Log into your Twitter Developer account (not your Twitter account). At top right of your browser window, you will see the name of your App. Move your mouse cursor over your App and select Apps from the dropdown menu. Click "Details" and then select "Keys and tokens" tab. You will see your keys and tokens.

In "user auth" mode, your app uses the two API keys and two access tokens in the requests. Your app can access your own Twitter data and public Twitter data. Your app can post to Twitter. In "app auth" mode, you only provide the two API keys. You only have access to public Twitter data. Your app cannot post to Twitter.

In [1]:
```
import tweepy
auth = tweepy.OAuthHandler("omitted",
                           "omitted")
auth.set_access_token("omitted",
                      "omitted")
api = tweepy.API(auth)
```

1. Use Twitter data to create a social network diagram using NetworkX for the College of Arts & Sciences (@GSUArtSci).

a. This social network is three layers deep. First, select 5 friends of “GSUArtSci”.

b. For each friend of “GSUArtSci”, select at most 3 friends. For example, if A is a friend of “GSUArtSci”, then select 3 friends of A.

c. For each friend of friend of “GSUArtSci”, select at most 3 friends. For example, if B is a friend of A who is a friend of “GSUArtSci”, select at most 3 friends of B.

d. There should be an edge between any two nodes that are friends.

e. Create a network visualization of the social network using either Plotly or python-graphviz.

f. Each node should include the screen name of the Twitter user.

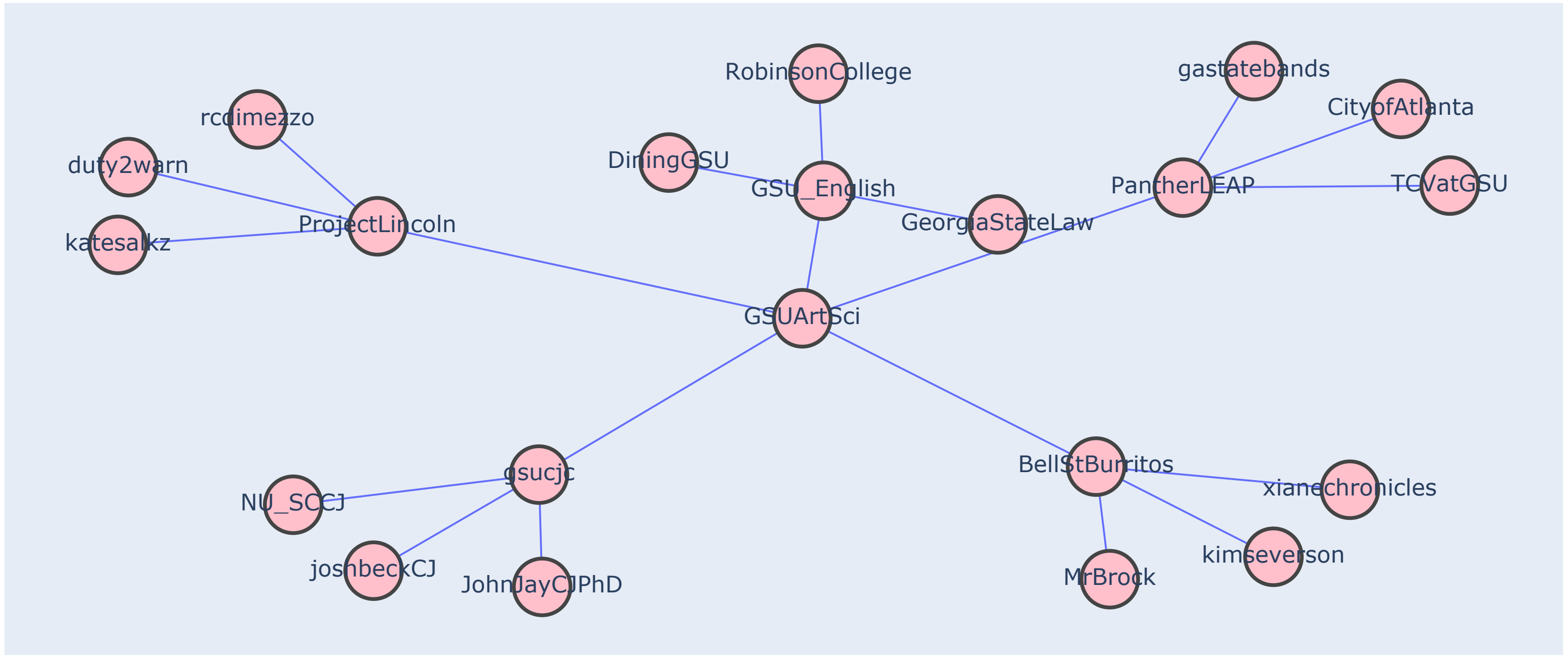
In [2]:
```
# get user information.
#import pprint
import pandas as pd

# Create an empty edge list
edge_list = pd.DataFrame(columns = ["source", "target"])
max_num_friends = 3

handle = "GSUArtSci"
# Get user information
user = api.get_user(handle)
friends = user.friends()

for friend in friends[0:5]:
    # Create an edge for this connection and add it to the edge list.
    edge_list = edge_list.append({'source' : user.screen_name,
                                  'target' : friend.screen_name} ,
                                 ignore_index=True)

    friends_of_friends = friend.friends()

    # Get friends of the friend and create edges for the connections.
    # Retrieve at most 3 friends of the friend
    for friend_of_friend in friends_of_friends[0:min(len(friends_of_friends), max_num_friends)]:
        edge_list = edge_list.append({'source' : friend.screen_name,
                                      'target' : friend_of_friend.screen_name} ,
                                     ignore_index=True)

edge_list
```

