Introduction

As a resource for social data, Twitter's platform has been used to measure the quality of life through sentiment analysis. This capstone project explores another methodological technique of using specific keyword terms to determine dominant topics, word patterns, and sentiment leanings in a geographical area. Focusing on New York City and Los Angeles for comparative analysis, the keyword term "why" will be used to build a Python analysis around topic modeling and sentiment analysis. With this approach, the analysis reveals social and cultural differences, the overall sentiment of tweets, and areas of interest to tweeters.

Contents

Install libraries

If you find that you are missing any libraries after importing them in the next step, please use this section to install them. For the nltk.download() functions, you are able to download them AFTER importing the NLTK library.

pip install searchtweets-v2pip install nltk#to import stop words from the English language nltk.download('stopwords')#to import word_tokenizer() nltk.download('punkt')#to import SentimentIntensityAnalyzer() nltk.download('vader_lexicon')pip install textblobpip install contractionspip install wordcloudpip install sklearnpip install plotlypip install geopandas

Python Libraries

In [1]:

```
import searchtweets as twitter

import pandas as pd
from pandas.io.json import json_normalize
import numpy as np

import re
import contractions
import string
import textwrap

import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS
from PIL import Image
import plotly.graph_objs as go
import plotly.express as px
import geopandas as gpd

import nltk
from nltk.corpus import stopwords
from nltk.probability import FreqDist
from nltk.stem import PorterStemmer, WordNetLemmatizer
```

```python
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk.tokenize import word_tokenize
from nltk.util import ngrams

from textblob import TextBlob

import sklearn
from sklearn import metrics
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import LatentDirichletAllocation
from sklearn.preprocessing import normalize

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import confusion_matrix, classification_report

import warnings
from warnings import simplefilter
warnings.filterwarnings('ignore')
simplefilter(action='ignore', category=FutureWarning)
```

Data Setup

Import Twitter Data

The file *(twitter_keys.yaml)* in this code needs to be edited with your own tokens. Twitter does not allow the sharing of tokens.

In [2]:
```python
#file contains API and Bearer tokens
search_args = twitter.load_credentials('twitter_keys.yaml',
                                       yaml_key='search_tweets_v2',
                                       env_overwrite=False)
```

Query Twitter Data

To set up search terms for query. Twitter offers documention on building a query.

In [ ]:
```python
#set up parameters for query search
query_whynyc = twitter.gen_request_parameters(
    query = 'why (place:01a9a39529b27f36 ' +
    'OR place:011add077f4d2da3' +
    'OR place:00c39537733fa112' +'
    'OR place:002e24c6736f069d' +
    'OR place:00c55f041e27dc51) -is:retweet -is:nullcast lang:en',
    results_per_call = 500,
    start_time = '2021-01-01',
    end_time = '2022-01-01',
    tweet_fields = 'id,created_at,text,geo',
    granularity=''
)

#140k to get all of 2021
whynyc_tweets = twitter.collect_results(
    query_whynyc,
    max_tweets=140000,
    result_stream_args=search_args
)
```

In [4]:
```
query_whyla = twitter.gen_request_parameters(
    query = 'why (place:3b77caf94bfc81fe) -is:retweet -is:nullcast lang:en',
    results_per_call = 500,
    start_time = '2021-01-01',
    end_time = '2022-01-01',
    tweet_fields = 'id,created_at,text,geo',
    granularity=''
)

#90k
whyla_tweets = twitter.collect_results(
    query_whyla,
    max_tweets=90000,
    result_stream_args=search_args
)
```

Convert JSON to Dataframe

In [5]:
```
whynyc_df = pd.json_normalize(whynyc_tweets, record_path=['data'])
whyla_df = pd.json_normalize(whyla_tweets, record_path=['data'])
```

Data Cleaning

In [6]:
```
def find_links(tweet):
    #function  extracts the links
    return re.findall('(http\S+|bit.ly/\S+)', tweet)

#def find_retweeted(tweet):
   # function finds and extracts retweeted twitter handles
    #return re.findall('(?<=RT\s)(@[A-Za-z0-9]+[A-Za-z0-9-_]+)', tweet)

def find_mentioned(tweet):
    #function finds and extracts the twitter handles of people mentioned
    return re.findall('(?<!RT\s)(@[A-Za-z0-9]+[A-Za-z0-9-_]+)', tweet)

def find_hashtags(tweet):
    #This function will extract hashtags
    return re.findall('(#[A-Za-z0-9]+[A-Za-z0-9-_]+)', tweet)
```

In [7]:
```
# make new columns for links, retweeted usernames, mentioned usernames and hashtags
whynyc_df['links'] = whynyc_df.text.apply(find_links)
#whynyc_df['retweeted'] = whynyc_df.text.apply(find_retweeted)
whynyc_df['mentioned'] = whynyc_df.text.apply(find_mentioned)
whynyc_df['hashtags'] = whynyc_df.text.apply(find_hashtags)

whyla_df['links'] = whyla_df.text.apply(find_links)
#whyla_df['retweeted'] = whyla_df.text.apply(find_retweeted)
whyla_df['mentioned'] = whyla_df.text.apply(find_mentioned)
whyla_df['hashtags'] = whyla_df.text.apply(find_hashtags)
```

In [8]:
```
#to clean up the ['text'] column

stopwords = nltk.corpus.stopwords.words('english')
lemmatizer = WordNetLemmatizer() #groups together similar words as a single term
punctuation = string.punctuation #'!'$%&\'()*+,-./:;<=>?[\\]^_`{|}~•@'
symbol = '—…«»“”‘’' #for symbols not captured in punctuation

def clean_df(tweet):
```

```python
    #remove parts of a tweet
    tweet = re.sub(r'http\S+', '', tweet) #removes links
    tweet = re.sub(r'bit.ly/\S+', '', tweet) #removes bitly links
    #tweet = re.sub('(RT\s@[A-Za-z0-9]+[A-Za-z0-9-_]+)', '', tweet) #removes retweeted use
    tweet = re.sub('(@[A-Za-z0-9]+[A-Za-z0-9-_]+)', '', tweet) #removes mentioned username
    #removes hashtags, for this analysis they are kept in
    #tweet = re.sub('(#[A-Za-z0-9]+[A-Za-z0-9-_]+)', '', tweet)

    #removing these that showup after data cleaning processing
    tweet = re.sub('&amp;', '&', tweet)
    tweet = re.sub('\n', '', tweet)

    #lower-case characters
    tweet = tweet.lower()

    #remove contractions
    tweet = contractions.fix(tweet)

    #remove numbers
    tweet = re.sub('([0-9]+)', '', tweet)

    #remove punctuation
    tweet = re.sub('['+ string.punctuation +']+', ' ', tweet)

    #remove symbols not captured in punctuation
    tweet = re.sub('['+ symbol +']+', ' ', tweet)

    #remove whitespace
    tweet = re.sub(r'^\s+|\s+$', '', tweet)
    tweet = re.sub(r'\s+', ' ', tweet)

    #tokenize words and remove stopwords
    tweet_token_list = [word for word in tweet.split(' ')#]
                        if word not in stopwords] # remove stopwords

    #apply word lemmatization
    tweet_token_list = [lemmatizer.lemmatize(word) if '#' not in word else word
                    for word in tweet_token_list]

    tweet = ' '.join(tweet_token_list)
    return tweet

#create a new column for the cleaned text column.
whynyc_df['corpora'] = whynyc_df.text.apply(clean_df)
whyla_df['corpora'] = whyla_df.text.apply(clean_df)
```

In [9]:

```python
#pull list of columns
list(whynyc_df)
```

Out[9]:

```
['text',
 'id',
 'created_at',
 'geo.place_id',
 'geo.coordinates.type',
 'geo.coordinates.coordinates',
 'withheld.copyright',
 'withheld.country_codes',
 'withheld.scope',
 'links',
 'mentioned',
 'hashtags',
 'corpora']
```

In [10]:

```
#reorder columns in dataframe
whynyc = whynyc_df[['id', 'created_at', 'text', 'corpora', 'geo.place_id']]
whyla = whyla_df[['id', 'created_at', 'text', 'corpora', 'geo.place_id']]

#created for a one-time analysis
#adding this to whynyc and whyla dataframes may cause kernel to crash due to size
whynyc_pot = whynyc_df[['mentioned', 'hashtags', 'links']]
whyla_pot = whyla_df[['mentioned', 'hashtags', 'links']]
```

Categorizing Sentiment on Tweets

Uses the NLTK and TextBlob libraries to calculate the polarity/sentiment (NLTK & TextBlob) and subjectivity (TextBlob only) scores of tweets.

In [11]:

```
sid = SentimentIntensityAnalyzer()
```

In [12]:

```
def add_sentiment(why_df):
    #pulling polarity scores from NLTK library
    sa_nltk_list = []
    for i in why_df['text']:
        sa_nltk_list.append((sid.polarity_scores(str(i)))['compound'])
    #why_df['score'] = pd.Series(sa_nltk_list, dtype='float64')

    #pulling subjectivity and polarity scores from TextBlob
    def subjectivity(text):
        return TextBlob(text).sentiment.subjectivity
    why_df['subjectivity'] = why_df['text'].apply(subjectivity)

    #Create a function to get the polarity
    sa_tb_list = []
    def polarity(text):
        return TextBlob(text).sentiment.polarity
    sa_tb_list = why_df['text'].apply(polarity)

    #average of NLTK and TextBlob's polarity scores via Numpy
    avg = []
    avg = np.mean(np.array([sa_nltk_list, sa_tb_list]), axis=0)
    why_df['score'] = pd.DataFrame(avg)

    #Categorizing sentiment scores
    def sentiment_category(sentiment):
        label = ''
        if(sentiment>0):
            label = 'positive'
        elif(sentiment == 0):
            label = 'neutral'
        else:
            label = 'negative'
        return(label)
    why_df['sentiment'] = why_df['score'].apply(sentiment_category)

    return why_df

whynyc = add_sentiment(whynyc)
whyla = add_sentiment(whyla)
```

Exploratory Analysis

Due to links, mentions, and hahtags accounting for less than 1/3 of the total tweets queried, this portion will only provide a basic idea of how much relevance the parts of a tweet (below) play a role. In a future project, a network analysis will come into play for this part.discourse.

