Project 5

Mohammed Aljubori

In [1]:
```
import networkx as nx
import matplotlib.pyplot as plt
import tweepy
import pandas as pd
import plotly.graph_objects as go
```

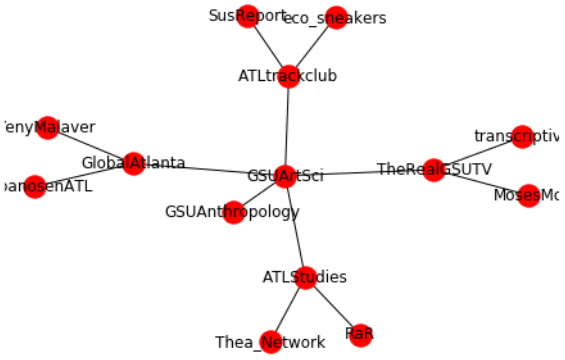
In [4]:
```
auth =tweepy.OAuthHandler("W4wasE9VfZt9E35aqGMwidkOo", "ySEJEo4DMlrD1Rkse6ybA669SdI1N8oeBwl3jasD5vl4PZc20L")
auth.set_access_token("1097579915845750784-ggURY9vLeghTYFSpcT6U9RgqkCEmNZ", "R32zETHx6sbM7hUQyODqvnKGcDNoLFMLe3kbW6BAY5jMA")
api = tweepy.API(auth)
```

1)
a.Select 5 friends of “GSUArtSci” and 5 followers of “GSUArtSci”.
b. For each friend of “GSUArtSci”, select at most 2 friends. For example, if A is a friend of “GSUArtSci”, then select 2 friends of A.

In [35]:
```
edge_list = pd.DataFrame(columns = ["source", "target"])

gsu_friends = api.friends("GSUArtSci")
for friend in gsu_friends[0:5]: # Only retrieve the first 5 friends
    print(friend.screen_name)
    print(friend.location)
    print("\n")
    edge_list = edge_list.append({'source' : "GSUArtSci", 'target' : friend.screen_name} , ignore_index=True)
    for friend_of_friend in api.friends(friend.screen_name)[0:2]:
        edge_list = edge_list.append({'source' : friend.screen_name, 'target' : friend_of_friend.screen_name} , ignore_index=True)

edge_list
edge_list
G=nx.from_pandas_edgelist(edge_list, 'source', 'target')
nx.draw(G, with_labels=True)
plt.show()
```

```
GlobalAtlanta
Atlanta


ATLtrackclub
Atlanta, GA


ATLStudies
ATL


GSUAnthropology



TheRealGSUTV
```

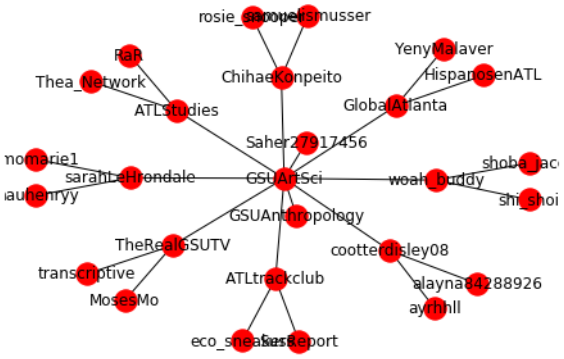
```
C:\Users\aljub\Anaconda3\lib\site-packages\networkx\drawing\nx_pylab.py:611: MatplotlibDeprecationWarning: isinstance(..., numbers.Number)
  if cb.is_numlike(alpha):
```



1)

c. For each follower of “GSUArtSci” , select at most 2 followers. For example, if B is a follower of “GSUArtSci”, then select 2 followers of B.
d. There should be an edge between any two nodes who are either friends or followers.

