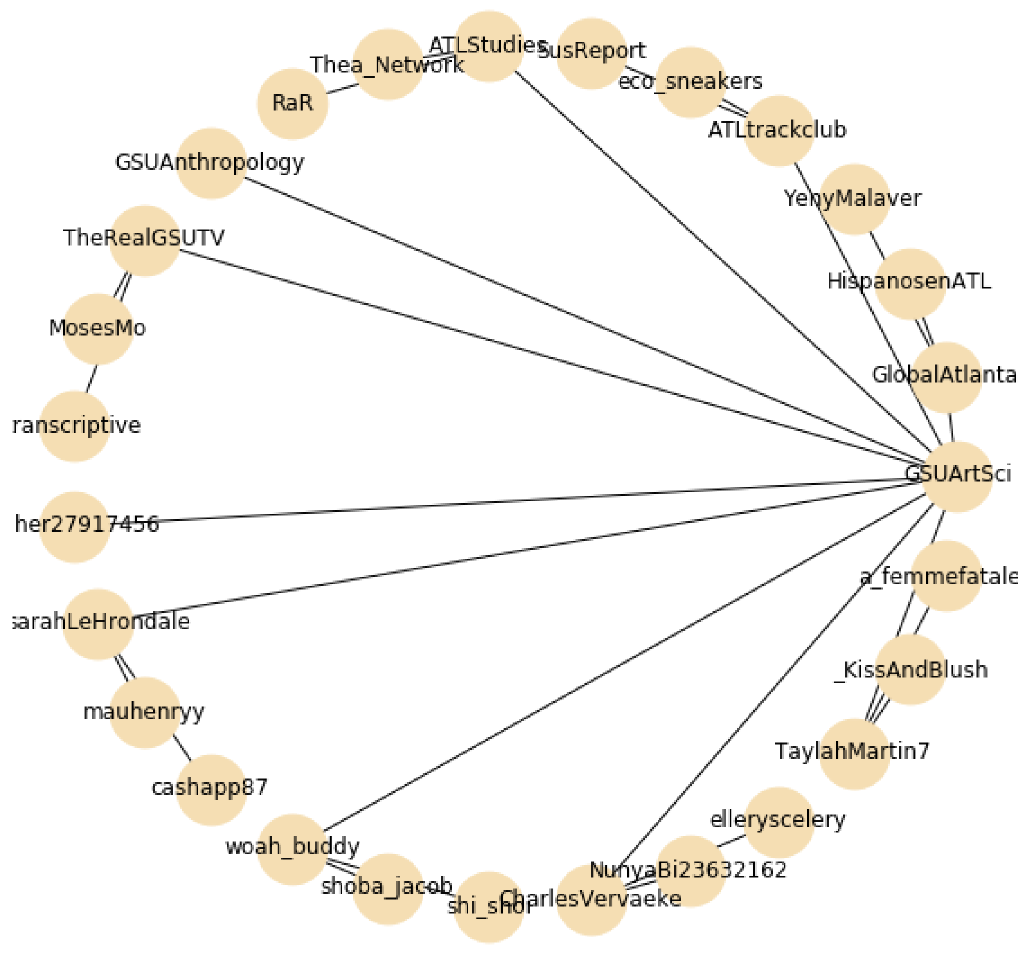
CSC6730 Project5 Ruixie Fang

In [19]:
```
import tweepy
auth = tweepy.OAuthHandler()
auth.set_access_token( )
api = tweepy.API(auth)


import pandas as pd
import plotly.offline as off
off.init_notebook_mode(connected=False)
import plotly.graph_objs as go
import networkx as nx
import numpy as np
import matplotlib.pyplot as plt
```