Out[2]:

	source	target
0	GSUArtSci	gsucjc
1	gsucjc	joshbeckCJ
2	gsucjc	JohnJayCJPhD
3	gsucjc	NU_SCCJ
4	GSUArtSci	BellStBurritos
5	BellStBurritos	MrBrock
6	BellStBurritos	kimseverson
7	BellStBurritos	xianechronicles
8	GSUArtSci	ProjectLincoln
9	ProjectLincoln	rcdimezzo
10	ProjectLincoln	duty2warn
11	ProjectLincoln	katesalkz
12	GSUArtSci	PantherLEAP
13	PantherLEAP	CityofAtlanta
14	PantherLEAP	gastatebands
15	PantherLEAP	TCVatGSU
16	GSUArtSci	GSU_English
17	GSU_English	GeorgiaStateLaw
18	GSU_English	DiningGSU
19	GSU_English	RobinsonCollege

In [3]:
```python
import plotly.graph_objects as go
import networkx as nx

G = nx.Graph()
G.add_nodes_from(list(set().union(edge_list.source, edge_list.target)))
G.add_edges_from(list(zip(edge_list.source, edge_list.target)))

pos = nx.spring_layout(G)

edge_x = []
edge_y = []
for edge in G.edges():
    x0 = pos[edge[0]][0]
    y0 = pos[edge[0]][1]
    x1 = pos[edge[1]][0]
    y1 = pos[edge[1]][1]
    edge_x.append(x0)
    edge_x.append(x1)
    edge_x.append(None)
    edge_y.append(y0)
    edge_y.append(y1)
    edge_y.append(None)

# Create a line plot to draw all the edges.
edge_trace = go.Scatter(
    x=edge_x,
    y=edge_y,
    mode='lines',
    line = dict(width = 1))

# Create a node list
node_x = []
node_y = []
for node in G.nodes():
    # Saving node coordinates to the node list.
    x = pos[node][0]
    y = pos[node][1]
    node_x.append(x)
    node_y.append(y)

# Create a scatter plot to draw all the nodes.
node_trace = go.Scatter(
    x=node_x,
    y=node_y,
    mode="markers + text",
    text = list(G.nodes),
    textposition = "middle center",
    hoverinfo = "text",
    marker=dict(
        size=30,
        color= "Pink",
        line_width=2))

fig = go.Figure(data=[edge_trace, node_trace],
             layout=go.Layout(
                title="A NetworkX Graph Rendered with Plotly",
                titlefont_size=16,
                showlegend=False,
                xaxis=dict(showgrid=False, zeroline=False, showticklabels=False),
                yaxis=dict(showgrid=False, zeroline=False, showticklabels=False))
               )
fig.show()
```

