NLTK

August 31, 2024

[1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

plt.style.use('ggplot')

import nltk
```

[2]:
```python
# read data
```

[3]:
```python
df = pd.read_csv('C:/Users/bendh/Downloads/ntlk/Reviews.csv')
```

[4]:
```python
df.head()
```

[4]:
```
   Id   ProductId          UserId                      ProfileName  \
0   1  B001E4KFG0  A3SGXH7AUHU8GW                       delmartian
1   2  B00813GRG4  A1D87F6ZCVE5NK                           dll pa
2   3  B000LQOCH0   ABXLMWJIXXAIN  Natalia Corres "Natalia Corres"
3   4  B000UA0QIQ  A395BORC6FGVXV                             Karl
4   5  B006K2ZZ7K  A1UQRSCLF8GW1T    Michael D. Bigham "M. Wassir"

   HelpfulnessNumerator  HelpfulnessDenominator  Score        Time  \
0                     1                       1      5  1303862400
1                     0                       0      1  1346976000
2                     1                       1      4  1219017600
3                     3                       3      2  1307923200
4                     0                       0      5  1350777600

                 Summary                                               Text
0  Good Quality Dog Food  I have bought several of the Vitality canned d…
1      Not as Advertised  Product arrived labeled as Jumbo Salted Peanut…
2  "Delight" says it all  This is a confection that has been around a fe…
3         Cough Medicine  If you are looking for the secret ingredient i…
4            Great taffy  Great taffy at a great price.  There was a wid…
```

[5]:
```python
df['Text'].values[0]
```

[5]: 'I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product looks more like a stew than a processed meat and it smells better. My Labrador is finicky and she appreciates this product better than most.'

[6]:
```
df['Text'][0]
```

[6]: 'I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product looks more like a stew than a processed meat and it smells better. My Labrador is finicky and she appreciates this product better than most.'

[7]:
```
df.shape
```

[7]: (568454, 10)

[8]:
```
df = df.head(500)
```

[9]:
```
df.shape
```

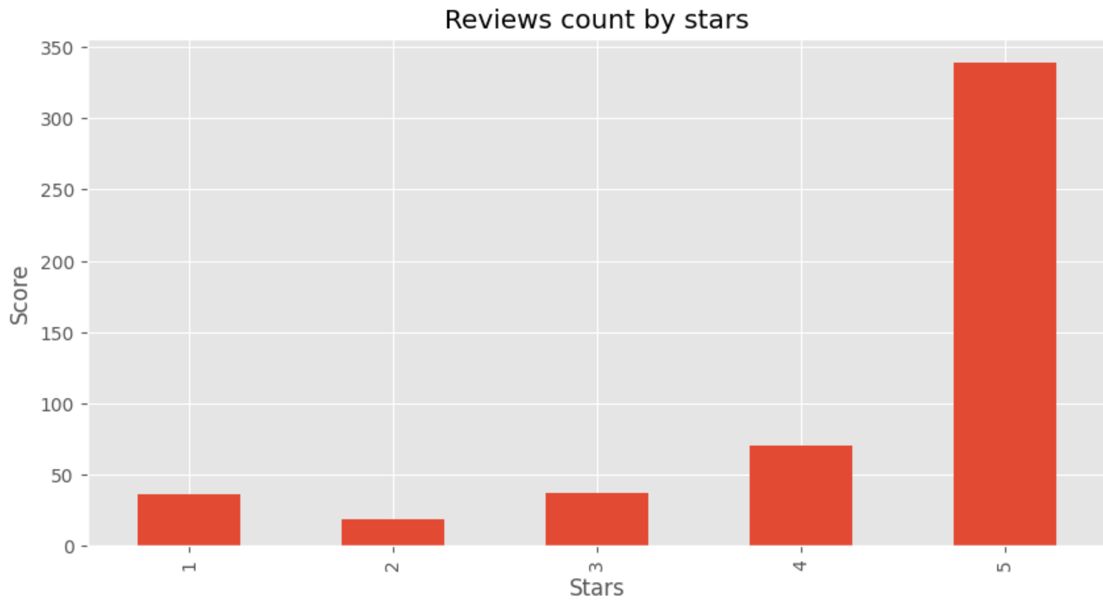
[9]: (500, 10)

[10]:
```
## Quick EDA
```

[11]:
```
df['Score'].value_counts()
```

[11]:
```
Score
5    339
4     70
3     37
1     36
2     18
Name: count, dtype: int64
```

[12]:
```
ax = df['Score'].value_counts().sort_index() \
                .plot(kind= 'bar',
                      title= 'Reviews count by stars',
                      figsize=(10,5))
ax.set_xlabel('Stars')
ax.set_ylabel('Score')
plt.show()
```



[13]:
```
##basic NLTK
```

[14]:
```
example = df['Text'][50]
print(example)
```

```
This oatmeal is not good. Its mushy, soft, I don't like it. Quaker Oats is the
way to go.
```

[15]:
```
print(nltk.data.path)
```

```
['C:\\Users\\bendh/nltk_data',
'c:\\users\\bendh\\appdata\\local\\programs\\python\\python39\\nltk_data', 'c:\\
users\\bendh\\appdata\\local\\programs\\python\\python39\\share\\nltk_data',
'c:\\users\\bendh\\appdata\\local\\programs\\python\\python39\\lib\\nltk_data',
'C:\\Users\\bendh\\AppData\\Roaming\\nltk_data', 'C:\\nltk_data',
'D:\\nltk_data', 'E:\\nltk_data']
```

[16]:
```
tokens = nltk.word_tokenize(example)
tokens[:10]
```

[16]:
```
['This', 'oatmeal', 'is', 'not', 'good', '.', 'Its', 'mushy', ',', 'soft']
```

[17]:
```
tagged = nltk.pos_tag(tokens)
print(tagged)
```

```
[('This', 'DT'), ('oatmeal', 'NN'), ('is', 'VBZ'), ('not', 'RB'), ('good',
'JJ'), ('.', '.'), ('Its', 'PRP$'), ('mushy', 'NN'), (',', ','), ('soft', 'JJ'),
(',', ','), ('I', 'PRP'), ('do', 'VBP'), ("n't", 'RB'), ('like', 'VB'), ('it',
```

```
'PRP'), ('.', '.'), ('Quaker', 'NNP'), ('Oats', 'NNPS'), ('is', 'VBZ'), ('the',
'DT'), ('way', 'NN'), ('to', 'TO'), ('go', 'VB'), ('.', '.')]
```

[18]:
```
tagged = nltk.pos_tag(tokens)
tagged[:10]
```

[18]:
```
[('This', 'DT'),
 ('oatmeal', 'NN'),
 ('is', 'VBZ'),
 ('not', 'RB'),
 ('good', 'JJ'),
 ('.', '.'),
 ('Its', 'PRP$'),
 ('mushy', 'NN'),
 (',', ','),
 ('soft', 'JJ')]
```

[19]:
```
entities = nltk.chunk.ne_chunk(tagged) #it specifies persons, locations,␣
↪organizations...etc (like Quaker is and ORG)
entities.pprint()
```