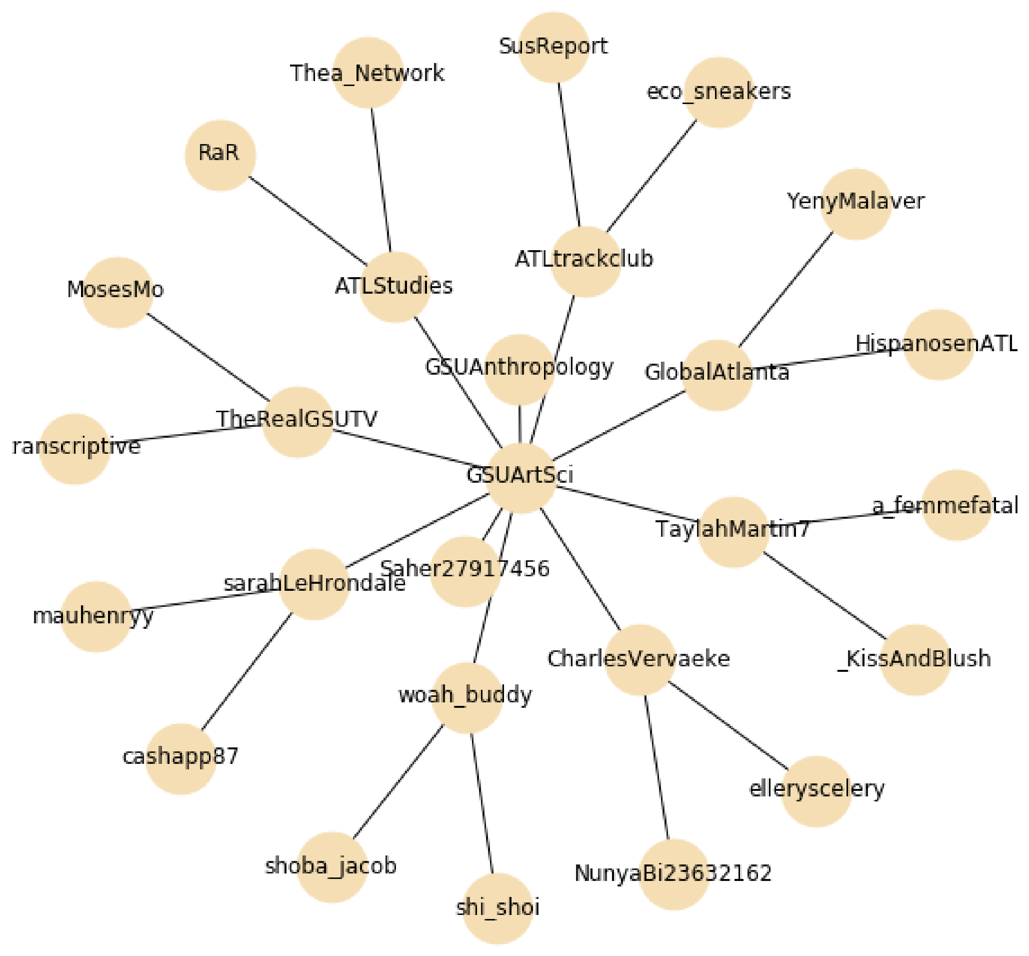
1.(30 points) Use Twitter data to create a social network diagram using either NetworkX or Plotly for the College of Arts & Sciences (@GSUArtSci). a.Select 5 friends of “GSUArtSci” and 5 followers of “GSUArtSci”. b.For each friend of “GSUArtSci”, select at most 2 friends. For example, if A is a friend of “GSUArtSci”, then select 2 friends of A. 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. e.Create TWO network visualizations with two different layouts. f.Each node should include the screen name of the Twitter user.

In [20]:
```
edge_list = pd.DataFrame(columns = ["source", "target"])
gsu_friends = api.friends("GSUArtSci")
for friend in gsu_friends[0:5]:
    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)
gsu_followers = api.followers("GSUArtSci")
for follower in gsu_followers[0:5]:
    edge_list = edge_list.append({'source' : follower.screen_name, 'target' : "GSUArtSci"} , ignore_index=True)
    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)
```

In [21]:
```
V=list(set().union(edge_list.source,edge_list.target))
Label=dict(zip(V,V))

g = nx.from_pandas_edgelist(edge_list,source="source", target="target")

pos=nx.shell_layout(g)

plt.figure(figsize=(10,10))
nx.draw_networkx_nodes(g,pos,nodelist=V,node_color='Wheat',node_size=1500)
nx.draw_networkx_edges(g,pos,width=1)
nx.draw_networkx_labels(g,pos,labels=Label)
plt.axis('off')
plt.show()
```