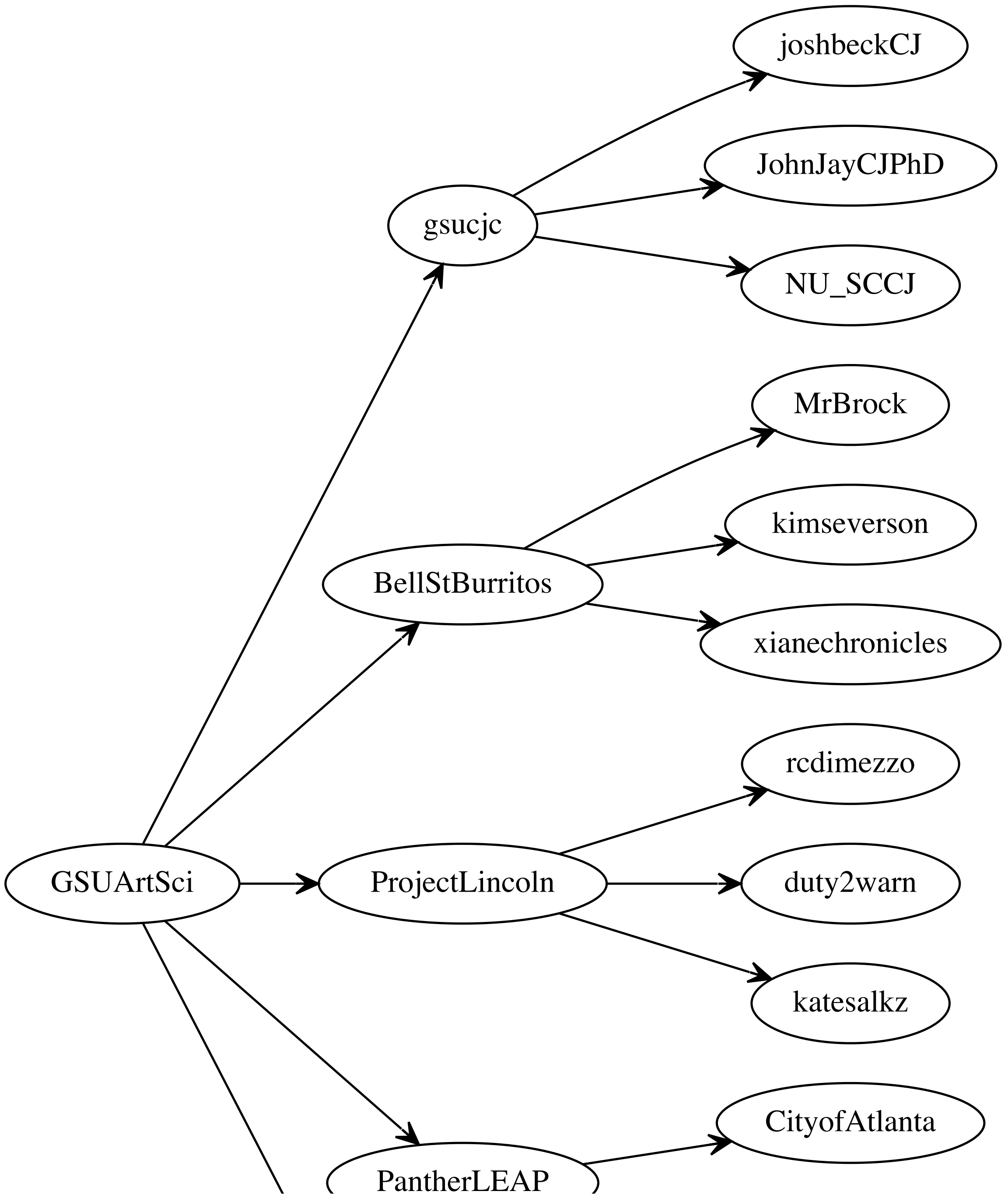
A NetworkX Graph Rendered with Plotly

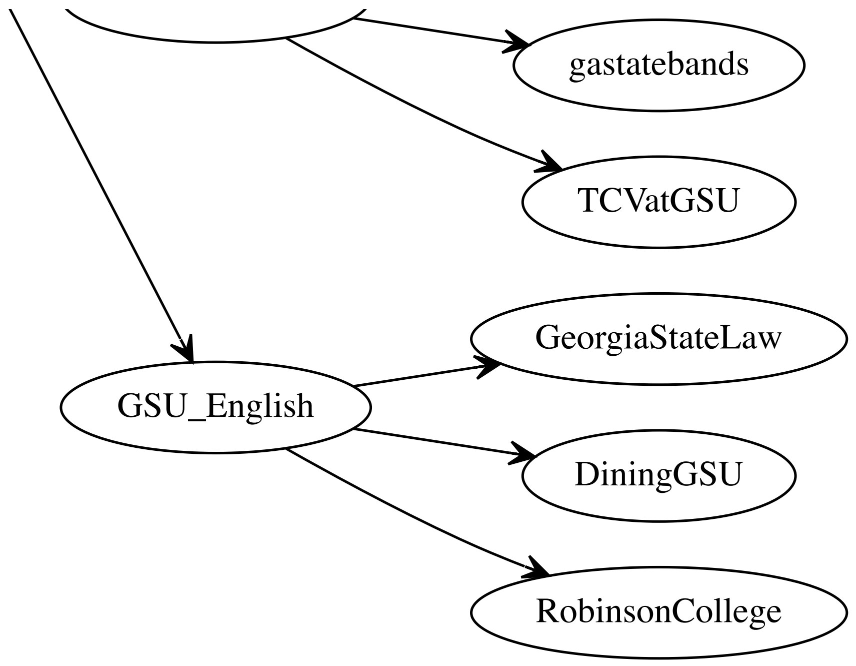


In [4]:
```python
import graphviz
import pandas as pd

GV = graphviz.Digraph(name = "social network",
                     filename='social_network.gv')

GV.attr("graph",
       rankdir = "LR",
       splines = "spline",
       label = "A social network created from Twitter data",
       labelloc = "t", # Place the graph label on top
       layout = "dot")

for i in range(0, len(edge_list)):
    GV.edge(tail_name = edge_list.iloc[i]["source"],
           head_name = edge_list.iloc[i]["target"],
          arrowhead = "vee")

GV
```

```
Out[4]:
```

A social network created from Twitter data





1. Retrieve the most recent tweets from CDC's Twitter account (@CDCgov). Collect at least 100 tweets (or as many as you can), excluding retweets.

a. Conduct sentiment analysis of the tweets. Calculate the average sentiment index for each day of the last 7 days, ending with the day you write the code.

b. Based on your data, draw a bar plot with Plotly Express (or Plotly) showing the sentiment index for the last 7 days.

