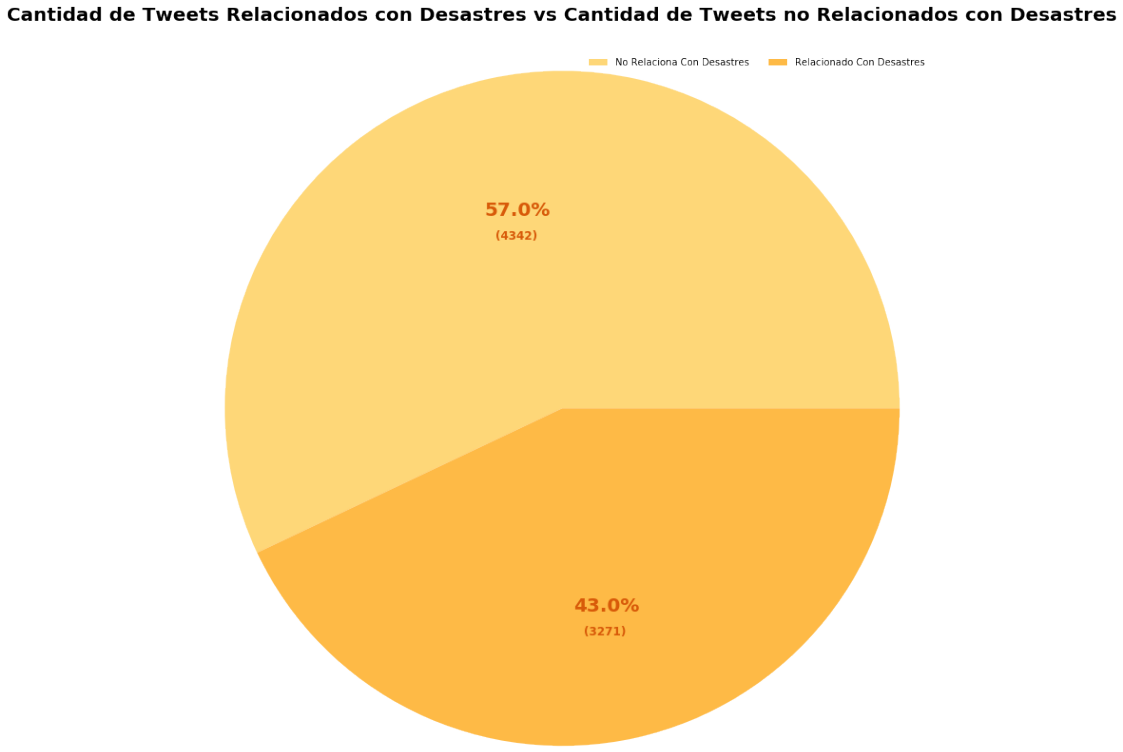
```
 3   text      7613 non-null   object
 4   target    7613 non-null   int64
 5   cuantity  7613 non-null   int64
dtypes: int64(3), object(3)
memory usage: 357.0+ KB
```

[4]:
```
target = tweets[['target', 'cuantity']]
target = target.groupby('target').sum()
target =target.reset_index()
target.head()
```

[4]:

	target	cuantity
0	0	4342
1	1	3271

[5]:
```
cmap = cm.get_cmap('YlOrBr')
saltos = np.linspace(0.3, 0.7, 20)
colores = cmap(saltos)

plt.figure(figsize=(14,14))
pie_chart = plt.subplot()


a,b, autotexts = pie_chart.pie(target['cuantity'], colors= [colores[0],␣
 ↪colores[5]], autopct='%1.1f%%')

for autotext in autotexts:
    autotext.set_color(color=colores[19])
    autotext.set_fontsize(20)
    autotext.set_fontweight('bold')

cantidad = target['cuantity']
pie_chart.text(-0.198, 0.5, '(' +str(cantidad[0])+')', color=colores[19],␣
 ↪fontweight='bold', fontsize=12)
pie_chart.text(0.065, -0.67, '(' +str(cantidad[1])+')', color=colores[19],␣
 ↪fontweight='bold', fontsize=12)
pie_chart.legend(["No Relaciona Con Desastres", "Relacionado Con Desastres"],␣
 ↪frameon=False, loc='best', ncol=2)
plt.title('Cantidad de Tweets Relacionados con Desastres vs Cantidad de Tweets␣
 ↪no Relacionados con Desastres', weight='bold', size=20, pad=30)
plt.axis('equal')
plt.show()
```



[6]:
```
tweets['location_especificada'] = (tweets['location']!='Not Especified')
```

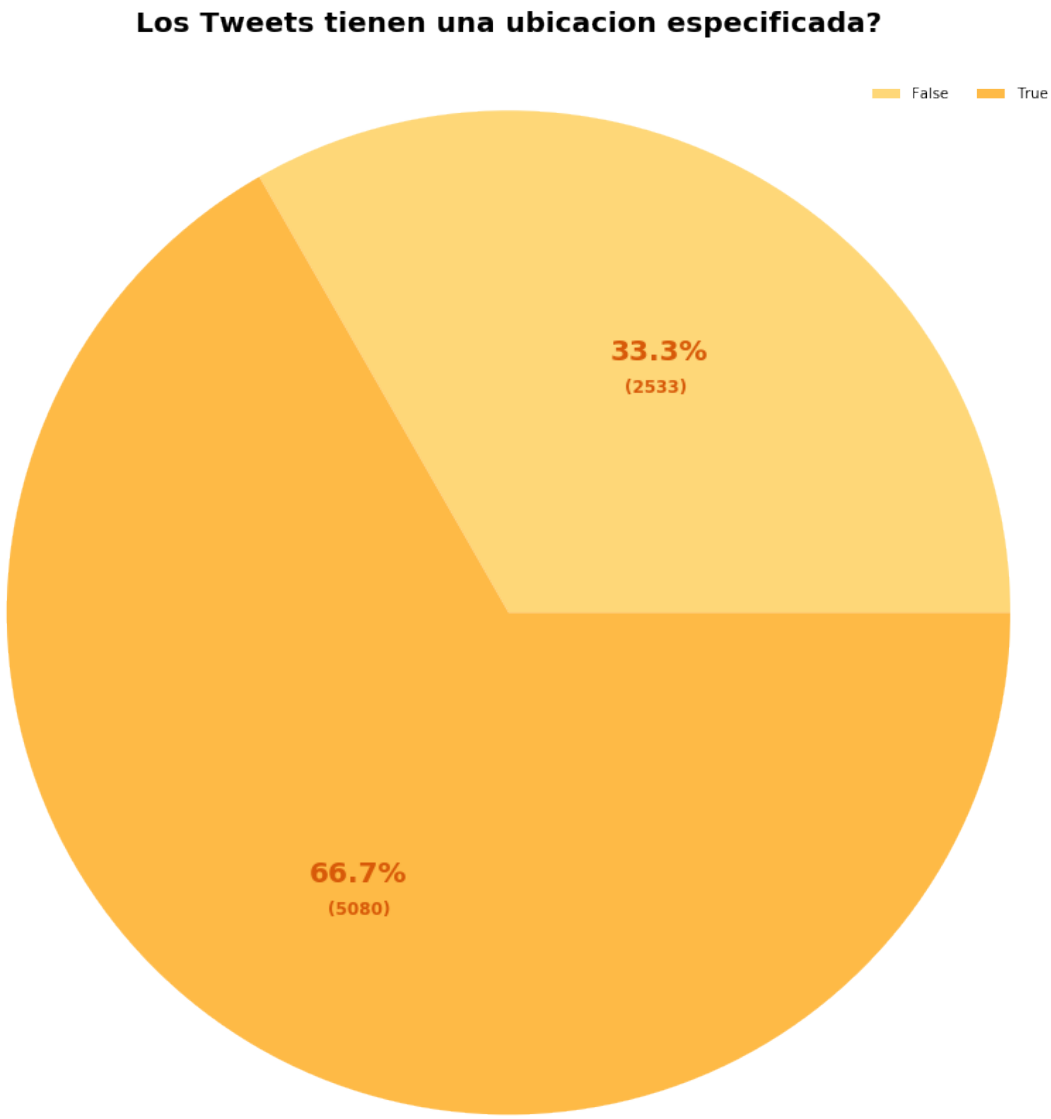
[7]:
```
fue_la_locacion_especificada = tweets[['cuantity', 'location_especificada']]
fue_la_locacion_especificada = fue_la_locacion_especificada.
 ↪groupby('location_especificada').sum()
fue_la_locacion_especificada = fue_la_locacion_especificada.reset_index()
fue_la_locacion_especificada
```

