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 trend and cultural differences, the overall sentiment of tweets, and areas of interest to tweeters.

Contents

Install libraries

pip install searchtweets-v2pip install nltknltk.download('stopwords')nltk.download('punkt')#to import nltk 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
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

In [2]:
```
#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

In [3]:
```
query_whynyc = twitter.gen_request_parameters(
    query = 'why (place:01a9a39529b27f36 OR place:011add077f4d2da3 OR place:00c39537733fa112 OR place:002e24c67
    results_per_call = 500,
    start_time = '2021-01-01',
    end_time = '2022-01-01',
    tweet_fields = 'id,created_at,text,geo',
    granularity=''
)

#140k
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-_]+)', tweet)

def find_mentioned(tweet):
    #function finds and extracts the twitter handles of people mentioned
    return re.findall('(?<!RT\s)(@[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 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]:
```
#removing links, mentions, hashtags from corpora column
stopwords = nltk.corpus.stopwords.words('english')
lemmatizer = WordNetLemmatizer()
punctuation = string.punctuation #'!"$%&\'()*+,-./:;<=>?[\\]^_`{|}~•@'
symbol = '...•"""' #for symbols not captured in punctuation

def clean_df(tweet):
    #remove parts of a tweet
    tweet = re.sub(r'(http\S+)', '', tweet)
    tweet = re.sub(r'bit.ly/\S+', '', tweet)
    #tweet = re.sub(r'(RT\s@[A-Za-z0-9-_]+[A-Za-z0-9-_]+)', '', tweet)
    tweet = re.sub(r'(@[A-Za-z0-9-_]+[A-Za-z0-9-_]+)', '', tweet)
    #tweet = re.sub(r'(#[A-Za-z0-9-_]+[A-Za-z0-9-_]+)', '', tweet)

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

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

    #remove contractions
    tweet = contractions.fix(tweet)

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

    #remove punctuation
    tweet = re.sub(r'[' + 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]:
```
#pull list of columns
list(whynyc_df)
```

Out[9]:
```
['id',
 'text',
 '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', 'hashtag', 'links']]
whyla_pot = whyla_df[['mentioned', 'hashtags', 'links']]
```

Categorizing Sentiment on 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')

    #polling 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%)

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

Out[13]:
```
138886
```

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

Out[14]:
```
89344
```

In [15]:
```
whynyc_pot['links'].value_counts()
```

Out[15]:
```
[]                                97361
[https://t.co/CiyzXjOgTz]            16
[https://t.co/UW7TkMtUIM]            13
[https://t.co/SPAVAK2eTW]            12
[https://t.co/Ef0jFoFAbo]            11
                                   ...
[https://t.co/D3QdYw2qGe]             1
[https://t.co/slR3zFrGH5]             1
[https://t.co/9ttmJbtdm5]             1
[https://t.co/dPxLlQNyXM]             1
[https://t.co/n0Mf9AhPBK]             1
Name: links, Length: 41321, dtype: int64
```

In [16]:
```
whynyc_pot['mentioned'].value_counts()
```

Out[16]:
```
[]                                                  79110
[@NYCTSubway]                                         312
[@YouTube]                                            174
[@MTA]                                                129
[@nypost]                                             101
                                                    ...
[@VP, @lizshuler, @AFLCIO]                              1
[@PaulMcG1994, @DiMarco_Pattaya]                        1
[@operaqueenie]                                         1
[@dcboyisangry, @jondunnsays, @JoyceWhiteVance]         1
[@cbaibix]                                              1
Name: mentioned, Length: 45930, dtype: int64
```

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

Out[17]:
```
[]                                     126520
[#RHOP]                                    85
[#Yankees]                                 65
[#LGM]                                     62
[#RHOA]                                    54
                                         ...
[#jasonblack]                               1
[#AmcManhattan]                             1
[#WWNXT, #TakeOver]                         1
[#RecallRonDeSantis, #RecallAbbott]         1
[#Stupidity]                                1
Name: hashtags, Length: 9164, dtype: int64
```

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

Out[18]:
```
[]                                                          64696
[https://t.co/H4JxLhbZow]                                      49
[https://t.co/QNaC8UUGfn]                                      19
[https://t.co/S5YKpBwbVm]                                      10
[https://t.co/Ob6sf4xrpK]                                       7
                                                             ...
[https://t.co/r4To8Aon3G]                                       1
[https://t.co/foF3Df4BcH]                                       1
[https://t.co/91jGhMWQyl]                                       1
[https://t.co/BRiGLEVktk, https://t.co/6zIFe7BZjU]              1
[https://t.co/dqnarpbqIs]                                       1
Name: links, Length: 24498, dtype: int64
```

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

Out[19]:
```
[]                                                       54450
[@YouTube]                                                 188
[@GavinNewsom]                                              44
[@Dodgers]                                                  43
[@thehill]                                                  43
                                                           ...
[@Nothing22280589, @londongal_28, @OccupyDemocrats]          1
[@amandaapittman]                                            1
[@RealRaider2055]                                            1
[@mikejschaefer]                                             1
[@BookSyrup]                                                 1
Name: mentioned, Length: 28525, dtype: int64
```

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

Out[20]:
```
[]                                                  82567
[#Dodgers]                                             69
[#FreeBritney]                                         48
[#RHOBH]                                               26
[#BB23]                                                25
                                                      ...
[#COVID, #vaxxing, #masking, #science]                  1
[#OscarIsaacs, #foxstudios, #xmenmovies]                1
[#DemonsSouls, #ps5]                                    1
[#ClimateAction, #StandWithGavin, #VoteNoOnRecall]      1
[#pointlessskip]                                        1
Name: hashtags, Length: 5539, dtype: int64
```

Text Analysis

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()
```

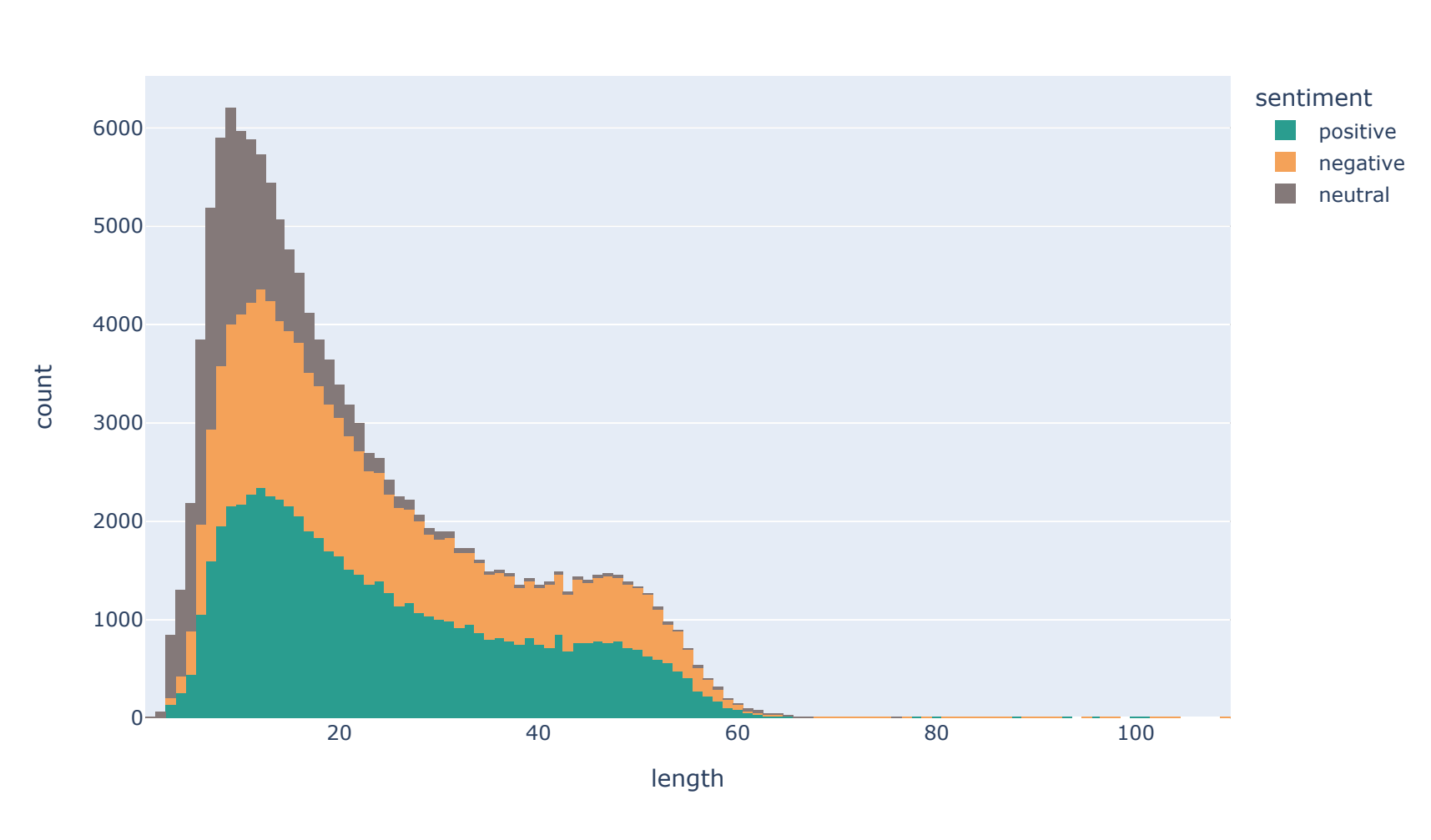
Out[22]:
```
count    138886.000000
mean         22.437315
std          14.227934
min           1.000000
25%          11.000000
50%          18.000000
75%          31.750000
max         109.000000
Name: length, dtype: float64
```