In [5]:
```
#pip install clean-text
#python -m pip install textblob
import pandas as pd
from cleantext import clean
from textblob import TextBlob
```

In [6]:
```
# We will analyze recent tweets from one user.
handle = "CDCgov"
user = api.get_user(handle)

tweets = tweepy.Cursor(api.user_timeline,
                        screen_name='CDCgov',
                        count=None,
                        max_id=None,
                        lang="en",
                        trim_user=True,
                        #exclude_replies=True,
                        exclude_retweeted=True,
                        contributor_details=False,
                        include_entities=False
                        ).items();
```

In [7]:
```
import datetime
import pytz
list_tweets = []
for tweet in tweets:
        # The content of a tweet is stored as a dictionary in a JSON structure.
        list_tweets.append(tweet._json)

df = pd.DataFrame(list_tweets)
for i in range(0, len(df)):
     df.loc[i, "datetime"] = datetime.datetime.strptime(df.loc[i, "created_at"],'%a %b %d %H:%M:%S +0000 %Y').repla
ce(tzinfo=pytz.UTC)
```

In [8]:
```
for i in range(0, len(df)):
    df.loc[i, "date"] = df.iloc[i]["datetime"].date()
```

In [9]:
```
df2 = df[['text','date']]
df2.head()
```
Out[9]:

	text	date
0	#HCPs: Mark your calendar for tomorrow’s CDC C...	2020-08-19
1	RT @CDCemergency: #DYK? You may spread #COVID1...	2020-08-19
2	RT @CDCEnvironment: Planning for hurricane sea...	2020-08-19
3	RT @CDCChronic: Are you feeling more tired tha...	2020-08-19
4	RT @CDC_DRH: Pregnancy-related deaths can occu...	2020-08-19

In [10]:
```
# Sentiment analysis with TextBlob
sentiment_objects = [TextBlob(tweet) for tweet in df2['text']]

# Get sentiment values "polarity"
sentiment_values = [[tweet.sentiment.polarity,
                     str(tweet)] for tweet in sentiment_objects]

sentiment_df = pd.DataFrame(sentiment_values, columns=["polarity", "text"])

sentiment_df.head()
```
Out[10]:

	polarity	text
0	0.00	#HCPs: Mark your calendar for tomorrow’s CDC C...
1	0.00	RT @CDCemergency: #DYK? You may spread #COVID1...
2	0.00	RT @CDCEnvironment: Planning for hurricane sea...
3	-0.05	RT @CDCChronic: Are you feeling more tired tha...
4	0.50	RT @CDC_DRH: Pregnancy-related deaths can occu...

```
In [11]: sentiment_df['date'] = df2['date']
         sentiment_df.head()
```