[7]:

	location_especificada	cuantity
0	False	2533
1	True	5080

[8]:
```
cmap = cm.get_cmap('YlOrBr')
saltos = np.linspace(0.3, 0.7, 20)
colores = cmap(saltos)

plt.figure(figsize=(14,14))
pie_chart = plt.subplot()


a, b,autotexts= pie_chart.pie(fue_la_locacion_especificada['cuantity'], colors=␣
 ↪[colores[0], colores[5]], autopct='%1.1f%%')
```

```
for autotext in autotexts:
    autotext.set_color(color=colores[19])
    autotext.set_fontsize(20)
    autotext.set_fontweight('bold')
cantidad = fue_la_locacion_especificada['cuantity']
pie_chart.text(0.23, 0.44, '(' +str(cantidad[0])+')', color=colores[19],␣
 ↪fontweight='bold', fontsize=12)
pie_chart.text(-0.36, -0.6, '(' +str(cantidad[1])+')', color=colores[19],␣
 ↪fontweight='bold', fontsize=12)
pie_chart.legend(fue_la_locacion_especificada['location_especificada'],␣
 ↪frameon=False, loc='best', ncol=2)
plt.title('Los Tweets tienen una ubicacion especificada?', weight='bold',␣
 ↪size=20, pad=30)
plt.axis('equal')
plt.show()
```



Creo una nueva columna en el df principal con la longitud del Tweet para relacionarlo posteriormente con la locacion

[9]:
```
tweets['text lenght'] = tweets['text'].str.len()
```

[10]:
```
tweets_by_loc = tweets[['location', 'target', 'text lenght', 'cuantity']]
tweets_by_loc = tweets_by_loc.groupby('location').sum()
tweets_by_loc.sample(n=10)
```

[10]:

location	target	text lenght	cuantity
Ellensburg to Spokane	1	105	1

```
Central Coast, California            1          140         1
online                               1          236         2
Maharashtra                          1          136         1
Milky Way galaxy                     0          119         1
Hattiesburg, MS                      0           72         1
Auckland, New Zealand                0          260         2
Ã Ã T: 27.9136024,-81.6078532       0          128         1
Top Secret                           1          126         1
Viejo                                0           45         1
```

[11]:
```
tweets_by_loc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3342 entries,    to Ã¥Ã¸\_(?)_/Ã¥Ã¸
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   target       3342 non-null   int64
 1   text lenght  3342 non-null   int64
 2   cuantity     3342 non-null   int64
dtypes: int64(3)
memory usage: 104.4+ KB
```

[12]:
```
#Calculo el factor de veracidad para cada tweet y lor ordeno por dicho factor␣
 ↪en orden descendiente
tweets_by_loc['Veracidad'] = tweets_by_loc['target'] / tweets_by_loc['cuantity']
tweets_by_loc = tweets_by_loc.sort_values('Veracidad', ascending = False)
tweets_by_loc.tail(10)
```

[12]:

location	target	text lenght	cuantity	Veracidad
Pittsburgh PA	0	136	1	0.0
Pittsburgh	0	27	1	0.0
Pittsburgh	0	141	2	0.0
Honolulu,Hawaii	0	48	1	0.0
BestCoast	0	114	1	0.0
Between Dire and Radiant	0	42	1	0.0
Between the worlds	0	70	1	0.0
Photo : Blue Mountains	0	139	1	0.0
Hooters on Peachtree	0	56	1	0.0
Edinburgh	0	482	4	0.0

[13]:
```
#Me quedo solo con las locaciones que tienen mas de 10 Tweets
does_the_location_has_more_than_10_tweets = (tweets_by_loc['cuantity'] > 10)
locations_with_more_than_10_tweets =␣
 ↪tweets_by_loc[does_the_location_has_more_than_10_tweets]
del locations_with_more_than_10_tweets['target']
```

```
#lar ordeno descendientemente por su factor de veracidad y reseteo el index a␣
 ↪fines de poder graficar
locations_with_more_than_10_tweets = locations_with_more_than_10_tweets.
 ↪sort_values('Veracidad', ascending =True)
locations_with_more_than_10_tweets = locations_with_more_than_10_tweets.
 ↪reset_index()
locations_with_more_than_10_tweets.head(10)
```

[13]:

	location	text lenght	cuantity	Veracidad
0	NYC	1406	12	0.166667
1	Everywhere	1357	15	0.200000
2	Florida	1540	14	0.214286
3	New York	9322	71	0.225352
4	Kenya	2635	20	0.250000
5	United Kingdom	1400	14	0.285714
6	Los Angeles, CA	2971	26	0.307692
7	London	4833	45	0.355556
8	Chicago	1241	11	0.363636
9	Seattle	1398	11	0.363636

[14]:
```
tweets_by_loc = tweets_by_loc.reset_index()
tweets_by_loc.tail()
```

[14]:

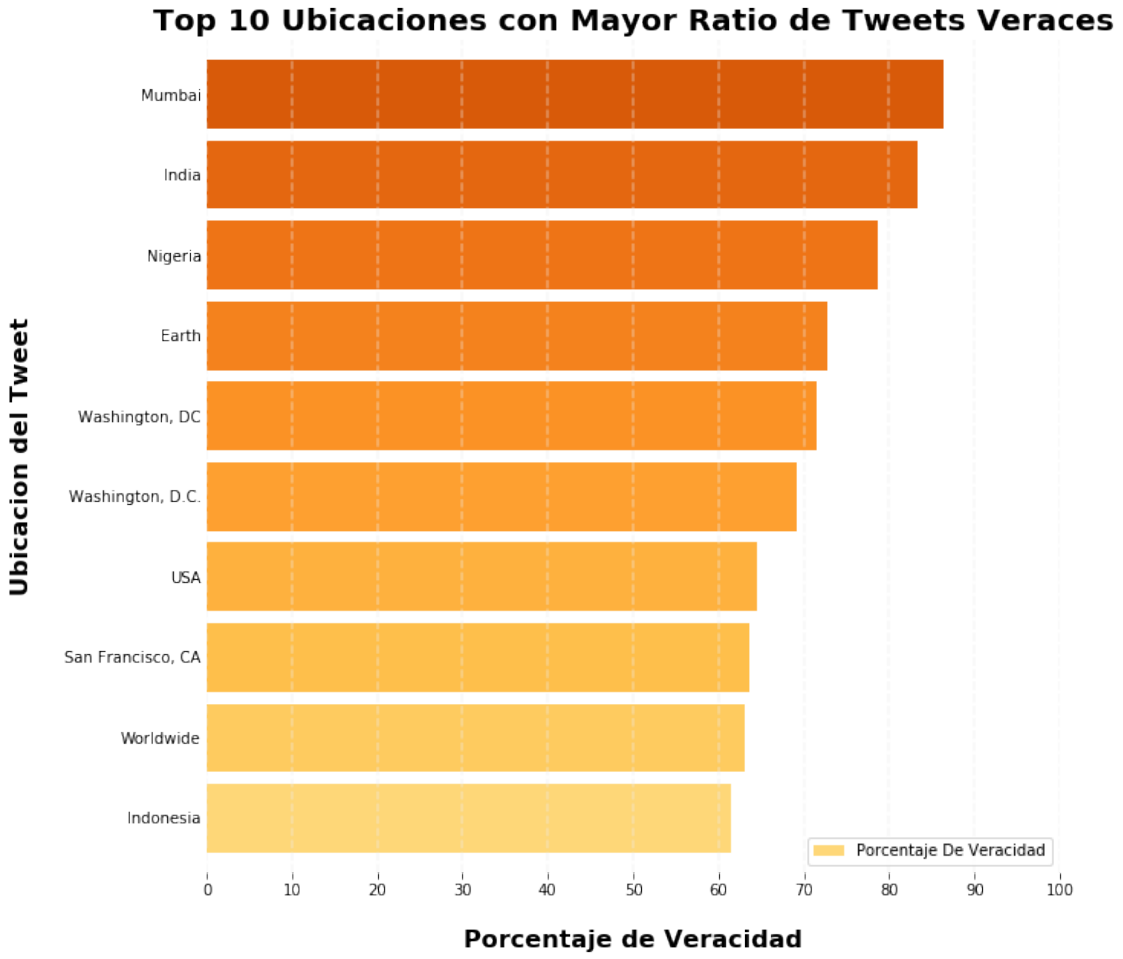
	location	target	text lenght	cuantity	Veracidad
3337	Between Dire and Radiant	0	42	1	0.0
3338	Between the worlds	0	70	1	0.0
3339	Photo : Blue Mountains	0	139	1	0.0
3340	Hooters on Peachtree	0	56	1	0.0
3341	Edinburgh	0	482	4	0.0

[15]:
```
#multiplico el factor de veracidad por 100 a fines de poder graficar, ahora es␣
 ↪un porcentaje de veracidad
locations_with_more_than_10_tweets['Porcentaje De Veracidad'] =␣
 ↪locations_with_more_than_10_tweets['Veracidad'] * 100
```

[16]:
```
saltos = np.linspace(0.3, 0.7, 10)
colores = (cm.get_cmap('YlOrBr'))(saltos)
locations_with_more_than_10_tweets_plot = locations_with_more_than_10_tweets.
 ↪tail(10).plot(kind='barh', y ='Porcentaje De Veracidad', x = 'location',␣
 ↪figsize=(10,10), color=colores, width=0.85)

plt.xticks(np.arange(0, 100+1, 10.0))
plt.tick_params(axis='y', length=0)

locations_with_more_than_10_tweets_plot.spines['right'].set_visible(False)
locations_with_more_than_10_tweets_plot.spines['top'].set_visible(False)
locations_with_more_than_10_tweets_plot.spines['left'].set_visible(False)
```

```
locations_with_more_than_10_tweets_plot.spines['bottom'].set_visible(False)

lineas = locations_with_more_than_10_tweets_plot.get_xticks()
for i in lineas:
      locations_with_more_than_10_tweets_plot.axvline(x=i, linestyle='--',␣
 ↪alpha=0.4, color='#eeeeee')

locations_with_more_than_10_tweets_plot.set_xlabel("Porcentaje de Veracidad",␣
 ↪labelpad=20, weight='bold', size=16)
locations_with_more_than_10_tweets_plot.set_ylabel("Ubicacion del Tweet",␣
 ↪labelpad=20, weight='bold', size=16)
locations_with_more_than_10_tweets_plot.set_title("Top 10 Ubicaciones con Mayor␣
 ↪Ratio de Tweets Veraces", weight='bold', size=20)
```

[16]: Text(0.5, 1.0, 'Top 10 Ubicaciones con Mayor Ratio de Tweets Veraces')



[17]:
```
tweets_by_loc.sort_values('target', ascending = False).head(11)
```

[17]:

	location	target	text lenght	cuantity	Veracidad
1429	Not Especified	1075	250161	2533	0.424398
1265	USA	67	11321	104	0.644231
1283	United States	27	5029	50	0.540000
1209	Nigeria	22	3387	28	0.785714
1203	India	20	2481	24	0.833333
1199	Mumbai	19	2755	22	0.863636
1498	New York	16	9322	71	0.225352
1440	London	16	4833	45	0.355556
1278	UK	16	3146	27	0.592593
1224	Washington, DC	15	2434	21	0.714286
1423	Canada	13	3082	29	0.448276

[18]:
```
#me quedo con las 10 locaciones con mayor cantidad de Tweets verdaderos
top_10_tweets_by_loc = (tweets_by_loc.loc[:,'target'] >= 13)
top_10 = tweets_by_loc.loc[top_10_tweets_by_loc]
top_10.sort_values('target', ascending = False).head(15)
```

[18]:

	location	target	text lenght	cuantity	Veracidad
1429	Not Especified	1075	250161	2533	0.424398
1265	USA	67	11321	104	0.644231
1283	United States	27	5029	50	0.540000
1209	Nigeria	22	3387	28	0.785714
1203	India	20	2481	24	0.833333
1199	Mumbai	19	2755	22	0.863636
1278	UK	16	3146	27	0.592593
1440	London	16	4833	45	0.355556
1498	New York	16	9322	71	0.225352
1224	Washington, DC	15	2434	21	0.714286
1423	Canada	13	3082	29	0.448276