1. Links (either to an image or a website)
2. Mentioned
3. Hashtags

	New York City	Los Angeles	% of tweets (NYC/LA)
Links accounts for	41,566 tweets	24,497 tweets	(30%/27%)
Mentioned tweeters account for	46,067 tweets	28,535 tweets	(33%/32%)
Hashtags account for	9,180 tweets	5,537 tweets	(7%/6%)

If you run into an error running this portion of the code, it's recommended to update the pandas library: *pip install pandas --upgrade*

In [13]:
```
len(whynyc)
```
Out[13]:
```
138773
```

In [14]:
```
len(whyla)
```
Out[14]:
```
89292
```

In [15]:
```
whynyc_pot['links'].value_counts()
```
Out[15]:
```
[]                               97296
[https://t.co/CiyzXjOgTz]           16
[https://t.co/UW7TkMtUIM]           13
[https://t.co/SPAVAK2eTW]           12
[https://t.co/Ef0jFoFAbo]           11
                                 ...
[https://t.co/zAErT5xZB8]            1
[https://t.co/RbYXteEiZo]            1
[https://t.co/8SKYBDzkCp]            1
[https://t.co/61DlVpQdzt]            1
[https://t.co/n0Mf9AhPBK]            1
Name: links, Length: 41275, dtype: int64
```

In [16]:
```
whynyc_pot['mentioned'].value_counts()
```
Out[16]:
```
[]                                          79144
[@NYCTSubway]                                 312
[@YouTube]                                    165
[@MTA]                                        129
[@nypost]                                     101
                                            ...
[@SarahLongwell25]                              1
[@dyorcanada, @RpsAgainstTrump]                 1
[@RealMeMP, @hbryant42]                         1
[@RealMeMP, @TwoLiterHero, @hbryant42]          1
[@cbaibix]                                      1
Name: mentioned, Length: 45835, dtype: int64
```

In [17]:
```
whynyc_pot['hashtags'].value_counts()
```
```
[]                                     126439
```

```
Out[17]: [#RHOP]                                          85
         [#Yankees]                                       65
         [#LGM]                                           62
         [#RHOA]                                          54
                                                        ...
         [#WWNXT, #TakeOver]                               1
         [#RecallRonDeSantis, #RecallAbbott]               1
         [#Exiles, #WhatIf]                                1
         [#Millions, #VERZUZ, #verzuztv]                   1
         [#Stupidity]                                      1
         Name: hashtags, Length: 9140, dtype: int64
```

```
In [18]: whyla_pot['links'].value_counts()
```

```
Out[18]: []                              64673
         [https://t.co/H4JxLhbZow]          49
         [https://t.co/QNaC8UUGfn]          19
         [https://t.co/S5YKpBwbVm]          10
         [https://t.co/Ob6sf4xrpK]           7
                                          ...
         [https://t.co/KHOQAozFbQ]           1
         [https://t.co/Tg1zGicgJt]           1
         [https://t.co/MIhsPcE6z8]           1
         [https://t.co/zaM35H9Pxp]           1
         [https://t.co/dqnarpbqIs]           1
         Name: links, Length: 24469, dtype: int64
```

```
In [19]: whyla_pot['mentioned'].value_counts()
```

```
Out[19]: []                    54446
         [@YouTube]              187
         [@GavinNewsom]           44
         [@Dodgers]               43
         [@thehill]               43
                                ...
         [@prestonsphoto]          1
         [@OhemaaEsther_]          1
         [@ProdByStreet]           1
         [@AndrewMoMoney]          1
         [@BookSyrup]              1
         Name: mentioned, Length: 28487, dtype: int64
```

```
In [20]: whyla_pot['hashtags'].value_counts()
```

```
Out[20]: []                                                    82525
         [#Dodgers]                                               69
         [#FreeBritney]                                           48
         [#RHOBH]                                                 26
         [#BB23]                                                  25
                                                               ...
         [#DemonsSouls, #ps5]                                      1
         [#ClimateAction, #StandWithGavin, #VoteNoOnRecall]        1
         [#StandWithGavin, #VoteNoOnRecall]                        1
         [#karma, #lakers, #goat]                                  1
         [#ShacarriRichardson]                                     1
         Name: hashtags, Length: 5535, dtype: int64
```

Text Analysis

This portion converts the 'created_at' column into datatime format with to_datatime() function in pandas. Length of tweets and a basic time series analysis is performed.

In [21]:

```
def format_why_df(why_df):
    #convert created_at column to datetime
    why_df['datetime'] = pd.to_datetime(why_df['created_at'], errors='coerce')

    #create a day column
    why_df['day'] = why_df['datetime'].dt.date

    #create a month column
    why_df['month'] = why_df['datetime'].dt.month

    #break up text column into length
    why_df['length']=why_df['text'].apply(lambda x:len(x.split()))
    return why_df

whynyc = format_why_df(whynyc)
whyla = format_why_df(whyla)
```

In [22]:

```
whynyc['length'].describe()
```

Out[22]:

```
count    138773.000000
mean         22.413640
std          14.222482
min           1.000000
25%          11.000000
50%          18.000000
75%          31.000000
max         109.000000
Name: length, dtype: float64
```

In [23]:

```
whyla['length'].describe()
```

Out[23]:

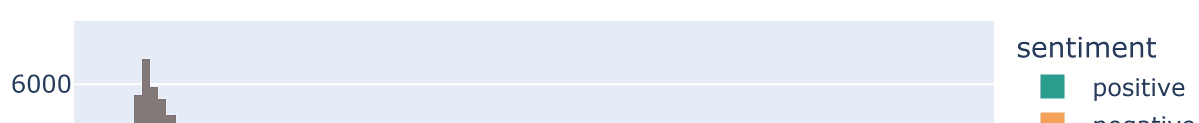
```
count    89292.000000
mean        21.403340
std         13.847574
min          1.000000
25%         11.000000
50%         17.000000
75%         29.000000
max        109.000000
Name: length, dtype: float64
```

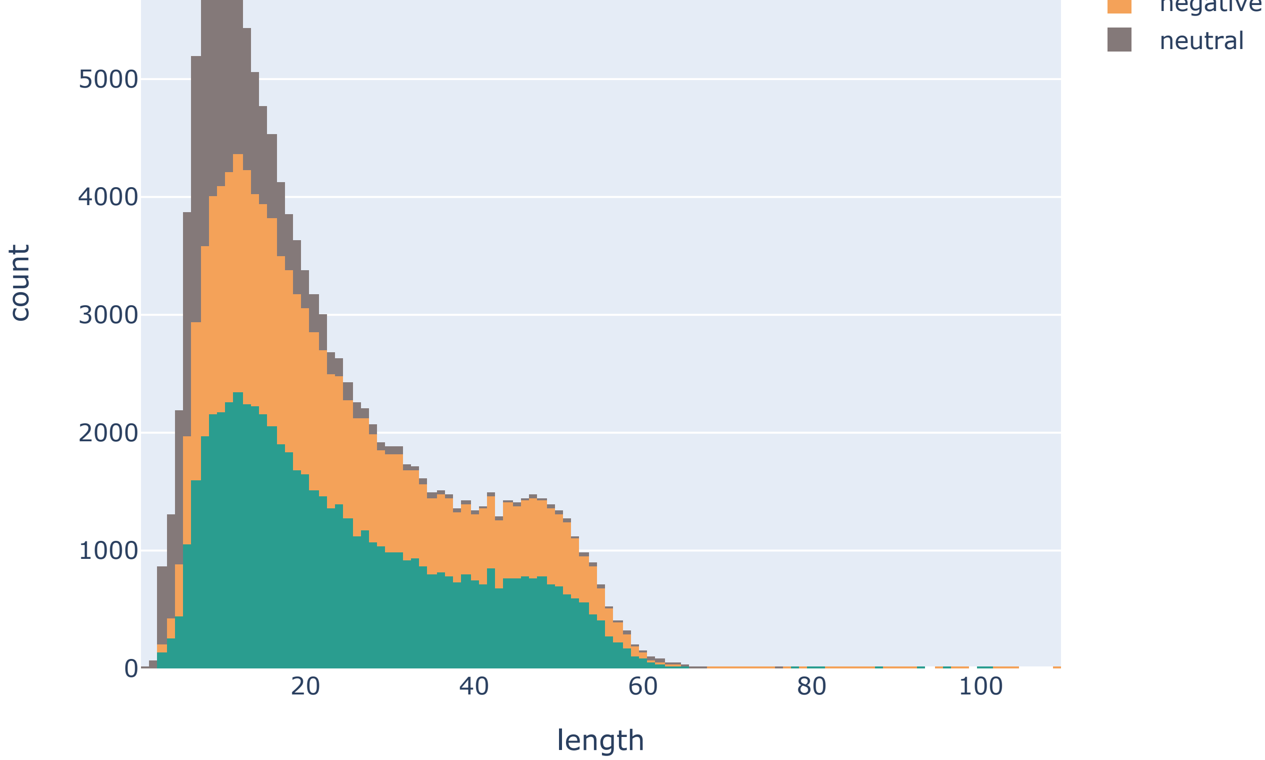
In [24]:

```
#set colors for tweets categorized as positive, neutral, or negative
sentiment_colors = {
    'positive': '#2A9D8F',
    'neutral': '#847979',
    'negative': '#F4A259'}
```

In [25]:

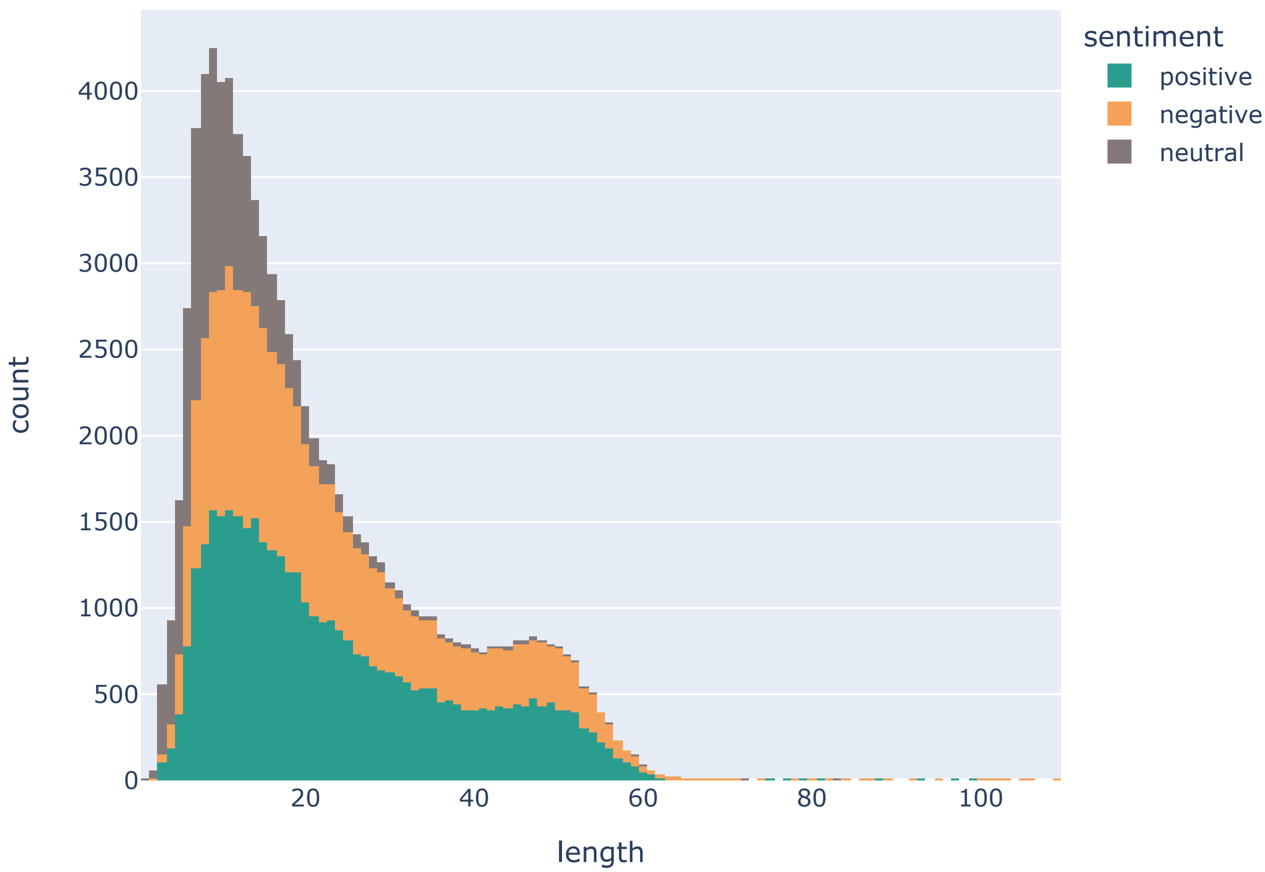
```
#creates a graph of length of tweets by sentiment in a histogram
px.histogram(
    whynyc,
    x='length',
    color='sentiment',
    color_discrete_map = sentiment_colors)
```