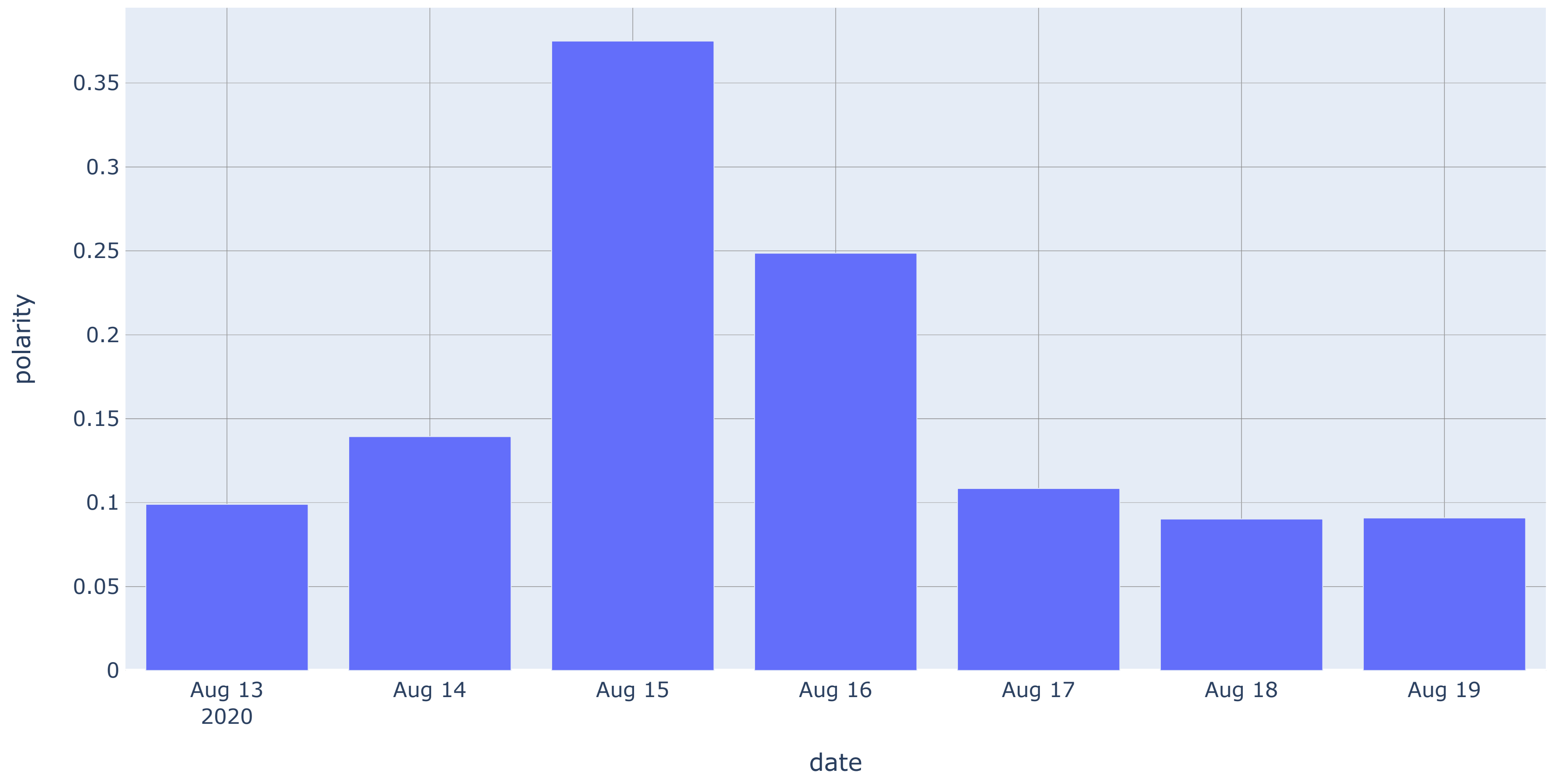
```
Out[11]:
   polarity                                               text        date
0      0.00       #HCPs: Mark your calendar for tomorrow’s CDC C...  2020-08-19
1      0.00  RT @CDCemergency: #DYK? You may spread #COVID1...  2020-08-19
2      0.00     RT @CDCEnvironment: Planning for hurricane sea...  2020-08-19
3     -0.05      RT @CDCChronic: Are you feeling more tired tha...  2020-08-19
4      0.50   RT @CDC_DRH: Pregnancy-related deaths can occu...  2020-08-19
```

*Calculate the average sentiment index for each day of the last 7 days, ending with the day you write the code.

```
In [12]: sentiment_avg = sentiment_df.groupby(['date']).agg({'polarity':['mean']}).reset_index()
         sentiment_avg.columns = sentiment_avg.columns.get_level_values(0)
         sentiment_7day = sentiment_avg.tail(7)
         sentiment_7day
```

```
Out[12]:
           date  polarity
349  2020-08-13  0.099007
350  2020-08-14  0.139359
351  2020-08-15  0.375000
352  2020-08-16  0.248611
353  2020-08-17  0.108433
354  2020-08-18  0.090235
355  2020-08-19  0.090873
```

In [13]:
```python
import plotly.express as px
fig = px.bar(sentiment_7day, x='date', y='polarity')
fig.show()
```



1. (20 points) Retrieve at least 100 (or as many as you can) tweets that contain #COVID19 and conduct the following data analysis and visualization.

a. Clean the text to remove all the URLs, email, number, etc. Remove all the stop words. Convert all words to lower case letters. See my lecture notes for an example.

b. Create a histogram plot using Plotly Express (or Plotly) to show the most frequently used words and their frequencies.

In [14]:
```
# For removing stop words
import nltk
from nltk.corpus import stopwords

keyword = "COVID19" + " -filter:retweets"
since_when = "2020-08-11"

# Use Tweepy's Cursor open to retrieve multiple pages of tweets.
tweets = tweepy.Cursor(api.search, q = keyword,
                       lang="en", since = since_when).items(100)

# Retrieve only texts
tweet_text = [tweet.text for tweet in tweets]
```

In [15]:
```python
import cleantext
import nltk
from nltk.corpus import stopwords
nltk.download("stopwords")
stop_words = set(stopwords.words('english'))

words = []
# Clean text and split into words
for i in range(len(tweet_text)):
        # Clean text with "cleantext"
    tweet_text[i] = cleantext.clean_words(tweet_text[i],
                                         all= False, # Execute all cleaning operations
                                         extra_spaces=True ,  # Remove extra white space
                                         stemming=True , # Stem the words
                                         stopwords=True ,# Remove stop words
                                         lowercase=True ,# Convert to lowercase
                                         numbers=True ,# Remove all digits
                                         punct=True ,# Remove all punctuations
                                         stp_lang='english'  # Language for stop words
                                         )

    #Split string into words
    words.append(list(tweet_text[i]))

# Flatten the word list to do frequency test.
# This is called a "bag of words".
words = [y for x in words for y in x]
words = [w for w in words if not w in stop_words]
```