In [36]:
```
gsu_followers = api.followers("GSUArtSci")
for follower in gsu_followers[0:5]:
    #print(follower.screen_name)
    #print(follower.location)
    #print("\n")

    # Create an edge for this connection
    edge_list = edge_list.append({'source' : follower.screen_name, 'target' : "GSUArtSci"} , ignore_index=True)
    # Get followers of the follower and create edges for the connections.
    for follower_of_follower in api.followers(follower.screen_name)[0:2]:
        edge_list = edge_list.append({'source' : follower_of_follower.screen_name, 'target' : follower.screen_name} , ignore_index=True

edge_list
G=nx.from_pandas_edgelist(edge_list, 'source', 'target')
nx.draw(G, with_labels=True)
plt.show()
```



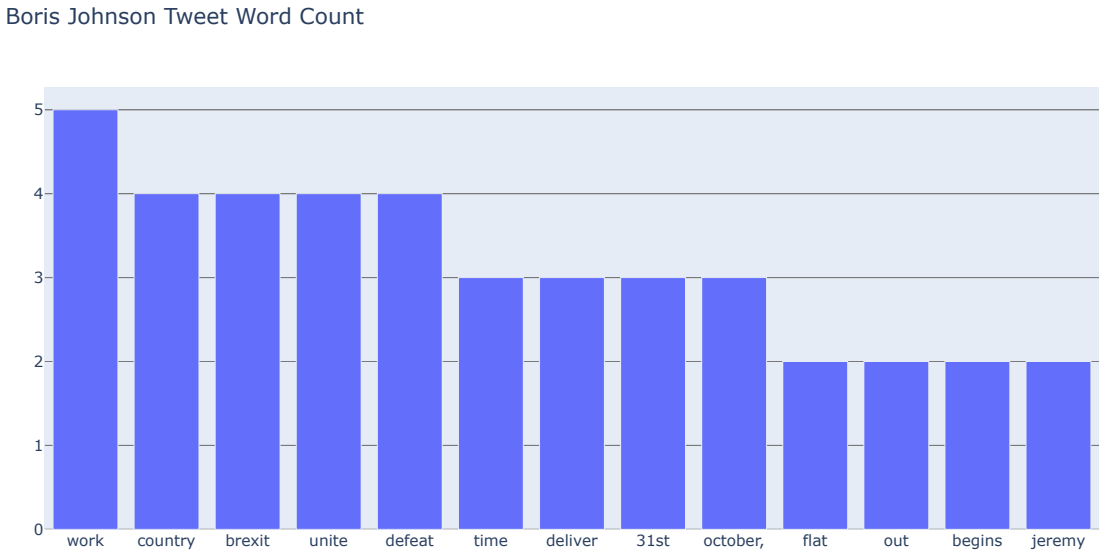
2)
Retrieve the most recent tweets from Boris Johnson’s Twitter account (@BorisJohnson). Collect as many tweets as you can, excluding retweets.
a. Find the 10 most frequently used words from the text and draw a bar chart using Plotly (not Seaborn).

In [29]:
```
gsu_tweets = api.user_timeline("BorisJohnson", include_rts=False)
for tweet in gsu_tweets:
    print(tweet.text)
```

```
I salute the work of our Armed Services. And that’s one of the reasons we’re today announcing a new Office for Vete… https://t.co/pHu
cahgC8h (https://t.co/pHucahgC8h)
Today in Scotland I toured a submarine at HMNB Clyde, and met some of the fantastic military personnel who do so mu… https://t.co/tyy
t1SD6hx (https://t.co/tyyt1SD6hx)
We in this Government will work flat out to give this country the leadership it deserves. That work begins now https://t.co/Nqd8SdBDA
R (https://t.co/Nqd8SdBDAR)
It’s time to get to work to deliver Brexit by 31st October, unite the party, defeat Jeremy Corbyn - and energise ou… https://t.co/WGB
h3IZG9J (https://t.co/WGBh3IZG9J)
Thank you all for the incredible honour you have done me. The time for campaigning is over and the time for work be… https://t.co/6RP
QhxzdCq (https://t.co/6RPQhxzdCq)
4 nations, 8000 miles, 16 regional hustings and hundreds of members’ events! Thank you everyone for your support th… https://t.co/ARd
9Lifz8P (https://t.co/ARd9Lifz8P)
With less than 150 hours before polls close, this is your last chance to post your ballot. Please vote for me so I… https://t.co/ep4I
aBz9E2 (https://t.co/ep4IaBz9E2)
Thanks for inviting me last night!

Less than 150 hours to go until polls close. If you haven't received your ball… https://t.co/a3W0sn14iF (https://t.co/a3W0sn14iF)
```

2)
i. Clean the text to remove all the URL, email, number, etc.
ii. Remove all the stop words.
iii. Convert all words to lower case letters.