In [26]:

```
px.histogram(
    whyla,
    x='length',
    color='sentiment',
    color_discrete_map = sentiment_colors)
```

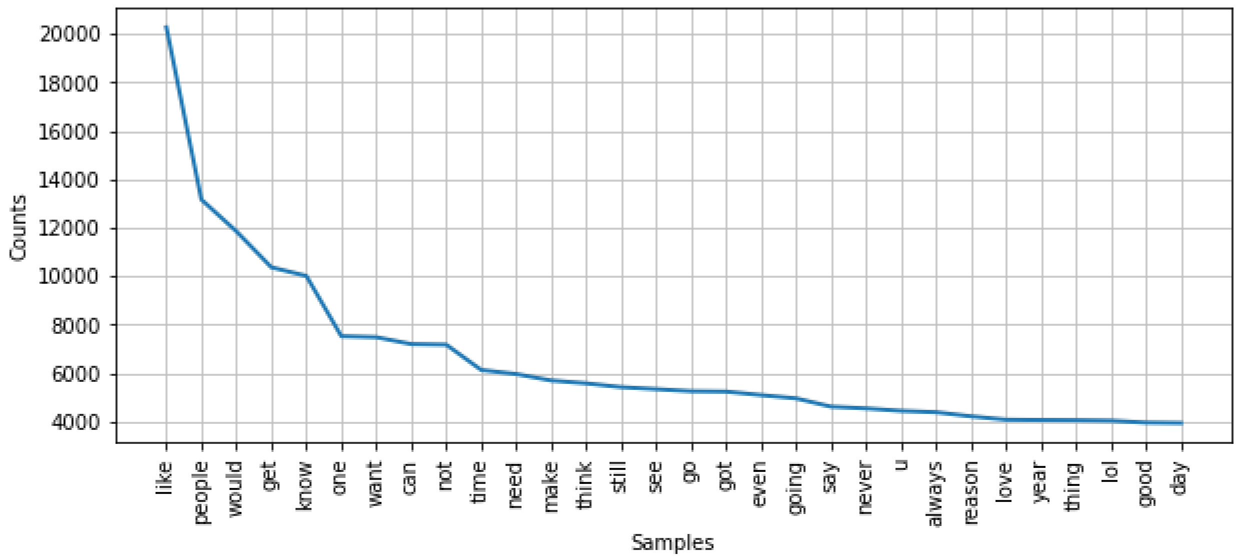


In [27]:

```
#convert dataframe to lists
whynyc_list = whynyc['corpora'].values.tolist()
whynyc_list = ' '.join(whynyc_list).lower()

whyla_list = whyla['corpora'].values.tolist()
whyla_list = ' '.join(whyla_list).lower()
```

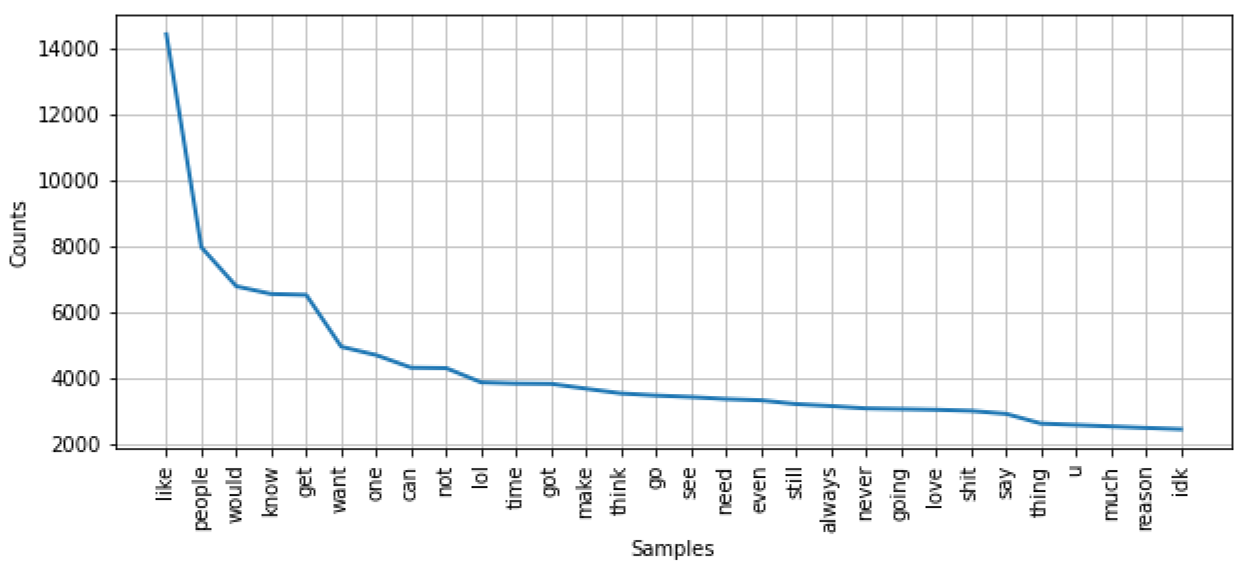
In [28]:

```
#create a frequency distribution and graph it
fdist_whynyc = FreqDist(word_tokenize(whynyc_list))
plt.figure(figsize=(10, 4))
fdist_whynyc.plot(30, cumulative=False)
plt.show()
```



In [29]:

```
fdist_whyla = FreqDist(word_tokenize(whyla_list))
plt.figure(figsize=(10, 4))
fdist_whyla.plot(30,cumulative=False)
plt.show()
```



Text Analysis - Word Clouds

Creates a word cloud based on the text ('corpora') data for NYC and LA.

In [32]:

```
def create_wordcloud(file_name, list_name):
    #pull the image file
    mask = np.array(Image.open(file_name))
    #function converts RGB values from 0 (black) to white (255)
    def transform_zeros(val):
        if val == 0:
            return 255
        else:
            return val
    #map and create a mask for image
    maskable_image = np.ndarray((mask.shape[0],mask.shape[1]), np.int32)
    for i in range(len(mask)):
        maskable_image[i] = list(map(transform_zeros, mask[i]))

    #create word cloud
    wordcloud = WordCloud(
        width = 3000,
        height = 2000,
        #random_state=1,
        background_color='white',
        colormap='twilight_r',
        contour_width = 1,
        contour_color = '#111954',
        collocations=True,
        stopwords = STOPWORDS,
        mask=maskable_image).generate(list_name)

    def plot_cloud(wordcloud):
        # Set figure size
        plt.figure(figsize=(15, 7))
        # Display image
        plt.imshow(wordcloud)
        # No axis details
        plt.axis('off');
    return plot_cloud(wordcloud)

create_wordcloud('new-york-city.png', whynyc_list)
create_wordcloud('los-angeles.png', whyla_list)
```

N-grams

N-grams are continuous sequences of a neighbouring sequences of terms in a document. This section will look at n-grams up to 4.

In [33]:

```
def get_ngrams(text, ngram_from=2, ngram_to=2, n=None, max_features=20000):

    vectorizer = TfidfVectorizer(ngram_range = (ngram_from, ngram_to),
                                 max_features = max_features,
                                 stop_words='english').fit(text)
    bag_of_words = vectorizer.transform(text)
    sum_words = bag_of_words.sum(axis = 0)
    words_freq = [(word, sum_words[0, i]) for word, i in vectorizer.vocabulary_.items()]
    words_freq = sorted(words_freq, key = lambda x: x[1], reverse = True)

    return words_freq[:n]
```

In [34]:

```
def ngrams_table(why_df):
    unigrams = pd.DataFrame(get_ngrams(why_df['corpora'], ngram_from=1, ngram_to=1, n=15))
    bigrams = pd.DataFrame(get_ngrams(why_df['corpora'], ngram_from=2, ngram_to=2, n=15))
    trigrams = pd.DataFrame(get_ngrams(why_df['corpora'], ngram_from=3, ngram_to=3, n=15))
    quadgrams = pd.DataFrame(get_ngrams(why_df['corpora'], ngram_from=4, ngram_to=4, n=15)

    ngrams = pd.concat([unigrams, bigrams, trigrams, quadgrams], axis = 1)
    ngrams.columns = ['unigrams', 'frequency', 'bigrams', 'frequency', 'trigrams', 'freque
    ngrams
    return ngrams

ngrams_nyc = ngrams_table(whynyc)
ngrams_la = ngrams_table(whyla)
```

In [35]:

```
ngrams_nyc
```

Out[35]:

unigrams	frequency	bigrams	frequency	trigrams	frequency	quadgrams	frequency

	unigrams	frequency	bigrams	frequency	trigrams	frequency	quadgrams	frequency
0	like	3526.358632	look like	655.783034	new york city	151.816242	new york new york	156.422971
1	people	2150.918389	new york	645.985518	new york new	115.668852	news network elected official	31.975772
2	know	1951.539763	feel like	501.050213	york new york	112.994978	network elected official silent	31.975772
3	want	1507.601631	make sense	397.817810	brooklyn new york	88.385581	elected official silent obvious	31.975772
4	lol	1220.547820	want know	311.547878	make make sense	77.332696	official silent obvious miscarriage	31.975772
5	time	1177.122034	people like	214.641075	manhattan new york	56.435142	silent obvious miscarriage justice	31.975772
6	got	1170.163288	year old	212.544547	gt gt gt	36.356726	obvious miscarriage justice social	31.975772
7	need	1161.383152	sound like	196.237768	idk feel like	33.265191	miscarriage justice social security	31.975772
8	think	1151.911341	year ago	183.308309	really want know	31.393707	justice social security irs	31.975772
9	make	1110.034728	social medium	179.510993	news network elected	31.205158	social security irs administration	31.975772
10	love	1059.343709	black people	160.755027	network elected official	31.205158	security irs administration ssi	31.975772
11	say	1031.696893	understand people	160.101779	elected official silent	31.205158	irs administration ssi veteran	31.975772
12	going	1013.371542	acting like	133.604973	official silent obvious	31.205158	administration ssi veteran deserve	31.975772
13	reason	944.012198	people think	133.345902	silent obvious miscarriage	31.205158	ssi veteran deserve date	31.975772
14	good	898.629182	like know	131.502554	obvious miscarriage justice	31.205158	veteran deserve date expect	31.975772