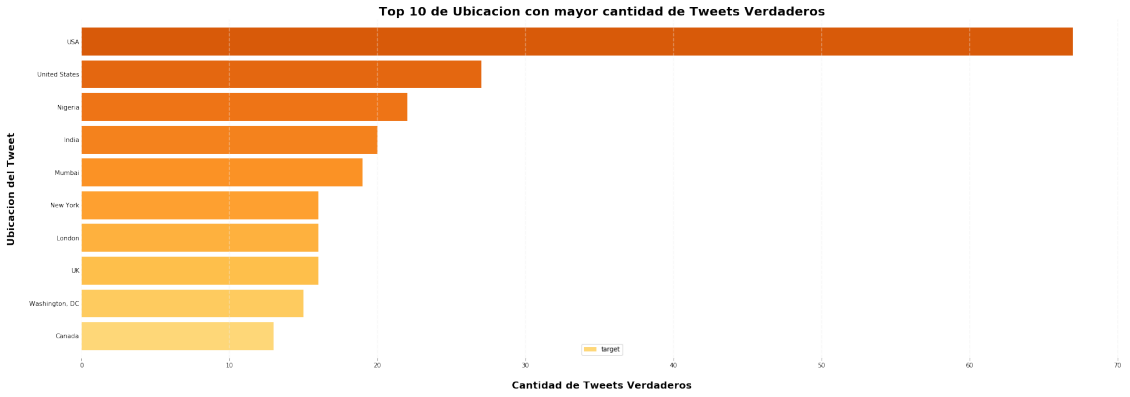
[19]:
```
specified_location = (tweets_by_loc.loc[:,'location'] != 'Not Especified')
top_10 = top_10.loc[specified_location]
top_10.head(15)
```

[19]:

	location	target	text lenght	cuantity	Veracidad
1199	Mumbai	19	2755	22	0.863636
1203	India	20	2481	24	0.833333
1209	Nigeria	22	3387	28	0.785714
1224	Washington, DC	15	2434	21	0.714286
1265	USA	67	11321	104	0.644231
1278	UK	16	3146	27	0.592593
1283	United States	27	5029	50	0.540000
1423	Canada	13	3082	29	0.448276
1440	London	16	4833	45	0.355556
1498	New York	16	9322	71	0.225352

[20]:
```
saltos = np.linspace(0.3, 0.7, 10)
colores = (cm.get_cmap('YlOrBr'))(saltos)
top_10 = top_10.sort_values('target')
top10_plot = top_10.plot(kind='barh', y ='target', x = 'location',␣
↪figsize=(30,10), color=colores, width=0.85)

plt.xticks(np.arange(0, 71, 10.0))
plt.tick_params(axis='y', length=0)

top10_plot.spines['right'].set_visible(False)
top10_plot.spines['top'].set_visible(False)
top10_plot.spines['left'].set_visible(False)
top10_plot.spines['bottom'].set_visible(False)

lineas = top10_plot.get_xticks()
for i in lineas:
      top10_plot.axvline(x=i, linestyle='--', alpha=0.4, color='#eeeeee')

top10_plot.set_xlabel("Cantidad de Tweets Verdaderos", labelpad=20,␣
↪weight='bold', size=16)
top10_plot.set_ylabel("Ubicacion del Tweet", labelpad=20, weight='bold',␣
↪size=16)
top10_plot.set_title("Top 10 de Ubicacion con mayor cantidad de Tweets␣
↪Verdaderos", weight='bold', size=20)
```

[20]:
```
Text(0.5, 1.0, 'Top 10 de Ubicacion con mayor cantidad de Tweets Verdaderos')
```



[21]:
```
#Creo una lista que para cada elemento del df responde a: es el elemento falso?
is_false = []
for item in tweets['target']:
    is_false.append(1 if item == 0 else 0)
```

[22]:
```
false_tweets = tweets[['location', 'cuantity']]
false_tweets['is false'] = is_false
false_tweets = false_tweets.groupby('location').sum()
false_tweets.sample(n=15)
```

```
/home/riedel/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[22]:

location	cuantity	is false
Manhattan	1	1
?????	4	2
Davao City	1	1
Manchester, UK	2	1
In my own world!!!	1	0
Fort Worth, Texas	1	0
ss	10	9
BodÃ Ã¼, Norge	1	1
beijing .China	1	1
Tafekop Ga-Matsepe	1	0
Chester Football Club	1	1
Paradise City	1	1
Planet Eyal, Shandral System	1	0
Washington, Krasnodar (Russia)	1	0
County Durham, United Kingdom	1	1

[23]:
```
#calculo el porcentaje de tweets que son falsos del total de tweets para cada␣
↪locacion
false_tweets['Porcentaje de Falsedad'] = (false_tweets['is false'] /␣
↪false_tweets['cuantity']) *100
false_tweets.head(10)
```

[23]:

location	cuantity	is false	Porcentaje de Falsedad
	1	0	0.0
Glasgow	1	1	100.0
Melbourne, Australia	1	1	100.0
News	1	0	0.0
Ã¥_	1	1	100.0
45Ã¥Â¡ 5'12.53N 14Ã¥Â¡ 7'24.93E	1	1	100.0
616 Â Ã Â¢ Kentwood , MI	1	0	0.0