In [2]:
```
from collections import Counter

data_set = "Today in Scotland I toured a submarine at HMNB Clyde, and met some of the fantastic military personnel who do so much to ke
# split() returns list of all the words in the string
split_it = data_set.split()

# Pass the split_it list to instance of Counter class.
Counter = Counter(split_it)

# most_common() produces k frequently encountered
# input values and their respective counts.
most_occur = Counter.most_common(30)

pf = pd.DataFrame(most_occur,columns = ['Word' , 'Number'])
data = pf.drop([0,1,2,3,4,6,11,12,13,14,17,20,21,22,23,24,29], axis=0)
fig = go.Figure([go.Bar(x=data['Word'], y=['Number'])])
data.reset_index()
fig = go.Figure([go.Bar(x=data['Word'].str.lower(), y=[5,4,4,4,4,3,3,3,3,2,2,2,2])])
fig.update_layout(title_text='Boris Johnson Tweet Word Count')
fig.show()
```



3.

Retrieve at least 20 (or as many as you can) tweets that contains #TheLionKing and conduct the following data analysis and visualization.

a. Conduct sentiment analysis of the tweets and draw a sentiment index lineplot with Plotly (not Seaborn).

In [5]:
```
# Search for a keyword and filter out retweets
keyword = "TheLionKing" + " -filter:retweets"
since_when = "2019-07-01"

tweets = tweepy.Cursor(api.search, q = keyword,
                       lang="en", since = since_when).items(20)

# Print out text, user name, and location
#for tweet in tweets:
#    print("text: %s" %(tweet.text))
#    print("screen_name: %s" %(tweet.user.screen_name))
#    print("location: %s\n" %(tweet.user.location))
```

3.

b. Clean the text to remove all the URL, email, number, etc.

In [6]:
```
# Clean text with package "cleantext"
# See https://pypi.org/project/clean-text/
# pip install clean-text[gpl]

from cleantext import clean

# For removing stop words
import nltk
from nltk.corpus import stopwords

# For frequency analysis
import collections

#pip install -U textblob
# For sentiment analysis
from textblob import TextBlob

keyword = "TheLionKing" + " -filter:retweets"
since_when = "2019-07-01"

# Search by keyword and time
tweets = tweepy.Cursor(api.search, q = keyword,
                       lang="en", since = since_when).items(20)

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

# Clean text and split into words
for i in range(len(tweet_text)):
    #Before cleaning
   # print("%s" %(tweet_text[i]))

    # Clean text with "cleantext"
    tweet_text[i] = clean(tweet_text[i],
                          no_urls=True,
                          no_emails=True,
                          no_numbers=True,
                          no_phone_numbers=True,
                          no_currency_symbols=True,
                          no_line_breaks=True,
                          no_punct=True,
                          replace_with_url="")

    #After cleaning
    #print("%s\n" %(tweet_text[i]))

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

# Flatten the word list to do frequency test
words = [y for x in words for y in x]

#print(words)
```