```
(S
  This/DT
  oatmeal/NN
  is/VBZ
  not/RB
  good/JJ
  ./.
  Its/PRP$
  mushy/NN
  ,/,
  soft/JJ
  ,/,
  I/PRP
  do/VBP
  n't/RB
  like/VB
  it/PRP
  ./.
  (ORGANIZATION Quaker/NNP Oats/NNPS)
  is/VBZ
  the/DT
  way/NN
  to/TO
  go/VB
  ./.)
```

[20]:
```
from nltk.sentiment import SentimentIntensityAnalyzer
# import library to monitor and display progress bars for loops and iterative␣
↪processes (tqdm)
from tqdm.notebook import tqdm
```

[21]:
```
#create our sentiment object
```

[22]:
```
sia = SentimentIntensityAnalyzer()
```

[23]:
```
sia.polarity_scores ('I am so happy!')
```

[23]:
```
{'neg': 0.0, 'neu': 0.318, 'pos': 0.682, 'compound': 0.6468}
```

[24]:
```
sia.polarity_scores ('This is the worst thing ever!')
```

[24]:
```
{'neg': 0.468, 'neu': 0.532, 'pos': 0.0, 'compound': -0.6588}
```

[25]:
```
sia.polarity_scores (example)
```

[25]:
```
{'neg': 0.22, 'neu': 0.78, 'pos': 0.0, 'compound': -0.5448}
```

[26]:
```
# Run the polarity score on the entire dataset
```

[27]:
```
#iterrows iterate through each row with the index i and total specifies the␣
↪total number of iterations
for i, row in tqdm (df.iterrows(), total =len(df)):
    text = row['Text']
    myid = row['Id']
    break # this prevents the loop to continue (so it stops at the first␣
↪iteration)
```

```
0%|          | 0/500 [00:00<?, ?it/s]
```

[28]:
```
res = {} # creates empty dictionary (not set, because for creating sets we use␣
↪set() not {})
for i, row in tqdm (df.iterrows(), total =len(df)):
    text = row['Text']
    myid = row['Id']
    res[myid] = sia.polarity_scores(text) #here myid is the key for the text␣
↪that is being analysed
```

```
0%|          | 0/500 [00:00<?, ?it/s]
```

[29]:
```
#result of the dictionary
```

[30]:
```
res
```

[30]:
```
{1: {'neg': 0.0, 'neu': 0.695, 'pos': 0.305, 'compound': 0.9441},
 2: {'neg': 0.138, 'neu': 0.862, 'pos': 0.0, 'compound': -0.5664},
```