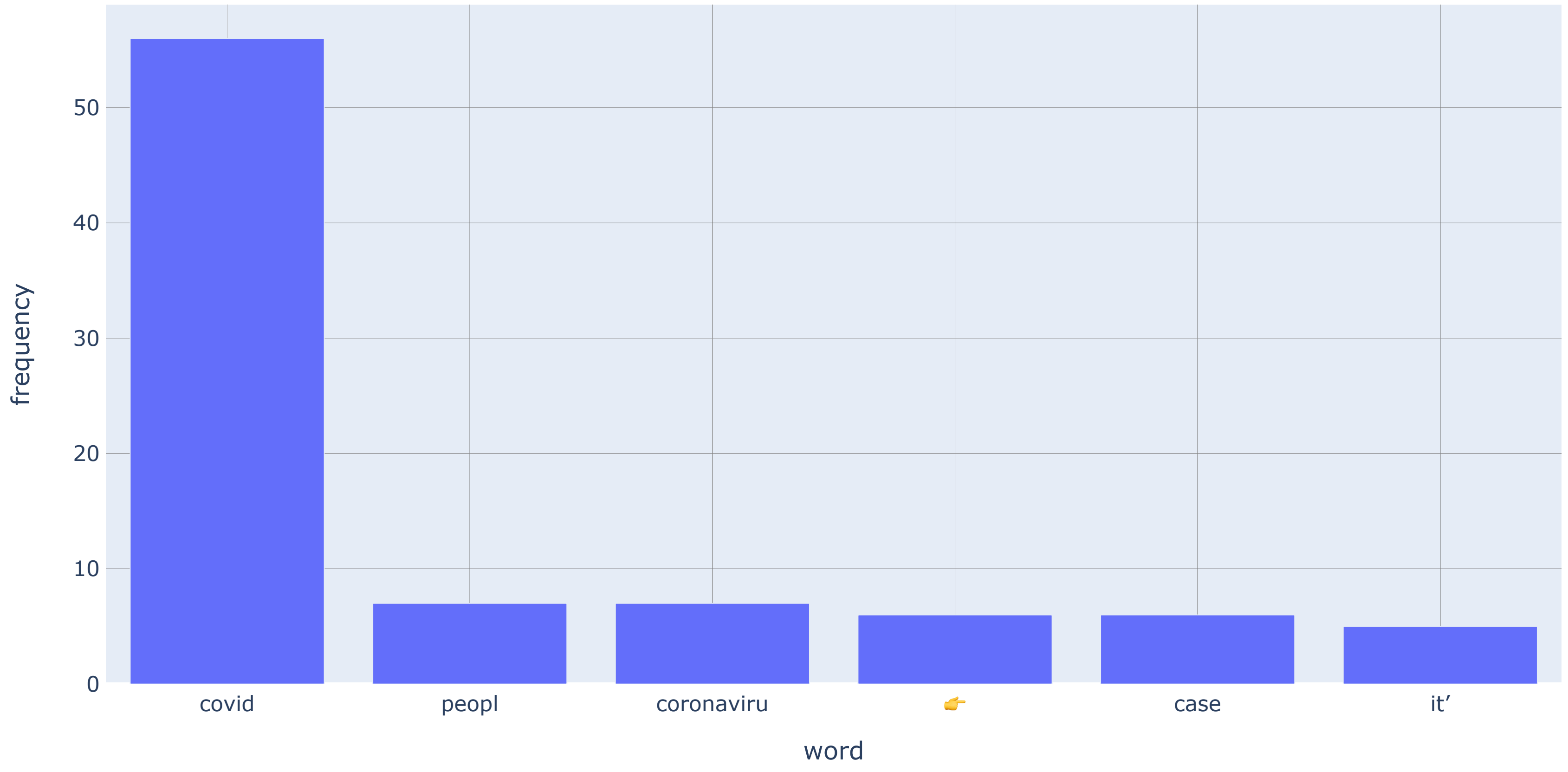
```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/jianqunkou/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [16]:
```python
import pandas as pd

df = pd.DataFrame(words, columns=["word"])

frequency = df["word"].value_counts()

word_frequency = pd.DataFrame({"word": frequency.index.tolist(),
                               "frequency": frequency.tolist()})

word_frequency.head(10)
```

Out[16]:

	word	frequency
0	covid	56
1	peopl	7
2	coronaviru	7
3	👉	6
4	case	6
5	it’	5
6	school	5
7	say	5
8	elect	5
9	take	5