In [36]:
```
ngrams_la
```

Out[36]:

	unigrams	frequency	bigrams	frequency	trigrams	frequency	quadgrams	frequency
0	like	2494.710708	look like	420.705109	los angeles california	248.723841	lt lt lt lt	55.803537
1	people	1344.269198	feel like	344.436889	cosmos graphically audiovisual	100.247752	graphically audiovisual face race	32.907430

	unigrams	frequency	bigrams	frequency	trigrams	frequency	quadgrams	frequency
2	know	1305.473135	los angeles	307.263053	lt lt lt	57.832749	audiovisual face race age	32.907430
3	lol	1103.700921	make sense	236.575670	make make sense	54.574224	face race age nationality	32.907430
4	want	1003.557985	angeles california	178.797667	graphically audiovisual face	31.199637	race age nationality exact	32.907430
5	got	865.285627	want know	167.804014	audiovisual face race	31.199637	age nationality exact location	32.907430
6	love	780.777020	people like	141.267188	face race age	31.199637	cosmos graphically audiovisual face	32.700536
7	think	755.747000	sound like	136.189164	race age nationality	31.199637	nationality exact location everybody	27.344332
8	time	748.595090	year old	132.020825	age nationality exact	31.199637	los angeles hollywood california	13.905519
9	make	727.506954	social medium	119.397240	nationality exact location	31.199637	al haqq nur graphically	11.941086
10	shit	713.857198	year ago	110.075647	exact location everybody	25.998256	haqq nur graphically audiovisual	11.941086
11	idk	713.544498	understand people	97.060446	idk feel like	24.236026	cosmos graphically audiovisual body	11.901906
12	need	681.686774	graphically audiovisual	87.964938	gt gt gt	19.437876	guest catch live weeknight	9.367723
13	say	670.784850	people think	83.777291	today feel like	17.177447	catch live weeknight et	9.199177
14	going	659.624205	cosmos graphically	81.720461	make feel like	17.032416	south los angeles california	9.098962

Sentiment EDA

This section is an exploratory data analysis of the sentiment on tweets.

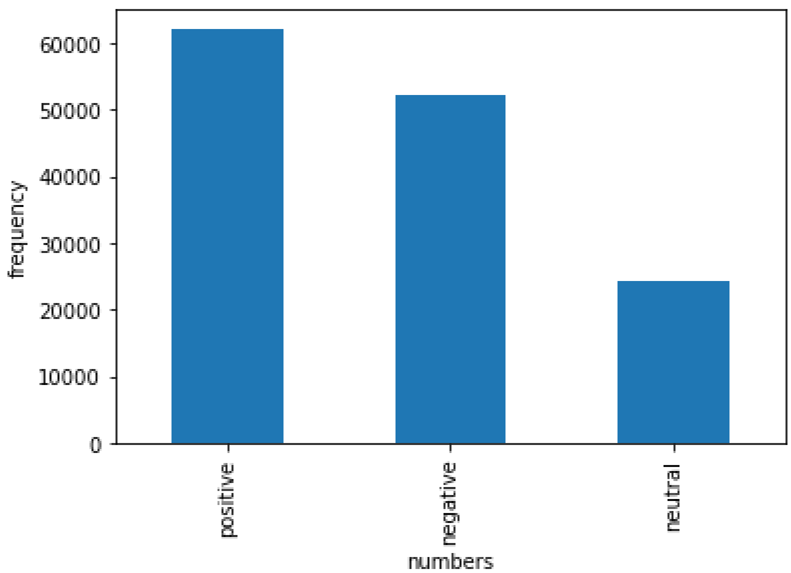
In [37]:
```
whynyc['score'].describe()
```

Out[37]:
```
count    138773.000000
mean          0.030560
std           0.330986
min          -0.991750
25%          -0.186600
50%           0.000000
75%           0.248300
max           0.982300
Name: score, dtype: float64
```

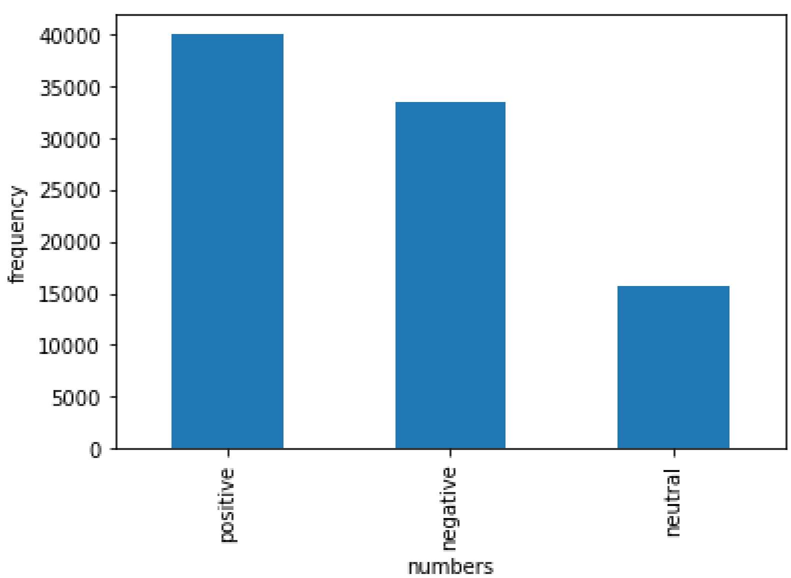
In [38]:
```
whynyc['sentiment'].describe()
```

Out[38]:
```
count       138773
unique           3
top       positive
freq         62135
Name: sentiment, dtype: object
```

In [39]:
```
fig, ax = plt.subplots()
whynyc['sentiment'].value_counts().plot(ax=ax, kind='bar', xlabel='numbers', ylabel='frequ
plt.show()
```

