Sentiment Analysis with VADER in NLTK for Shakespeare’s Poems:

1.1-1.3

In [12]:

```
from os import walk
import os,re
from pprint import pprint
import statistics as st

# Simply Provide the path to the folder
# Reads all files recursively
folder_path =r"Full_Folder_Path"
os.chdir(folder_path)
filenames = next(walk(folder_path), (None, None, []))[2]
```

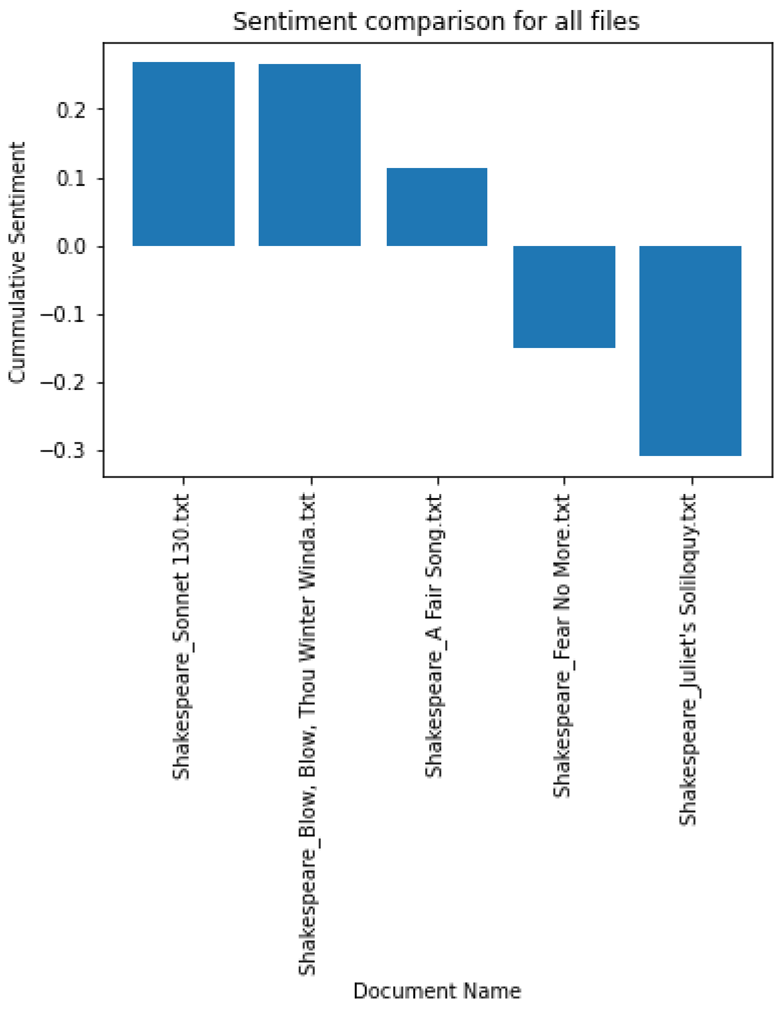
In [13]:

```
# Preprocess poems:

import string
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import collections

sid = SentimentIntensityAnalyzer()

import numpy as np

def GetSentiment(file_name):

    f = open(file_name,'r')
    l=f.read().split('\n')

    for i in l:
        if i in string.punctuation:
            l.remove(i)
    lst=[]
    for line in l:
        s = sid.polarity_scores(line)
        if s['compound']!= 0:
            lst.append(s['compound'])

    return st.mean(lst)

# Assemble & Sort results:
dict={}
for file in filenames:
    dict[file]=round(GetSentiment(file),4)


sorted_dict = collections.OrderedDict(sorted(dict.items(), key=lambda x: x[1],reverse=T
pprint(sorted_dict)


# Visualize Results:
import matplotlib.pyplot as plt

keys = sorted_dict.keys()
values = sorted_dict.values()
```

```
plt.bar(keys, values)
plt.title('Sentiment comparison for all files')
plt.xticks(rotation=90)
plt.ylabel('Cummulative Sentiment')
plt.xlabel('Document Name')
```

```
OrderedDict([('Shakespeare_Sonnet 130.txt', 0.2695),
             ('Shakespeare_Blow, Blow, Thou Winter Winda.txt', 0.2656),
             ('Shakespeare_A Fair Song.txt', 0.1131),
             ('Shakespeare_Fear No More.txt', -0.1511),
             ("Shakespeare_Juliet's Soliloquy.txt", -0.3099)])
```

```
Out[13]: Text(0.5, 0, 'Document Name')
```



1.3

The program is robust (Verified and read the poems to compare rankings), convenient (simply provide the path to the folder where files/poems are stored), and additional visualization has also been added to compare output.

1.4.1

The program is robust (Verified and read the poems to compare rankings), convenient (simply provide the path to the folder where files/poems are stored), and additional visualization has also been added to compare output.

1.4.2

After visually inspecting the output, the compound score is the best choice to measure the overall sentiment of the poem/document. It contains both the magnitude and direction (+/-) of the sentiment 'vector'. The overall sentiment is the average of line/verse sentiments. An example is shown in the plot where Shakespeare_Sonnet and Shakespeare_Juliet's Soliloquy are of similar magnitudes but opposite sentiments where we can read and verify that the latter is obviously a tragedy.

1.4.3

A potential application is to predict stock market response based on sentiments related to statistically significant variable including the Federal Government, COVID Cases, International Politics, and News Media... For example, during the start of COVID-19 pharmaceutical companies were racing and competing to get the vaccine, and therefore sentiment analysis on the medical news, cases, and each vaccine developer, will certainly guide in making the right investment. Similarly, one could use sentiment analysis to predict the direction of oil, travel, and hospitality stocks given the rise/drop of COVID-19 cases and even vaccine development progress/news.

```
In [ ]:
```