In [23]:
```
whyla['length'].describe()
```

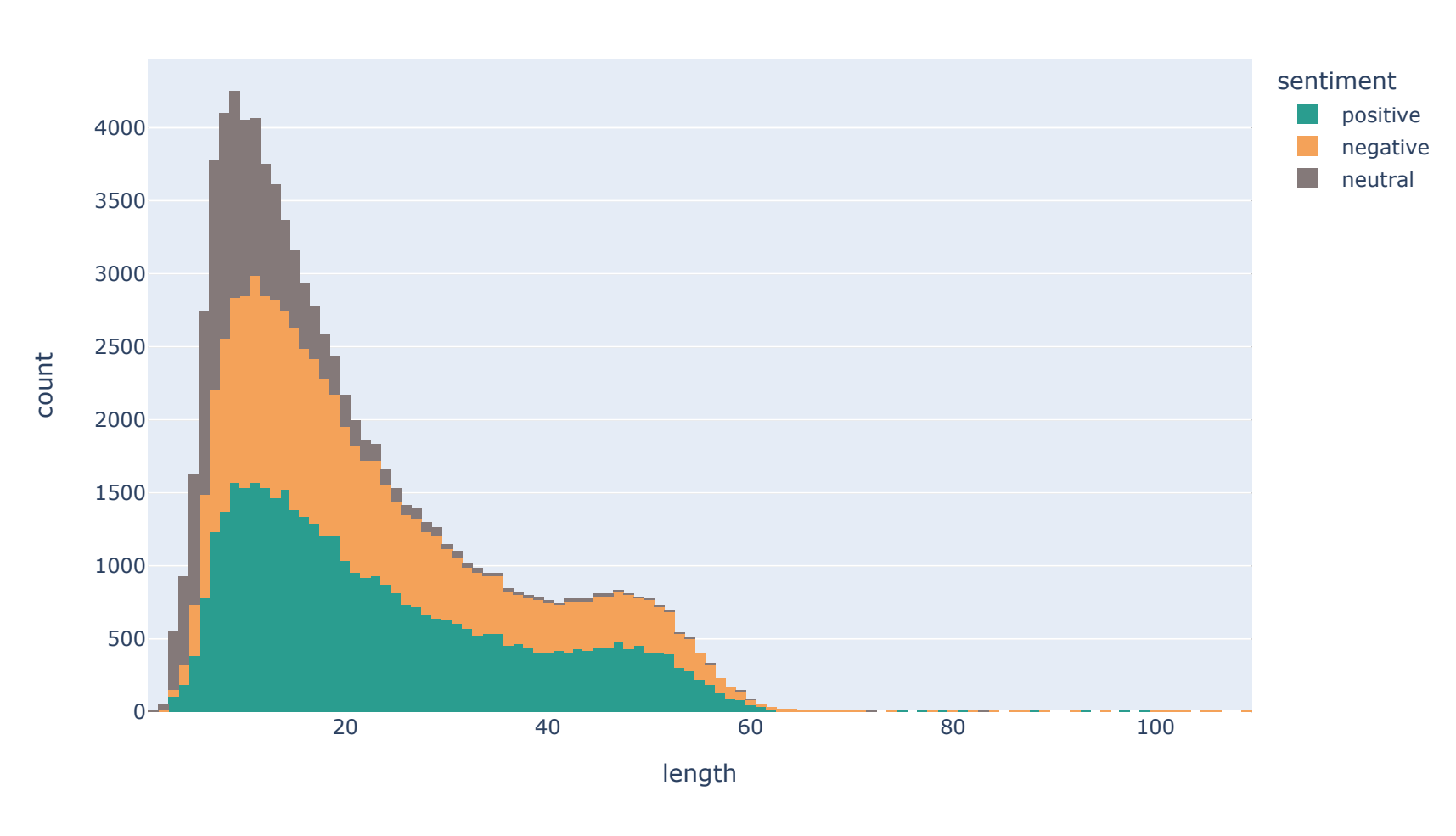
Out[23]:
```
count    89344.000000
mean        21.401986
std         13.847466
min          1.000000
25%         11.000000
50%         17.000000
75%         29.000000
max        109.000000
Name: length, dtype: float64
```

In [24]:
```
sentiment_colors = {
    'positive': '#2A9D8F',
    'neutral': '#847979',
    'negative': '#F4A259'}
```

In [25]:
```
px.histogram(whynyc, x='length', color='sentiment', color_discrete_map = sentiment_colors)
```

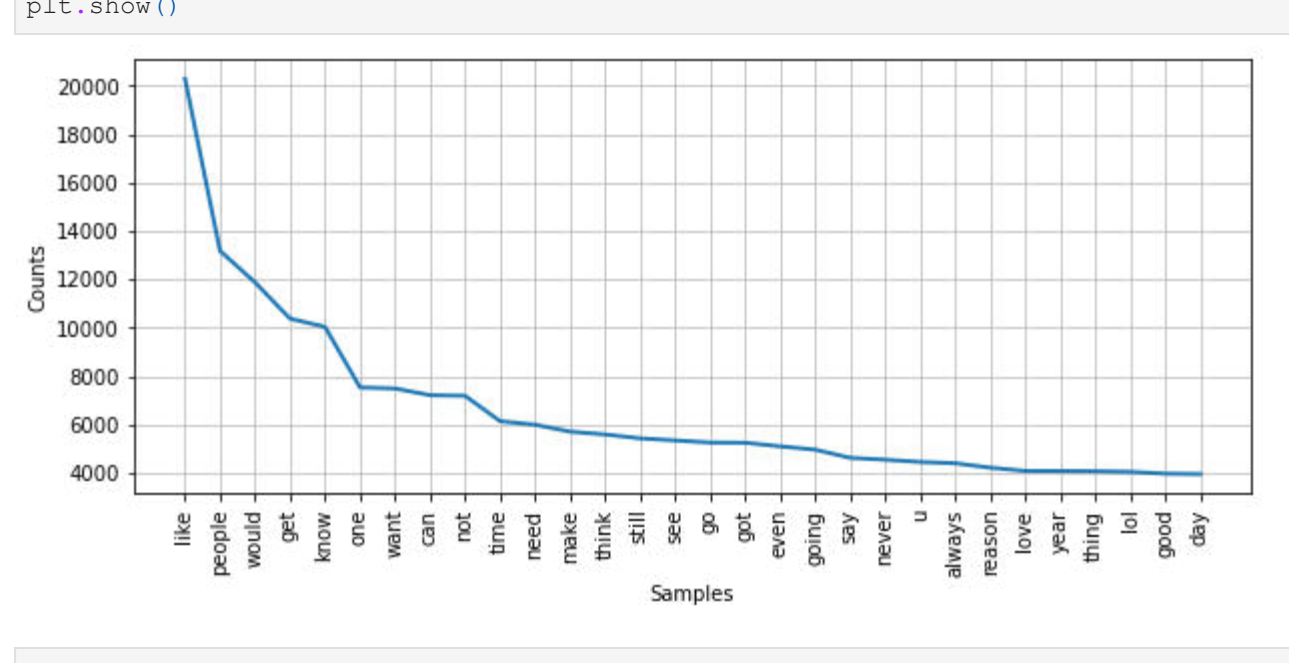


In [26]:
```
px.histogram(whyla, x='length', color='sentiment', color_discrete_map = sentiment_colors)
```