```
3: {'neg': 0.091, 'neu': 0.754, 'pos': 0.155, 'compound': 0.8265},
4: {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0},
5: {'neg': 0.0, 'neu': 0.552, 'pos': 0.448, 'compound': 0.9468},
6: {'neg': 0.029, 'neu': 0.809, 'pos': 0.163, 'compound': 0.883},
7: {'neg': 0.034, 'neu': 0.693, 'pos': 0.273, 'compound': 0.9346},
8: {'neg': 0.0, 'neu': 0.52, 'pos': 0.48, 'compound': 0.9487},
9: {'neg': 0.0, 'neu': 0.851, 'pos': 0.149, 'compound': 0.6369},
10: {'neg': 0.0, 'neu': 0.705, 'pos': 0.295, 'compound': 0.8313},
11: {'neg': 0.017, 'neu': 0.846, 'pos': 0.137, 'compound': 0.9746},
12: {'neg': 0.113, 'neu': 0.887, 'pos': 0.0, 'compound': -0.7579},
13: {'neg': 0.031, 'neu': 0.923, 'pos': 0.046, 'compound': 0.296},
14: {'neg': 0.0, 'neu': 0.355, 'pos': 0.645, 'compound': 0.9466},
15: {'neg': 0.104, 'neu': 0.632, 'pos': 0.264, 'compound': 0.6486},
16: {'neg': 0.0, 'neu': 0.861, 'pos': 0.139, 'compound': 0.5719},
17: {'neg': 0.097, 'neu': 0.694, 'pos': 0.209, 'compound': 0.7481},
18: {'neg': 0.0, 'neu': 0.61, 'pos': 0.39, 'compound': 0.8883},
19: {'neg': 0.012, 'neu': 0.885, 'pos': 0.103, 'compound': 0.8957},
20: {'neg': 0.0, 'neu': 0.863, 'pos': 0.137, 'compound': 0.6077},
21: {'neg': 0.0, 'neu': 0.865, 'pos': 0.135, 'compound': 0.6249},
22: {'neg': 0.0, 'neu': 0.739, 'pos': 0.261, 'compound': 0.9153},
23: {'neg': 0.0, 'neu': 0.768, 'pos': 0.232, 'compound': 0.7687},
24: {'neg': 0.085, 'neu': 0.771, 'pos': 0.143, 'compound': 0.2617},
25: {'neg': 0.038, 'neu': 0.895, 'pos': 0.068, 'compound': 0.3939},
26: {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0},
27: {'neg': 0.128, 'neu': 0.872, 'pos': 0.0, 'compound': -0.296},
28: {'neg': 0.04, 'neu': 0.808, 'pos': 0.152, 'compound': 0.5956},
29: {'neg': 0.022, 'neu': 0.669, 'pos': 0.309, 'compound': 0.9913},
30: {'neg': 0.017, 'neu': 0.846, 'pos': 0.137, 'compound': 0.9746},
31: {'neg': 0.041, 'neu': 0.692, 'pos': 0.267, 'compound': 0.9713},
32: {'neg': 0.0, 'neu': 0.484, 'pos': 0.516, 'compound': 0.9153},
33: {'neg': 0.069, 'neu': 0.839, 'pos': 0.092, 'compound': 0.7103},
34: {'neg': 0.024, 'neu': 0.72, 'pos': 0.256, 'compound': 0.9779},
35: {'neg': 0.0, 'neu': 0.874, 'pos': 0.126, 'compound': 0.9091},
36: {'neg': 0.024, 'neu': 0.821, 'pos': 0.155, 'compound': 0.7622},
37: {'neg': 0.0, 'neu': 0.754, 'pos': 0.246, 'compound': 0.9196},
38: {'neg': 0.0, 'neu': 0.938, 'pos': 0.062, 'compound': 0.4457},
39: {'neg': 0.05, 'neu': 0.846, 'pos': 0.104, 'compound': 0.7638},
40: {'neg': 0.0, 'neu': 0.856, 'pos': 0.144, 'compound': 0.8114},
41: {'neg': 0.033, 'neu': 0.82, 'pos': 0.147, 'compound': 0.9301},
42: {'neg': 0.03, 'neu': 0.848, 'pos': 0.122, 'compound': 0.9435},
43: {'neg': 0.0, 'neu': 0.588, 'pos': 0.412, 'compound': 0.9441},
44: {'neg': 0.0, 'neu': 0.685, 'pos': 0.315, 'compound': 0.9161},
45: {'neg': 0.031, 'neu': 0.778, 'pos': 0.191, 'compound': 0.8421},
46: {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0},
47: {'neg': 0.0, 'neu': 0.737, 'pos': 0.263, 'compound': 0.9169},
48: {'neg': 0.0, 'neu': 0.868, 'pos': 0.132, 'compound': 0.4404},
49: {'neg': 0.0, 'neu': 0.821, 'pos': 0.179, 'compound': 0.747},
```

50: {'neg': 0.056, 'neu': 0.865, 'pos': 0.079, 'compound': 0.2363},
51: {'neg': 0.22, 'neu': 0.78, 'pos': 0.0, 'compound': -0.5448},
52: {'neg': 0.047, 'neu': 0.735, 'pos': 0.218, 'compound': 0.9194},
53: {'neg': 0.09, 'neu': 0.858, 'pos': 0.052, 'compound': -0.8259},
54: {'neg': 0.075, 'neu': 0.925, 'pos': 0.0, 'compound': -0.3612},
55: {'neg': 0.0, 'neu': 0.857, 'pos': 0.143, 'compound': 0.8761},
56: {'neg': 0.071, 'neu': 0.708, 'pos': 0.221, 'compound': 0.8908},
57: {'neg': 0.029, 'neu': 0.694, 'pos': 0.277, 'compound': 0.908},
58: {'neg': 0.0, 'neu': 0.701, 'pos': 0.299, 'compound': 0.91},
59: {'neg': 0.0, 'neu': 0.611, 'pos': 0.389, 'compound': 0.9323},
60: {'neg': 0.0, 'neu': 0.638, 'pos': 0.362, 'compound': 0.8807},
61: {'neg': 0.0, 'neu': 0.9, 'pos': 0.1, 'compound': 0.4404},
62: {'neg': 0.0, 'neu': 0.741, 'pos': 0.259, 'compound': 0.8442},
63: {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0},
64: {'neg': 0.055, 'neu': 0.765, 'pos': 0.179, 'compound': 0.9817},
65: {'neg': 0.046, 'neu': 0.75, 'pos': 0.205, 'compound': 0.8674},
66: {'neg': 0.04, 'neu': 0.822, 'pos': 0.138, 'compound': 0.5165},
67: {'neg': 0.057, 'neu': 0.869, 'pos': 0.073, 'compound': 0.492},
68: {'neg': 0.183, 'neu': 0.776, 'pos': 0.041, 'compound': -0.9116},
69: {'neg': 0.135, 'neu': 0.71, 'pos': 0.155, 'compound': -0.0096},
70: {'neg': 0.344, 'neu': 0.52, 'pos': 0.136, 'compound': -0.7345},
71: {'neg': 0.036, 'neu': 0.916, 'pos': 0.048, 'compound': 0.2228},
72: {'neg': 0.078, 'neu': 0.701, 'pos': 0.222, 'compound': 0.9733},
73: {'neg': 0.025, 'neu': 0.653, 'pos': 0.323, 'compound': 0.9787},
74: {'neg': 0.093, 'neu': 0.762, 'pos': 0.144, 'compound': 0.9665},
75: {'neg': 0.0, 'neu': 0.872, 'pos': 0.128, 'compound': 0.2263},
76: {'neg': 0.106, 'neu': 0.768, 'pos': 0.126, 'compound': 0.1098},
77: {'neg': 0.019, 'neu': 0.898, 'pos': 0.083, 'compound': 0.5647},
78: {'neg': 0.034, 'neu': 0.798, 'pos': 0.168, 'compound': 0.8303},
79: {'neg': 0.0, 'neu': 0.763, 'pos': 0.237, 'compound': 0.7814},
80: {'neg': 0.087, 'neu': 0.589, 'pos': 0.324, 'compound': 0.8636},
81: {'neg': 0.0, 'neu': 0.723, 'pos': 0.277, 'compound': 0.9098},
82: {'neg': 0.0, 'neu': 0.663, 'pos': 0.337, 'compound': 0.9041},
83: {'neg': 0.04, 'neu': 0.794, 'pos': 0.165, 'compound': 0.9957},
84: {'neg': 0.055, 'neu': 0.767, 'pos': 0.178, 'compound': 0.8642},
85: {'neg': 0.109, 'neu': 0.676, 'pos': 0.214, 'compound': 0.8431},
86: {'neg': 0.035, 'neu': 0.698, 'pos': 0.267, 'compound': 0.9487},
87: {'neg': 0.019, 'neu': 0.855, 'pos': 0.126, 'compound': 0.8797},
88: {'neg': 0.05, 'neu': 0.735, 'pos': 0.215, 'compound': 0.7424},