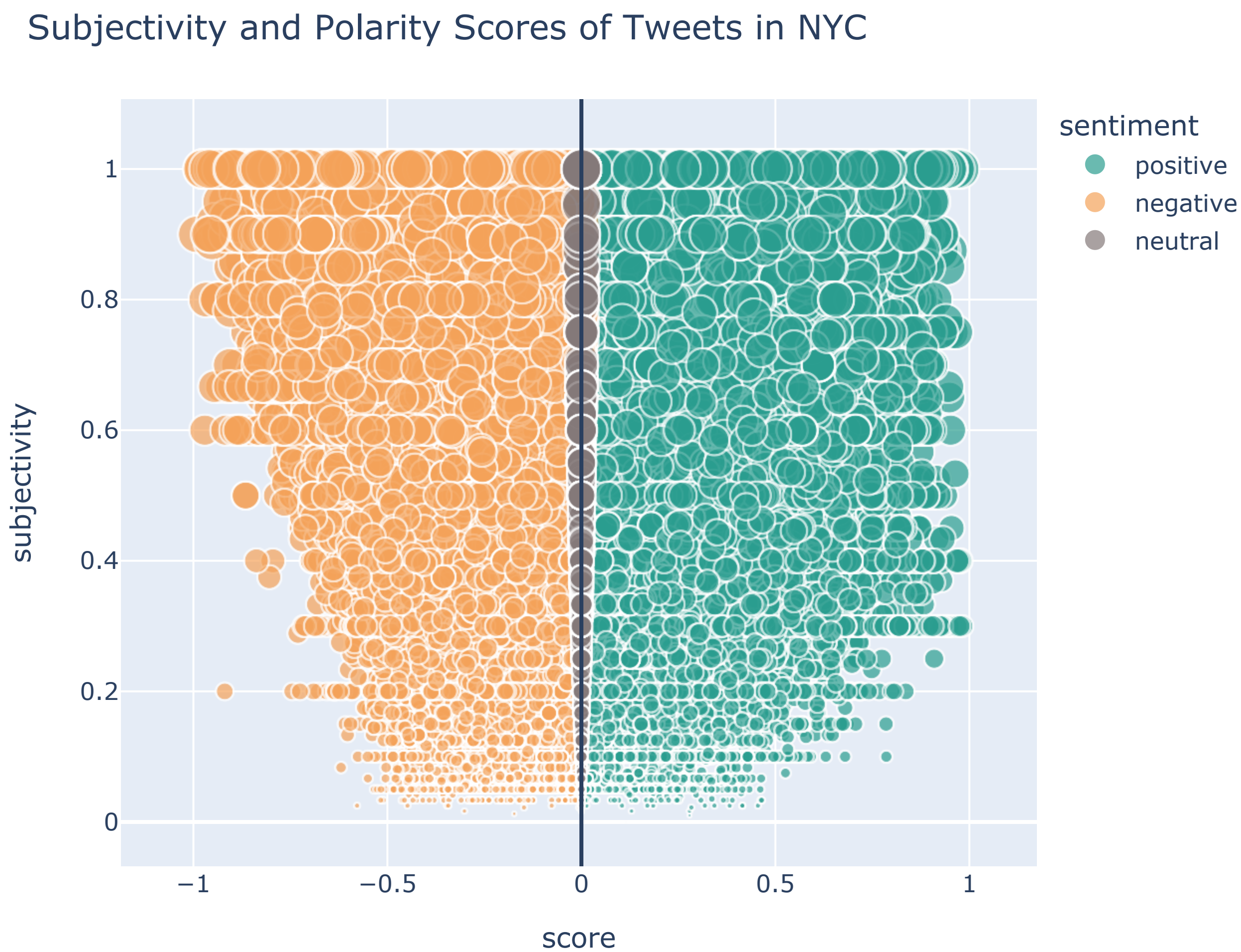


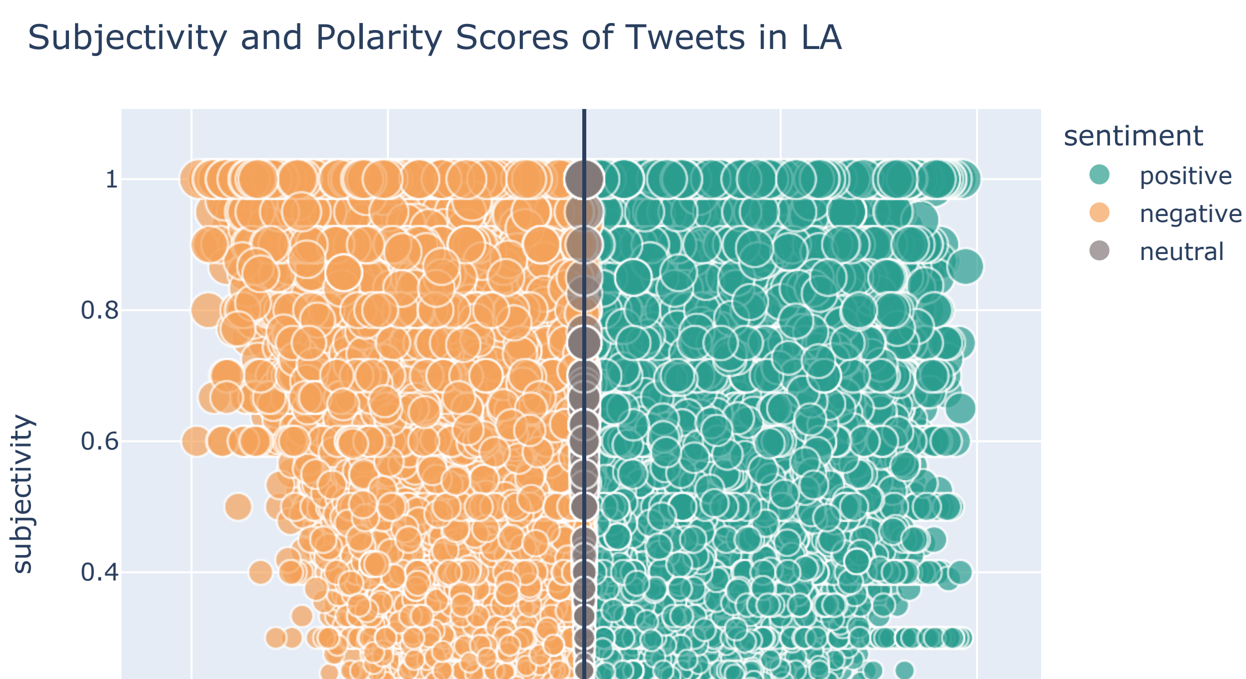
In [40]:
```
fig, ax = plt.subplots()
whyla['sentiment'].value_counts().plot(ax=ax, kind='bar', xlabel='numbers', ylabel='freque
plt.show()
```

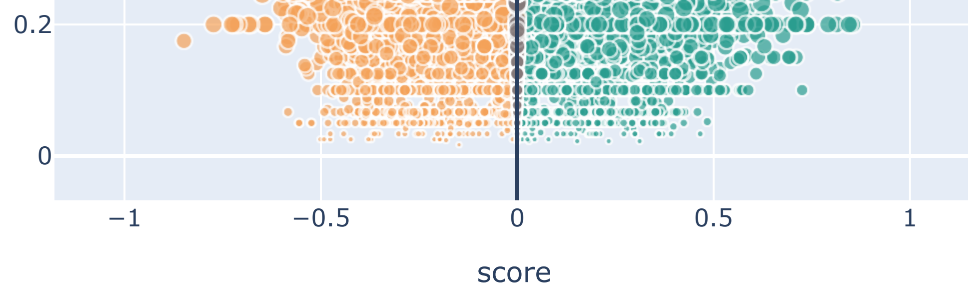


In [41]:
```
def pol_and_sub_of_tweets(title, why_df):
    # plot the polarity and subjectivity
    fig = px.scatter(why_df,
                     x='score',
                     y='subjectivity',
                     color = 'sentiment',
                     color_discrete_map = sentiment_colors,
                     size='subjectivity',
                     hover_name=why_df.text.apply(lambda txt: '<br>'.join(textwrap.wrap(tx
```

```
    #add a vertical line at x=0 for Netural Reviews
    fig.update_layout(title=title,
                      shapes=[dict(type= 'line',
                                   yref= 'paper', y0= 0, y1= 1,
                                   xref= 'x', x0= 0, x1= 0)])
    return fig.show()

pol_and_sub_of_tweets('Subjectivity and Polarity Scores of Tweets in NYC', whynyc)
pol_and_sub_of_tweets('Subjectivity and Polarity Scores of Tweets in LA', whyla)
```





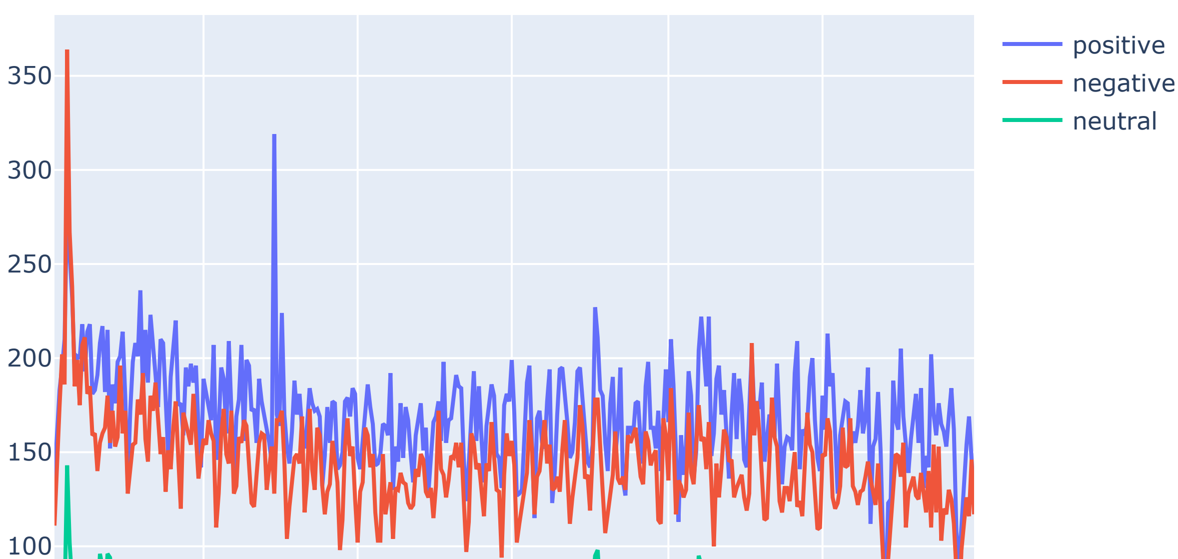


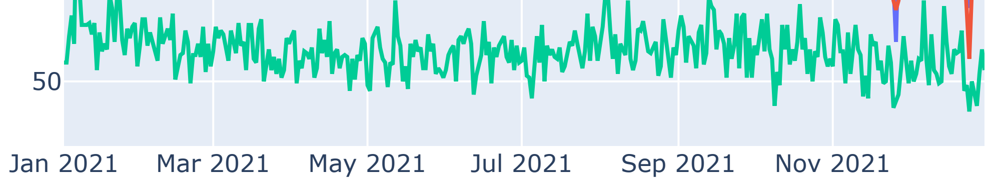
In [42]:

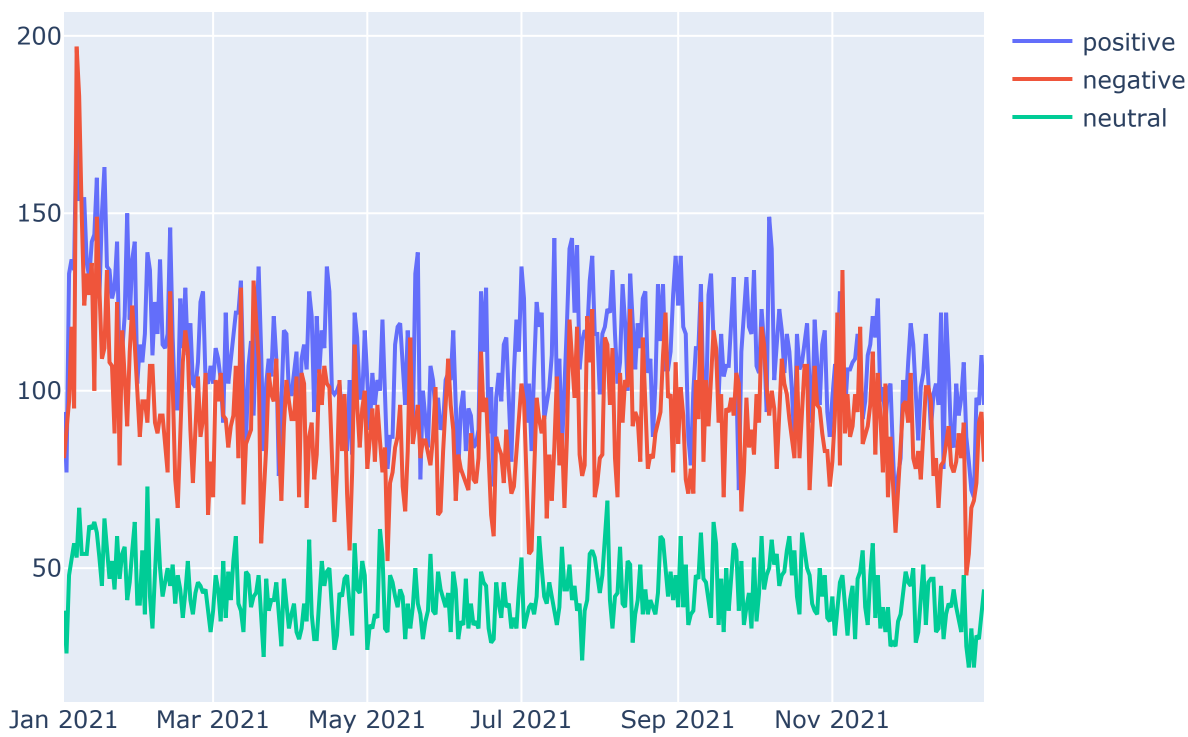
```
def sentiment_count(why_df):
    sent_count = why_df.groupby('day').sentiment.value_counts()
    sent_count = sent_count.to_frame(name='count')
    sent_count.reset_index(inplace=True)
    return sent_count

whynyc_sent = sentiment_count(whynyc)
whyla_sent = sentiment_count(whyla)
```

In [43]:

```
def sentiment_plotly(why_df):
    fig = go.Figure()
    for c in why_df['sentiment'].unique()[:3]:
        dfp = why_df[why_df['sentiment']==c].pivot(
            index='day',
            columns='sentiment',
            values='count')

        fig.add_traces(
            go.Scatter(
                x=dfp.index,
                y=dfp[c],
                mode='lines',
                name = c))
    return fig.show()

sentiment_plotly(whynyc_sent)
sentiment_plotly(whyla_sent)
```







The next few word clouds explore the spikes in sentiment on particular days.