```
? ??????? ? ( ?? Ã¥Â¡ ? ? ? Ã¥Â¡)          1          0                   0.0
?currently writing a book?                  1          1                 100.0
Alberta                                     1          0                   0.0
```

[24]:
```
#de las locaciones que tengoo me quedo solo con las que tienen mas de 10 Tweets
has_more_tweets_than_10_tweets = (false_tweets['cuantity'] > 10)
locations_with_more_than_10_false_tweets =␣
↪false_tweets[has_more_tweets_than_10_tweets]
locations_with_more_than_10_false_tweets =␣
↪locations_with_more_than_10_false_tweets.sort_values('Porcentaje de␣
↪Falsedad', ascending =True)
locations_with_more_than_10_false_tweets =␣
↪locations_with_more_than_10_false_tweets.reset_index()
locations_with_more_than_10_false_tweets.head(10)
```

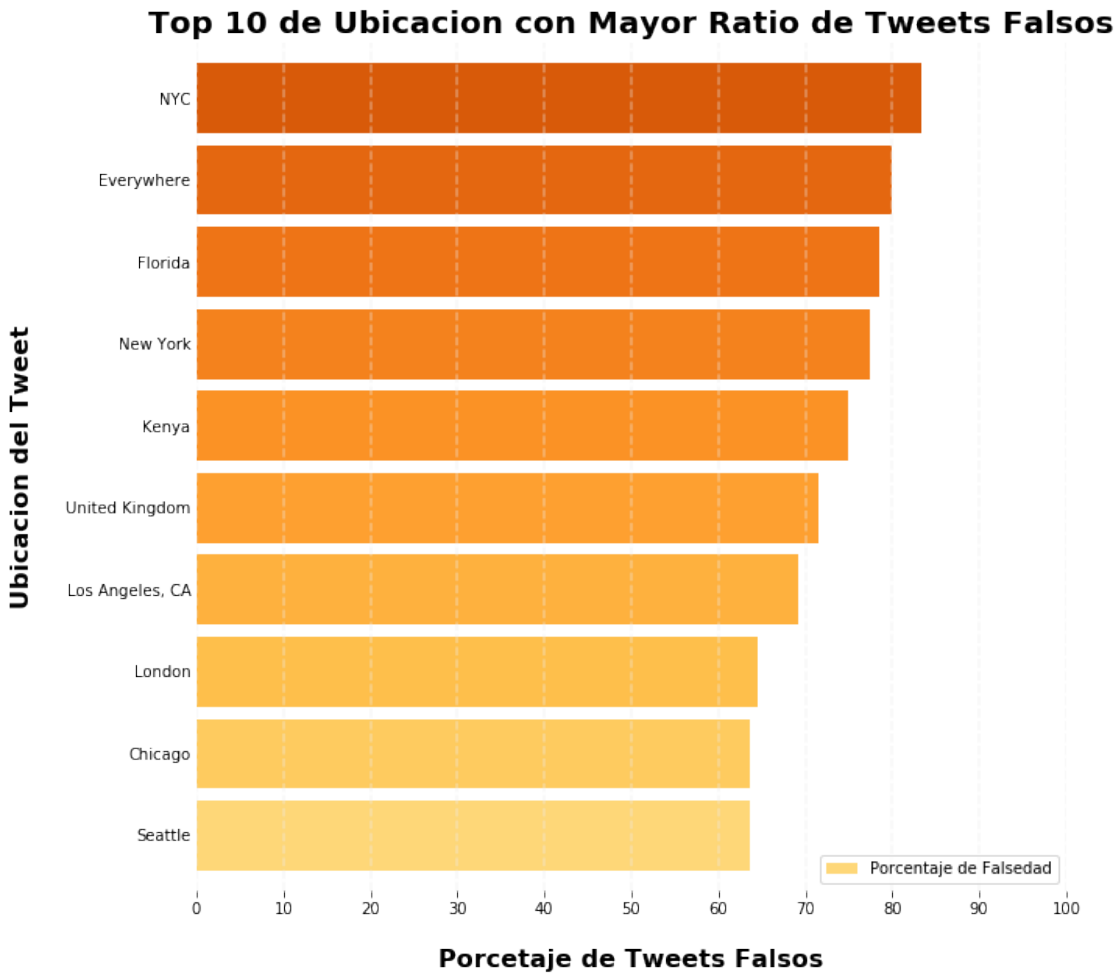
[24]:

	location	cuantity	is false	Porcentaje de Falsedad
0	Mumbai	22	3	13.636364
1	India	24	4	16.666667
2	Nigeria	28	6	21.428571
3	Earth	11	3	27.272727
4	Washington, DC	21	6	28.571429
5	Washington, D.C.	13	4	30.769231
6	USA	104	37	35.576923
7	San Francisco, CA	11	4	36.363636
8	Worldwide	19	7	36.842105
9	Indonesia	13	5	38.461538

[25]:
```
saltos = np.linspace(0.3, 0.7, 10)
colores = (cm.get_cmap('YlOrBr'))(saltos)
top10_plot = locations_with_more_than_10_false_tweets.tail(10).
↪plot(kind='barh', y ='Porcentaje de Falsedad', x = 'location',␣
↪figsize=(10,10), color=colores, width=0.85)

plt.xticks(np.arange(0, 101, 10.0))
plt.tick_params(axis='y', length=0)

top10_plot.spines['right'].set_visible(False)
top10_plot.spines['top'].set_visible(False)
top10_plot.spines['left'].set_visible(False)
top10_plot.spines['bottom'].set_visible(False)

lineas = top10_plot.get_xticks()
for i in lineas:
      top10_plot.axvline(x=i, linestyle='--', alpha=0.4, color='#eeeeee')

top10_plot.set_xlabel("Porcetaje de Tweets Falsos", labelpad=20, weight='bold',␣
↪size=16)
```

```
top10_plot.set_ylabel("Ubicacion del Tweet", labelpad=20, weight='bold',␣
 ↪size=16)
top10_plot.set_title("Top 10 de Ubicacion con Mayor Ratio de Tweets Falsos",␣
 ↪weight='bold', size=20)
```

[25]: Text(0.5, 1.0, 'Top 10 de Ubicacion con Mayor Ratio de Tweets Falsos')

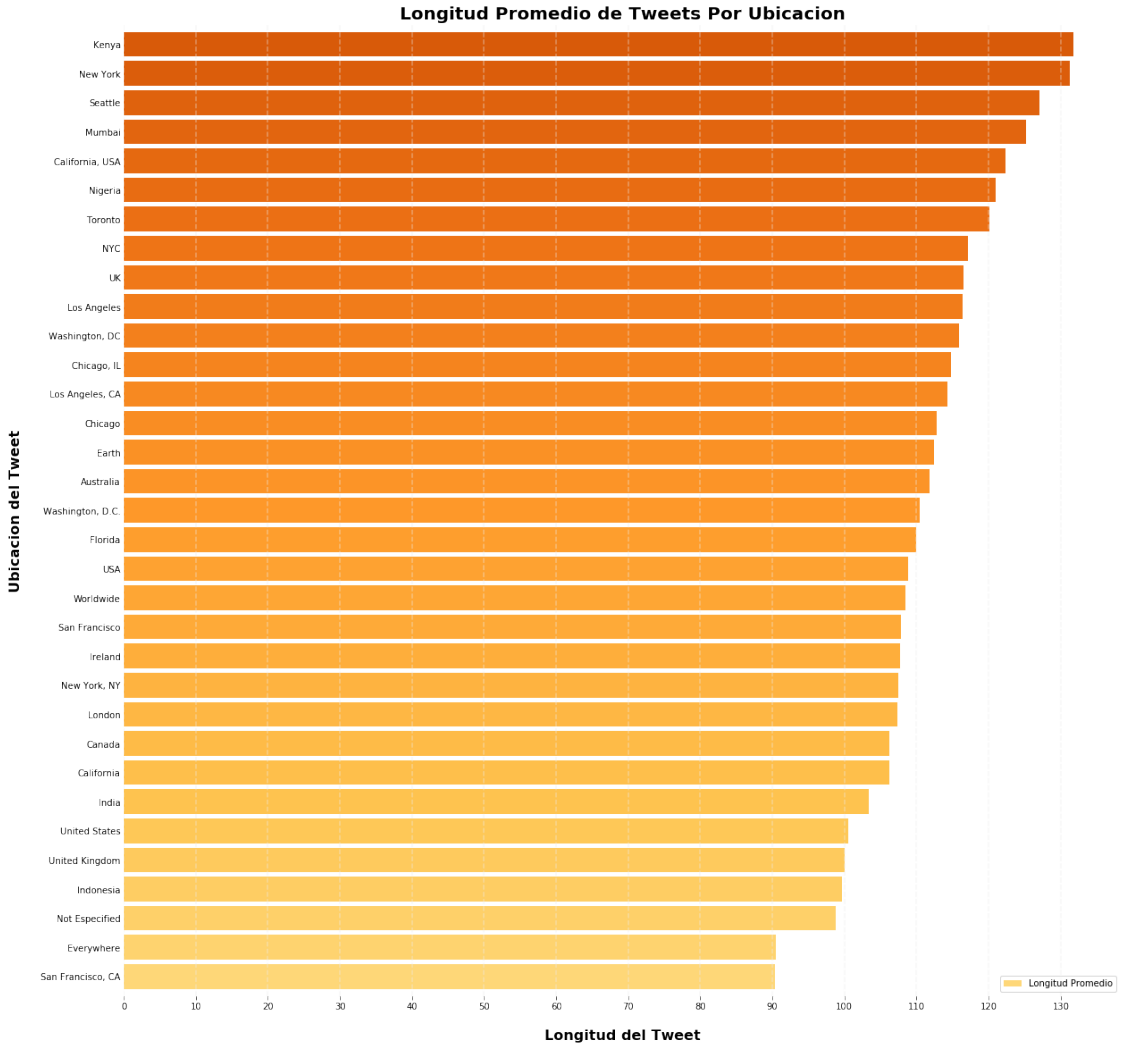


[26]:
```
#De las locaciones con mas de 10 tweets, calculo cual es el promedio de␣
 ↪longitud del texto para cada locacion
locations_with_more_than_10_tweets['Longitud Promedio'] =␣
 ↪locations_with_more_than_10_tweets['text lenght'] /␣
 ↪locations_with_more_than_10_tweets['cuantity']
locations_with_more_than_10_tweets = locations_with_more_than_10_tweets.
 ↪sort_values('Longitud Promedio')
locations_with_more_than_10_tweets.head()
```

[26]:
```
              location  text lenght  cuantity  Veracidad  \
25  San Francisco, CA          994        11   0.636364
1           Everywhere         1357        15   0.200000
11     Not Especified       250161      2533   0.424398
23           Indonesia         1296        13   0.615385
5      United Kingdom          1400        14   0.285714