89: {'neg': 0.048, 'neu': 0.762, 'pos': 0.19, 'compound': 0.9716},
90: {'neg': 0.029, 'neu': 0.645, 'pos': 0.326, 'compound': 0.9554},
91: {'neg': 0.0, 'neu': 0.833, 'pos': 0.167, 'compound': 0.7351},
92: {'neg': 0.0, 'neu': 0.837, 'pos': 0.163, 'compound': 0.6249},
93: {'neg': 0.069, 'neu': 0.663, 'pos': 0.268, 'compound': 0.8255},
94: {'neg': 0.01, 'neu': 0.781, 'pos': 0.208, 'compound': 0.9882},
95: {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0},
96: {'neg': 0.031, 'neu': 0.732, 'pos': 0.237, 'compound': 0.9273},

97: {'neg': 0.0, 'neu': 0.818, 'pos': 0.182, 'compound': 0.982},
98: {'neg': 0.053, 'neu': 0.793, 'pos': 0.154, 'compound': 0.7729},
99: {'neg': 0.024, 'neu': 0.91, 'pos': 0.066, 'compound': 0.5106},
100: {'neg': 0.173, 'neu': 0.735, 'pos': 0.092, 'compound': -0.5267},
101: {'neg': 0.0, 'neu': 0.807, 'pos': 0.193, 'compound': 0.7717},
102: {'neg': 0.103, 'neu': 0.752, 'pos': 0.145, 'compound': 0.2285},
103: {'neg': 0.0, 'neu': 0.75, 'pos': 0.25, 'compound': 0.9287},
104: {'neg': 0.0, 'neu': 0.859, 'pos': 0.141, 'compound': 0.7249},
105: {'neg': 0.051, 'neu': 0.577, 'pos': 0.372, 'compound': 0.9313},
106: {'neg': 0.0, 'neu': 0.696, 'pos': 0.304, 'compound': 0.9603},
107: {'neg': 0.0, 'neu': 0.791, 'pos': 0.209, 'compound': 0.5719},
108: {'neg': 0.0, 'neu': 0.804, 'pos': 0.196, 'compound': 0.9503},
109: {'neg': 0.059, 'neu': 0.676, 'pos': 0.265, 'compound': 0.9116},
110: {'neg': 0.014, 'neu': 0.764, 'pos': 0.222, 'compound': 0.9841},
111: {'neg': 0.059, 'neu': 0.879, 'pos': 0.062, 'compound': 0.0176},
112: {'neg': 0.0, 'neu': 0.81, 'pos': 0.19, 'compound': 0.8769},
113: {'neg': 0.037, 'neu': 0.786, 'pos': 0.177, 'compound': 0.9946},
114: {'neg': 0.0, 'neu': 0.631, 'pos': 0.369, 'compound': 0.8779},
115: {'neg': 0.027, 'neu': 0.727, 'pos': 0.245, 'compound': 0.9379},
116: {'neg': 0.0, 'neu': 0.645, 'pos': 0.355, 'compound': 0.872},
117: {'neg': 0.0, 'neu': 0.892, 'pos': 0.108, 'compound': 0.6573},
118: {'neg': 0.0, 'neu': 0.781, 'pos': 0.219, 'compound': 0.9751},
119: {'neg': 0.05, 'neu': 0.872, 'pos': 0.079, 'compound': 0.8972},
120: {'neg': 0.013, 'neu': 0.785, 'pos': 0.203, 'compound': 0.9828},
121: {'neg': 0.026, 'neu': 0.759, 'pos': 0.215, 'compound': 0.9509},
122: {'neg': 0.102, 'neu': 0.822, 'pos': 0.076, 'compound': -0.3626},
123: {'neg': 0.025, 'neu': 0.803, 'pos': 0.172, 'compound': 0.9022},
124: {'neg': 0.017, 'neu': 0.795, 'pos': 0.188, 'compound': 0.9769},
125: {'neg': 0.079, 'neu': 0.67, 'pos': 0.252, 'compound': 0.9678},
126: {'neg': 0.035, 'neu': 0.87, 'pos': 0.095, 'compound': 0.5709},
127: {'neg': 0.0, 'neu': 0.721, 'pos': 0.279, 'compound': 0.9258},
128: {'neg': 0.067, 'neu': 0.633, 'pos': 0.299, 'compound': 0.9022},
129: {'neg': 0.043, 'neu': 0.728, 'pos': 0.229, 'compound': 0.8142},
130: {'neg': 0.114, 'neu': 0.676, 'pos': 0.21, 'compound': 0.6721},
131: {'neg': 0.0, 'neu': 0.755, 'pos': 0.245, 'compound': 0.8658},
132: {'neg': 0.135, 'neu': 0.76, 'pos': 0.105, 'compound': -0.3612},
133: {'neg': 0.046, 'neu': 0.772, 'pos': 0.181, 'compound': 0.7902},
134: {'neg': 0.02, 'neu': 0.878, 'pos': 0.103, 'compound': 0.8082},
135: {'neg': 0.0, 'neu': 0.877, 'pos': 0.123, 'compound': 0.4215},
136: {'neg': 0.0, 'neu': 0.9, 'pos': 0.1, 'compound': 0.6503},
137: {'neg': 0.0, 'neu': 0.695, 'pos': 0.305, 'compound': 0.9661},
138: {'neg': 0.0, 'neu': 0.689, 'pos': 0.311, 'compound': 0.8591},
139: {'neg': 0.15, 'neu': 0.773, 'pos': 0.077, 'compound': -0.4199},
140: {'neg': 0.043, 'neu': 0.833, 'pos': 0.125, 'compound': 0.835},
141: {'neg': 0.098, 'neu': 0.787, 'pos': 0.114, 'compound': 0.2023},
142: {'neg': 0.0, 'neu': 0.782, 'pos': 0.218, 'compound': 0.7814},
143: {'neg': 0.0, 'neu': 0.763, 'pos': 0.237, 'compound': 0.9296},

144: {'neg': 0.059, 'neu': 0.667, 'pos': 0.274, 'compound': 0.9653},
145: {'neg': 0.058, 'neu': 0.841, 'pos': 0.102, 'compound': 0.6124},
146: {'neg': 0.144, 'neu': 0.677, 'pos': 0.178, 'compound': 0.6341},
147: {'neg': 0.087, 'neu': 0.783, 'pos': 0.13, 'compound': 0.7567},
148: {'neg': 0.058, 'neu': 0.867, 'pos': 0.075, 'compound': 0.1533},
149: {'neg': 0.04, 'neu': 0.833, 'pos': 0.127, 'compound': 0.6956},
150: {'neg': 0.0, 'neu': 0.709, 'pos': 0.291, 'compound': 0.9231},
151: {'neg': 0.0, 'neu': 0.564, 'pos': 0.436, 'compound': 0.9858},
152: {'neg': 0.0, 'neu': 0.784, 'pos': 0.216, 'compound': 0.765},
153: {'neg': 0.0, 'neu': 0.775, 'pos': 0.225, 'compound': 0.7269},
154: {'neg': 0.12, 'neu': 0.76, 'pos': 0.12, 'compound': 0.2502},
155: {'neg': 0.0, 'neu': 0.647, 'pos': 0.353, 'compound': 0.9803},
156: {'neg': 0.0, 'neu': 0.768, 'pos': 0.232, 'compound': 0.9681},
157: {'neg': 0.191, 'neu': 0.809, 'pos': 0.0, 'compound': -0.7269},
158: {'neg': 0.071, 'neu': 0.514, 'pos': 0.415, 'compound': 0.8934},
159: {'neg': 0.065, 'neu': 0.893, 'pos': 0.042, 'compound': -0.4721},
160: {'neg': 0.081, 'neu': 0.779, 'pos': 0.14, 'compound': 0.4194},
161: {'neg': 0.0, 'neu': 0.644, 'pos': 0.356, 'compound': 0.9117},
162: {'neg': 0.106, 'neu': 0.894, 'pos': 0.0, 'compound': -0.5504},
163: {'neg': 0.072, 'neu': 0.652, 'pos': 0.276, 'compound': 0.9517},
164: {'neg': 0.047, 'neu': 0.869, 'pos': 0.085, 'compound': 0.4199},
165: {'neg': 0.025, 'neu': 0.752, 'pos': 0.223, 'compound': 0.8957},
166: {'neg': 0.032, 'neu': 0.717, 'pos': 0.251, 'compound': 0.9597},
167: {'neg': 0.0, 'neu': 0.657, 'pos': 0.343, 'compound': 0.9098},
168: {'neg': 0.05, 'neu': 0.905, 'pos': 0.045, 'compound': -0.1154},
169: {'neg': 0.186, 'neu': 0.74, 'pos': 0.074, 'compound': -0.5283},
170: {'neg': 0.141, 'neu': 0.832, 'pos': 0.028, 'compound': -0.7721},
171: {'neg': 0.0, 'neu': 0.854, 'pos': 0.146, 'compound': 0.6476},
172: {'neg': 0.04, 'neu': 0.844, 'pos': 0.116, 'compound': 0.6808},