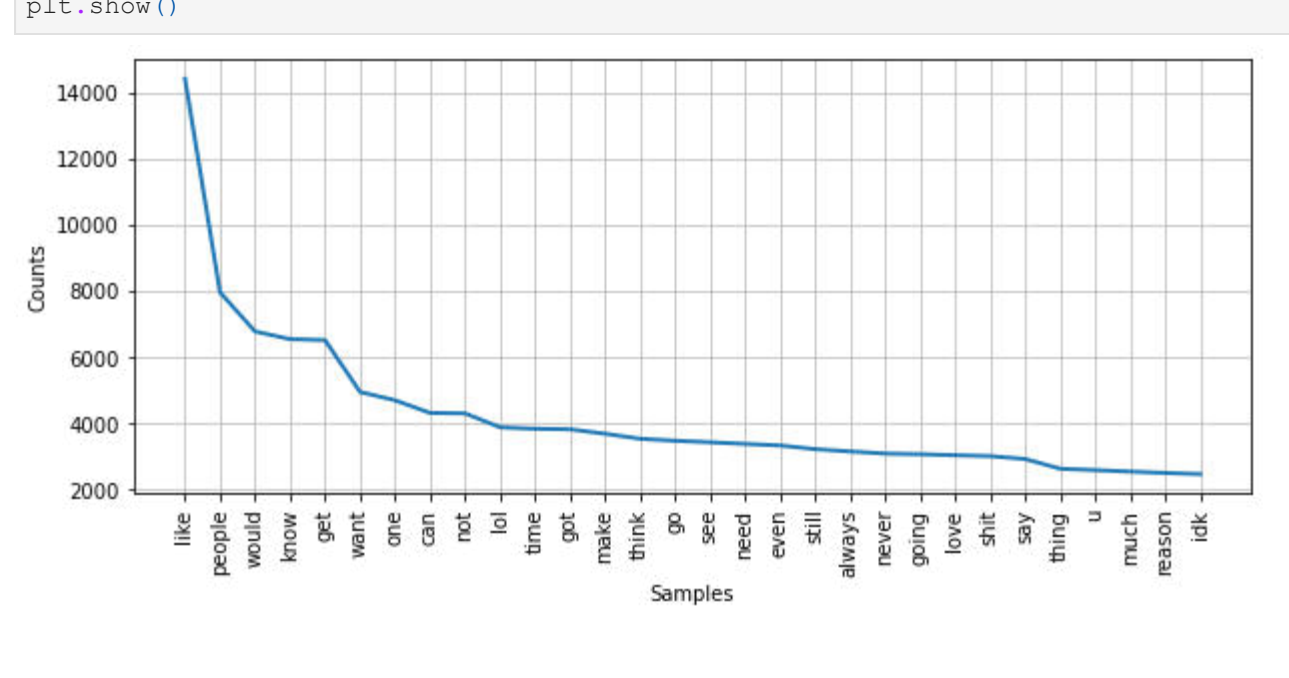


In [27]:
```
#convert dataframe to lists
whynyc_list = whynyc['corpora'].values.tolist()
whynyc_list = ' '.join(whynyc_list).lower()

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

In [30]:
```
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]))

    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

In [31]:
```
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 [32]:
```
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', 'frequency', 'quadgrams', 'f
    ngrams
    return ngrams

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

In [33]:
```
ngrams_nyc
```

Out[33]:

	unigrams	frequency	bigrams	frequency	trigrams	frequency	quadgrams	frequency
0	like	3529.134437	look like	663.753302	new york city	151.470388	new york new york	156.424098
1	people	2155.107287	new york	642.976160	new york new	115.670007	news network elected official	31.975772
2	know	1956.359802	feel like	507.129674	york new york	112.996076	network elected official silent	31.975772
3	want	1510.708458	make sense	398.085112	brooklyn new york	87.817685	elected official silent obvious	31.975772
4	lol	1222.646746	want know	314.946995	make make sense	78.753550	official silent obvious miscarriage	31.975772
5	time	1179.093271	people like	215.473221	manhattan new york	56.847758	silent obvious miscarriage justice	31.975772
6	got	1170.154142	year old	214.833320	gt gt gt	36.189982	obvious miscarriage justice social	31.975772
7	need	1164.732432	sound like	195.317658	idk feel like	33.265314	miscarriage justice social security	31.975772
8	think	1153.762136	year ago	187.132847	really want know	31.393852	justice social security irs	31.975772
9	make	1112.067675	social medium	178.868629	news network elected	31.205158	social security irs administration	31.975772
10	love	1060.079614	black people	160.050376	network elected official	31.205158	security irs administration ssi	31.975772
11	say	1031.997556	understand people	159.638005	elected official silent	31.205158	irs administration ssi veteran	31.975772
12	going	1013.232131	like know	135.994399	official silent obvious	31.205158	administration ssi veteran deserve	31.975772
13	reason	943.120180	acting like	133.882774	silent obvious miscarriage	31.205158	ssi veteran deserve date	31.975772
14	good	901.036878	people think	133.700021	obvious miscarriage justice	31.205158	veteran deserve date expect	31.975772

In [34]:
```
ngrams_la
```

Out[34]:

	unigrams	frequency	bigrams	frequency	trigrams	frequency	quadgrams	frequency
0	like	2499.620279	look like	423.889082	los angeles california	248.712247	lt lt lt lt	55.803544
1	people	1344.985599	feel like	353.107215	cosmos graphically audiovisual	100.248971	graphically audiovisual face race	32.907557
2	know	1303.699416	los angeles	306.655385	lt lt lt	57.832755	audiovisual face race age	32.907557
3	lol	1107.359648	make sense	237.456033	make make sense	55.159320	face race age nationality	32.907557
4	want	1004.126975	angeles california	178.055481	graphically audiovisual face	31.199730	race age nationality exact	32.907557
5	got	862.895499	want know	165.832352	audiovisual face race	31.199730	age nationality exact location	32.907557
6	love	781.008396	people like	145.078652	face race age	31.199730	cosmos graphically audiovisual face	32.700657
7	think	755.096647	sound like	138.559658	race age nationality	31.199730	nationality exact location everybody	27.344379
8	time	748.275575	year old	131.546071	age nationality exact	31.199730	los angeles hollywood california	13.905531
9	make	727.414897	social medium	118.785631	nationality exact location	31.199730	al haqq nur graphically	11.941129
10	idk	715.780248	year ago	110.466998	exact location everybody	25.998278	haqq nur graphically audiovisual	11.941129
11	shit	713.766732	understand people	98.619342	idk feel like	24.236062	cosmos graphically audiovisual body	11.901923
12	need	683.379039	graphically audiovisual	88.351261	gt gt gt	19.437893	guest catch live weeknight	9.367749
13	say	670.838977	people think	83.344778	make feel like	17.547511	catch live weeknight et	9.199203
14	going	659.337095	cosmos graphically	82.137094	today feel like	17.176137	south los angeles california	9.098979

Sentiment EDA

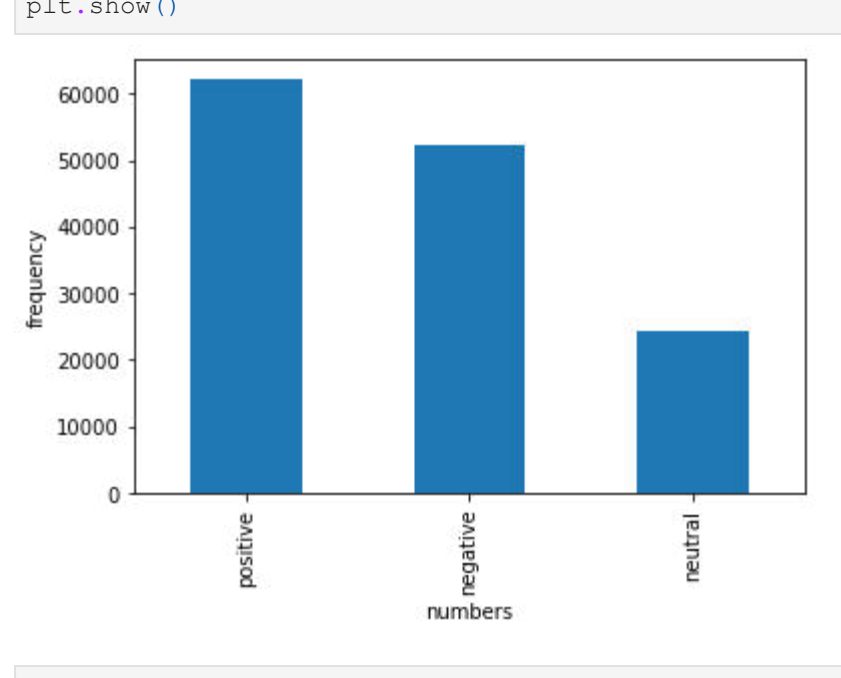
In [35]:
```
whynyc['score'].describe()
```

Out[35]:
```
count    138886.000000
mean          0.030624
std           0.331123
min          -0.991750
25%          -0.186600
50%           0.000000
75%           0.248782
max           0.982300
Name: score, dtype: float64
```