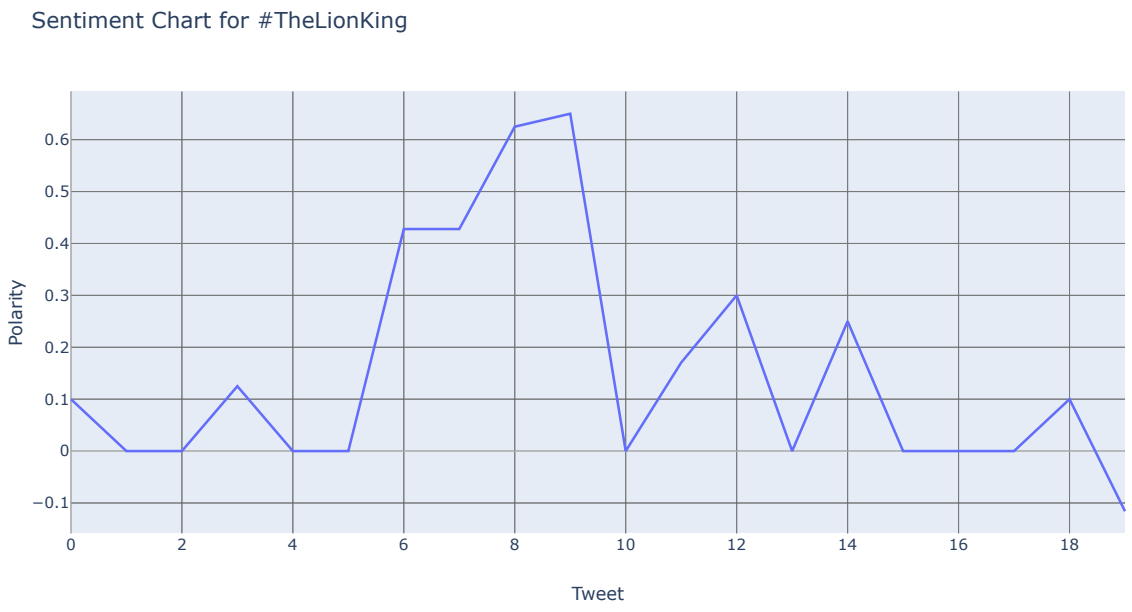
In [7]:
```
# Sentiment analysis with Textblob package.

sentiment_objects = [TextBlob(tweet) for tweet in tweet_text]

sentiment_values = [[tweet.sentiment.polarity, str(tweet)] for tweet in sentiment_objects]

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

sentiment_df.head()
```

Out[7]:

	polarity	tweet
0	0.100	join us this saturday at <number> am for this ...
1	0.000	the waiter at bwws name is legit simba thelion...
2	0.000	hakuna matoddler it means no silence for the r...
3	0.125	congratulations jonfavs for being the first pe...
4	0.000	thebeat979fm justdolapo beyonce shattawalegh m...

Graph for number 3

In [8]:
```
sentiment_df
zz = sentiment_df.reset_index()
fig = go.Figure(data=go.Scatter(x= zz['index'] , y=zz["polarity"]))
fig.update_layout(title='Sentiment Chart for #TheLionKing',
                  xaxis_title='Tweet',
                  yaxis_title='Polarity')

fig.show()
```



4. Retrieve captions from the following YouTube videos, conduct sentiment analysis and draw the sentiment index timeline using Plotly (not Seaborn).

In [10]:
```
from pytube import YouTube
```

In [16]:
```
yt = YouTube('https://www.youtube.com/watch?v=JtJdZrmeYKc')
```

In [26]:
```
caption = yt.captions.get_by_language_code("en")
```

In [32]:
```
caption_srt = caption.generate_srt_captions()
text_file = open("YouTube_caption.txt", "w")
text_file.write(caption_srt)
text_file.close()
```

In [31]:
```
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])
```

In [39]:
```
caption_df = pd.DataFrame(nested, columns = ["index", "time", "text", "line_break"])
caption_df = caption_df.drop(columns = ["line_break"])
caption_df.head()
```

Out[39]:

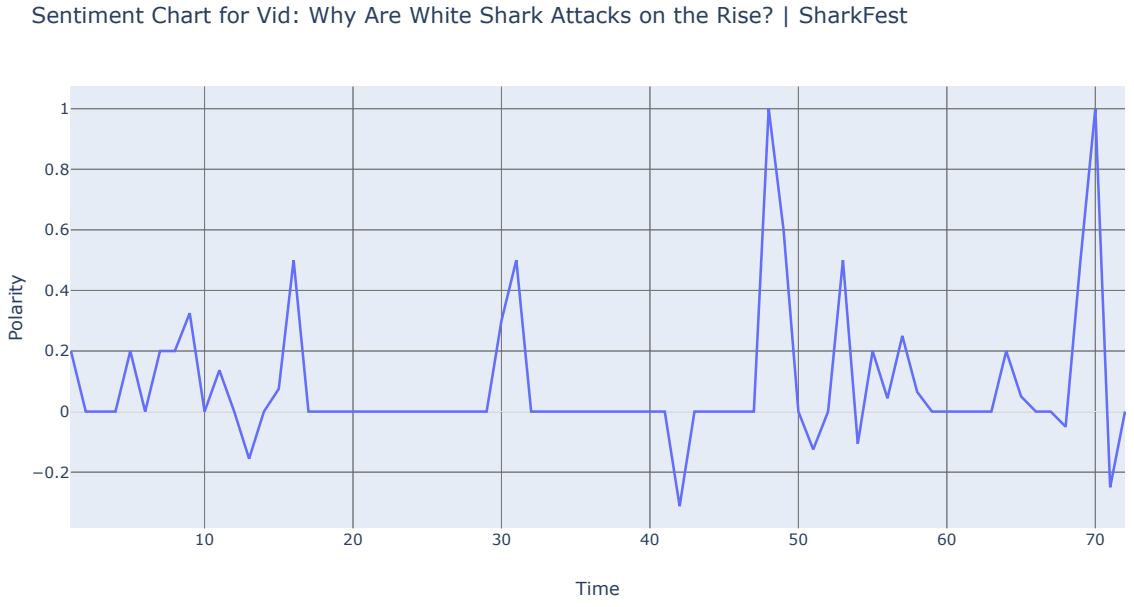
	index	time	text
0	1	00:00:00,633 --> 00:00:02,035	[dramatic music]
1	2	00:00:02,102 --> 00:00:03,436	NARRATOR: Great whites are the most
2	3	00:00:03,503 --> 00:00:07,107	feared predator in the ocean.
3	4	00:00:07,173 --> 00:00:11,211	They typically hunt large mammals, like seals,
4	5	00:00:11,277 --> 00:00:15,181	sea lions, and whales.

```
Sentiment analysis with TextBlob
```

In [38]:
```
from textblob import TextBlob
sentiment_objects = [TextBlob(caption) for caption in caption_df["text"]]
sentiment_values = [[sentiment_obj.sentiment.polarity, str(sentiment_obj)] for sentiment_obj in sentiment_objects]
caption_df["polarity"] = [sentiment_obj.sentiment.polarity for sentiment_obj in sentiment_objects]
```

4.
a. (15 points) Create a sentiment timeline for this video:
https://www.youtube.com/watch?v=JtJdZrmeYKc (https://www.youtube.com/watch?v=JtJdZrmeYKc)

In [49]:
```
fig = go.Figure(data=go.Scatter(x= caption_df['index'] , y=caption_df["polarity"]))
fig.update_layout(title='Sentiment Chart for Vid: Why Are White Shark Attacks on the Rise? | SharkFest',
                  xaxis_title='Time',
                  yaxis_title='Polarity')
fig.show()
```



4a. (15 points) Create a sentiment timeline for this video:
https://www.youtube.com/watch?v=JtJdZrmeYKc (https://www.youtube.com/watch?v=JtJdZrmeYKc)

In [45]:
```
yt = YouTube('https://www.youtube.com/watch?v=zRAFNSgk1Ns&t=42s')
caption = yt.captions.get_by_language_code("en")
# Get the captions in the more readable SRT format (see https://www.speechpad.com/captions/srt).
caption_srt = caption.generate_srt_captions()

#print(caption_srt)

#save the caption to a file named Output.txt
text_file = open("YouTube_caption.txt", "w")
text_file.write(caption_srt)
text_file.close()
```