173: {'neg': 0.0, 'neu': 0.763, 'pos': 0.237, 'compound': 0.8906},
174: {'neg': 0.022, 'neu': 0.788, 'pos': 0.189, 'compound': 0.9901},
175: {'neg': 0.04, 'neu': 0.722, 'pos': 0.237, 'compound': 0.9782},
176: {'neg': 0.0, 'neu': 0.874, 'pos': 0.126, 'compound': 0.7579},
177: {'neg': 0.0, 'neu': 0.938, 'pos': 0.062, 'compound': 0.4215},
178: {'neg': 0.058, 'neu': 0.794, 'pos': 0.148, 'compound': 0.6249},
179: {'neg': 0.2, 'neu': 0.63, 'pos': 0.171, 'compound': 0.1203},
180: {'neg': 0.048, 'neu': 0.829, 'pos': 0.122, 'compound': 0.7458},
181: {'neg': 0.076, 'neu': 0.767, 'pos': 0.156, 'compound': 0.6085},
182: {'neg': 0.0, 'neu': 0.433, 'pos': 0.567, 'compound': 0.9667},
183: {'neg': 0.088, 'neu': 0.743, 'pos': 0.169, 'compound': 0.943},
184: {'neg': 0.0, 'neu': 0.857, 'pos': 0.143, 'compound': 0.9577},
185: {'neg': 0.11, 'neu': 0.593, 'pos': 0.297, 'compound': 0.6597},
186: {'neg': 0.189, 'neu': 0.811, 'pos': 0.0, 'compound': -0.5994},
187: {'neg': 0.016, 'neu': 0.842, 'pos': 0.142, 'compound': 0.9944},
188: {'neg': 0.0, 'neu': 0.824, 'pos': 0.176, 'compound': 0.6983},
189: {'neg': 0.0, 'neu': 0.843, 'pos': 0.157, 'compound': 0.8868},
190: {'neg': 0.0, 'neu': 0.934, 'pos': 0.066, 'compound': 0.3506},

191: {'neg': 0.148, 'neu': 0.64, 'pos': 0.212, 'compound': 0.4926},
192: {'neg': 0.0, 'neu': 0.75, 'pos': 0.25, 'compound': 0.9062},
193: {'neg': 0.055, 'neu': 0.728, 'pos': 0.217, 'compound': 0.8756},
194: {'neg': 0.031, 'neu': 0.735, 'pos': 0.234, 'compound': 0.9595},
195: {'neg': 0.082, 'neu': 0.483, 'pos': 0.435, 'compound': 0.8299},
196: {'neg': 0.0, 'neu': 0.761, 'pos': 0.239, 'compound': 0.9538},
197: {'neg': 0.0, 'neu': 0.917, 'pos': 0.083, 'compound': 0.4738},
198: {'neg': 0.0, 'neu': 0.904, 'pos': 0.096, 'compound': 0.4153},
199: {'neg': 0.0, 'neu': 0.701, 'pos': 0.299, 'compound': 0.8268},
200: {'neg': 0.0, 'neu': 0.811, 'pos': 0.189, 'compound': 0.7178},
201: {'neg': 0.039, 'neu': 0.888, 'pos': 0.072, 'compound': 0.6381},
202: {'neg': 0.064, 'neu': 0.597, 'pos': 0.339, 'compound': 0.9531},
203: {'neg': 0.0, 'neu': 0.688, 'pos': 0.312, 'compound': 0.8225},
204: {'neg': 0.061, 'neu': 0.814, 'pos': 0.125, 'compound': 0.8728},
205: {'neg': 0.0, 'neu': 0.882, 'pos': 0.118, 'compound': 0.6249},
206: {'neg': 0.0, 'neu': 0.754, 'pos': 0.246, 'compound': 0.9368},
207: {'neg': 0.0, 'neu': 0.59, 'pos': 0.41, 'compound': 0.8779},
208: {'neg': 0.051, 'neu': 0.8, 'pos': 0.15, 'compound': 0.8436},
209: {'neg': 0.05, 'neu': 0.82, 'pos': 0.13, 'compound': 0.8913},
210: {'neg': 0.045, 'neu': 0.761, 'pos': 0.194, 'compound': 0.9893},
211: {'neg': 0.075, 'neu': 0.755, 'pos': 0.171, 'compound': 0.9218},
212: {'neg': 0.051, 'neu': 0.821, 'pos': 0.129, 'compound': 0.9529},
213: {'neg': 0.051, 'neu': 0.838, 'pos': 0.11, 'compound': 0.4404},
214: {'neg': 0.095, 'neu': 0.883, 'pos': 0.022, 'compound': -0.9726},
215: {'neg': 0.0, 'neu': 0.891, 'pos': 0.109, 'compound': 0.6476},
216: {'neg': 0.0, 'neu': 0.798, 'pos': 0.202, 'compound': 0.7964},
217: {'neg': 0.078, 'neu': 0.922, 'pos': 0.0, 'compound': -0.296},
218: {'neg': 0.015, 'neu': 0.884, 'pos': 0.101, 'compound': 0.9736},
219: {'neg': 0.059, 'neu': 0.774, 'pos': 0.167, 'compound': 0.9424},
220: {'neg': 0.031, 'neu': 0.702, 'pos': 0.267, 'compound': 0.9812},
221: {'neg': 0.027, 'neu': 0.909, 'pos': 0.064, 'compound': 0.25},
222: {'neg': 0.068, 'neu': 0.666, 'pos': 0.266, 'compound': 0.9883},
223: {'neg': 0.0, 'neu': 0.779, 'pos': 0.221, 'compound': 0.9623},
224: {'neg': 0.0, 'neu': 0.607, 'pos': 0.393, 'compound': 0.923},
225: {'neg': 0.152, 'neu': 0.739, 'pos': 0.109, 'compound': -0.25},
226: {'neg': 0.064, 'neu': 0.794, 'pos': 0.141, 'compound': 0.7951},
227: {'neg': 0.139, 'neu': 0.754, 'pos': 0.108, 'compound': -0.3774},
228: {'neg': 0.106, 'neu': 0.718, 'pos': 0.176, 'compound': 0.5475},
229: {'neg': 0.0, 'neu': 0.837, 'pos': 0.163, 'compound': 0.6486},
230: {'neg': 0.025, 'neu': 0.854, 'pos': 0.121, 'compound': 0.6478},
231: {'neg': 0.03, 'neu': 0.726, 'pos': 0.244, 'compound': 0.9281},
232: {'neg': 0.0, 'neu': 0.904, 'pos': 0.096, 'compound': 0.8144},
233: {'neg': 0.0, 'neu': 0.807, 'pos': 0.193, 'compound': 0.8126},
234: {'neg': 0.103, 'neu': 0.729, 'pos': 0.169, 'compound': 0.2481},
235: {'neg': 0.0, 'neu': 0.805, 'pos': 0.195, 'compound': 0.8655},
236: {'neg': 0.11, 'neu': 0.792, 'pos': 0.098, 'compound': -0.4786},
237: {'neg': 0.041, 'neu': 0.793, 'pos': 0.166, 'compound': 0.9387},

238: {'neg': 0.029, 'neu': 0.798, 'pos': 0.174, 'compound': 0.9936},
239: {'neg': 0.064, 'neu': 0.7, 'pos': 0.236, 'compound': 0.9677},
240: {'neg': 0.0, 'neu': 0.72, 'pos': 0.28, 'compound': 0.765},
241: {'neg': 0.066, 'neu': 0.71, 'pos': 0.223, 'compound': 0.9553},
242: {'neg': 0.0, 'neu': 0.765, 'pos': 0.235, 'compound': 0.807},
243: {'neg': 0.0, 'neu': 0.76, 'pos': 0.24, 'compound': 0.9344},
244: {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0},
245: {'neg': 0.081, 'neu': 0.63, 'pos': 0.289, 'compound': 0.765},
246: {'neg': 0.072, 'neu': 0.825, 'pos': 0.103, 'compound': 0.682},
247: {'neg': 0.075, 'neu': 0.633, 'pos': 0.292, 'compound': 0.9757},
248: {'neg': 0.0, 'neu': 0.869, 'pos': 0.131, 'compound': 0.7717},
249: {'neg': 0.0, 'neu': 0.602, 'pos': 0.398, 'compound': 0.9351},
250: {'neg': 0.0, 'neu': 0.75, 'pos': 0.25, 'compound': 0.7184},
251: {'neg': 0.047, 'neu': 0.781, 'pos': 0.172, 'compound': 0.9476},
252: {'neg': 0.076, 'neu': 0.924, 'pos': 0.0, 'compound': -0.4823},
253: {'neg': 0.107, 'neu': 0.893, 'pos': 0.0, 'compound': -0.4767},
254: {'neg': 0.0, 'neu': 0.801, 'pos': 0.199, 'compound': 0.9698},
255: {'neg': 0.091, 'neu': 0.736, 'pos': 0.172, 'compound': 0.4118},
256: {'neg': 0.103, 'neu': 0.699, 'pos': 0.198, 'compound': 0.9805},