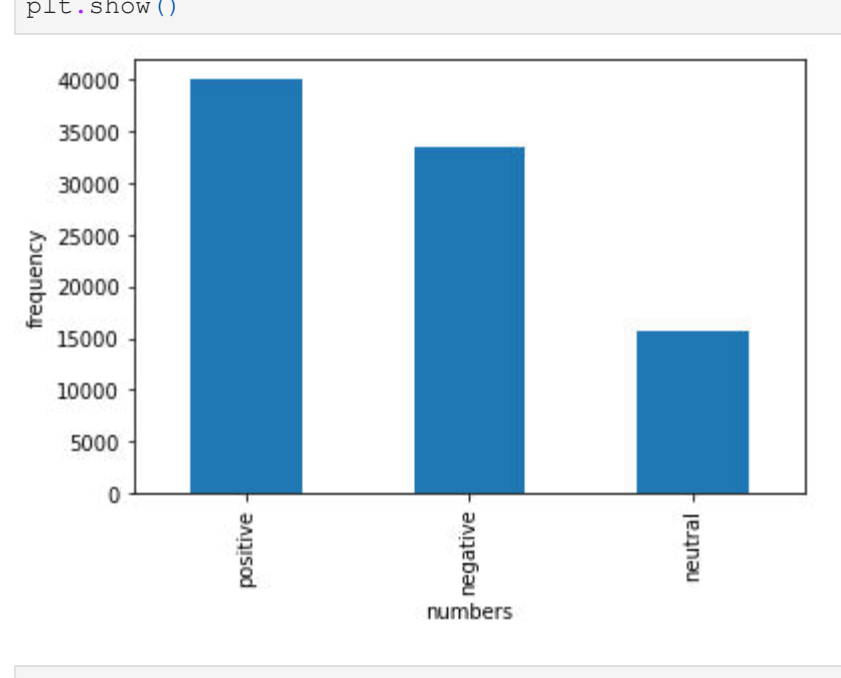
In [36]:
```
whynyc['sentiment'].describe()
```

Out[36]:
```
count       138886
unique           3
top       positive
freq         62204
Name: sentiment, dtype: object
```

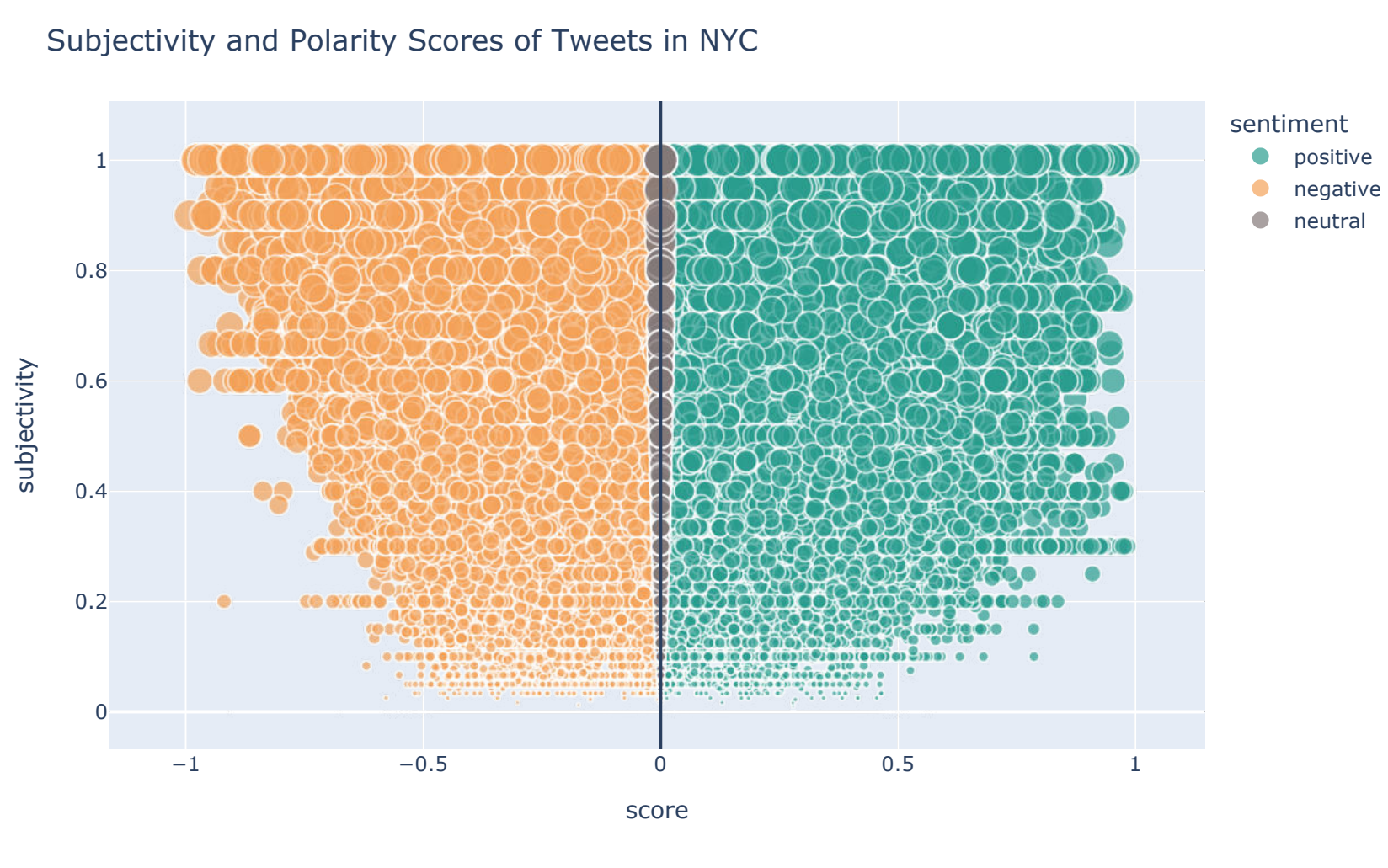
In [37]:
```
fig, ax = plt.subplots()
whynyc['sentiment'].value_counts().plot(ax=ax, kind='bar', xlabel='numbers', ylabel='frequency')
plt.show()
```

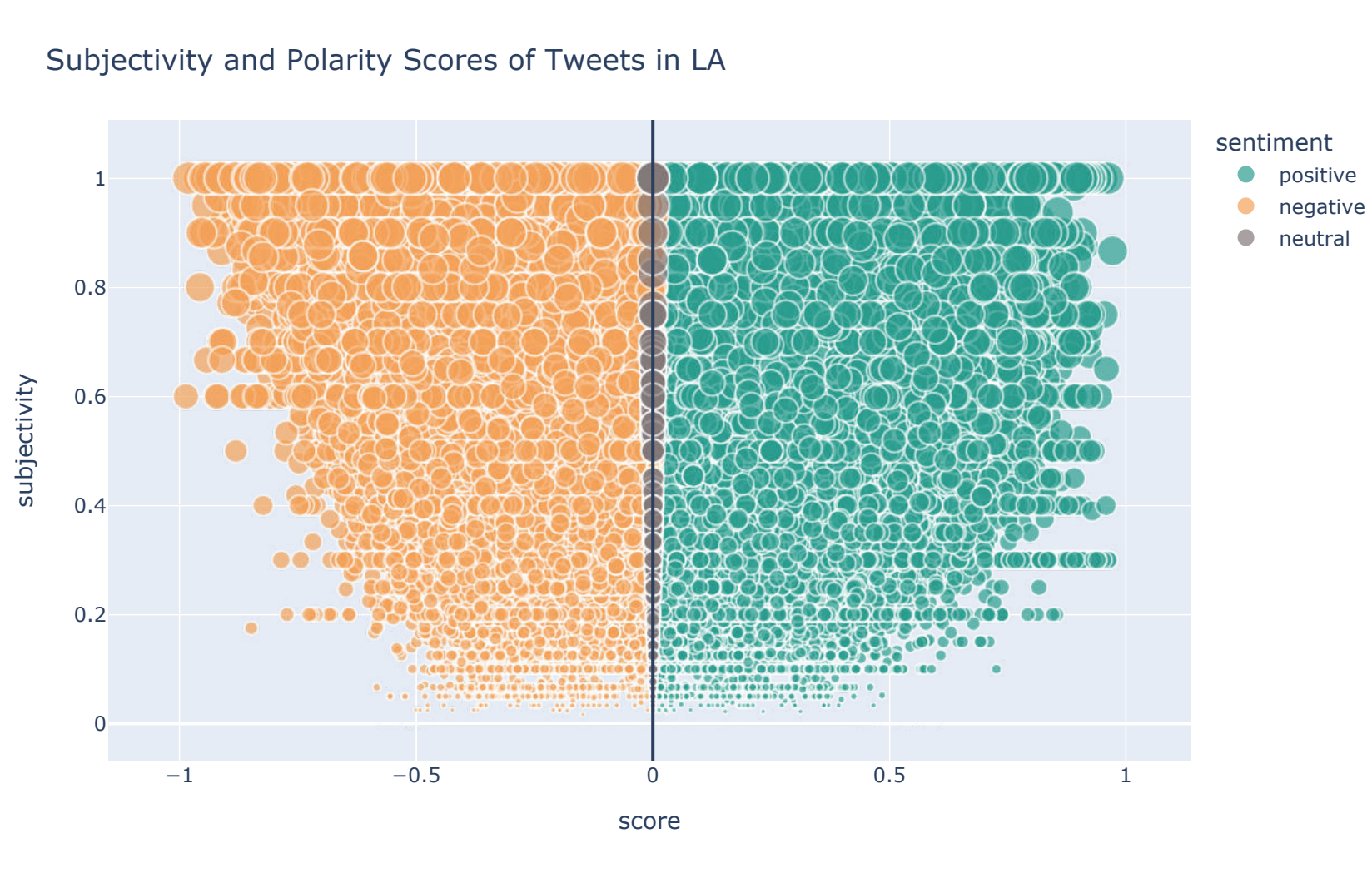


In [38]:
```
fig, ax = plt.subplots()
whyla['sentiment'].value_counts().plot(ax=ax, kind='bar', xlabel='numbers', ylabel='frequency')
plt.show()
```