In [44]:

```
nyc_neg_wc_j6 = whynyc[(whynyc['datetime']>='2021-01-05') & (whynyc['datetime']<='2021-01-
wordcloud = WordCloud(max_font_size=50, max_words=500, background_color='white').generate
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

In [45]:

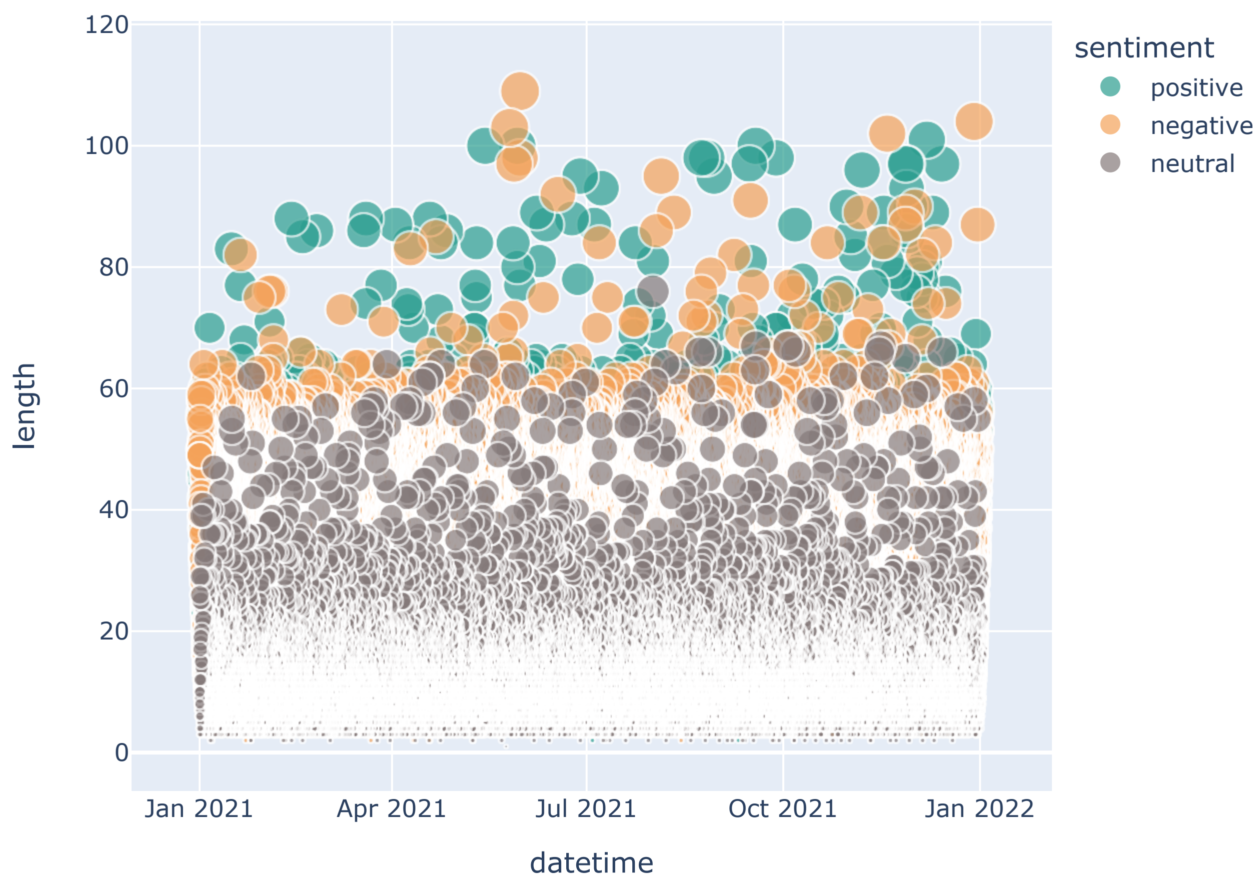
```
la_neg_wc_j6 = whyla[(whyla['datetime']>='2021-01-05') & (whyla['datetime']<='2021-01-08')
wordcloud = WordCloud(max_font_size=50, max_words=500, background_color='white').generate
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

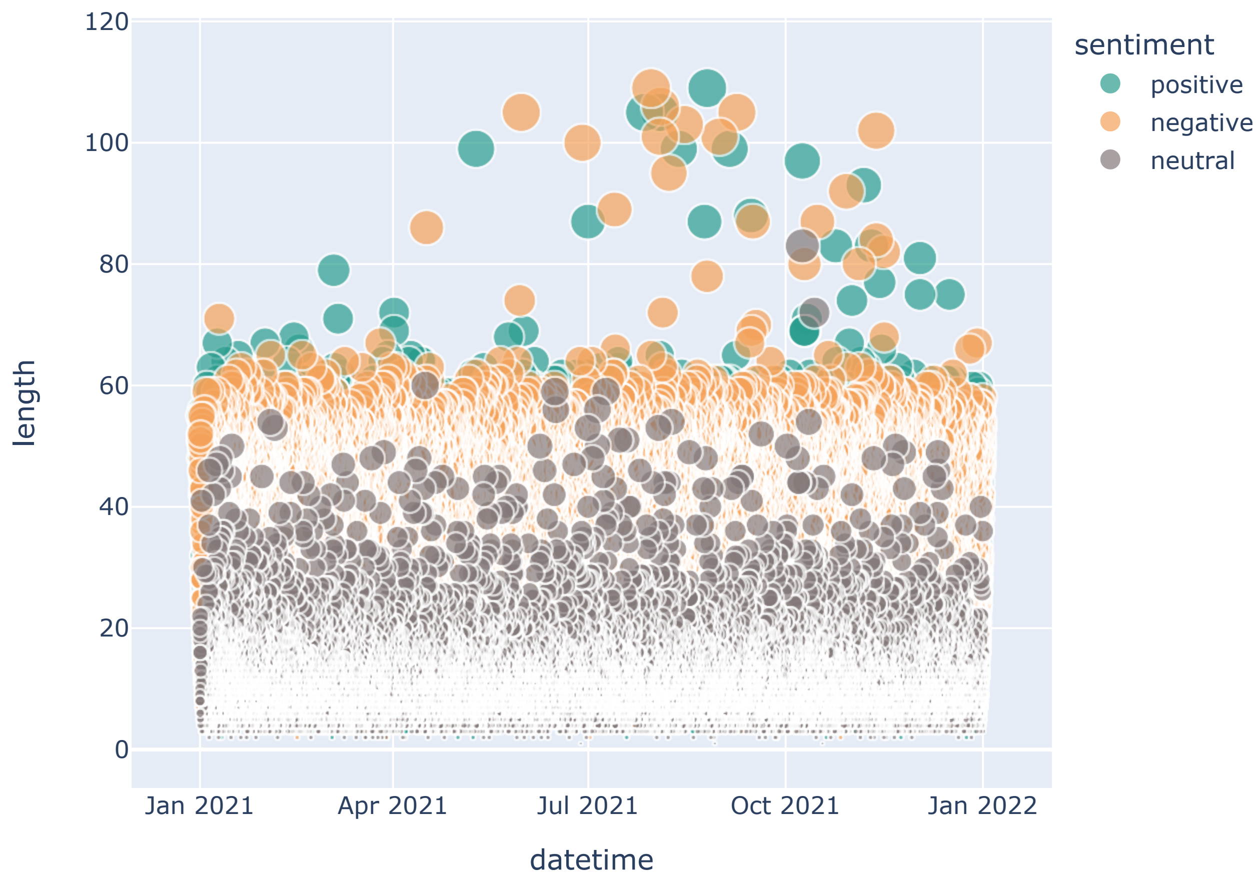
In [46]:

```
positive_wc_m29 = whynyc[(whynyc['datetime']>='2021-03-29') & (whynyc['datetime']<='2021-(
wordcloud = WordCloud(max_font_size=50, max_words=500, background_color='white').generate
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

In [47]:

```
def tweets_timeline_plotly(why_df):
    fig = px.scatter(why_df,
                     x='datetime',
                     y='length',
                     #range_y=[50, 120],
                     color_discrete_map = sentiment_colors,
                     color='sentiment',
                     size='length',
                     hover_name=why_df.text.apply(lambda txt: '<br>'.join(textwrap.wrap(txt, w
                    )
    return fig.show()

tweets_timeline_plotly(whynyc)
tweets_timeline_plotly(whyla)
```





Topic Modeling

Topic modeling is a text mining tool to reveal semantic structures of a body of text to reveal abstract topics that occur. It is a probabalistic model that will document which specific topic has certain words appearing more frequently than others. From the scikit-learn library, the Latent Dirichlet Allocation and TfidfVectorizer are used to build this model.

In [48]:

```
def why_lda_model(why_df):
    vectorizer = TfidfVectorizer(max_df=0.9, min_df=25, token_pattern='\w+|\$[\d\.]+|\S+',

    # apply transformation
    tf = vectorizer.fit_transform(why_df['corpora']).toarray()

    # tf_feature_names tells us what word each column in the matric represents
    tf_feature_names = vectorizer.get_feature_names()

    number_of_topics = 30

    model = LatentDirichletAllocation(n_components=number_of_topics, random_state=0)
    model.fit(tf)

    #creates a table of the topics and weight of the document
    def display_topics(model, feature_names, no_top_words):
        topic_dict = {}
        for topic_idx, topic in enumerate(model.components_):
            topic_dict['Topic %d words' % (topic_idx)]= ['{}'.format(feature_names[i])
```

```
                         for i in topic.argsort()[:-no_top_words - 1:-1]]
                topic_dict['Topic %d weights' % (topic_idx)]= ['{:.1f}'.format(topic[i])
                         for i in topic.argsort()[:-no_top_words - 1:-1]]
        return pd.DataFrame(topic_dict)

    no_top_words = 15
    return display_topics(model, tf_feature_names, no_top_words).T

whynyc_lda = why_lda_model(whynyc)
whyla_lda = why_lda_model(whyla)
```

In [49]:

```
def collect_topics(x, why_df):
    #creates a dataframe of the extracted topics from the previous cell box
    topic_df = pd.DataFrame()

    #extracts the first column of topics for every other row
    topic_df['topic'] = x.iloc[::2, :1].reset_index(drop=True)
    #extracts the first column of weights for every other row
    topic_df['weight'] = x.iloc[1::2, :1].reset_index(drop=True)
    #combines the other columns of topics sans the calculated weight
    topic_df['subtopics'] = x.iloc[::2, 1:].apply(lambda x: ', '.join(x[x.notnull()]), axi

    #calculates the overall average of sentiment (polarity) based on the topic extracted i
    values = []
    for word in topic_df['topic']:
        temp_list = why_df.loc[why_df['text'].str.contains(word, case=False)].reset_index
        mean_of_topic = temp_list['score'].mean()
        values.append(mean_of_topic)
        values = [0 if x != x else x for x in values]
        topic_df['sentiment'] = pd.DataFrame(values)
        topic_df = topic_df.sort_values(by=['weight'], ascending=False, ignore_index=True)

        #categorizw the sentiment based on score
        def sentiment_category(sentiment):
            label = ''
            if(sentiment>0):
                label = 'positive'
            elif(sentiment == 0):
                label = 'neutral'
            else:
                label = 'negative'
            return(label)

    topic_df['category'] = topic_df['sentiment'].apply(sentiment_category)
    return topic_df

whynyc_topics = collect_topics(whynyc_lda, whynyc)
whyla_topics = collect_topics(whyla_lda, whyla)
```

In [50]:

```
#to display all the items in the subtopics column
pd.set_option('display.max_colwidth', 0)
```

In [51]:

```
whynyc_topics
```

Out[51]:

	topic	weight	subtopics	sentiment	category
0	leave	84.1	like, alone, favorite, get, go, degree, therapy, dressed, felt, shoot, cannot, tiktok, slap, need	-0.277968	negative
1	like	645.9	look, look like, 😭, tweet, idk, tired, thought, waste, time, 😭😭😭😭, delete, like 😭, answer, better	0.057003	positive

	topic	weight	subtopics	sentiment	category
2	would	432.4	like, feel, feel like, rn, cannot, loud, nice, would want, lying, 😒, music, want, know, lmfaoooo	0.048527	positive
3	love	417.8	get, fall, said, morning, never, married, like, rid, follower, born, much, see, reason, woman	-0.043739	negative
4	know	407.8	want, mask, people, want know, wearing, wear, vaccinated, really, get, go, cannot, even, wearing mask, wear mask	0.279071	positive
5	still	368.9	figure, ya, bruh, thing, trying, like, angry, taste, yea, scared, get, tag, trying figure, jesus	0.028544	positive
6	tho	318.2	lmao, wtf, like, 😩, acting, good, acting like, birthday, fucking, shit, 😅, twitter, kill, fr	-0.035360	negative
7	lol	298.8	lmfao, cute, hard, nah, old, year old, year, lmaoooo, like, damn, blocked, monday, 😂😂😂😂, could	0.024331	positive
8	trump	291.7	vote, republican, people, u, state, democrat, get, biden, need, election, american, country, would, party	0.375361	positive
9	new	274.4	york, new york, hate, wait, people, brooklyn, like, much, dick, tweeting, uber, keep, goat, dm	-0.104492	negative
10	make	260.9	night, sense, make sense, last, last night, awake, business, like, mind, drunk, next, show, tv, think	-0.038797	negative
11	funny	245.3	yes, weird, tf, sleep, please, single, 😂, dream, 🤔, like, flight, horny, go, explain	0.005126	positive
12	need	227.6	like, sound, know, sound like, people, police, yelling, cop, asking, officer, feel, cat, help, saying	0.043599	positive
13	train	223.4	yankee, omg, bus, care, bike, get, would, exactly, 🤣🤣🤣, car, give, one, people, bitcoin	0.129061	positive
14	nigga	222.8	bitch, like, wonder, mad, 🤣, got, people, act, tryna, understand, smell, never, always, cannot	0.041052	positive
15	fuck	207.4	cold, th, see, would, back, going, money, like, short, come, win, cannot, dj, shoe	0.380031	positive
16	black	201.2	think, white, lie, people, always, woman, friend, would, 👀, like, know, black woman, white people, black people	0.099756	positive
17	thank	185.2	trending, ago, 😂😂, everyone, year, year ago, tl, looking, like, explain, face, cannot, watch, quit	0.224718	positive
18	anyone	182.6	expensive, hot, buy, album, would anyone, laughing, coffee, would, app, customer, cannot, apple, service, song	0.056771	positive
19	cry	182.3	yeah, keep, 🥺, medium, social, social medium, oh, call, hell, 🤨, sad, like, gym, posting	0.051902	positive
20	say	163.1	would, like, god, happen, ever, 🥲, ask, choose, gt, would ever, bad, someone, thing, something	-0.081617	negative
21	want	160.2	reason, never, obsessed, one, people, understood, get, see, nyc, never understood, like, 😒, lmfaooo, everything	0.046054	positive
22	game	151.8	time, like, one, people, vibe, play, cannot, around, fox, loved, get, joe, mess, basketball	-0.088285	negative
23	people	143.5	first, place, many, first place, get, would, sending, hear, like, year, cannot, one, got, time	-0.060194	negative

	topic	weight	subtopics	sentiment	category
24	question	142.2	would, surprised, people, trust, serious, say, 💀, talking, would say, one, get, laugh, make, 🤣🤣	0.053680	positive
25	bother	130.2	though, even, sure, right, know, park, perfect, craving, wing, like, called, even bother, nicki, lately	-0.005719	negative
26	playing	127.5	bro, 😂😂😂, like, seriously, hungry, news, happening, sexy, date, deserve, start, drake, team, official	0.003907	positive
27	men	121.6	😭😭😭, day, wake, like, exist, ad, school, work, gay, high, home, every, mother, dating	0.021006	positive
28	follow	120.6	vaccine, best, lmaooo, covid, like, dying, use, man, people, number, ice, would, one, mandate	0.050645	positive
29	one	101.2	way, tell, u, take, ugh, told, mean, cannot, time, shirt, stick, people, would, get	0.035458	positive

In [52]:

```
whyla_topics
```

Out[52]:

	topic	weight	subtopics	sentiment	category
0	old	99.9	reason, good, year old, one, year, big, cannot, trash, ❤️, would, idea, excuse, mention, people	0.039953	positive
1	😭😭	90.5	surprised, catch, service, 😌, get, hold, bus, time, store, emotional, rude, customer, leg, leaving	0.070636	positive
2	like	572.5	feel, feel like, want, people, know, medium, social, 🤣, people like, social medium, miss, idk, 👀, 😂😂😂😂	0.025305	positive
3	hating	54.9	jail, get, back, people, reason, help, tl, money, terrible, would, ice, horny, make, vote	-0.198236	negative
4	look	332.6	look like, like, night, last, single, last night, obsessed, cat, episode, listening, video, youtube, never, new	-0.170898	negative
5	lol	278.5	love, omg, ago, vaccine, idk, haha, covid, getting, laughing, friday, year, watching, ok, home	0.067624	positive
6	tho	247.8	wait, 😂😂😂, smh, lie, like, bro, craving, date, sudden, raining, beautiful, sexy, lol, going	0.004948	positive
7	would	210.4	game, play, team, player, drunk, win, accurate, get, want, time, everyone, waste, yeah, nba	0.033147	positive
8	funny	208.6	lmfao, ugh, lying, figure, trying, jesus, would, birthday, happy, angry, album, g, make, mask	0.146233	positive
9	people	192.6	cannot, would, way, choose, u, american, yelling, country, get, america, want, use, say, like	-0.031138	negative
10	cry	186.3	make, sense, 😂, make sense, cute, first, place, remind, first place, know, delete, need, app, thing	0.001955	positive
11	los	170.3	angeles, los angeles, california, hot, angeles california, los angeles california, bother, gay, like, ugly, explains, tweet, dream, post	-0.017948	negative
12	thank	162.9	😩, pay, understand, texas, 🥴, never understand, never, degree, hungry, like, outside, following, people, attention	0.022733	positive
13	white	154.6	sleep, trump, know, people, police, like, black, fat, cop, like 😂, going, cannot, racist, election	0.133784	positive

	topic	weight	subtopics	sentiment	category
14	lmao	148.9	like, 😭😭😭, tf, though, shit, 💀, really, keep, 😔, friend, lmaoo, got, act, always	0.133784	positive
15	n	146.5	😭, anyone, 😅, show, ask, tell, hate, would anyone, face, exist, graphically show, graphically show audiovisual, graphically, audiovisual	0.283310	positive
16	mad	145.4	every, see, every time, time, call, 🤨, like, one, happening, r, cannot, get, want, day	-0.042893	negative
17	go	138.1	😂😂, hard, 🤣🤣🤣, instagram, fr, like, stop, ppl, dodger, showing, got, everywhere, want go, mood	0.104447	positive
18	talking	137.0	drink, like, laugh, got, nobody, one, 😆, early, bag, broke, bout, called, know, cousin	-0.090194	negative
19	fuck	132.7	know, bruh, wear, ya, love, wit, shoe, short, ion, shit, like, much, tweeting, care	0.005987	positive
20	wonder	129.9	weird, hurt, even, playing, yet, people, 🤣🤣, get, taking, clue, like, hahaha, know, rid	0.067825	positive
21	yes	127.4	awake, cold, nigga, much, 🙄, traffic, la, hoe, always, dating, married, get, queen, still	0.061327	positive
22	sad	121.9	expensive, woman, men, lt, black, happen, gt, 🙃, as, kill, question, bad, grown, skin	0.385785	positive
23	sound	119.0	🥲, 🥺, sound like, like, suck, man, voice, song, dying, 🤦🏻‍♀️, stuck, along, head, know	0.267948	positive
24	car	114.1	like, trending, oh, text, drive, uber, god, driver, dat, excited, ride, yo, dawg, get	0.422365	positive
25	like	108.5	acting, mean, picture, taste, exactly, take, 😐, acting like, fly, looking, brand, beat, day, anxiety	0.017654	positive
26	never	105.9	wake, would, want, screaming, curious, one, understood, hear, vaccinated, like, know, seen, get, pizza	0.129670	positive
27	get	103.7	still, sure, say, nice, lmaoooo, cannot, people, 😳, best, republican, stupid, crypto, another, need	-0.270488	negative
28	favorite	103.5	school, people, many, wearing, would, one, mask, high, long, spend, know, say, time, would say	0.043717	positive
29	think	102.9	let, u, loud, like, lol, drinking, let u, girl, 🤦🏽‍♂️, like lol, bring, would, tear, quiet	0.280529	positive

Topic Bubble Map

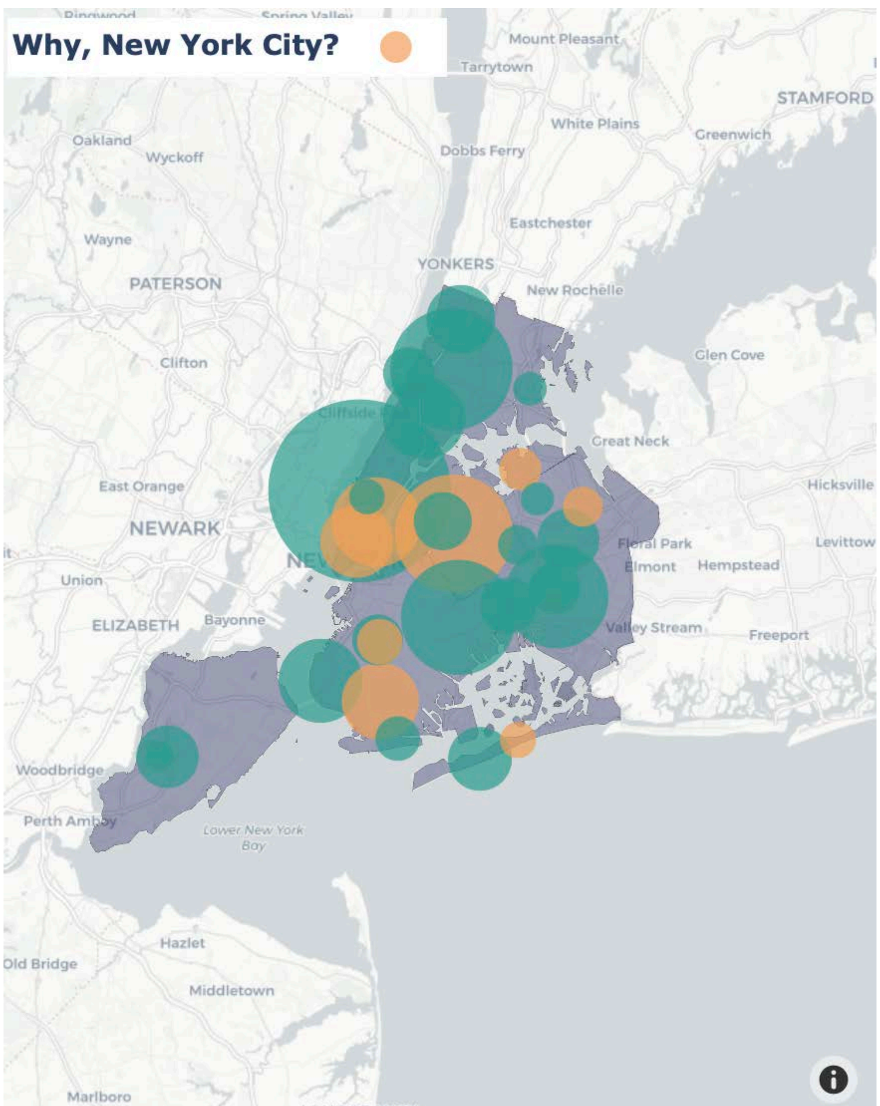
This data visualization creates a topic bubble map based on the results in the *Topic Modeling* section. Geopandas is used to generate a set of random latitude and longitude coordinates within the geographical boundaries of the area of interest. Plotly is used to graph everything together.

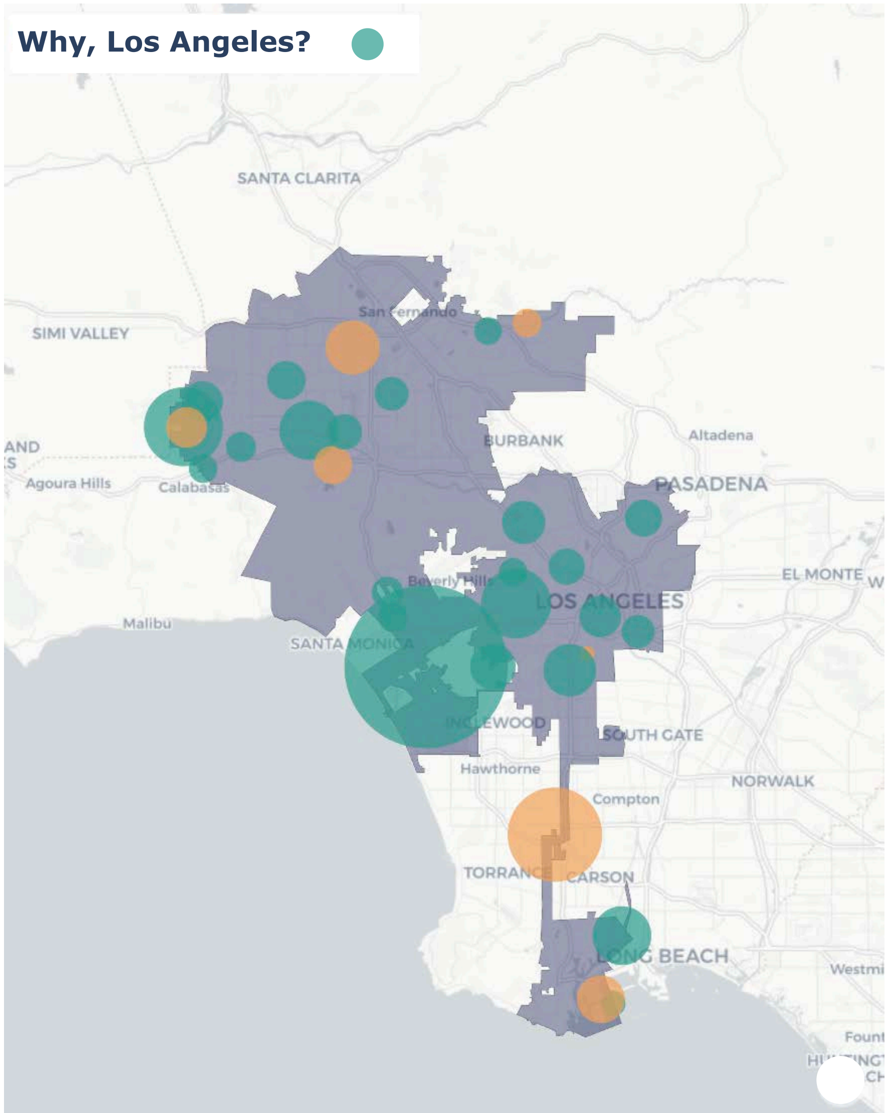
In [55]:

```
def topic_map(file_name, why_df, export_name, title):
    #read GeoJSON file
    gdf_polys = gpd.read_file(file_name)

    # find the bounds of your geodataframe
    x_min, y_min, x_max, y_max = gdf_polys.total_bounds
    # set sample size
    n = 100
```

```
# generate random data within the bounds
x = np.random.uniform(x_min, x_max, n)
y = np.random.uniform(y_min, y_max, n)

# convert them to a points GeoSeries
gdf_points = gpd.GeoSeries(gpd.points_from_xy(x, y))
# only keep those points within polygons
gdf_points = gdf_points[gdf_points.within(gdf_polys.unary_union)]

#reset thet index of both dataframes and add the lat and lon columns
gdf_points = gdf_points.reset_index(drop=True)
why_df = why_df.reset_index(drop=True)
why_df['lon'] = gdf_points.geometry.apply(lambda p: p.x)
why_df['lat'] = gdf_points.geometry.apply(lambda p: p.y)

#create the geographical scatter plot
fig = px.scatter_mapbox(
    why_df,
    lat=why_df['lat'],
    lon=why_df['lon'],
    zoom=8.5,
    hover_name=why_df['topic'],
    text = why_df.subtopics.apply(lambda txt: '<br>'.join(textwrap.wrap(txt, width=35)
    width = 600,
    height = 700,
)

#set color of scatter points
def SetColor(x):
    if(x < 0):
        return '#F4A259'
    elif(x == 0):
        return '#847979'
    elif(x > 0):
        return '#2A9D8F'

#set scatter points
fig.update_traces(
    mode='markers+text',

    marker=dict(
        color= list(map(SetColor, why_df['sentiment'])),
        size=why_df['weight'].astype(float)/7,
    ),

    showlegend=True,
    hovertemplate= '<b>Topic: '+ why_df.topic + '</b><br><br>'
        + 'The associated words with this topic are: <br>%{text}<br><br>'
        + 'The overall sentiment is '
        + why_df.category + '.',
)

#update and customize map
fig.update_layout(
    mapbox = {
        'style': 'carto-positron',
        'layers': [
            {
            'source': file_name,
            'type': 'fill',
                'below': 'traces',
                'color': '#111954',
                'opacity': 0.4,
                'line': {'width': 5}
            }
        ]},
```

```
        hoverlabel=dict(
            bgcolor='white',
            bordercolor='white',
            font=dict(color='black'),
            font_size=12,
            font_family='Helvetica',
            align='left'
        ),
        legend_title_text = '<b>'+ title + '</b>',
        legend=dict(
            orientation='h',
            yanchor='top',
            y=0.99,
            xanchor='left',
            x=0.01
        ),

    )

    #write Plotly graph to an HTML file
    fig.write_html(export_name, full_html=False, include_plotlyjs='cdn')

    fig.show()
    return topic_map

nyc_map = topic_map('nyc.geojson', whynyc_topics, 'nyc.html', 'Why, New York City?')
la_map = topic_map('la.geojson', whyla_topics, 'la.html', 'Why, Los Angeles?')
```





Accuracy of Sentiment Analysis

This purpose of this section is to determine whether or not the methodology behind the sentiment analysis is accurate using Logisitic Regression and Multinomial Naive Bayes from the sklearn library.

For a recap, to get both the subjectivity and polarity scores on tweets, the NLTK and TextBlob libraries were used. NLTK does not have a number associate with their subjectivity library; therefore, the mean was calculated for the NLTK and TextBlob polarity scores. Lastly, this was converted into 'positive' (scores above 0), 'negative' (scores less than 0) and 'neutral' (scores equal to 0) values. A manual look-through of the dataset, there are some tweets that appear to be miscategorized between the sentiment categories.

In [56]:
```python
def model_accuracy(df_name, why_df):
    X_train, X_test, y_train, y_test = train_test_split(
        why_df['text'],
        why_df['sentiment'],
        test_size=0.2,
        random_state=24)

    vectorizer = TfidfVectorizer(ngram_range=(1,3), stop_words='english')
    X_train = vectorizer.fit_transform(X_train)
    X_test = vectorizer.transform(X_test)

    #Logistic Regression
    lr = LogisticRegression()
    lr.fit(X_train, y_train)

    predictions = lr.predict(X_test)
    confusion_matrix(predictions,y_test)
    print(classification_report(predictions,y_test))

    accuracy = metrics.accuracy_score(predictions, y_test)
    print(str('For the ' + df_name + ' dataset, the accuracy for Logistic Regression with

    #Naive Bayes (Multinomial)
    mnb = MultinomialNB()
    mnb.fit(X_train,y_train)
    predictions_mnb = mnb.predict(X_test)
    confusion_matrix(predictions_mnb,y_test)
    print(classification_report(predictions_mnb,y_test))
    accuracy_mnb = metrics.accuracy_score(predictions_mnb, y_test)

    return print(str('For the ' + df_name + ' dataset, the accuracy for Multinomial Naive

model_accuracy('whynyc', whynyc)
model_accuracy('whyla', whyla)
```

```
              precision    recall  f1-score   support

    negative       0.77      0.78      0.77     10299
     neutral       0.58      0.83      0.68      3419
    positive       0.87      0.77      0.82     14037

    accuracy                           0.78     27755
   macro avg       0.74      0.79      0.76     27755
weighted avg       0.80      0.78      0.79     27755

For the whynyc dataset, the accuracy for Logistic Regression with TfidfVectorizer is 78.1
8%
              precision    recall  f1-score   support

    negative       0.51      0.81      0.62      6488
     neutral       0.01      0.97      0.01        29
    positive       0.95      0.56      0.71     21238

    accuracy                           0.62     27755
   macro avg       0.49      0.78      0.45     27755
weighted avg       0.85      0.62      0.69     27755

For the whynyc dataset, the accuracy for Multinomial Naive Bayes with TfidfVectorizer is 6
1.97%
              precision    recall  f1-score   support

    negative       0.76      0.79      0.77      6507
     neutral       0.56      0.81      0.66      2143
    positive       0.87      0.76      0.81      9209
```

```
    accuracy                           0.78     17859
   macro avg       0.73      0.79      0.75     17859
weighted avg       0.79      0.78      0.78     17859

For the whyla dataset, the accuracy for Logistic Regression with TfidfVectorizer is 77.66%
              precision    recall  f1-score   support

    negative       0.51      0.83      0.63      4126
     neutral       0.01      1.00      0.02        25
    positive       0.96      0.56      0.71     13708

    accuracy                           0.62     17859
   macro avg       0.49      0.80      0.45     17859
weighted avg       0.85      0.62      0.69     17859

For the whyla dataset, the accuracy for Multinomial Naive Bayes with TfidfVectorizer is 6
2.22%
```