257: {'neg': 0.034, 'neu': 0.664, 'pos': 0.302, 'compound': 0.9463},
258: {'neg': 0.105, 'neu': 0.816, 'pos': 0.079, 'compound': -0.3489},
259: {'neg': 0.04, 'neu': 0.841, 'pos': 0.119, 'compound': 0.8883},
260: {'neg': 0.0, 'neu': 0.833, 'pos': 0.167, 'compound': 0.8824},
261: {'neg': 0.0, 'neu': 0.613, 'pos': 0.387, 'compound': 0.9493},
262: {'neg': 0.0, 'neu': 0.54, 'pos': 0.46, 'compound': 0.9153},
263: {'neg': 0.106, 'neu': 0.706, 'pos': 0.188, 'compound': 0.5849},
264: {'neg': 0.098, 'neu': 0.875, 'pos': 0.026, 'compound': -0.9218},
265: {'neg': 0.051, 'neu': 0.802, 'pos': 0.147, 'compound': 0.872},
266: {'neg': 0.0, 'neu': 0.619, 'pos': 0.381, 'compound': 0.902},
267: {'neg': 0.0, 'neu': 0.862, 'pos': 0.138, 'compound': 0.4926},
268: {'neg': 0.062, 'neu': 0.911, 'pos': 0.028, 'compound': -0.7067},
269: {'neg': 0.0, 'neu': 0.767, 'pos': 0.233, 'compound': 0.8176},
270: {'neg': 0.032, 'neu': 0.794, 'pos': 0.174, 'compound': 0.9354},
271: {'neg': 0.0, 'neu': 0.839, 'pos': 0.161, 'compound': 0.5927},
272: {'neg': 0.062, 'neu': 0.863, 'pos': 0.074, 'compound': 0.2609},
273: {'neg': 0.052, 'neu': 0.817, 'pos': 0.132, 'compound': 0.7003},
274: {'neg': 0.0, 'neu': 0.733, 'pos': 0.267, 'compound': 0.7346},
275: {'neg': 0.037, 'neu': 0.693, 'pos': 0.271, 'compound': 0.9421},
276: {'neg': 0.132, 'neu': 0.711, 'pos': 0.157, 'compound': 0.3303},
277: {'neg': 0.0, 'neu': 0.523, 'pos': 0.477, 'compound': 0.9542},
278: {'neg': 0.025, 'neu': 0.809, 'pos': 0.167, 'compound': 0.937},
279: {'neg': 0.072, 'neu': 0.641, 'pos': 0.288, 'compound': 0.8565},
280: {'neg': 0.066, 'neu': 0.859, 'pos': 0.075, 'compound': 0.1666},
281: {'neg': 0.049, 'neu': 0.823, 'pos': 0.127, 'compound': 0.6438},
282: {'neg': 0.0, 'neu': 0.754, 'pos': 0.246, 'compound': 0.8016},
283: {'neg': 0.028, 'neu': 0.934, 'pos': 0.038, 'compound': 0.1779},
284: {'neg': 0.032, 'neu': 0.792, 'pos': 0.176, 'compound': 0.9852},

285: {'neg': 0.0, 'neu': 0.864, 'pos': 0.136, 'compound': 0.5255},
286: {'neg': 0.0, 'neu': 0.898, 'pos': 0.102, 'compound': 0.7917},
287: {'neg': 0.0, 'neu': 0.857, 'pos': 0.143, 'compound': 0.919},
288: {'neg': 0.035, 'neu': 0.801, 'pos': 0.163, 'compound': 0.9676},
289: {'neg': 0.054, 'neu': 0.745, 'pos': 0.2, 'compound': 0.9557},
290: {'neg': 0.039, 'neu': 0.697, 'pos': 0.264, 'compound': 0.8439},
291: {'neg': 0.104, 'neu': 0.705, 'pos': 0.191, 'compound': 0.6257},
292: {'neg': 0.052, 'neu': 0.745, 'pos': 0.203, 'compound': 0.9434},
293: {'neg': 0.09, 'neu': 0.705, 'pos': 0.205, 'compound': 0.8636},
294: {'neg': 0.034, 'neu': 0.757, 'pos': 0.209, 'compound': 0.9823},
295: {'neg': 0.0, 'neu': 0.887, 'pos': 0.113, 'compound': 0.4939},
296: {'neg': 0.12, 'neu': 0.781, 'pos': 0.099, 'compound': -0.7095},
297: {'neg': 0.025, 'neu': 0.737, 'pos': 0.239, 'compound': 0.9566},
298: {'neg': 0.0, 'neu': 0.811, 'pos': 0.189, 'compound': 0.8781},
299: {'neg': 0.0, 'neu': 0.681, 'pos': 0.319, 'compound': 0.8934},
300: {'neg': 0.078, 'neu': 0.735, 'pos': 0.187, 'compound': 0.9637},
301: {'neg': 0.0, 'neu': 0.632, 'pos': 0.368, 'compound': 0.9661},
302: {'neg': 0.148, 'neu': 0.625, 'pos': 0.227, 'compound': 0.5849},
303: {'neg': 0.014, 'neu': 0.705, 'pos': 0.281, 'compound': 0.9763},
304: {'neg': 0.076, 'neu': 0.791, 'pos': 0.133, 'compound': 0.25},
305: {'neg': 0.058, 'neu': 0.778, 'pos': 0.165, 'compound': 0.5734},
306: {'neg': 0.15, 'neu': 0.773, 'pos': 0.077, 'compound': -0.9037},
307: {'neg': 0.097, 'neu': 0.781, 'pos': 0.122, 'compound': 0.4733},
308: {'neg': 0.0, 'neu': 0.649, 'pos': 0.351, 'compound': 0.894},
309: {'neg': 0.0, 'neu': 0.796, 'pos': 0.204, 'compound': 0.9695},
310: {'neg': 0.0, 'neu': 0.774, 'pos': 0.226, 'compound': 0.9287},
311: {'neg': 0.031, 'neu': 0.657, 'pos': 0.312, 'compound': 0.9644},
312: {'neg': 0.087, 'neu': 0.913, 'pos': 0.0, 'compound': -0.4939},
313: {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0},
314: {'neg': 0.018, 'neu': 0.914, 'pos': 0.069, 'compound': 0.4971},
315: {'neg': 0.024, 'neu': 0.828, 'pos': 0.148, 'compound': 0.6897},
316: {'neg': 0.06, 'neu': 0.772, 'pos': 0.168, 'compound': 0.9109},
317: {'neg': 0.0, 'neu': 0.823, 'pos': 0.177, 'compound': 0.5783},
318: {'neg': 0.07, 'neu': 0.839, 'pos': 0.091, 'compound': 0.6785},
319: {'neg': 0.0, 'neu': 0.904, 'pos': 0.096, 'compound': 0.3716},
320: {'neg': 0.0, 'neu': 0.758, 'pos': 0.242, 'compound': 0.7717},
321: {'neg': 0.065, 'neu': 0.562, 'pos': 0.373, 'compound': 0.886},
322: {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0},
323: {'neg': 0.05, 'neu': 0.69, 'pos': 0.26, 'compound': 0.7712},
324: {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0},
325: {'neg': 0.213, 'neu': 0.514, 'pos': 0.274, 'compound': 0.3185},
326: {'neg': 0.0, 'neu': 0.688, 'pos': 0.312, 'compound': 0.8979},
327: {'neg': 0.075, 'neu': 0.726, 'pos': 0.199, 'compound': 0.9373},
328: {'neg': 0.064, 'neu': 0.594, 'pos': 0.342, 'compound': 0.9581},
329: {'neg': 0.163, 'neu': 0.708, 'pos': 0.129, 'compound': -0.8462},
330: {'neg': 0.029, 'neu': 0.856, 'pos': 0.115, 'compound': 0.5709},
331: {'neg': 0.0, 'neu': 0.837, 'pos': 0.163, 'compound': 0.6249},

332: {'neg': 0.115, 'neu': 0.885, 'pos': 0.0, 'compound': -0.4588},
333: {'neg': 0.0, 'neu': 0.689, 'pos': 0.311, 'compound': 0.9732},
334: {'neg': 0.0, 'neu': 0.662, 'pos': 0.338, 'compound': 0.9719},
335: {'neg': 0.0, 'neu': 0.886, 'pos': 0.114, 'compound': 0.6124},
336: {'neg': 0.046, 'neu': 0.8, 'pos': 0.154, 'compound': 0.6796},
337: {'neg': 0.078, 'neu': 0.651, 'pos': 0.271, 'compound': 0.8506},
338: {'neg': 0.0, 'neu': 0.765, 'pos': 0.235, 'compound': 0.9008},
339: {'neg': 0.0, 'neu': 0.734, 'pos': 0.266, 'compound': 0.784},
340: {'neg': 0.078, 'neu': 0.823, 'pos': 0.098, 'compound': 0.4416},
341: {'neg': 0.069, 'neu': 0.782, 'pos': 0.149, 'compound': 0.8499},