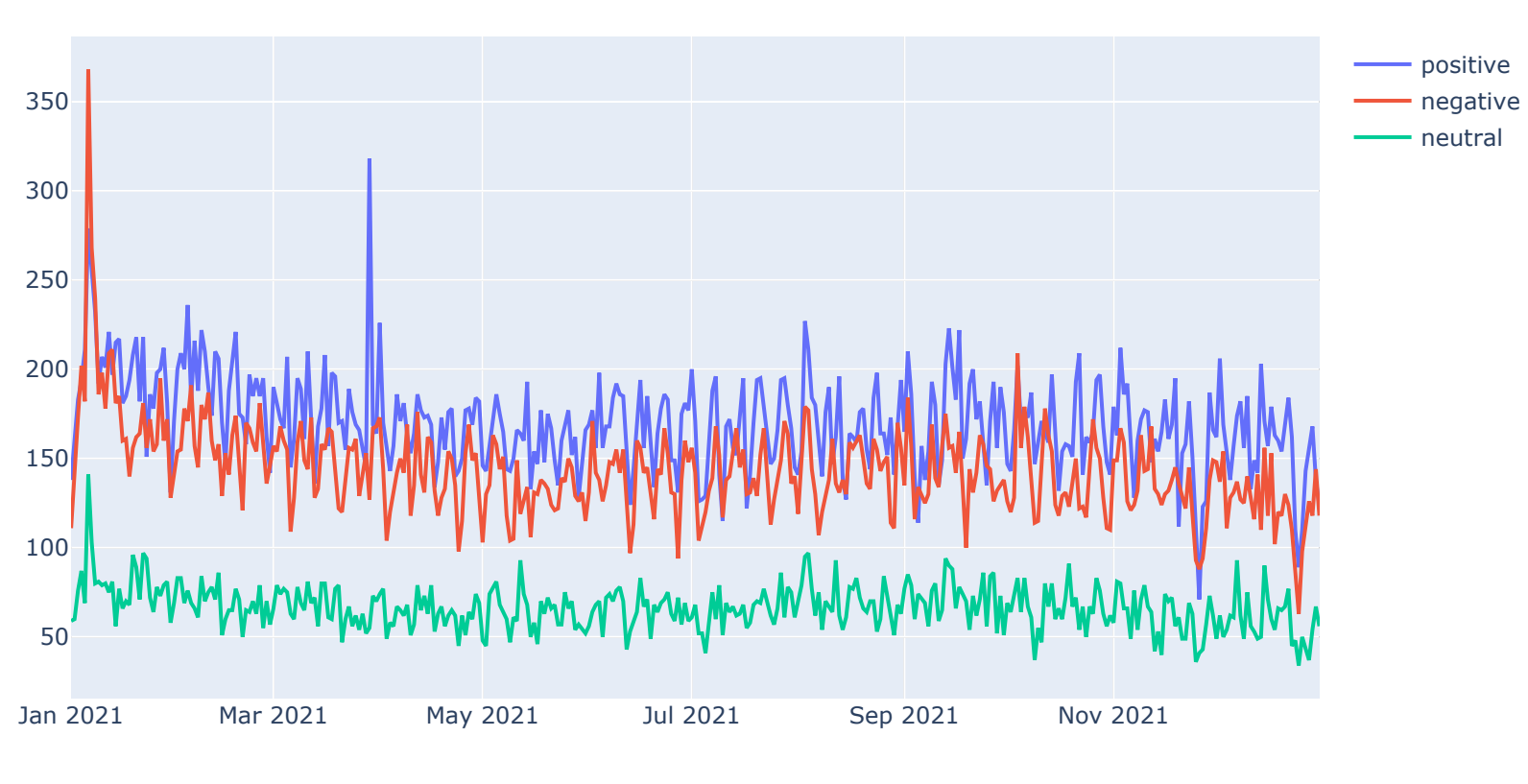
In [39]:
```
def pol_and_sub_of_tweets(title, why_df):
    # plot the polarity and subjectivity
    fig = px.scatter(why_df,
                     x='score',
                     y='subjectivity',
                     color = 'sentiment',
                     color_discrete_map = sentiment_colors,
                     size='subjectivity',
                     hover_name=why_df.text.apply(lambda txt: '<br>'.join(textwrap.wrap(txt, width=35))))

    #add a vertical layout via 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)
```