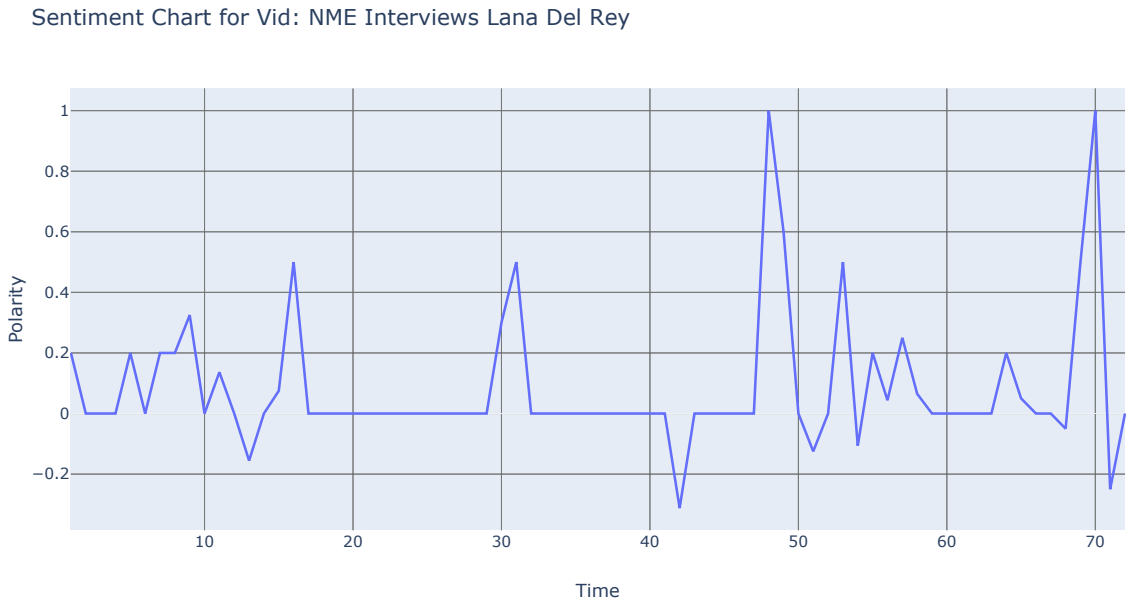
In [46]:
```
# Create a dataframe with indices, time, and texts in separate columns.

#Split SRT file into lines.
caption_lines = caption_srt.splitlines()
# print(caption_lines)

#Create a nested list so we can create data frame out of it.
nested = []
#There are four lines for each item on the list.
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])

#print(nested)

# Create a data frame of the captions
caption_df = pd.DataFrame(nested, columns = ["index", "time", "text", "line_break"])

# Delete the the last column because it's empty.
caption_df = caption_df.drop(columns = ["line_break"])

# Now we have a dataframe with three columns: index, time, and text.
caption_df.head()
```

Out[46]:

	index	time	text
0	1	00:00:00,350 --> 00:00:13,190	I think one thing that really affected
1	2	00:00:08,180 --> 00:00:15,740	me was when I was 11 I saw Kurt Cobain
2	3	00:00:13,190 --> 00:00:18,080	singing heart-shaped box on MTV I saw
3	4	00:00:15,740 --> 00:00:19,880	like the video for it that was something
4	5	00:00:18,080 --> 00:00:22,100	that really struck me I think when I was

In [47]:
```
# Send the text to TextBlob for sentiment analysis
sentiment_objects = [TextBlob(caption) for caption in caption_df["text"]]

# Retrieve sentiment values
sentiment_values = [[sentiment_obj.sentiment.polarity, str(sentiment_obj)] for sentiment_obj in sentiment_objects]

# Add a "polarity" column to the dataframe
caption_df["polarity"] = [sentiment_obj.sentiment.polarity for sentiment_obj in sentiment_objects]
```

In [50]:
```
fig = go.Figure(data=go.Scatter(x= caption_df['index'] , y=caption_df["polarity"]))
fig.update_layout(title='Sentiment Chart for Vid: NME Interviews Lana Del Rey',
                  xaxis_title='Time',
                  yaxis_title='Polarity')

fig.show()
```



In [ ]: