VADER ANALYSIS

In [1]:

```
#Import necessary packages
import pandas as pd
import numpy as np
import seaborn as sns
import re
import matplotlib.pyplot as plt
from nltk.sentiment.vader import SentimentIntensityAnalyzer #Package for the vader method o
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator #Package for wordcloud to s
```

In [3]:

```
#READ the dataset
data = pd.read_csv("omicron1.csv")
# data = data[:7000]
data.head(5)
```

Out[3]:

	id	user_name	user_location	user_description	user_created	user_followers	use
0	1.491840e+18	Nathan Joyner	Los Angeles, CA	Global Venture Captial and Private Equity/Busi...	18/05/2015 20:52	49	
1	1.491840e+18	Gatherer Thompson	Corporate	I'm with the people who are with everyone. A s...	10/05/2009 23:01	639	
2	1.491840e+18	Nathan Joyner	Los Angeles, CA	Global Venture Captial and Private Equity/Busi...	18/05/2015 20:52	49	
3	1.491840e+18	Nathan Joyner	Los Angeles, CA	Global Venture Captial and Private Equity/Busi...	18/05/2015 20:52	49	
4	1.491840e+18	Brownyard Group	Bay Shore, NY	We offer liability coverage for the Security G...	08/12/2009 16:41	375	

In [4]:

```
#Check the data_hashtags and try cleaning it
data["hashtags"].head(3)
```

Out[4]:

```
0                                                 NaN
1    ['CDC', 'vaccinated', 'Omicron', 'hospital']
2                                                 NaN
Name: hashtags, dtype: object
```

In [6]:

```
#TRYED Cleaning the hashtags using #Regular expression and clean to make the data usable

data["hashtags_filled"]= data["hashtags"].fillna('None') # Fill NaN with None
#Clean HASHTAGS to get new values that will be useful for chart
def Convert(hashtag1):
    hashtag1 = re.sub('[\[\]\.*?<.*?>+\'\,]','',hashtag1)
    split_hashtag1= hashtag1.split(" ")
    return split_hashtag1
data["hashtags_new"] = data["hashtags_filled"].apply(Convert)
data["hashtags_new"].loc[10:20]
# words_list
```

Out[6]:

```
10                                  [None]
11    [Covid, pandemic, Ukraine, Omicron]
12                  [COVID, EuropeanUnion]
13                                  [None]
14                                  [None]
15                                  [None]
16                                  [None]
17                     [Omicron, Covid19]
18                                  [None]
19                         [voc, omicron]
20                                  [None]
Name: hashtags_new, dtype: object
```

In [7]:

```
## Cleans the words in the hashtag and put in a set, so that we can reference it.

words_list = []
rough = []
def Converted():
    for hashtag in data["hashtags_new"]:
        if isinstance(hashtag, list):
            for words in hashtag:
                words_list.append(words)
#                 for letters in words:
        else:
            words_list.append(hashtag)
    return words_list
Converted()
set_list = set(words_list)
# set_list

##Thoughts: Rank hashtags then use top ten to check for hashtags
# def Sorted_hashtag():
#     if isinstance(hashtag1, list):
#         if hashtag1 is in set_list:
#             hashtag1 =
#Unable to do it in the end
```

In [9]:

```
#Check data for missing values
data.isna().sum()
# from the summary; user_loaction, user_description, hashtags all have missing values
```

Out[9]:

```
id                    0
user_name             0
user_location      3612
user_description   1330
user_created          0
user_followers        0
user_friends          0
user_favourites       0
user_verified         0
date                  0
text                  0
hashtags           6827
source                0
retweets              0
favorites             0
is_retweet            0
hashtags_filled       0
hashtags_new          0
dtype: int64
```

In [11]:

```
data.head(3)
```

Out[11]:

	id	user_name	user_location	user_description	user_created	user_followers	use
0	1.491840e+18	Nathan Joyner	Los Angeles, CA	Global Venture Captial and Private Equity/Busi...	18/05/2015 20:52	49	
1	1.491840e+18	Gatherer Thompson	Corporate	I'm with the people who are with everyone. A s...	10/05/2009 23:01	639	
2	1.491840e+18	Nathan Joyner	Los Angeles, CA	Global Venture Captial and Private Equity/Busi...	18/05/2015 20:52	49	

In [12]:

```
#Importing necessary packages for sentiment analysis and cleaning of the "text" column

import nltk
import re #Imports regular expression
nltk.download('stopwords') #Put stopwords and ensure it's english
stemmer = nltk.SnowballStemmer("english")
from nltk.corpus import stopwords
import string
stopword=set(stopwords.words('english'))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

In [13]:

```
#Create a function that cleans the text column using #"re"# and ensure it's usable for anal

def clean(text):
    text = str(text).lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    text = [word for word in text.split(' ') if word not in stopword]
    text=" ".join(text)
    text = [stemmer.stem(word) for word in text.split(' ')]
    text=" ".join(text)
    return text
data["text"] = data["text"].apply(clean)
```

In [14]:

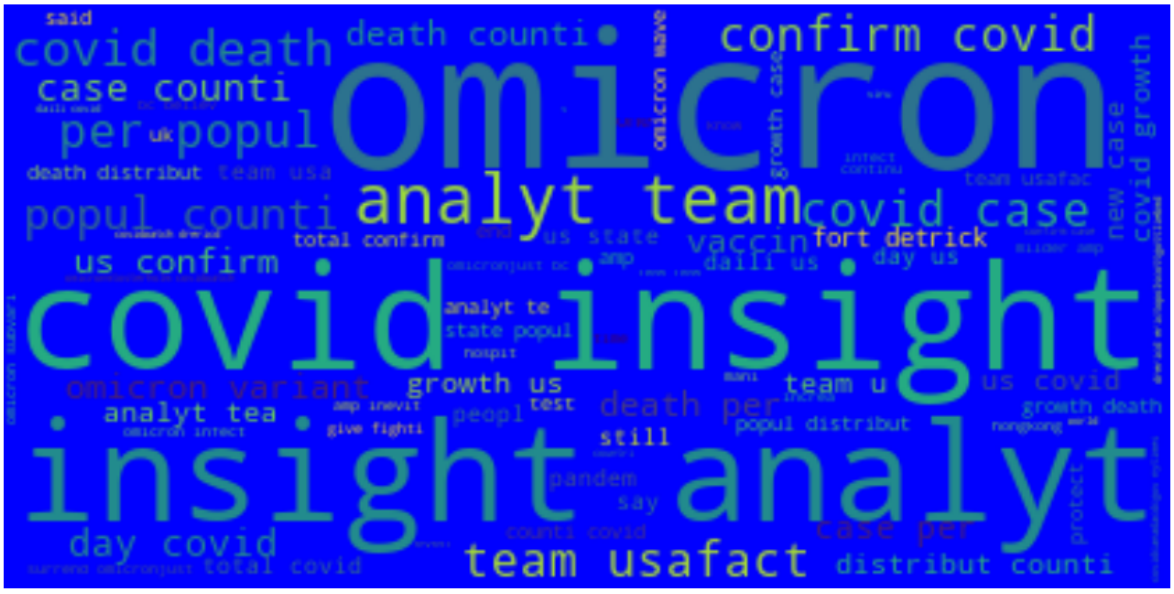
```
#Shows the cleaned data["text"] and know if it's usable for a wordcloud

data["text"]
```

Out[14]:

```
0         daili us confirm covid case counti     covid i...
1         yaschamounk cdc say number fulli vaccin omicro...
2         daili us confirm covid case counti la     covi...
3         daili us confirm covid case counti la     covi...
4          winterolymp alreadi underway us olympian look...
                              ...
17041     us head back offic environ here articl call  w...
17042     long time it great feel watch  live intern cri...
17043     newzealand longcovidkid mask ventil school pol...
17044     patient coinfect differ variant yes common pos...
17045     dcyellowcab pay via electron payment contactle...
Name: text, Length: 17046, dtype: object
```

In [15]:

```
#Plots a WordCloud for better visualization of most common words in text so that it can be

text = " ".join(i for i in data.text)
stopwords = set(STOPWORDS)
wordcloud = WordCloud(stopwords=stopwords, background_color="blue").generate(text)
plt.figure( figsize=(15,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



In [16]:

```
#Plots a WordCloud for better visualization of most common words in hashtags so that it can

tested = " ".join(i for i in data.hashtags_filled)
stopwords = set(STOPWORDS)
wordcloud = WordCloud(stopwords=stopwords, background_color="red").generate(tested)
plt.figure( figsize=(15,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

In [17]:

```
#Gets the result of the sentiments and puts in new columsn which can be better anlaysis

nltk.download('vader_lexicon')
sentiments = SentimentIntensityAnalyzer()
data["Positive"] = [sentiments.polarity_scores(i)["pos"] for i in data["text"]]
data["Negative"] = [sentiments.polarity_scores(i)["neg"] for i in data["text"]]
data["Neutral"] = [sentiments.polarity_scores(i)["neu"] for i in data["text"]]
data1 = data[["text", "Positive", "Negative", "Neutral"]]
print(data1.head())
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
```

	text	Positive	Negative	Neutral
0	daili us confirm covid case counti covid i...	0.000	0.0	1.000
1	yaschamounk cdc say number fulli vaccin omicro...	0.126	0.0	0.874
2	daili us confirm covid case counti la covi...	0.000	0.0	1.000
3	daili us confirm covid case counti la covi...	0.000	0.0	1.000
4	winterolymp alreadi underway us olympian look...	0.000	0.0	1.000

In [18]:

```
#Creates a function that helps to show the end result of everyone's tweet,
# showing that many persons were neutral to the Omicron Variant  using the Vader Model


x = sum(data["Positive"])
y = sum(data["Negative"])
z = sum(data["Neutral"])

def sentiment_score(a, b, c):
    if (a>b) and (a>c):
        print("Positive 😊 ")
    elif (b>a) and (b>c):
        print("Negative 😠 ")
    else:
        print("Neutral 🙂 ")
sentiment_score(x, y, z)
```

```
Neutral 🙂
```

In [19]:

```
new = data[["Positive","Negative","Neutral"]]
new["Score"] = new[["Positive","Negative","Neutral"]].mean(axis =1)
new
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
```

Out[19]:

	Positive	Negative	Neutral	Score
0	0.000	0.000	1.000	0.333333
1	0.126	0.000	0.874	0.333333
2	0.000	0.000	1.000	0.333333
3	0.000	0.000	1.000	0.333333
4	0.000	0.000	1.000	0.333333
...	...	...	...	...
17041	0.184	0.000	0.816	0.333333
17042	0.382	0.000	0.618	0.333333
17043	0.000	0.000	1.000	0.333333
17044	0.231	0.000	0.769	0.333333
17045	0.000	0.123	0.877	0.333333

17046 rows × 4 columns

In [20]:

```
##Tried to clean the followers_list to show which followers are most rampant
#i.e Grouping followers into class that can be used

def Follower_List(followers):
    if followers <= 1000:
        followers = 1000
    elif followers <= 10000 and followers > 1000:
        followers = 10000
    elif followers <= 50000 and followers > 10000:
        followers = 50000
    else:
        followers = 100000
    return followers
data["followers_sorted"] = data["user_followers"].apply(Follower_List)
data["followers_sorted"]
```

Out[20]:

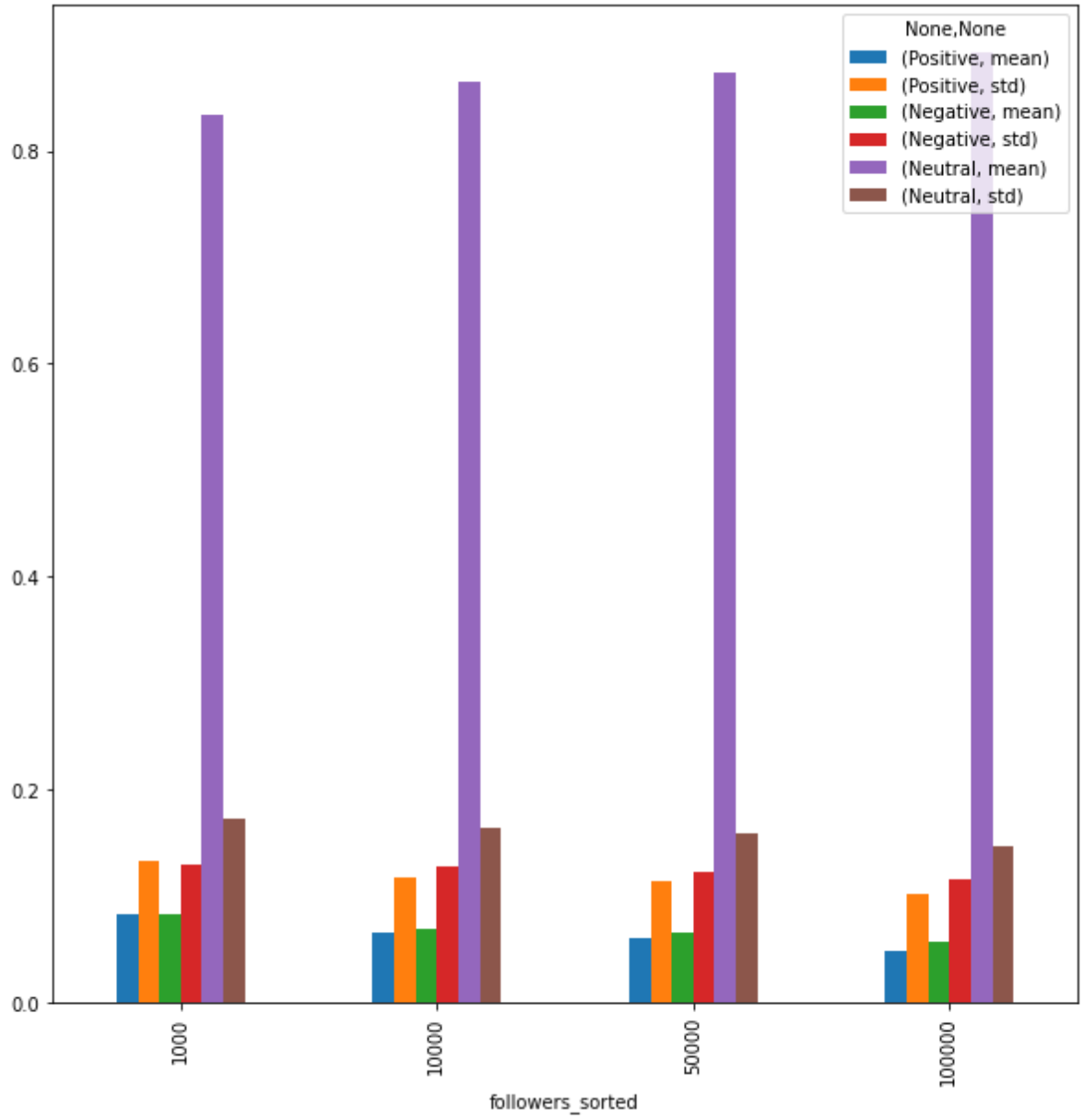
```
0         1000
1         1000
2         1000
3         1000
4         1000
         ...
17041     1000
17042     1000
17043    50000
17044    50000
17045    10000
Name: followers_sorted, Length: 17046, dtype: int64
```

In [21]:

```
#TRIED sorted the data to mean, std in order to check based on the mean and std
#To understand how the std and mean helps

check = ["Positive", "Negative", "Neutral"]
nest1 = ["mean", "std"]
new1= data.groupby("followers_sorted")[check].agg(nest1)
new1.plot(kind = "bar",figsize= (10,10))
new1
```

Out[21]:

	Positive		Negative		Neutral	
	mean	std	mean	std	mean	std
followers_sorted						
1000	0.082959	0.132792	0.083477	0.130641	0.833464	0.172254
10000	0.066458	0.117777	0.069140	0.128445	0.864404	0.165082
50000	0.060710	0.114642	0.065255	0.123520	0.874034	0.158396
100000	0.048929	0.102310	0.057937	0.116655	0.893131	0.146949

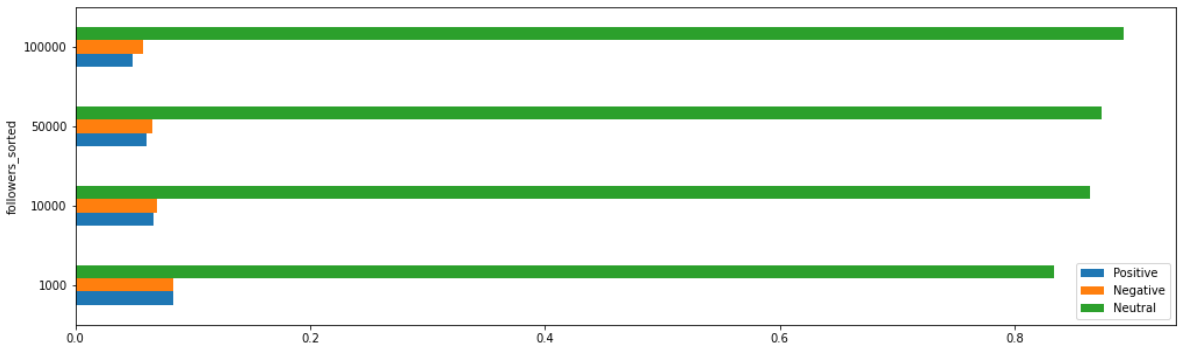


In [111]:

```
#TRIED sorted the data to mean, std in order to check based on the mean and std
#To understand how the  mean helps using a barh
check = ["Positive", "Negative", "Neutral"]
nest = ["mean"]
new1= data.groupby("followers_sorted")[check].mean()
new1.plot(kind = "barh", figsize= (17,5))
new1
```

Out[111]:

followers_sorted	Positive	Negative	Neutral
1000	0.082959	0.083477	0.833464
10000	0.066458	0.069140	0.864404
50000	0.060710	0.065255	0.874034
100000	0.048929	0.057937	0.893131



In [24]:

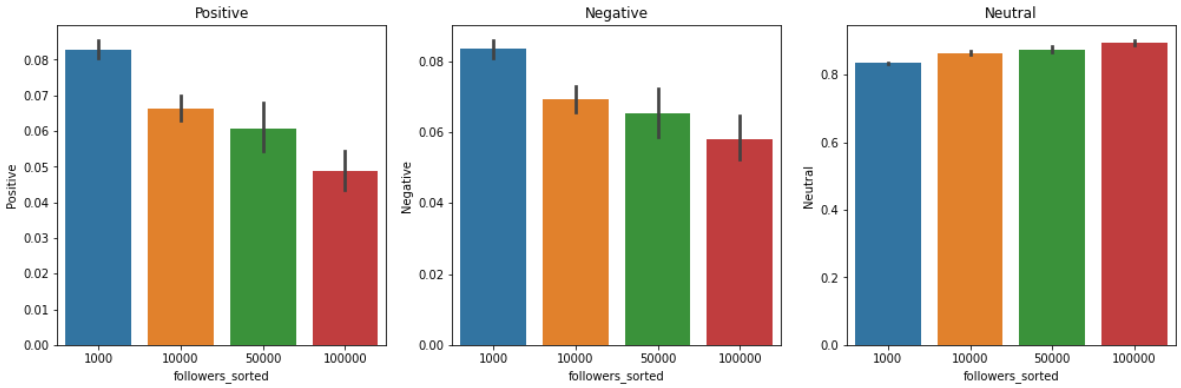
```
#Used Vader model to create a function that finalizes each tweet and know if
# it's Negative, Neutral and Positive
score_list_score = []
def sentiment_score_set():
    for a,b,c in zip(data["Positive"].values,data["Negative"].values,data["Neutral"].values
        if a>b and a>c:
            score1 = "Positive"
        elif b>a and b>c:
            score1 = "Negative"
        else:
            score1 = "Neutral"
        score_list_score.append(score1)
    data["Score1"] = score_list_score

sentiment_score_set() #Calls the function

set(score_list_score)
```

Out[24]:

```
{'Negative', 'Neutral', 'Positive'}
```

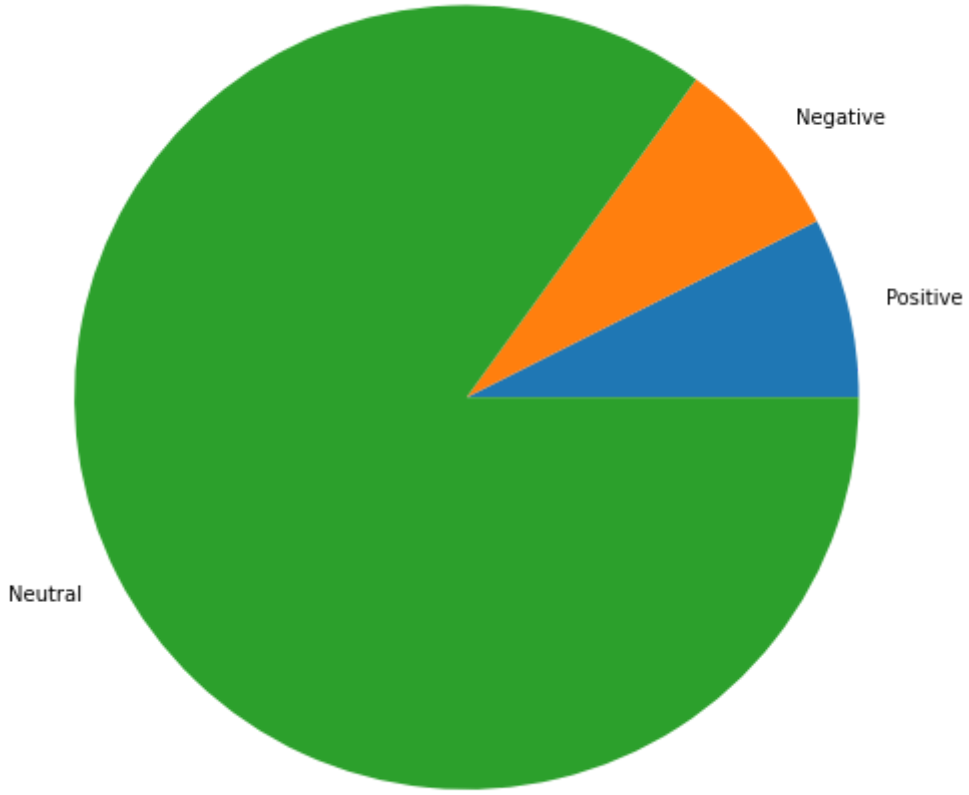
In [26]:

```
#Using Sns to plot charts that are useful for analysis
#The Sns charts uses the followers to show the analysis of positive, negative and Neutral T

fig, axs = plt.subplots(1, 3, figsize= (17,5))
sns.barplot(data = data, x="followers_sorted", y= data["Positive"], ax = axs[0])
sns.barplot(data = data, x="followers_sorted", y= data["Negative"], ax = axs[1])
sns.barplot(data = data, x="followers_sorted", y= data["Neutral"], ax = axs[2])
axs[0].set_title("Positive")
axs[1].set_title("Negative")
axs[2].set_title("Neutral")
plt.show()
```



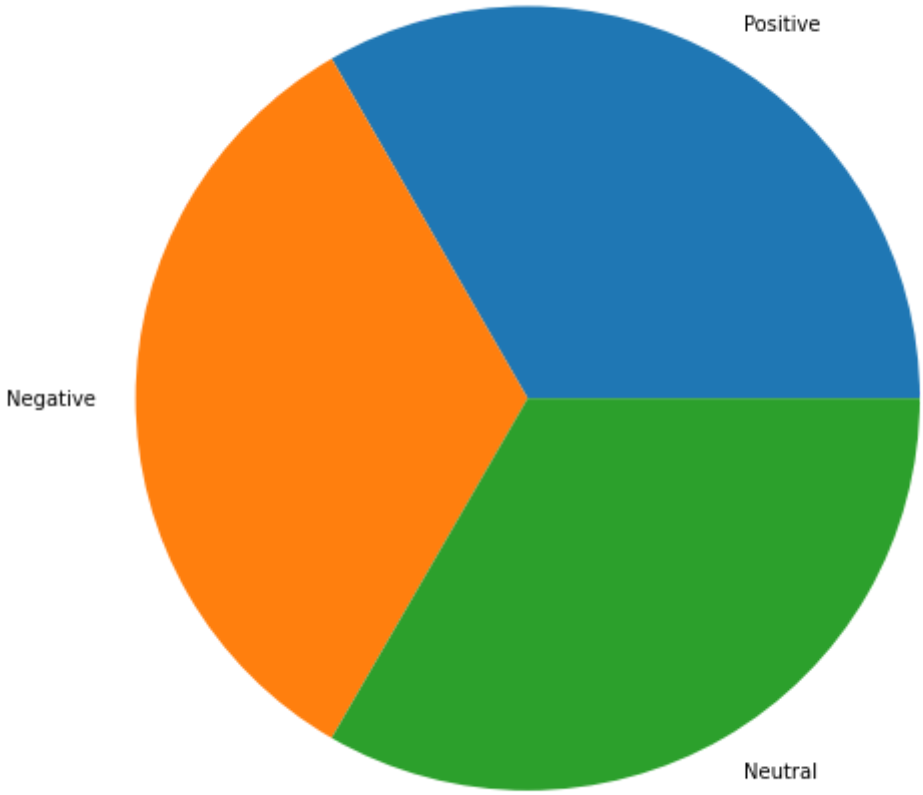
In [27]:

```
#Plots a pie Chart of the the tweets anlaysis of Positive, Negative and Neutral
# Using Vader Model

fig1 = plt.figure(figsize =(12, 9))
lists = ["Positive", "Negative", "Neutral"]
values1 = [sum(data["Positive"]), sum(data["Negative"]), sum(data["Neutral"])]
plt.pie(values1, labels =lists)
plt.show()
```



In [50]:

```
#Pie Chart for analysis of the tweets
# Using Vader Model by followers_sorted
fig1 = plt.figure(figsize =(12, 9))
lists = ["Positive", "Negative", "Neutral"]
values1 = [len([data[data["Score1"]=="Positive"]["followers_sorted"]]), len([data[data["Sco
plt.pie(values1, labels =lists)
plt.show()
```



Roberta Pretrained Model

In [28]:

```
#Installs Transformers for Roberta PreTrained Model

!pip install transformers
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) http
s://us-python.pkg.dev/colab-wheels/public/simple/ (https://us-python.pkg.de
v/colab-wheels/public/simple/)
Collecting transformers
  Downloading transformers-4.24.0-py3-none-any.whl (5.5 MB)
     |████████████████████████████████| 5.5 MB 5.2 MB/s
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-pac
kages (from transformers) (3.8.0)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.
7/dist-packages (from transformers) (2022.6.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.7/dist-
packages (from transformers) (6.0)
Collecting tokenizers!=0.11.3,<0.14,>=0.11.1
  Downloading tokenizers-0.13.1-cp37-cp37m-manylinux_2_17_x86_64.manylinux20
14_x86_64.whl (7.6 MB)
     |████████████████████████████████| 7.6 MB 51.3 MB/s
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-
packages (from transformers) (1.21.6)
Collecting huggingface-hub<1.0,>=0.10.0
  Downloading huggingface_hub-0.10.1-py3-none-any.whl (163 kB)
     |████████████████████████████████| 163 kB 24.8 MB/s
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.
7/dist-packages (from transformers) (4.13.0)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist-p
ackages (from transformers) (4.64.1)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-pac
kages (from transformers) (2.23.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/d
ist-packages (from transformers) (21.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/
python3.7/dist-packages (from huggingface-hub<1.0,>=0.10.0->transformers)
 (4.1.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/py
thon3.7/dist-packages (from packaging>=20.0->transformers) (3.0.9)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-pa
ckages (from importlib-metadata->transformers) (3.10.0)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /u
sr/local/lib/python3.7/dist-packages (from requests->transformers) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist
-packages (from requests->transformers) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.
7/dist-packages (from requests->transformers) (2022.9.24)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.
7/dist-packages (from requests->transformers) (3.0.4)
Installing collected packages: tokenizers, huggingface-hub, transformers
Successfully installed huggingface-hub-0.10.1 tokenizers-0.13.1 transformers
-4.24.0
```

In [29]:

```
#Import necessary packages which will be used for analysis

from transformers import AutoTokenizer
from transformers import AutoModelForSequenceClassification
from scipy.special import softmax
```

In [30]:

```
#Picks the model
MODEL = f"cardiffnlp/twitter-roberta-base-sentiment"
tokenizer = AutoTokenizer.from_pretrained(MODEL)
```

```
Downloading:   0%|          | 0.00/747 [00:00<?, ?B/s]

Downloading:   0%|          | 0.00/899k [00:00<?, ?B/s]

Downloading:   0%|          | 0.00/456k [00:00<?, ?B/s]

Downloading:   0%|          | 0.00/150 [00:00<?, ?B/s]
```

In [31]:

```
#Imports torch from pytorch

import torch
```

In [32]:

```
#Runs the model

model = AutoModelForSequenceClassification.from_pretrained(MODEL)
```

```
Downloading:   0%|          | 0.00/499M [00:00<?, ?B/s]
```

In [33]:

```
#Tries the model on an example text

example = "This oatmeal is not good. Its mushy, soft, I don't like it. Quaker Oats is the w
encoded_text = tokenizer(example, return_tensors='pt')
output = model(**encoded_text)
scores = output[0][0].detach().numpy()
scores = softmax(scores)
scores_dict = {
    'roberta_neg' : scores[0],
    'roberta_neu' : scores[1],
    'roberta_pos' : scores[2]
}
print(scores_dict)
```

```
{'roberta_neg': 0.9763551, 'roberta_neu': 0.020687476, 'roberta_pos': 0.0029
573715}
```

In [44]:

```
#Creates a function that runs the model and would be important
# for itterating through the dataset

def polarity_scores_roberta(test_robert):
    encoded_text = tokenizer(test_robert, return_tensors='pt')
    output = model(**encoded_text)
    scores = output[0][0].detach().numpy()
    scores = softmax(scores)
    scores_dict = {
    'roberta_neg' : scores[0],
    'roberta_neu' : scores[1],
    'roberta_pos' : scores[2]
    }

    return scores_dict
```

In [45]:

```
#Applies the function on the dataset and stores in the dataframe
data["Roberta_Scores"] =data["text"].apply(polarity_scores_roberta)
```

In [47]:

```
#Prints the results
data["Roberta_Scores"]
```

Out[47]:

```
0        {'roberta_neg': 0.050619192, 'roberta_neu': 0....
1        {'roberta_neg': 0.17633206, 'roberta_neu': 0.7...
2        {'roberta_neg': 0.062076133, 'roberta_neu': 0....
3        {'roberta_neg': 0.062076133, 'roberta_neu': 0....
4        {'roberta_neg': 0.030927537, 'roberta_neu': 0....
                              ...
17041    {'roberta_neg': 0.034048945, 'roberta_neu': 0....
17042    {'roberta_neg': 0.0056102853, 'roberta_neu': 0...
17043    {'roberta_neg': 0.08681164, 'roberta_neu': 0.8...
17044    {'roberta_neg': 0.038842432, 'roberta_neu': 0....
17045    {'roberta_neg': 0.020825654, 'roberta_neu': 0....
Name: Roberta_Scores, Length: 17046, dtype: object
```

In [76]:

```
#Itterates through the result and stores in a list

nulla = data["Roberta_Scores"].to_dict()
roberta_neg = []
roberta_neu = []
roberta_pos = []
for i in nulla.values():
  roberta_neg.append(i["roberta_neg"])
  roberta_neu.append(i["roberta_neu"])
  roberta_pos.append(i["roberta_pos"])
```

In [83]:

```
# Stores the data from the list in a dataFrame
data["Roberta_NEG"] = roberta_neg
data["Roberta_NEU"] = roberta_neu
data["Roberta_POS"] = roberta_pos
data.columns
```

Out[83]:

```
Index(['id', 'user_name', 'user_location', 'user_description', 'user_create
d',
       'user_followers', 'user_friends', 'user_favourites', 'user_verified',
       'date', 'text', 'hashtags', 'source', 'retweets', 'favorites',
       'is_retweet', 'hashtags_filled', 'hashtags_new', 'Positive', 'Negativ
e',
       'Neutral', 'followers_sorted', 'Score1', 'Roberta_Scores',
       'Vader_Score', 'Roberta_NEG', 'Roberta_NEU', 'Roberta_POS'],
      dtype='object')
```

In [109]:

```
#Creates a function that helps to show the end result of everyone's tweet,
# showing that many persons were neutral to the Omicron Variant
# using the Roberta Pre Trained Model

x = sum(data["Roberta_POS"])
y = sum(data["Roberta_NEG"])
z = sum(data["Roberta_NEU"])

def sentiment_score(a, b, c):
    if (a>b) and (a>c):
        print("Positive 😊 ")
    elif (b>a) and (b>c):
        print("Negative 😠 ")
    else:
        print("Neutral 🙂 ")
sentiment_score(x, y, z)
```

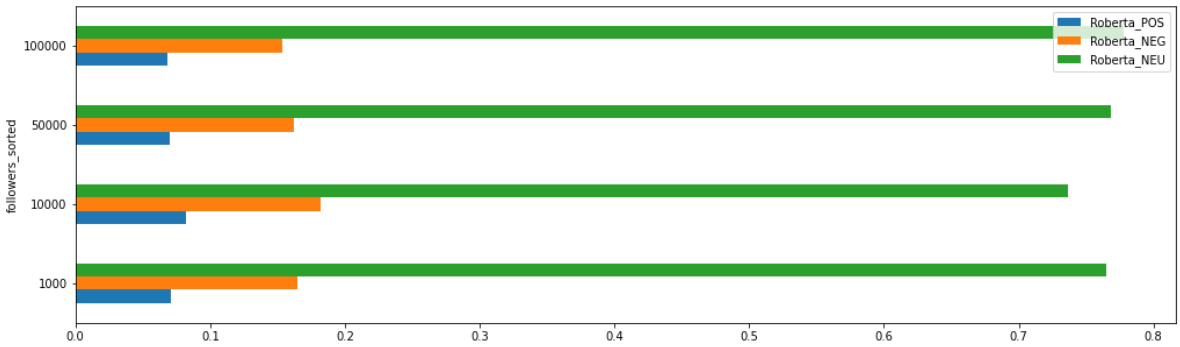
```
Neutral 🙂
```

In [110]:

```
#TRIED sorted the data to mean in order to check based on the mean and std
#To understand how the mean helps using a barh using the result
# from Roberta Model

check = ["Roberta_POS", "Roberta_NEG", "Roberta_NEU"]
new1= data.groupby("followers_sorted")[check].mean()
new1.plot(kind = "barh", figsize= (17,5))
new1
```

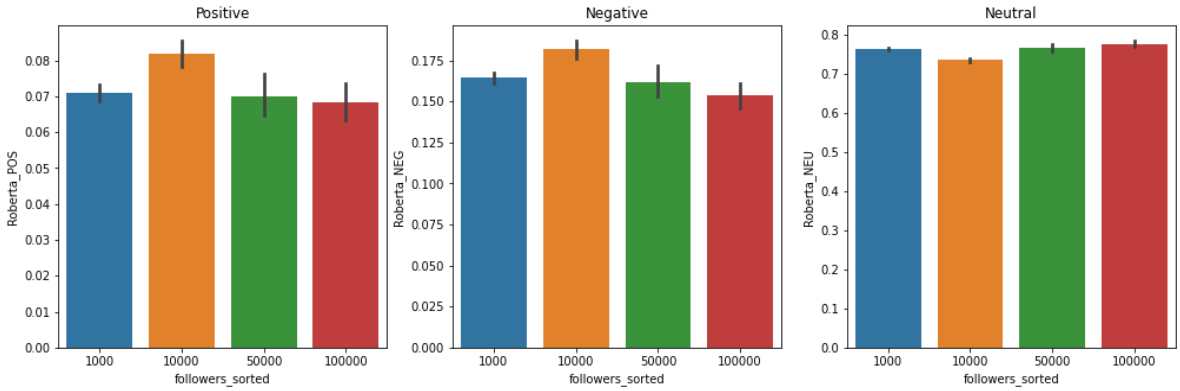
Out[110]:

followers_sorted	Roberta_POS	Roberta_NEG	Roberta_NEU
1000	0.070910	0.164321	0.764769
10000	0.081902	0.181656	0.736442
50000	0.070077	0.161917	0.768006
100000	0.068437	0.153583	0.777981



In [106]:

```
#Using Sns to plot charts that are useful for analysis
#The Sns charts uses the followers to show the analysis of
# positive, negative and Neutral Tweets Roberta PreTrained Model
fig, axs = plt.subplots(1, 3, figsize= (17,5))
sns.barplot(data = data, x="followers_sorted", y= data["Roberta_POS"], ax = axs[0])
sns.barplot(data = data, x="followers_sorted", y= data["Roberta_NEG"], ax = axs[1])
sns.barplot(data = data, x="followers_sorted", y= data["Roberta_NEU"], ax = axs[2])
axs[0].set_title("Positive")
axs[1].set_title("Negative")
axs[2].set_title("Neutral")
plt.show()
```



COMPARISON OF BOTH MODELS AND PREDICTIONS

In [51]:

```
#Imports models for Comparisons
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
```

In [52]:

```
data["Vader_Score"]= new["Score"]
data.columns
```

Out[52]:

```
Index(['id', 'user_name', 'user_location', 'user_description', 'user_create
d',
       'user_followers', 'user_friends', 'user_favourites', 'user_verified',
       'date', 'text', 'hashtags', 'source', 'retweets', 'favorites',
       'is_retweet', 'hashtags_filled', 'hashtags_new', 'Positive', 'Negativ
e',
       'Neutral', 'followers_sorted', 'Score1', 'Roberta_NEG', 'Roberta_NE
U',
       'Roberta_Scores', 'Vader_Score'],
      dtype='object')
```

In [60]:

```
#Using Vader Model and Knn for Prediction
v_model = KNeighborsClassifier(n_neighbors = 3)
x_data_v = data[["Positive", "Negative", "Neutral","user_followers"]]
y_data_v = data[["Score1"]]
x_train_v, x_test_v, y_train_v, y_test_v = train_test_split(x_data_v, y_data_v, test_size =
v_model.fit(x_train_v, y_train_v)
v_predicted = v_model.predict(x_test_v)
print("Model_Accuracy_Vader =", accuracy_score(v_predicted, y_test_v))
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neighbors/_classification.py:
198: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using ra
vel().
  return self._fit(X, y)

Model_Accuracy_Vader = 0.9828420589529256
```

In [95]:

```
#Using Roberta Model and Knn for Prediction
r_model = KNeighborsClassifier(n_neighbors = 3)
x_data_r = data[["Roberta_NEG","Roberta_NEU","Roberta_POS","user_followers"]]
y_data_r = data["Score1"]
x_train_r, x_test_r, y_train_r, y_test_r = train_test_split(x_data_r, y_data_r, test_size =
r_model.fit(x_train_r, y_train_r)
r_predicted = r_model.predict(x_test_r)
print("Model_Accuracy_Roberta =", accuracy_score(r_predicted, y_test_r))
```

```
Model_Accuracy_Roberta = 0.9809356210588063
```

VaderModel vs Roberta Model

In [105]:

```
x_data_rr = data[["Roberta_NEG","Roberta_NEU","Roberta_POS"]]
x_data_vv = data[["Negative", "Neutral","Positive"]]
print("Roberta_Model vs Vader_Model : = ", r2_score(x_data_rr, x_data_vv))
print("Vader_Model vs Roberta_Model: = ", r2_score(x_data_vv, x_data_rr))
```

```
Roberta_Model vs Vader_Model : =  -0.25561924584471485
Vader_Model vs Roberta_Model: =  -0.472897175628578
```

In [ ]:

```
```