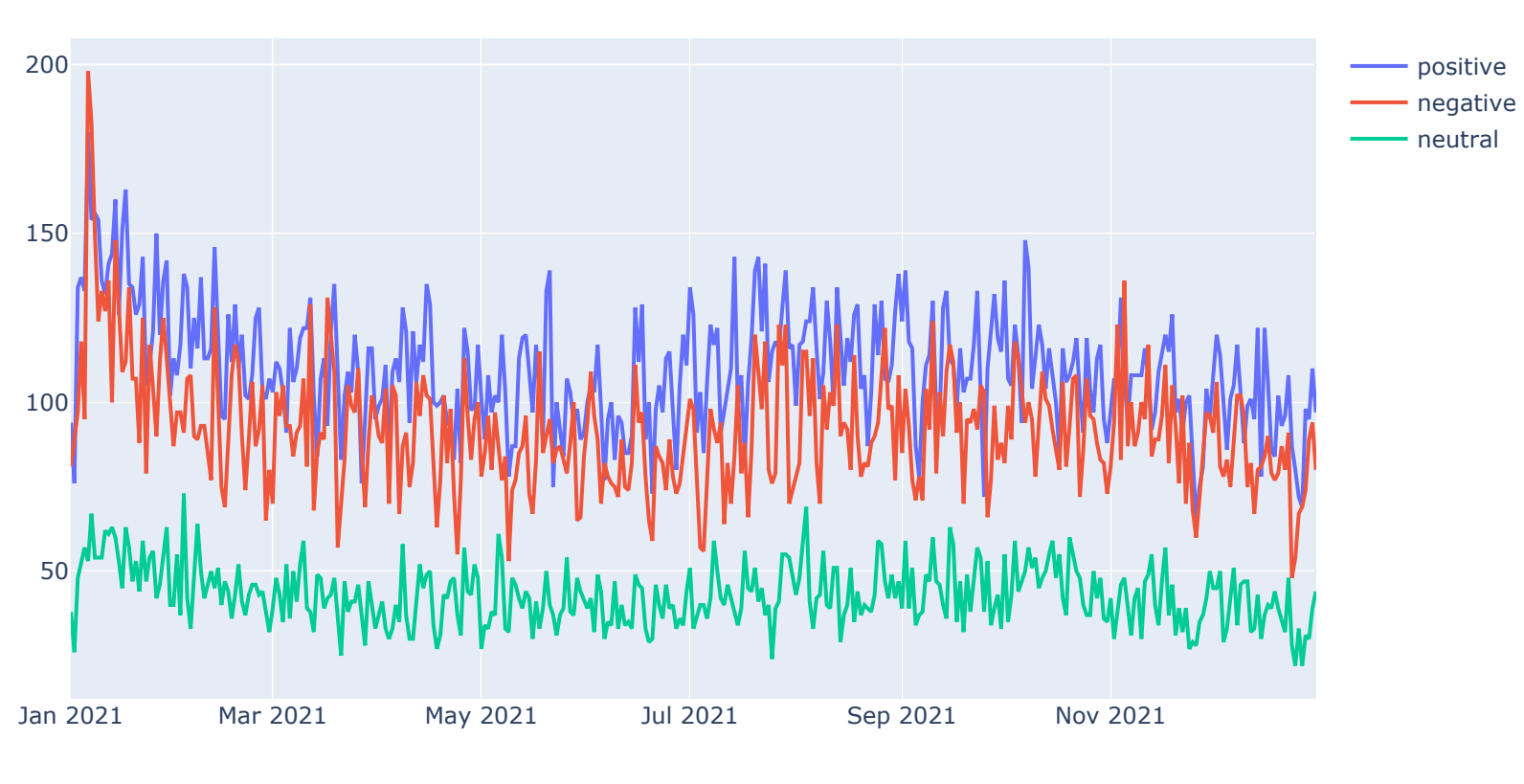




In [40]:
```
def sentiment_count(why_df):
    sent_count = why_df.groupby('day').sentiment.value_counts()
    sent_count = sent_count.to_frame(name='count')
    sent_count.reset_index(inplace=True)
    return sent_count

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

In [41]:
```
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)
```



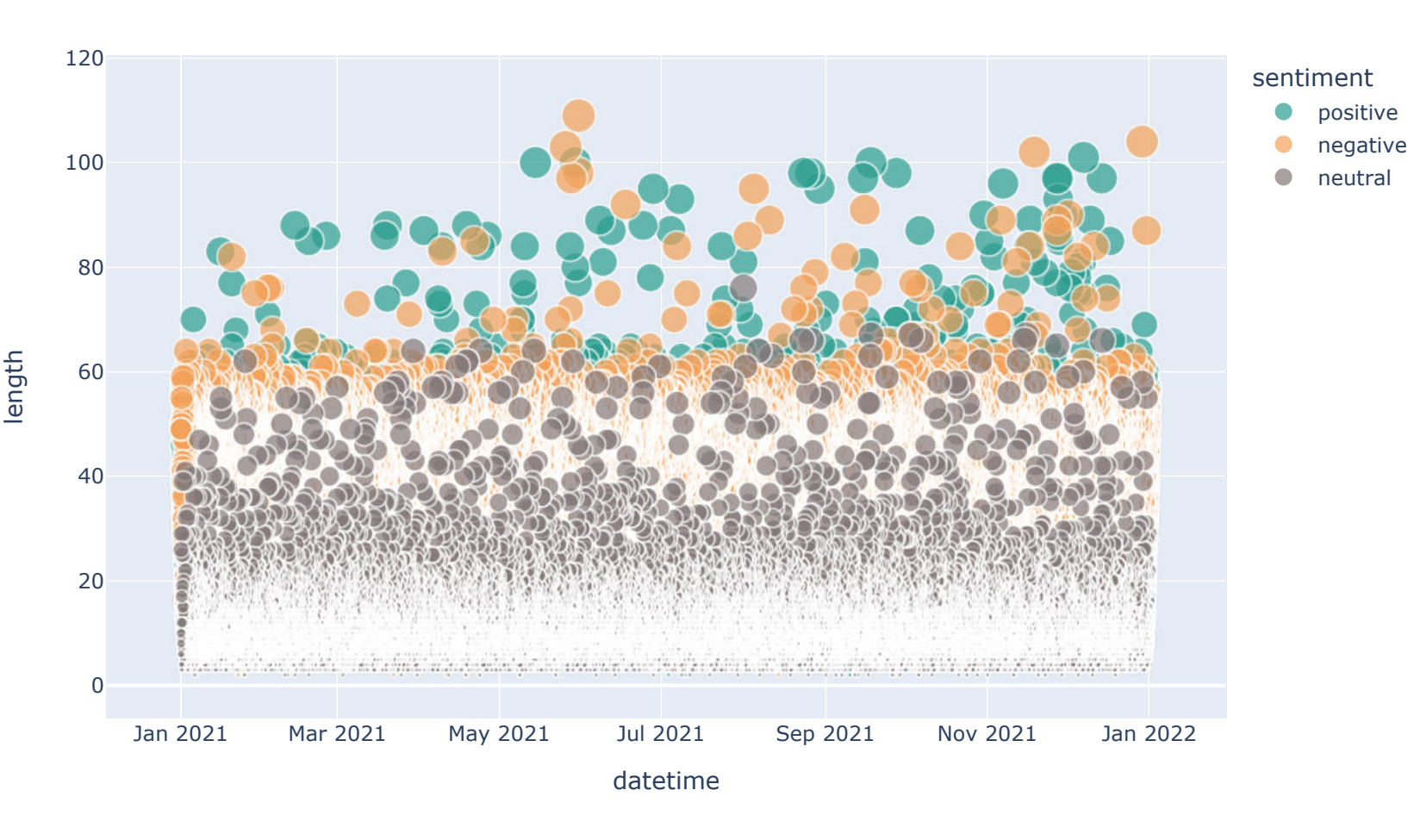


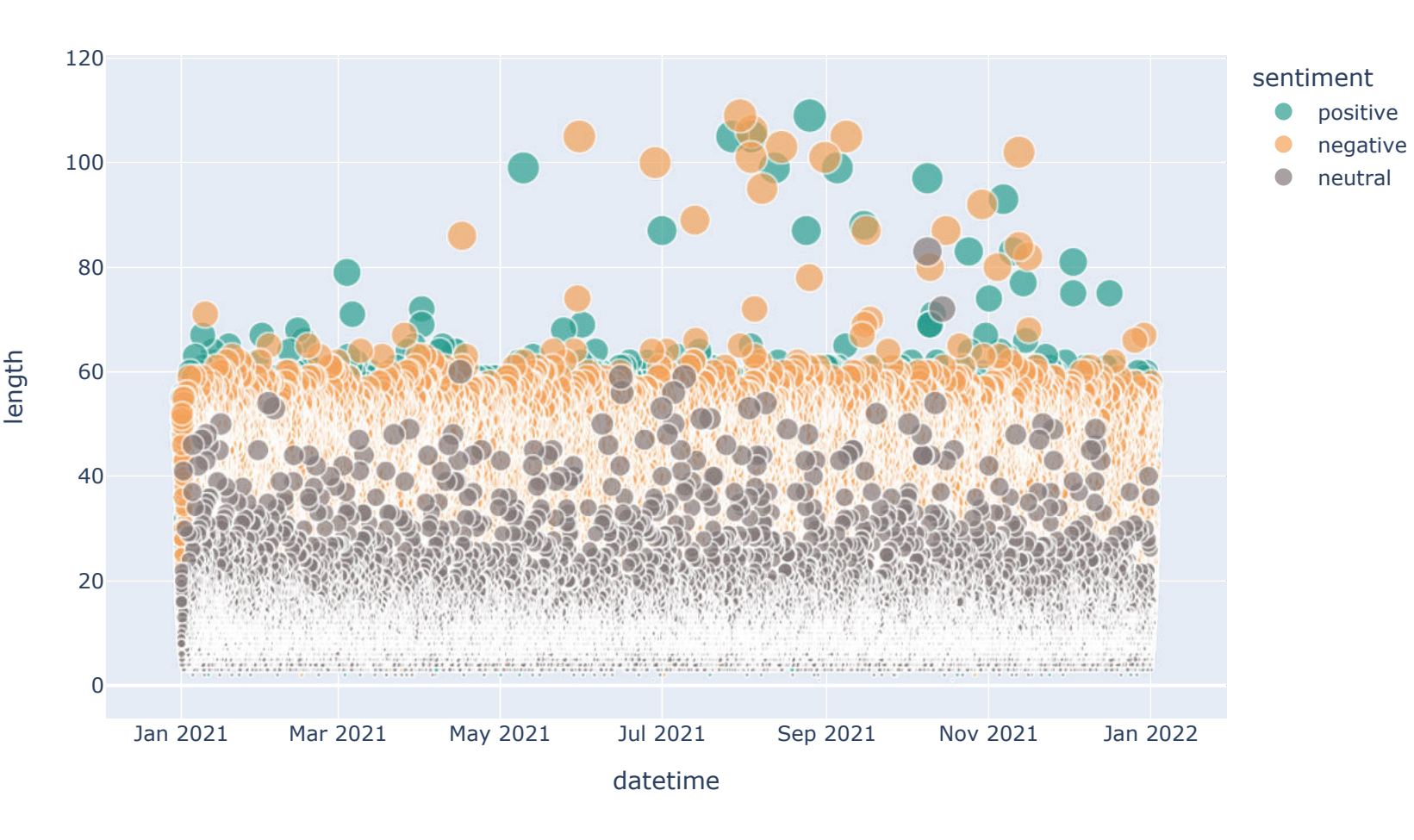
In [42]:
```
nyc_neg_wc_j6 = whynyc[(whynyc['datetime']>='2021-01-05') & (whynyc['datetime']<='2021-01-08') & (whynyc['senti
wordcloud = WordCloud(max_font_size=50, max_words=500, background_color='white').generate(str(nyc_neg_wc_j6['co
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

In [43]:
```
la_neg_wc_j6 = whyla[(whyla['datetime']>='2021-01-05') & (whyla['datetime']<='2021-01-08') & (whyla['sentiment'
wordcloud = WordCloud(max_font_size=50, max_words=500, background_color='white').generate(str(la_neg_wc_j6['cor
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

In [44]:
```
positive_wc_m29 = whynyc[(whynyc['datetime']>='2021-03-29') & (whynyc['datetime']<='2021-03-30') & (whynyc['sen
wordcloud = WordCloud(max_font_size=50, max_words=500, background_color='white').generate(str(positive_wc_m29['
plt.figure(figsize=(10, 7))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

In [45]:
```
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, width=35)))
                    )
    return fig.show()

tweets_timeline_plotly(whynyc)
tweets_timeline_plotly(whyla)
```





Topic Modeling

In [46]:
```
def why_lda_model(why_df):
    vectorizer = TfidfVectorizer(max_df=0.9, min_df=25, token_pattern='\w+|\$[\d\.]+|\S+', ngram_range=(1,3))#m

    # 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)

    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 [47]:
```
def collect_topics(x, why_df):
    topic_df = pd.DataFrame()

    topic_df['topic'] = x.iloc[::2, :1].reset_index(drop=True)
    topic_df['weight'] = x.iloc[1::2, :1].reset_index(drop=True)
    topic_df['subtopics'] = x.iloc[::2, 1:].apply(lambda x: ', '.join(x[x.notnull()]), axis = 1).reset_index(dr

    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)

        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 [48]:
```
pd.set_option('display.max_colwidth', 0)
```