342: {'neg': 0.041, 'neu': 0.657, 'pos': 0.302, 'compound': 0.8731},
343: {'neg': 0.0, 'neu': 0.912, 'pos': 0.088, 'compound': 0.4939},
344: {'neg': 0.11, 'neu': 0.678, 'pos': 0.211, 'compound': 0.8053},
345: {'neg': 0.101, 'neu': 0.627, 'pos': 0.273, 'compound': 0.9758},
346: {'neg': 0.044, 'neu': 0.725, 'pos': 0.231, 'compound': 0.8319},
347: {'neg': 0.0, 'neu': 0.608, 'pos': 0.392, 'compound': 0.9694},
348: {'neg': 0.093, 'neu': 0.752, 'pos': 0.155, 'compound': 0.7667},
349: {'neg': 0.0, 'neu': 0.678, 'pos': 0.322, 'compound': 0.908},
350: {'neg': 0.071, 'neu': 0.861, 'pos': 0.068, 'compound': -0.0258},
351: {'neg': 0.0, 'neu': 0.715, 'pos': 0.285, 'compound': 0.9177},
352: {'neg': 0.064, 'neu': 0.727, 'pos': 0.209, 'compound': 0.7337},
353: {'neg': 0.0, 'neu': 0.893, 'pos': 0.107, 'compound': 0.802},
354: {'neg': 0.0, 'neu': 0.888, 'pos': 0.112, 'compound': 0.6604},
355: {'neg': 0.0, 'neu': 0.802, 'pos': 0.198, 'compound': 0.6892},
356: {'neg': 0.05, 'neu': 0.734, 'pos': 0.215, 'compound': 0.8008},
357: {'neg': 0.027, 'neu': 0.835, 'pos': 0.138, 'compound': 0.8805},
358: {'neg': 0.0, 'neu': 0.895, 'pos': 0.105, 'compound': 0.631},
359: {'neg': 0.164, 'neu': 0.694, 'pos': 0.142, 'compound': 0.283},
360: {'neg': 0.0, 'neu': 0.705, 'pos': 0.295, 'compound': 0.954},
361: {'neg': 0.033, 'neu': 0.785, 'pos': 0.182, 'compound': 0.9441},
362: {'neg': 0.228, 'neu': 0.772, 'pos': 0.0, 'compound': -0.734},
363: {'neg': 0.0, 'neu': 0.891, 'pos': 0.109, 'compound': 0.8802},
364: {'neg': 0.0, 'neu': 0.742, 'pos': 0.258, 'compound': 0.8088},
365: {'neg': 0.033, 'neu': 0.621, 'pos': 0.346, 'compound': 0.9334},
366: {'neg': 0.076, 'neu': 0.768, 'pos': 0.156, 'compound': 0.4434},
367: {'neg': 0.0, 'neu': 0.685, 'pos': 0.315, 'compound': 0.9366},
368: {'neg': 0.038, 'neu': 0.84, 'pos': 0.122, 'compound': 0.8016},
369: {'neg': 0.064, 'neu': 0.871, 'pos': 0.066, 'compound': 0.0258},
370: {'neg': 0.0, 'neu': 0.913, 'pos': 0.087, 'compound': 0.7703},
371: {'neg': 0.012, 'neu': 0.86, 'pos': 0.128, 'compound': 0.9923},
372: {'neg': 0.087, 'neu': 0.643, 'pos': 0.27, 'compound': 0.6912},
373: {'neg': 0.11, 'neu': 0.748, 'pos': 0.142, 'compound': 0.1264},
374: {'neg': 0.0, 'neu': 0.588, 'pos': 0.412, 'compound': 0.9168},
375: {'neg': 0.0, 'neu': 0.728, 'pos': 0.272, 'compound': 0.9472},
376: {'neg': 0.054, 'neu': 0.69, 'pos': 0.256, 'compound': 0.8962},
377: {'neg': 0.0, 'neu': 0.796, 'pos': 0.204, 'compound': 0.874},
378: {'neg': 0.046, 'neu': 0.793, 'pos': 0.161, 'compound': 0.9341},

379: {'neg': 0.063, 'neu': 0.524, 'pos': 0.413, 'compound': 0.9709},
380: {'neg': 0.036, 'neu': 0.695, 'pos': 0.269, 'compound': 0.9468},
381: {'neg': 0.074, 'neu': 0.715, 'pos': 0.212, 'compound': 0.8349},
382: {'neg': 0.318, 'neu': 0.515, 'pos': 0.167, 'compound': -0.7184},
383: {'neg': 0.0, 'neu': 0.905, 'pos': 0.095, 'compound': 0.6369},
384: {'neg': 0.027, 'neu': 0.78, 'pos': 0.193, 'compound': 0.9913},
385: {'neg': 0.0, 'neu': 0.767, 'pos': 0.233, 'compound': 0.8065},
386: {'neg': 0.0, 'neu': 0.774, 'pos': 0.226, 'compound': 0.9796},
387: {'neg': 0.0, 'neu': 0.839, 'pos': 0.161, 'compound': 0.8625},
388: {'neg': 0.089, 'neu': 0.75, 'pos': 0.161, 'compound': 0.8201},
389: {'neg': 0.088, 'neu': 0.537, 'pos': 0.375, 'compound': 0.755},
390: {'neg': 0.031, 'neu': 0.764, 'pos': 0.205, 'compound': 0.9183},
391: {'neg': 0.248, 'neu': 0.636, 'pos': 0.116, 'compound': -0.8174},
392: {'neg': 0.0, 'neu': 0.642, 'pos': 0.358, 'compound': 0.8591},
393: {'neg': 0.0, 'neu': 0.661, 'pos': 0.339, 'compound': 0.8481},
394: {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0},
395: {'neg': 0.0, 'neu': 0.83, 'pos': 0.17, 'compound': 0.8016},
396: {'neg': 0.0, 'neu': 0.502, 'pos': 0.498, 'compound': 0.9677},
397: {'neg': 0.0, 'neu': 0.638, 'pos': 0.362, 'compound': 0.9682},
398: {'neg': 0.046, 'neu': 0.703, 'pos': 0.251, 'compound': 0.867},
399: {'neg': 0.0, 'neu': 0.8, 'pos': 0.2, 'compound': 0.9885},
400: {'neg': 0.0, 'neu': 0.787, 'pos': 0.213, 'compound': 0.7644},
401: {'neg': 0.234, 'neu': 0.556, 'pos': 0.211, 'compound': 0.0},
402: {'neg': 0.093, 'neu': 0.813, 'pos': 0.095, 'compound': 0.0258},
403: {'neg': 0.215, 'neu': 0.697, 'pos': 0.088, 'compound': -0.6351},
404: {'neg': 0.194, 'neu': 0.771, 'pos': 0.035, 'compound': -0.9058},
405: {'neg': 0.0, 'neu': 0.691, 'pos': 0.309, 'compound': 0.8172},
406: {'neg': 0.019, 'neu': 0.702, 'pos': 0.279, 'compound': 0.9622},
407: {'neg': 0.0, 'neu': 0.954, 'pos': 0.046, 'compound': 0.6249},
408: {'neg': 0.036, 'neu': 0.772, 'pos': 0.192, 'compound': 0.9477},
409: {'neg': 0.0, 'neu': 0.713, 'pos': 0.287, 'compound': 0.9257},
410: {'neg': 0.05, 'neu': 0.758, 'pos': 0.192, 'compound': 0.8316},
411: {'neg': 0.016, 'neu': 0.879, 'pos': 0.105, 'compound': 0.8681},
412: {'neg': 0.0, 'neu': 0.802, 'pos': 0.198, 'compound': 0.8555},
413: {'neg': 0.0, 'neu': 0.815, 'pos': 0.185, 'compound': 0.7777},
414: {'neg': 0.0, 'neu': 0.914, 'pos': 0.086, 'compound': 0.4118},
415: {'neg': 0.0, 'neu': 0.722, 'pos': 0.278, 'compound': 0.8902},
416: {'neg': 0.0, 'neu': 0.594, 'pos': 0.406, 'compound': 0.9612},
417: {'neg': 0.07, 'neu': 0.799, 'pos': 0.131, 'compound': 0.9222},
418: {'neg': 0.166, 'neu': 0.809, 'pos': 0.025, 'compound': -0.8957},
419: {'neg': 0.0, 'neu': 0.784, 'pos': 0.216, 'compound': 0.8876},
420: {'neg': 0.148, 'neu': 0.815, 'pos': 0.037, 'compound': -0.5983},
421: {'neg': 0.035, 'neu': 0.754, 'pos': 0.211, 'compound': 0.9561},
422: {'neg': 0.0, 'neu': 0.861, 'pos': 0.139, 'compound': 0.4404},
423: {'neg': 0.223, 'neu': 0.68, 'pos': 0.096, 'compound': -0.3314},
424: {'neg': 0.055, 'neu': 0.687, 'pos': 0.258, 'compound': 0.9106},
425: {'neg': 0.017, 'neu': 0.821, 'pos': 0.161, 'compound': 0.9576},