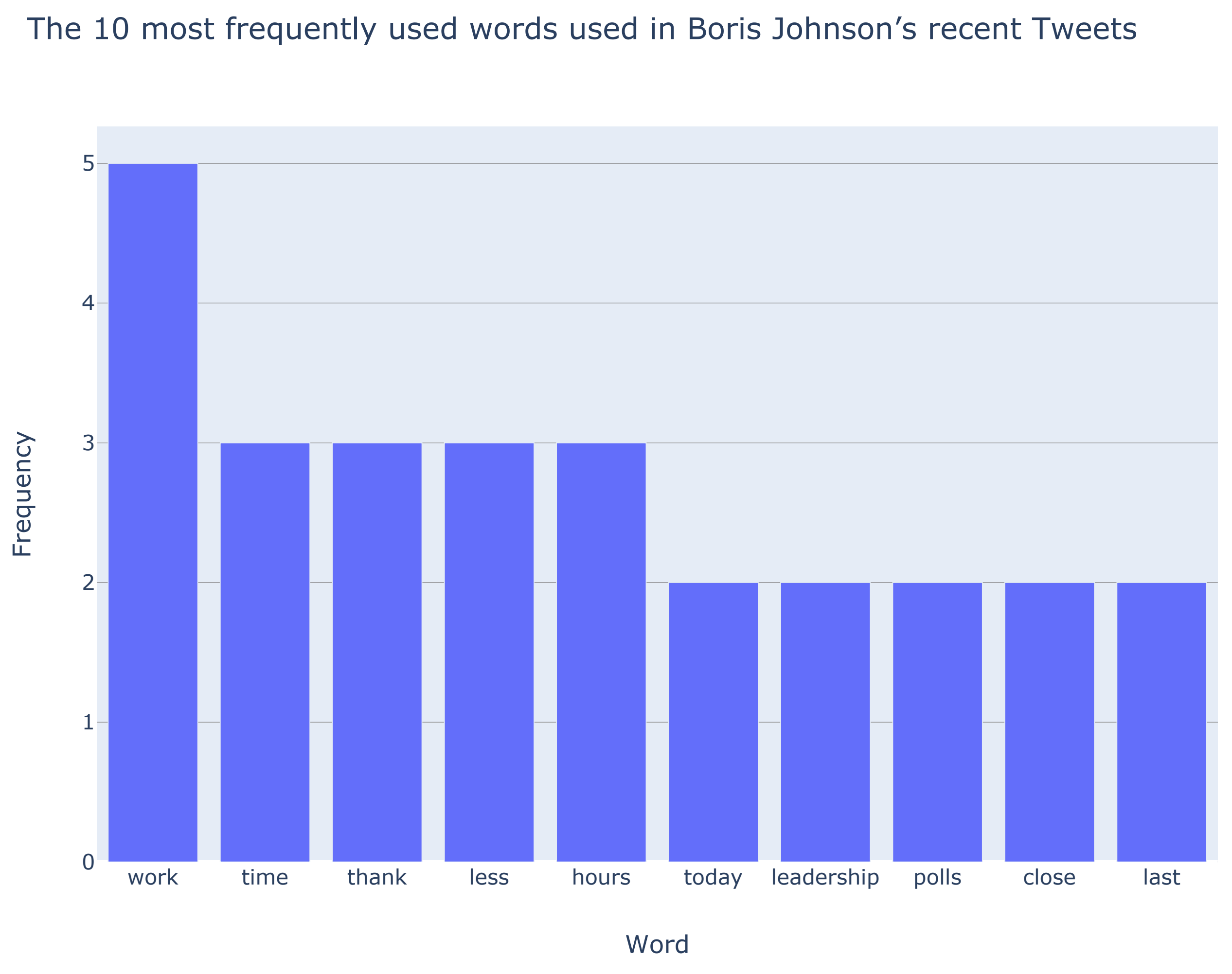
In [22]:
```
pos1=nx.kamada_kawai_layout(g)

plt.figure(figsize=(10,10))
nx.draw_networkx_nodes(g,pos1,nodelist=V,node_color='Wheat',node_size=1500)
nx.draw_networkx_edges(g,pos1,width=1)
nx.draw_networkx_labels(g,pos1,labels=Label)
plt.axis('off')
plt.show()
```



2.(20 points) 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). 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.

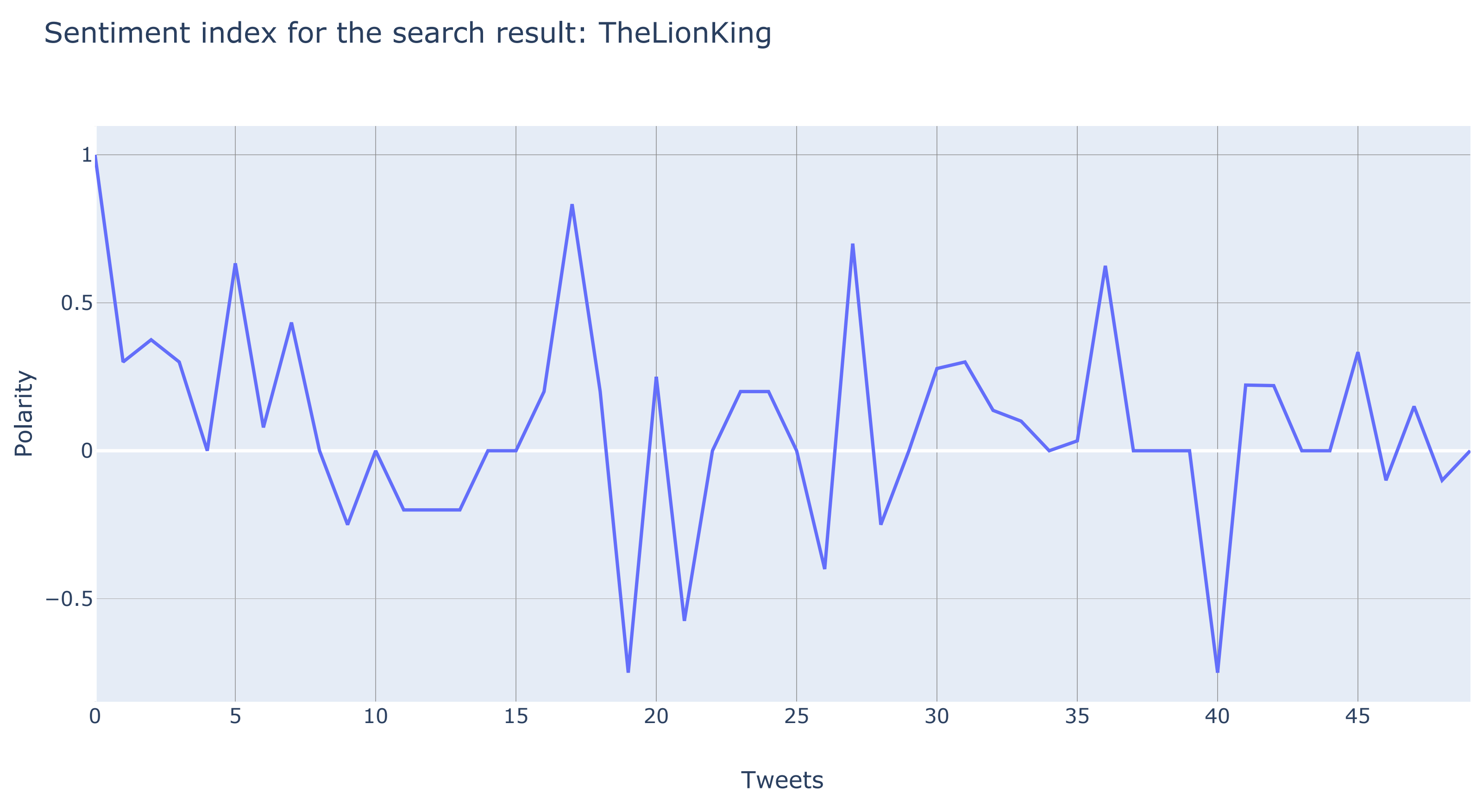
```
In [23]: from cleantext import clean
         import nltk
         from nltk.corpus import stopwords
         import collections

         tweets = api.user_timeline("BorisJohnson",include_rts=False)

         # Clean text
         tweet_text = [tweet.text for tweet in tweets]
         words = []
         for i in range(len(tweet_text)):
             tweet_text[i] = clean(tweet_text[i],
                                   no_urls=True,
                                   lower=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="",
                                   replace_with_number="")
             words.append(tweet_text[i].split())

         words = [y for x in words for y in x]

         # Remove stop words
         nltk.download("stopwords")
         stop_words = set(stopwords.words('english'))
         words = [w for w in words if not w in stop_words]

         #frequency analysis
         word_counts = collections.Counter(words)
         word_frequency = pd.DataFrame(word_counts.most_common(10), columns = ["word", "frequency"])

         #draw bar chart
         tracea = [go.Bar(
             x=word_frequency.word,
             y=word_frequency.frequency
             )]
         layouta = go.Layout(
             title="The 10 most frequently used words used in Boris Johnson’s recent Tweets",
             autosize=False,
             width=800,
             height=600,
             xaxis=dict(title="Word"),
             yaxis=dict(title="Frequency")
         )
         figa = go.Figure(data=tracea,layout=layouta)
         off.iplot(figa, filename="basic-bar")
```

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



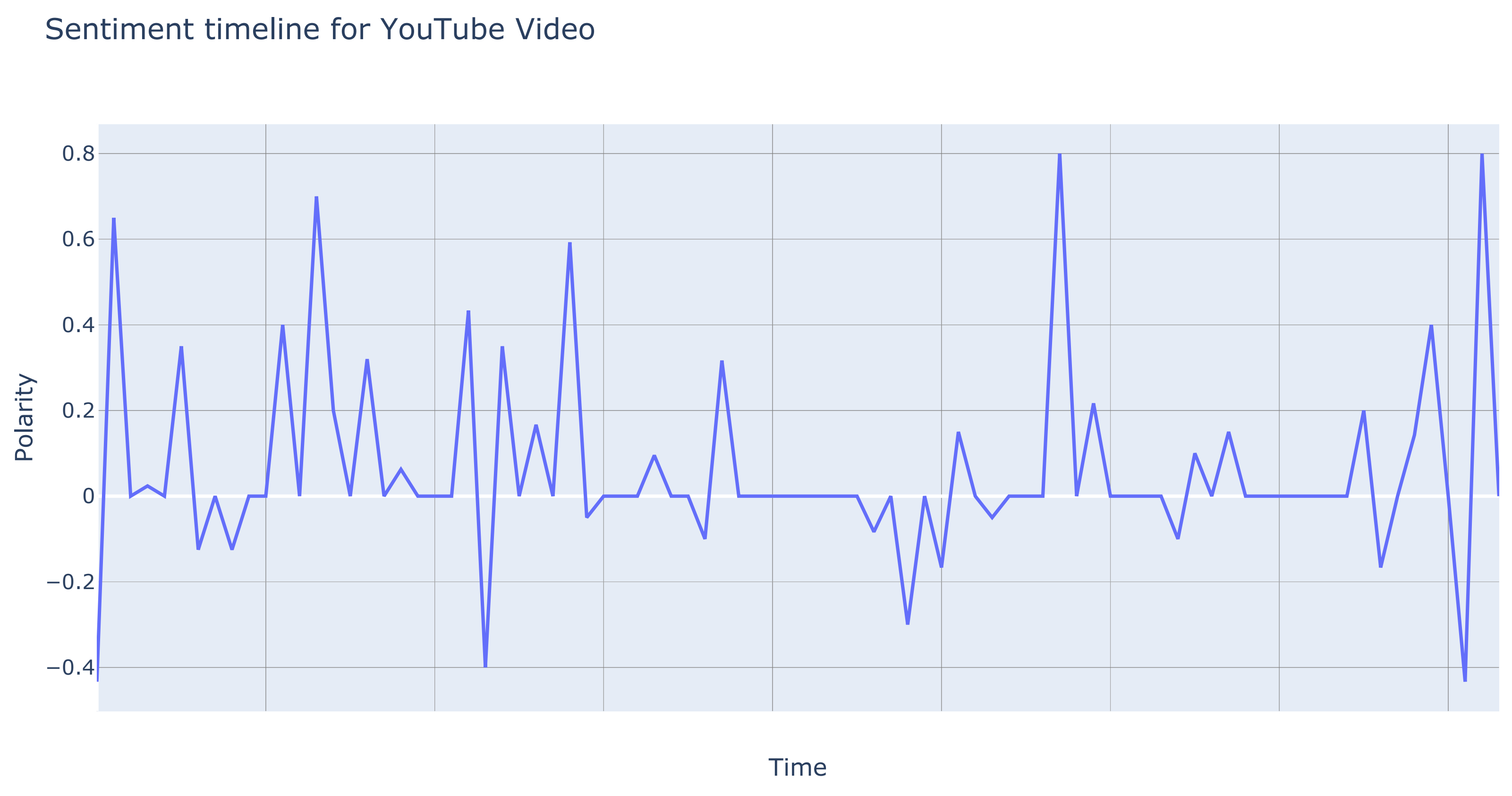
3.(20 points) Retrieve at least 20 (or as many as you can) tweets that contains #TheLionKing and conduct the following data analysis and visualization. a.Conduct sentiment analysis of the tweets and draw a sentiment index lineplot with Plotly (not Seaborn). b.Clean the text to remove all the URL, email, number, etc.

In [24]:
```python
from textblob import TextBlob

# Search for a keyword
keyword = "TheLionKing" + " -filter:retweets"

#Clean the text to remove all the URL, email, number, etc.
tweets = tweepy.Cursor(api.search, q = keyword,
                       lang="en").items(50)
tweet_text = [tweet.text for tweet in tweets]
words = []
for i in range(len(tweet_text)):
    tweet_text[i] = clean(tweet_text[i],
                          no_urls=True,
                          lower=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="",
                          replace_with_number="")

# 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"])

trace = go.Scatter(
        x = sentiment_df.index,
        y = sentiment_df.polarity,
        mode="lines"
    )

data=[trace]

layout = go.Layout(
        title="Sentiment index for the search result: " + "TheLionKing",
        xaxis=dict(title="Tweets"),
        yaxis=dict(title="Polarity")
    )

fig = go.Figure(data=data, layout=layout)
off.iplot(fig)
```

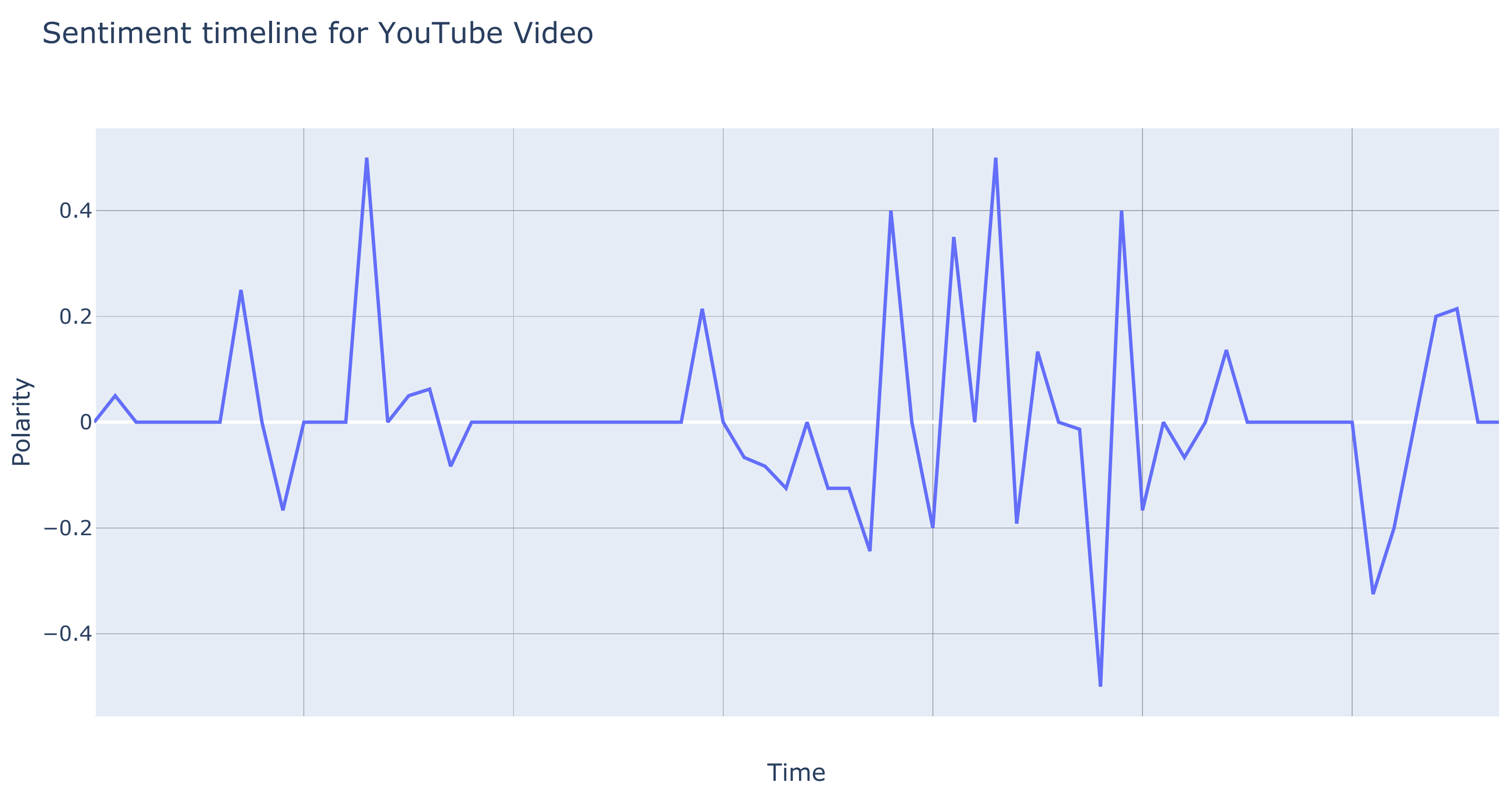


4.Retrieve captions from the following YouTube videos, conduct sentiment analysis and draw the sentiment index timeline using Plotly (not Seaborn). a.(15 points) Create a sentiment timeline for this video: [https://www.youtube.com/watch?v=JtJdZrmeYKc (https://www.youtube.com/watch?v=JtJdZrmeYKc)](https://www.youtube.com/watch?v=JtJdZrmeYKc) b.(15 points) Create a sentiment timeline for a YouTube video of your choice.

In [25]:
```python
from pytube import YouTube

def sa (yt):
    yt.captions.all()
    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_objects]
    caption_df["polarity"] = [sentiment_obj.sentiment.polarity for sentiment_obj in sentiment_objects]

    trace = go.Scatter(
        x = caption_df.index,
        y = caption_df.polarity,
        mode="lines"
    )

    data=[trace]

    layout = go.Layout(
        title="Sentiment timeline for YouTube Video",
        xaxis=dict(title="Time",showticklabels=False),
        yaxis=dict(title="Polarity")
    )

    fig = go.Figure(data=data, layout=layout)
    return fig

yt = YouTube("https://www.youtube.com/watch?v=JtJdZrmeYKc")
fig1 = sa(yt)
off.iplot(fig1)
```



```
In [26]: yt = YouTube("https://www.youtube.com/watch?v=n4crvs-KTBw")
         fig2 = sa(yt)
         off.iplot(fig2)
```


Sentiment timeline for YouTube Video
Polarity
Time