In [49]:
```
whynyc_topics
```

Out[49]:

	topic	weight	subtopics	sentiment	category
0	even	94.4	dollar, would, like, time, next, smoke, million, dj, 😒, people, pop, house, close, would even	-0.317683	negative
1	like	780.0	look, look like, say, going, would say, people, would, something, god, bed, feel, 😭😭😭😭, looking, see	0.023620	positive
2	get	376.2	vaccine, people, covid, vaccinated, yes, lmaooo, obsessed, cannot, still, like, lying, test, getting, would	-0.178955	negative
3	know	343.0	u, let, let u, reason, need, yet, gt, would, link, better, remind, tag, still, c	-0.060287	negative
4	would	341.0	watch, game, like, want, favorite, mad, would want, season, someone, hungry, dog, saying, stand, one	0.073104	positive
5	new	338.8	york, new york, cold, see, love, weak, tiktok, fall, hurt, york new, new york new, york new york, hating, city	-0.043682	negative
6	make	312.8	sense, make sense, brooklyn, surprised, nyc, awake, tell, would, park, cat, like, new, closed, people	0.140215	positive
7	lol	310.8	tho, lmfao, wear, would, night, tf, last night, 😂😂, last, really, mask, know, still, way	0.129006	positive
8	black	290.5	white, woman, people, like, men, black people, know, go, black woman, white people, twitter, let, would, love	0.050566	positive
9	hard	244.3	love, post, 😭😭, know, people, though, like, tl, yea, find, would, wow, understand, bro	0.225627	positive
10	still	237.6	game, playing, play, team, trending, year, like, every, acting, music, album, player, lmaoooo, 😅	0.379856	positive
11	😂	217.9	wait, ya, got, like, seen, red, lord, never, mood, cannot, shoe, head, flag, get	0.024341	positive
12	cry	217.5	want, friend, first, time, best, one, place, first place, birthday, exactly, idea, like, christmas, ever	0.041979	positive
13	fuck	212.1	alone, people, sad, business, shit, like, men, give, miss, leave, wake, mind, bitch, ugly	-0.047157	negative
14	hate	197.4	people, old, year, year old, bruh, get, 💀, would, like, much, know, done, need, choose	0.004948	positive
15	train	194.9	wtf, name, one, minute, like, write, time, tryna, jersey, got, day, going, lie, st	-0.278111	negative
16	trump	194.4	vote, think, republican, democrat, bother, election, president, biden, even, would, voted, people, party, gop	0.016578	positive
17	reason	192.1	like, care, nigga, medium, social, much, social medium, people, one, act, 🙂, bike, early, island	0.048847	positive
18	cannot	191.7	😂, 🥺, like, nice, bring, police, get, like 😂, tweeting, want, student, 🥴, thing, need	-0.017544	negative
19	funny	188.9	thing, 😂😂😂, would, everyone, like, said, thought, get, else, fat, trust, never, know, think	0.037708	positive
20	thank	186.6	wonder, ask, question, keep, sexy, ion, deserve, people, high, official, stop, blocked, start, check	0.101356	positive
21	weird	185.6	lmao, hot, understand, eating, eat, fucking, dumb, people, like, never, never understand, uber, drink, go	0.023949	positive
22	feel	181.8	like, need, feel like, lmaoo, expensive, arrested, gun, people, capitol, cop, bus, see, law, evil	0.020407	positive
23	anyone	172.4	follow, cute, would anyone, would, yo, ad, following, want, like, phone, instagram, timeline, single, tweet	0.056983	positive
24	rn	164.4	wearing, mask, good, know, want, answer, want know, question, knicks, man, 🥴🥴, asking, wearing mask, hell	-0.086510	negative
25	sound	159.8	like, smh, sound like, idk, 🥴, 👀, sleep, got, posting, angry, queen, lol, dress, jail	0.053926	positive
26	omg	159.4	go, yeah, would, matter, people, mean, right, back, bad, like, sing, think, cannot, understand	0.279285	positive
27	😂😂😂	138.8	shit, 🥴🥴🥴, always, call, like, know, sending, dm, cuz, got, bout, oh, send, happen	0.049752	positive
28	sure	130.9	ago, long, loud, take, happening, year ago, year, cannot, make, explain, please, month, know, hour	0.041041	positive
29	news	119.6	serious, smell, friday, word, k, accurate, like, episode, one, fox, would, people, drag, could	-0.035088	negative

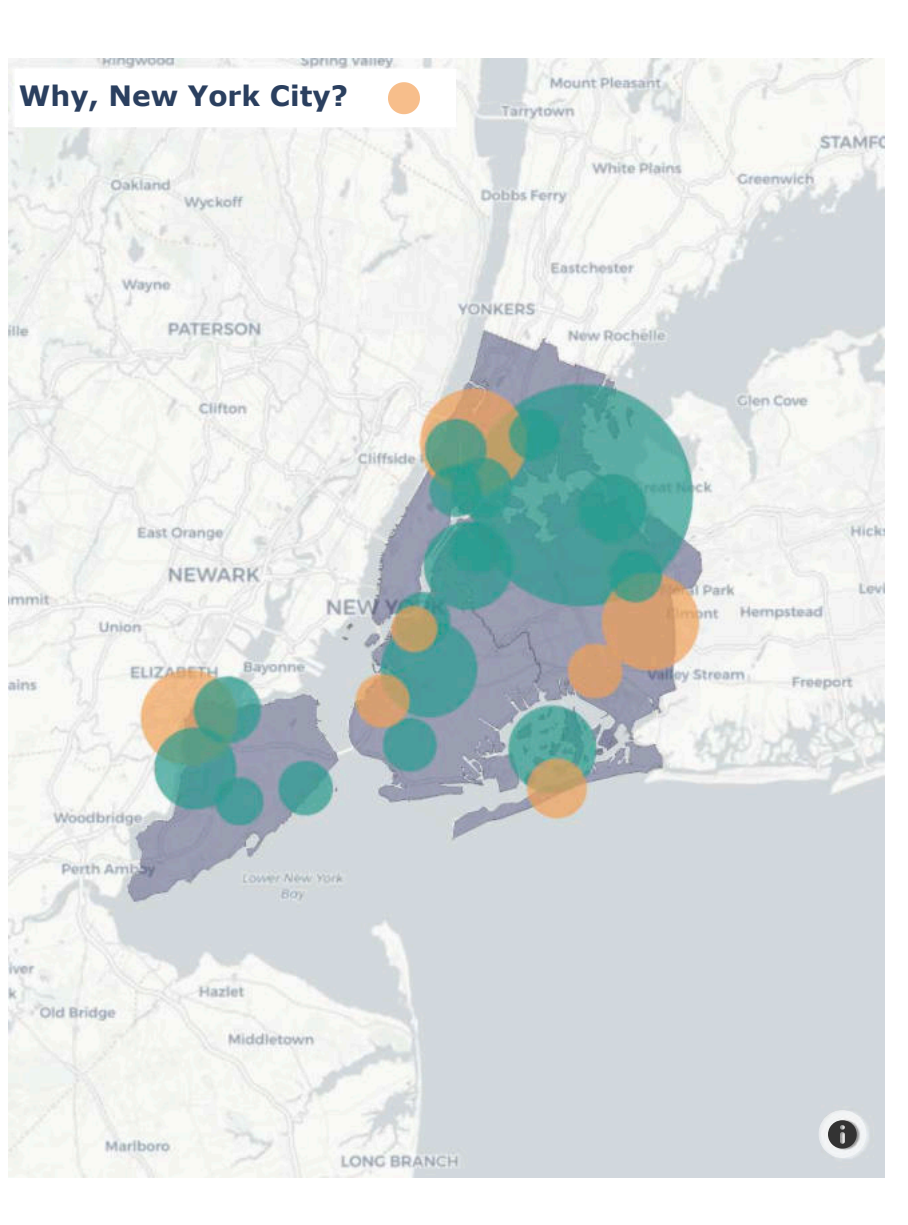
In [50]:
```
whyla_topics
```

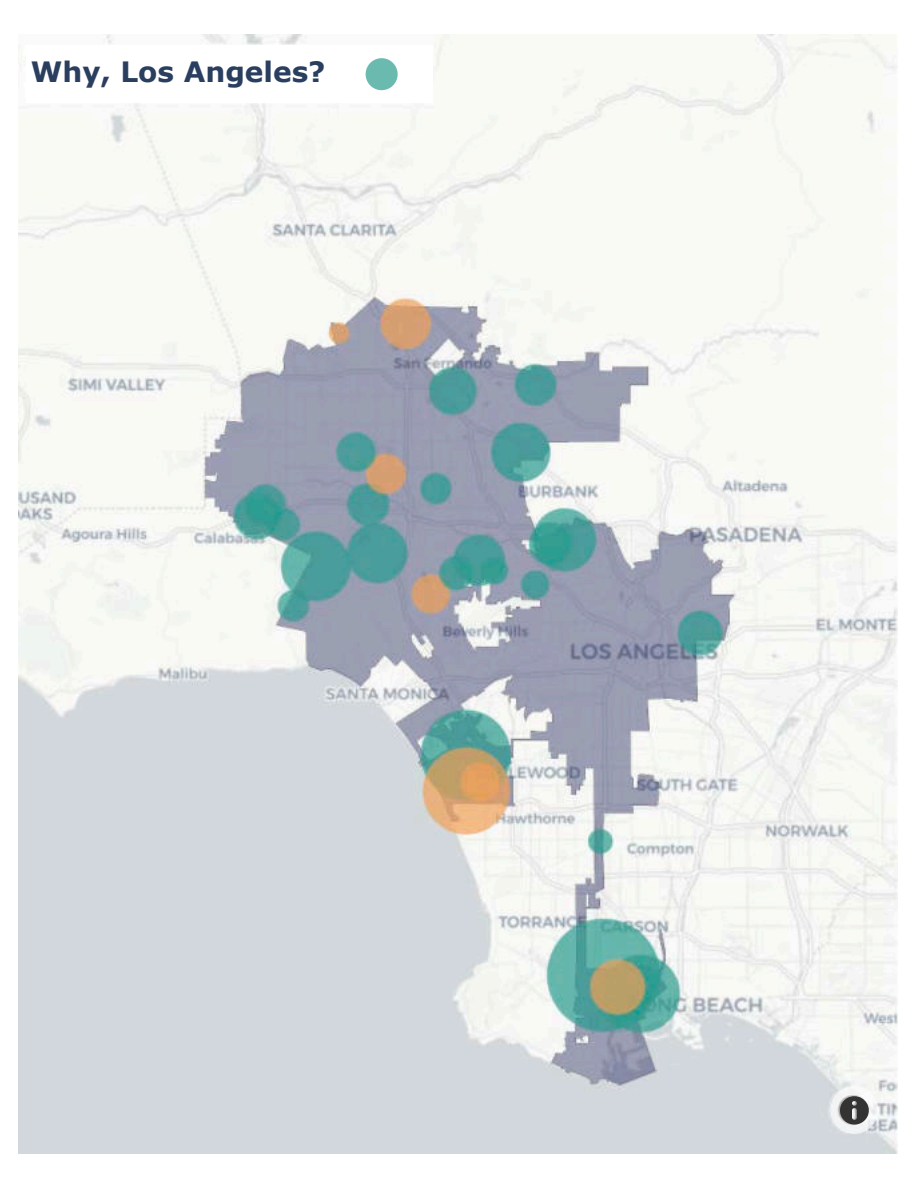
Out[50]:

	topic	weight	subtopics	sentiment	category
0	n	99.8	get, man, fr, play, want, show, take, home, stay, paid, pop, quiet, body, audiovisual	0.281390	positive
1	want	89.7	ugh, lt, exist, lakers, shirt, leave, alone, si, shoot, take, mood, know, nap, want see	0.031099	positive
2	oh	76.0	🙄, like, woman, kill, hair, deserve, dating, drunk, seeing, app, think, ice, know, get	-0.270990	negative
3	like	398.8	yes, hard, lying, voice, lmao, damn, dat, idk, said, 😅, looking, bag, beautiful, hahaha	0.044748	positive
4	lol	318.5	would, hot, 💀, wtf, like, cat, much, bring, mean, text, back, care, ever, 🙃	0.133775	positive
5	look	307.6	look like, love, like, place, first, first place, happening, face, god, exact, lol, 😩, name, scene	-0.089606	negative
6	people	274.4	like, ppl, get, surprised, know, white, black, lmaoooo, understood, want, never, waste, think, 🤦🏾‍♂️	0.042845	positive
7	feel	247.1	feel like, like, sleep, nice, wonder, cannot, call, yelling, christmas, 😡, would, 🥰, make, dying	0.284120	positive
8	tho	224.9	tweet, drinking, 😂, tired, traffic, 👀, like, coffee, sure, 😭, ion, walk, ❤️, know	0.057700	positive
9	funny	214.5	😂, idk, laughing, haha, like, smell, apple, hoe, felt, know, asleep, link, 🥴, mind	0.001898	positive
10	year	210.5	go, ago, awake, old, week, every, time, year old, every time, like, last, account, year ago, news	0.104492	positive
11	trump	196.4	republican, biden, people, would, president, vote, american, election, understand, democrat, taste, state, get, know	-0.298720	negative
12	thank	183.4	would, still, anyone, matter, perfect, would anyone, people, true, vaccinated, 😒, want, think, thing, ew	0.053451	positive
13	know	180.0	wait, like, good, people, 😂, happen, feeling, thing, listening, screaming, many, tweeting, people like, following	-0.051599	negative
14	let	170.8	los, angeles, los angeles, california, u, let u, angeles california, los angeles california, 😒, like, get, go, like 😂, would say	0.026291	positive
15	make	165.3	sense, make sense, one, shit, like, playing, album, way, remind, act, figure, go, trying, instagram	0.047818	positive
16	cry	160.5	weird, reason, sad, even, sound, like, 😂😂😂😂, sound like, low, one, picture, feed, lol, even know	0.039520	positive
17	night	154.0	last, mad, last night, understand, cute, texas, people, never, craving, never understand, c, earth, understand people, always	0.033076	positive
18	hate	151.4	omg, cannot, 😭😭😭, wake, early, like, annoying, people, stop, stupid, everyone, bro, morning, degree	0.057781	positive
19	fuck	148.9	expensive, 🥺, got, lmaooo, smh, pic, fuckin, like, wit, date, shit, go, would, get	0.033321	positive
20	rn	147.6	school, high, tax, people, need, would, like, job, hollywood, want, cannot, high school, real, life	0.057095	positive
21	😂😂😂	143.9	lmao, drink, see, blocked, photo, yeah, pretty, hear, store, late, want, blue, girl, one	-0.005581	negative
22	time	140.8	favorite, ya, accurate, like, 🤦🏾‍♂️, jesus, one, really, first time, know, first, tell, facebook, fly	0.030835	positive
23	tf	140.5	ask, gay, get, vaccine, bed, already, ugly, like, know, keep, would, raining, curious, people	-0.018719	negative
24	mask	138.5	wearing, wear, 😒, bother, like, lmaoo, going, nigga, cannot, even, pull, jail, still, people	-0.072360	negative
25	😂😂	123.4	men, eat, obsessed, think, hungry, need, like, short, close, lmfaooo, idea, grown, woman, dress	0.386189	positive
26	twitter	116.4	though, 😐, loud, anyone, tell, bruh, lie, 🙄, exactly, like, music, would, even, mr	0.026961	positive
27	single	114.1	medium, lmfao, like, social, acting, social medium, follow, ok, acting like, right, like lol, people, really, law	0.067568	positive
28	🤔	112.7	say, cold, always, hurt, laugh, fat, 🥴🥴🥴, hating, like, trust, get, seems, smoke, as	0.055678	positive
29	trending	102.8	game, friend, suck, best, birthday, day, tomorrow, weekend, test, covid, like, get, friday, still	0.072979	positive

Topic Bubble Map

In [59]:
```
def topic_map(file_name, why_df, export_name, title):
    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,
    )

    def SetColor(x):
        if(x < 0):
            return '#F4A259'
        elif(x == 0):
            return '#847979'
        elif(x > 0):
            return '#2A9D8F'

    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 + '.',
    )

    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
        ),

    )

    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

In [52]:
```
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 TfidfVectorizer is {:

    #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 Bayes with TfidfVecto

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

```
              precision    recall  f1-score   support

    negative       0.78      0.78      0.78     10248
     neutral       0.58      0.83      0.68      3428
    positive       0.88      0.78      0.82     14102

    accuracy                           0.79     27778
   macro avg       0.74      0.80      0.76     27778
weighted avg       0.80      0.79      0.79     27778

For the whynyc dataset, the accuracy for Logistic Regression with TfidfVectorizer is 78.70%
              precision    recall  f1-score   support

    negative       0.51      0.81      0.63      6514
     neutral       0.01      1.00      0.01        31
    positive       0.96      0.56      0.71     21233

    accuracy                           0.62     27778
   macro avg       0.49      0.79      0.45     27778
weighted avg       0.85      0.62      0.69     27778

For the whynyc dataset, the accuracy for Multinomial Naive Bayes with TfidfVectorizer is 62.38%
              precision    recall  f1-score   support

    negative       0.77      0.79      0.78      6569
     neutral       0.56      0.81      0.66      2164
    positive       0.88      0.77      0.82      9136

    accuracy                           0.78     17869
   macro avg       0.74      0.79      0.75     17869
weighted avg       0.80      0.78      0.79     17869

For the whyla dataset, the accuracy for Logistic Regression with TfidfVectorizer is 78.14%
              precision    recall  f1-score   support

    negative       0.51      0.83      0.63      4174
     neutral       0.01      0.93      0.02        27
    positive       0.96      0.56      0.71     13668

    accuracy                           0.62     17869
   macro avg       0.49      0.77      0.45     17869
weighted avg       0.85      0.62      0.69     17869

For the whyla dataset, the accuracy for Multinomial Naive Bayes with TfidfVectorizer is 62.15%
```