    Porcentaje De Veracidad  Longitud Promedio
25                63.636364          90.363636
1                 20.000000          90.466667
11                42.439795          98.760758
23                61.538462          99.692308
5                 28.571429         100.000000
```

[27]:
```
saltos = np.linspace(0.3, 0.7, 33)
colores = (cm.get_cmap('YlOrBr'))(saltos)
locations_with_more_than_10_tweets_plot = locations_with_more_than_10_tweets.
 ↪plot(kind='barh', y ='Longitud Promedio', x = 'location', figsize=(20,20),␣
 ↪color=colores, width=0.85)

plt.xticks(np.arange(0, 130+1, 10.0))
plt.tick_params(axis='y', length=0)

locations_with_more_than_10_tweets_plot.spines['right'].set_visible(False)
locations_with_more_than_10_tweets_plot.spines['top'].set_visible(False)
locations_with_more_than_10_tweets_plot.spines['left'].set_visible(False)
locations_with_more_than_10_tweets_plot.spines['bottom'].set_visible(False)

lineas = locations_with_more_than_10_tweets_plot.get_xticks()
for i in lineas:
      locations_with_more_than_10_tweets_plot.axvline(x=i, linestyle='--',␣
 ↪alpha=0.4, color='#eeeeee')

locations_with_more_than_10_tweets_plot.set_xlabel("Longitud del Tweet",␣
 ↪labelpad=20, weight='bold', size=16)
locations_with_more_than_10_tweets_plot.set_ylabel("Ubicacion del Tweet",␣
 ↪labelpad=20, weight='bold', size=16)
locations_with_more_than_10_tweets_plot.set_title("Longitud Promedio de Tweets␣
 ↪Por Ubicacion", weight='bold', size=20)
```

[27]:
```
Text(0.5, 1.0, 'Longitud Promedio de Tweets Por Ubicacion')
```



[28]:
```
#Creo un df en donde la columna de cantidad me indica las veces que aparecio␣
 ↪cierta keyword y la ordeno descendentemente por las keywords que mas salieron
keywords_mas_usadas = tweets.groupby('keyword').sum().sort_values('cuantity',␣
 ↪ascending = False).head(30)
keywords_mas_usadas = keywords_mas_usadas.reset_index()
keywords_mas_usadas.head()
```

[28]:

	keyword	id	target	cuantity	location_especificada	\
0	Not Especified	326326	42	61	0.0	
1	fatalities	233406	26	45	32.0	
2	deluge	133465	6	42	29.0	
3	armageddon	19748	5	42	32.0	
4	sinking	357380	8	41	23.0	

	text lenght

```
0         4520
1         4884
2         4682
3         4301
4         4268
```

[29]:
```
#creo una serie en la que me quedo solo con las 30 keywords mas usadas
keywords_mas_usadas_serie = pd.Series(keywords_mas_usadas['keyword'].values)
keywords_mas_usadas_serie
```

[29]:
```
0     Not Especified
1          fatalities
2              deluge
3          armageddon
4             sinking
5              damage
6                harm
7         body%20bags
8            evacuate
9                fear
10           outbreak
11              siren
12            twister
13          windstorm
14           collided
15           sinkhole
16               sunk
17           hellfire
18             weapon
19            weapons
20             famine
21          explosion
22          whirlwind
23         earthquake
24         derailment
25           wreckage
26          collision
27             flames
28            wrecked
29          ambulance
dtype: object
```

[30]:
```
cantidad_de_tweets_por_locacion = tweets.groupby('location').sum().
 ↪sort_values('cuantity', ascending = False)
tiene_mas_de_10_tweets = (cantidad_de_tweets_por_locacion['cuantity'] > 10)
locaciones_con_mas_de_10_tweets =␣
 ↪cantidad_de_tweets_por_locacion[tiene_mas_de_10_tweets]
```

```
locaciones_con_mas_de_10_tweets = locaciones_con_mas_de_10_tweets.reset_index()
locaciones_con_mas_de_10_tweets.head()
```

[30]:

	location	id	target	cuantity	location_especificada	\
0	Not Especified	13961318	1075	2533	0.0	
1	USA	618271	67	104	104.0	
2	New York	301155	16	71	71.0	
3	United States	304076	27	50	50.0	
4	London	248126	16	45	45.0	

	text lenght
0	250161
1	11321
2	9322
3	5029
4	4833

[31]:
```
#creo un df llamado scatter (para hacer un scatter plot) donde me quedo solo␣
 ↪con los tweets que usan las 30 keywords mas usadas
does_tweet_has_frecuent_keyword = tweets['keyword'].
 ↪isin(keywords_mas_usadas_serie)
scatter = tweets[does_tweet_has_frecuent_keyword]
scatter['keyword'].value_counts()
```

[31]:
```
Not Especified    61
fatalities        45
deluge            42
armageddon        42
harm              41
body%20bags       41
sinking           41
damage            41
siren             40
outbreak          40
twister           40
windstorm         40
evacuate          40
collided          40
fear              40
weapons           39
wreckage          39
famine            39
flames            39
derailment        39
collision         39
wrecked           39
weapon            39
```

```
sunk              39
sinkhole          39
whirlwind         39
earthquake        39
hellfire          39
explosion         39
ambulance         38
Name: keyword, dtype: int64
```

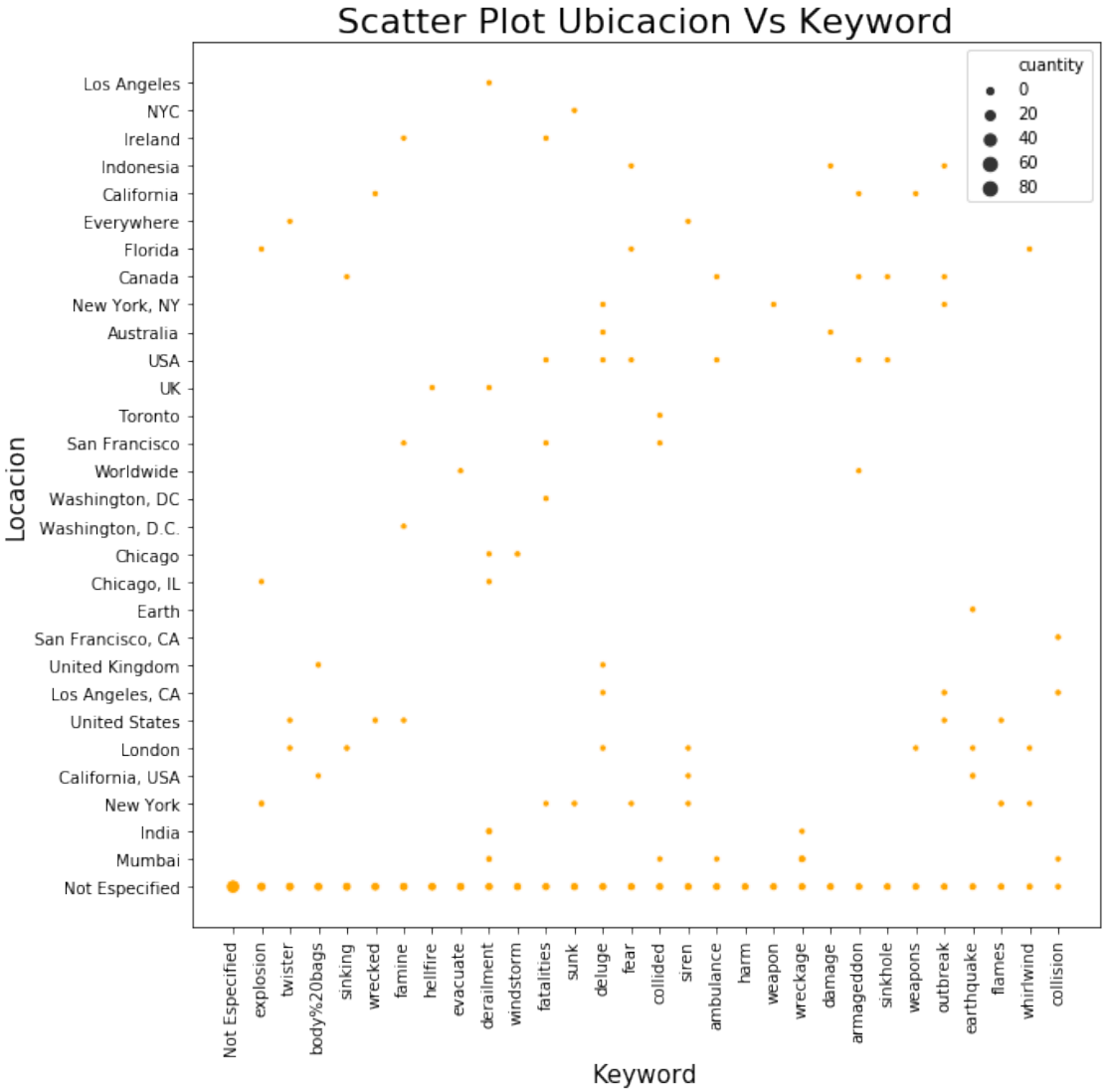
[32]:
```
#de los Tweets que tienen las keywords mas usadas me quedo solo con aquellos␣
 ↪que tienen mas de 10 tweets
does_location_has_more_than_10_tweets = scatter['location'].
 ↪isin(locaciones_con_mas_de_10_tweets['location'])
scatter = scatter[does_location_has_more_than_10_tweets]
del scatter['id']
del scatter['text lenght']
del scatter['target']
scatter.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 545 entries, 0 to 7612
Data columns (total 5 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   keyword                545 non-null    object
 1   location               545 non-null    object
 2   text                   545 non-null    object
 3   cuantity               545 non-null    int64
 4   location_especificada  545 non-null    bool
dtypes: bool(1), int64(1), object(3)
memory usage: 21.8+ KB
```

[33]:
```
#los agrupo por location y keyword y sumo, con lo que consigo saber que tanto␣
 ↪aparece cada par (locacion, keyword)
scatter = scatter.groupby(['location', 'keyword']).sum().
 ↪sort_values('cuantity', ascending=False)
scatter = scatter.reset_index()
```

[34]:
```
import seaborn as sns
plt.figure(figsize=(10,10))
plt.xticks(rotation=90)
g = sns.scatterplot(x = "keyword", y = "location", size='cuantity',␣
 ↪color='orange',  data= scatter)
g.set_title("Scatter Plot Ubicacion Vs Keyword", fontsize=22)
g.set_xlabel("Keyword",fontsize=15)
g.set_ylabel("Locacion", fontsize=15)
```

[34]: Text(0, 0.5, 'Locacion')



[35]:
```
tweets.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7613 entries, 0 to 7612
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   id                    7613 non-null   int64
 1   keyword               7613 non-null   object
 2   location              7613 non-null   object
 3   text                  7613 non-null   object
 4   target                7613 non-null   int64
```

```
 5   cuantity                7613 non-null   int64
 6   location_especificada   7613 non-null   bool
 7   text lenght             7613 non-null   int64
dtypes: bool(1), int64(4), object(3)
memory usage: 423.9+ KB
```

[36]:
```
tweets_por_locacion = tweets.groupby('location').sum()
tweets_por_locacion = tweets_por_locacion.sort_values('cuantity', ascending =␣
↪True)
tweets_por_locacion = tweets_por_locacion.reset_index()
tweets_por_locacion
```

[36]:

	location	id	target	cuantity	location_especificada	text lenght
0		9703	1	1	1.0	56
1	Redondo Beach, CA	4479	1	1	1.0	114
2	Regalo Island	10352	0	1	1.0	29
3	Republic of the Philippines	2813	1	1	1.0	91
4	Republica Dominicana	5491	1	1	1.0	135
…	…	…	…	…	…	…
3337	London	248126	16	45	45.0	4833
3338	United States	304076	27	50	50.0	5029
3339	New York	301155	16	71	71.0	9322
3340	USA	618271	67	104	104.0	11321
3341	Not Especified	13961318	1075	2533	0.0	250161

[3342 rows x 6 columns]

[37]:
```
tweets_por_locacion.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3342 entries, 0 to 3341
Data columns (total 6 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   location                3342 non-null   object
```

```
 1   id                     3342 non-null   int64
 2   target                 3342 non-null   int64
 3   cuantity               3342 non-null   int64
 4   location_especificada  3342 non-null   float64
 5   text lenght            3342 non-null   int64
dtypes: float64(1), int64(4), object(1)
memory usage: 156.8+ KB
```

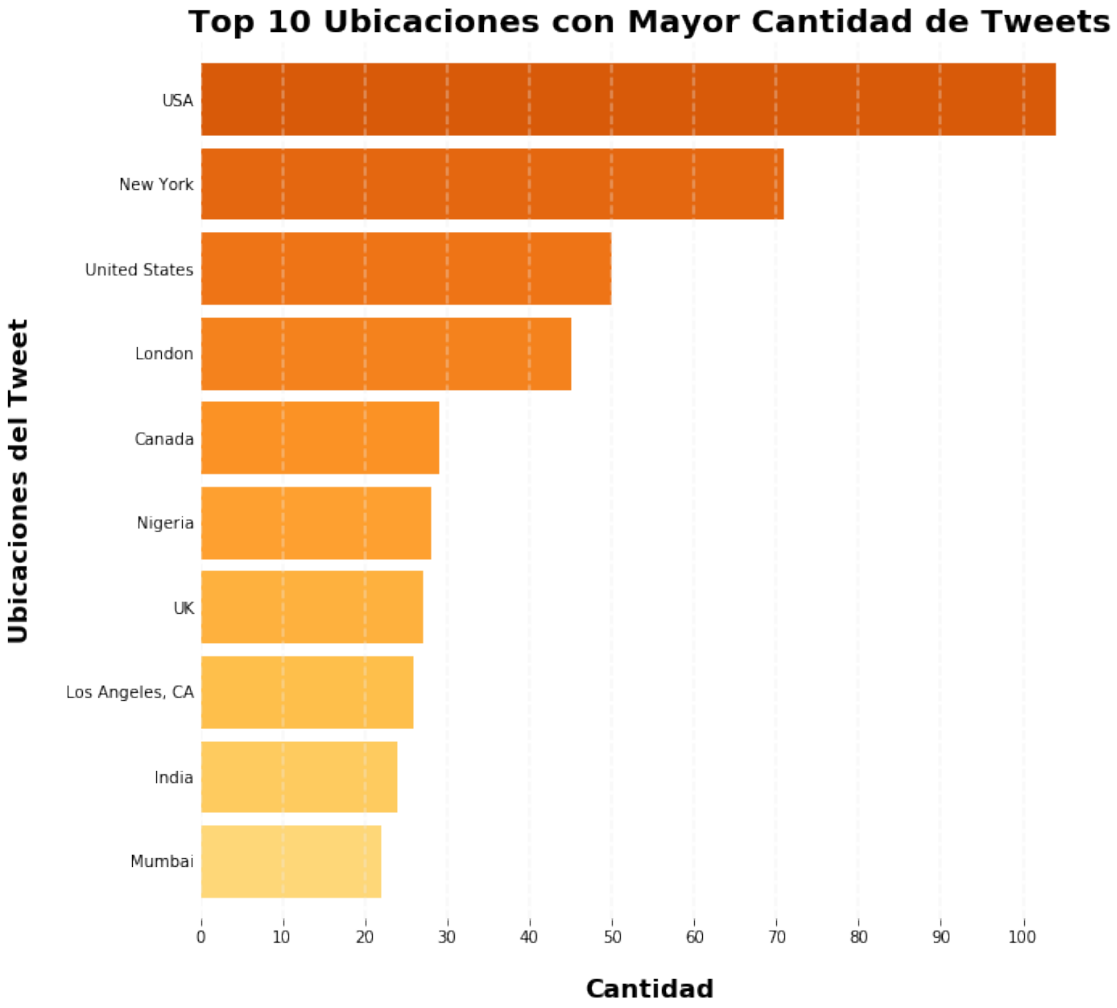
[38]:
```
specified_location = (tweets_por_locacion.loc[:,'location'] != 'Not Especified')
tweets_por_locacion = tweets_por_locacion.loc[specified_location]
tweets_por_locacion.head(15)
```

[38]:

	location	id	target	cuantity	location_especificada	text lenght
0		9703	1	1	1.0	56
1	Redondo Beach, CA	4479	1	1	1.0	114
2	Regalo Island	10352	0	1	1.0	29
3	Republic of the Philippines	2813	1	1	1.0	91
4	Republica Dominicana	5491	1	1	1.0	135
5	Reston, VA, USA	9469	1	1	1.0	90
6	Rheinbach / Germany	6085	0	1	1.0	49
7	Rhode Island	5017	0	1	1.0	121
8	RhodeIsland	3285	0	1	1.0	88
9	Rhyme Or Reason?	8738	1	1	1.0	102
10	Richardson TX	10415	0	1	1.0	147
11	Richmond Heights, OH	4239	0	1	1.0	112
12	Richmond, VA	3642	0	1	1.0	112
13	Right here	2867	0	1	1.0	140
14	Right next to Compton	6106	0	1	1.0	133

[39]:
```
saltos = np.linspace(0.3, 0.7, 10)
colores = (cm.get_cmap('YlOrBr'))(saltos)
tweets_por_locacion_plot = tweets_por_locacion.tail(10).plot(kind='barh', y␣
↪='cuantity', x = 'location', figsize=(10,10), color=colores, width=0.85,␣
↪legend = False)

plt.xticks(np.arange(0, 100+1, 10.0))
plt.tick_params(axis='y', length=0)

tweets_por_locacion_plot.spines['right'].set_visible(False)
tweets_por_locacion_plot.spines['top'].set_visible(False)
tweets_por_locacion_plot.spines['left'].set_visible(False)
tweets_por_locacion_plot.spines['bottom'].set_visible(False)

lineas = tweets_por_locacion_plot.get_xticks()
for i in lineas:
      tweets_por_locacion_plot.axvline(x=i, linestyle='--', alpha=0.4,␣
↪color='#eeeeee')

tweets_por_locacion_plot.set_xlabel("Cantidad", labelpad=20, weight='bold',␣
↪size=16)
tweets_por_locacion_plot.set_ylabel("Ubicaciones del Tweet", labelpad=20,␣
↪weight='bold', size=16)
tweets_por_locacion_plot.set_title("Top 10 Ubicaciones con Mayor Cantidad de␣
↪Tweets", weight='bold', size=20)
```

[39]:
```
Text(0.5, 1.0, 'Top 10 Ubicaciones con Mayor Cantidad de Tweets')
```


Top 10 Ubicaciones con Mayor Cantidad de Tweets
Ubicaciones del Tweet
USA
New York
United States
London
Canada
Nigeria
UK
Los Angeles, CA
India
Mumbai
0
10
20
30
40
50
60
70
80
90
100
Cantidad