```
In [17]: # Only choose the most frequent words
         import plotly.express as px
         most_frequent_words = word_frequency.head(6)
         fig = px.bar(most_frequent_words, x='word', y='frequency')
         fig.show()
```



1. Retrieve captions from the following YouTube videos, conduct sentiment analysis, and draw a line plot showing the sentiment index over time using Plotly Express (or Plotly).

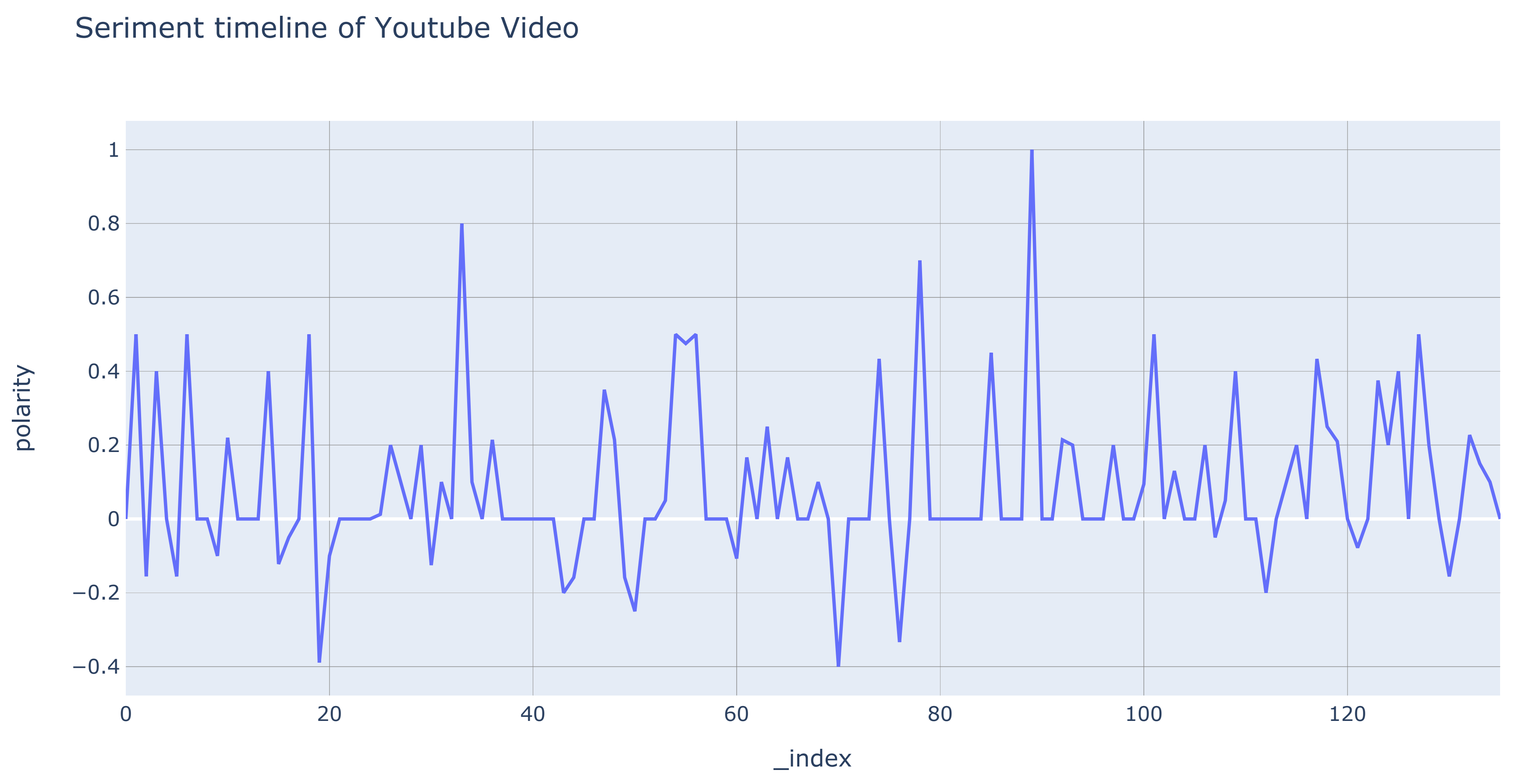
a. (15 points) Create a sentiment timeline for this video:

```
https://www.youtube.com/watch?v=6Af6b_wyiwI
```

b. (15 points) Create a sentiment timeline for a YouTube video of your choice.

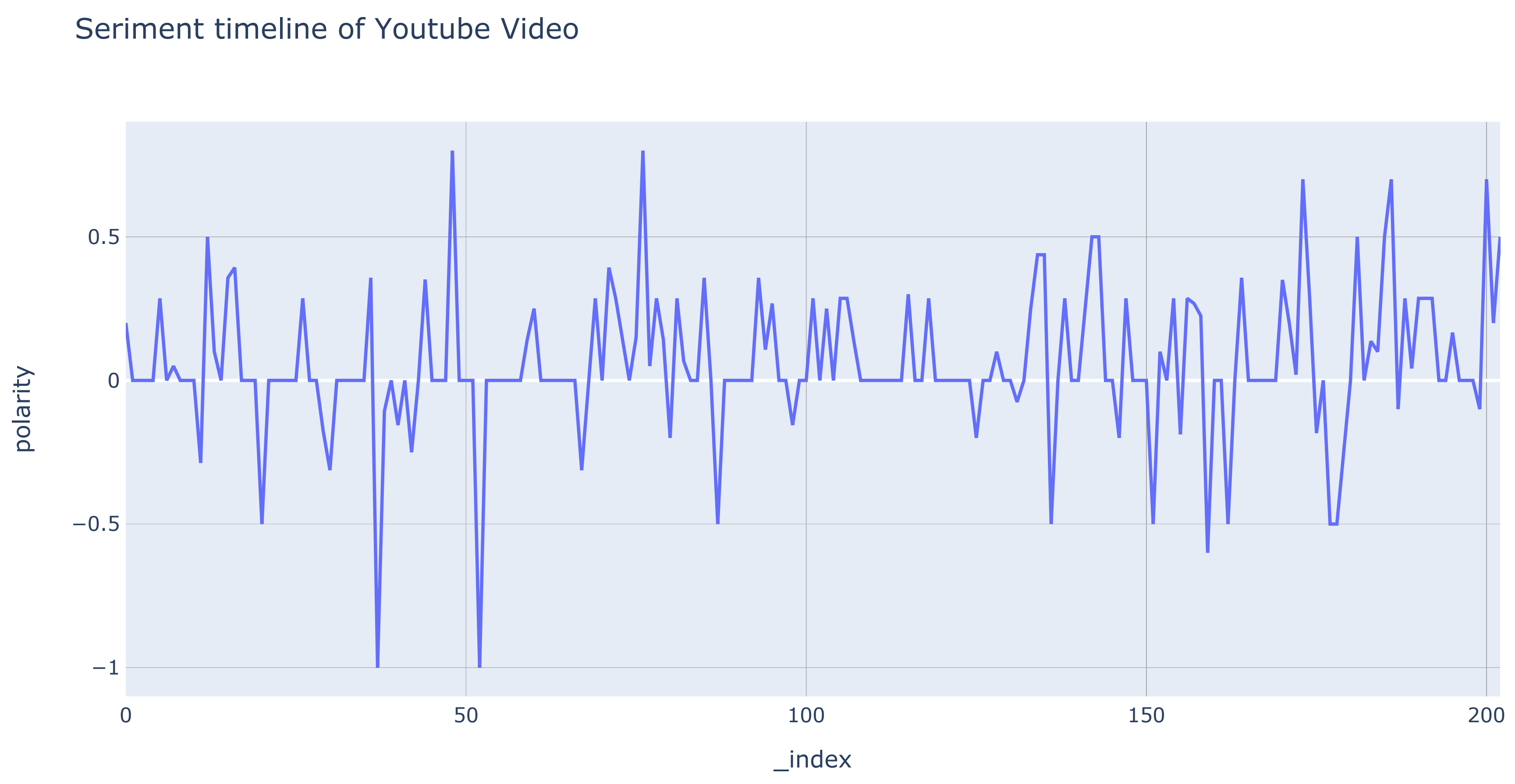
In [18]:
```python
# pip install pytube3
from pytube import YouTube
import pandas as pd
from textblob import TextBlob



def youtube (youTubeURL):
    yt = YouTube(youTubeURL)
    caption = yt.captions.get_by_language_code("en")
    caption_srt = caption.generate_srt_captions()
    text_file = open("YouTube_caption.txt", "w")
    text_file.write(caption_srt)
    text_file.close()

    caption_lines = caption_srt.splitlines()

    nested = []
    num_lines_per_item = 4
    for ix in range(0, len(caption_lines) - num_lines_per_item, num_lines_per_item):
        nested.append(caption_lines[ix:ix + num_lines_per_item])

    caption_df = pd.DataFrame(nested, columns = ["index", "time", "text", "line_break"])
    caption_df = caption_df.drop(columns = ["line_break"])

    sentiment_objects = [TextBlob(caption) for caption in caption_df["text"]]
    sentiment_values = [[sentiment_obj.sentiment.polarity, str(sentiment_obj)] for sentiment_obj in sentiment_objec
ts]
    caption_df["polarity"] = [sentiment_obj.sentiment.polarity for sentiment_obj in sentiment_objects]

    fig = px.line(caption_df, x=caption_df.index, y='polarity',
                  title='Seriment timeline of Youtube Video')
    return fig.show()

youTubeURL = "https://www.youtube.com/watch?v=6Af6b_wyiwI"

fig = youtube (youTubeURL)
```

```
/Users/jianqunkou/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:10: DeprecationWarning:

Call to deprecated function get_by_language_code (This object can be treated as a dictionary, i.e. captions['e
n']).
```



```
In [19]: youTubeURL = "https://www.youtube.com/watch?v=o4gEmLpxHHk"

         fig = youtube (youTubeURL)
```

```
/Users/jianqunkou/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:10: DeprecationWarning:

Call to deprecated function get_by_language_code (This object can be treated as a dictionary, i.e. captions['e
n']).
```



```
In [ ]:
```