```
426: {'neg': 0.0, 'neu': 0.806, 'pos': 0.194, 'compound': 0.7717},
427: {'neg': 0.029, 'neu': 0.817, 'pos': 0.154, 'compound': 0.7845},
428: {'neg': 0.0, 'neu': 0.761, 'pos': 0.239, 'compound': 0.9337},
429: {'neg': 0.0, 'neu': 0.739, 'pos': 0.261, 'compound': 0.9741},
430: {'neg': 0.0, 'neu': 0.617, 'pos': 0.383, 'compound': 0.9876},
431: {'neg': 0.04, 'neu': 0.786, 'pos': 0.174, 'compound': 0.9847},
432: {'neg': 0.0, 'neu': 0.73, 'pos': 0.27, 'compound': 0.9516},
433: {'neg': 0.083, 'neu': 0.751, 'pos': 0.166, 'compound': 0.8044},
434: {'neg': 0.108, 'neu': 0.593, 'pos': 0.299, 'compound': 0.8655},
435: {'neg': 0.0, 'neu': 0.771, 'pos': 0.229, 'compound': 0.9179},
436: {'neg': 0.0, 'neu': 0.829, 'pos': 0.171, 'compound': 0.8519},
437: {'neg': 0.0, 'neu': 0.926, 'pos': 0.074, 'compound': 0.7383},
438: {'neg': 0.0, 'neu': 0.887, 'pos': 0.113, 'compound': 0.6369},
439: {'neg': 0.0, 'neu': 0.728, 'pos': 0.272, 'compound': 0.87},
440: {'neg': 0.072, 'neu': 0.781, 'pos': 0.147, 'compound': 0.9307},
441: {'neg': 0.078, 'neu': 0.793, 'pos': 0.129, 'compound': 0.5176},
442: {'neg': 0.054, 'neu': 0.69, 'pos': 0.257, 'compound': 0.9683},
443: {'neg': 0.0, 'neu': 0.616, 'pos': 0.384, 'compound': 0.9603},
444: {'neg': 0.044, 'neu': 0.898, 'pos': 0.058, 'compound': 0.1882},
445: {'neg': 0.055, 'neu': 0.873, 'pos': 0.072, 'compound': 0.0935},
446: {'neg': 0.077, 'neu': 0.78, 'pos': 0.143, 'compound': 0.3699},
447: {'neg': 0.042, 'neu': 0.763, 'pos': 0.195, 'compound': 0.9883},
448: {'neg': 0.0, 'neu': 0.713, 'pos': 0.287, 'compound': 0.967},
449: {'neg': 0.0, 'neu': 0.737, 'pos': 0.263, 'compound': 0.8531},
450: {'neg': 0.0, 'neu': 0.845, 'pos': 0.155, 'compound': 0.6908},
451: {'neg': 0.034, 'neu': 0.743, 'pos': 0.223, 'compound': 0.9873},
452: {'neg': 0.054, 'neu': 0.782, 'pos': 0.164, 'compound': 0.9337},
453: {'neg': 0.0, 'neu': 0.5, 'pos': 0.5, 'compound': 0.943},
454: {'neg': 0.0, 'neu': 0.603, 'pos': 0.397, 'compound': 0.8811},
455: {'neg': 0.0, 'neu': 0.699, 'pos': 0.301, 'compound': 0.9619},
456: {'neg': 0.082, 'neu': 0.854, 'pos': 0.064, 'compound': -0.4854},
457: {'neg': 0.0, 'neu': 0.684, 'pos': 0.316, 'compound': 0.926},
458: {'neg': 0.0, 'neu': 0.564, 'pos': 0.436, 'compound': 0.9642},
459: {'neg': 0.045, 'neu': 0.717, 'pos': 0.239, 'compound': 0.8455},
460: {'neg': 0.066, 'neu': 0.743, 'pos': 0.19, 'compound': 0.9481},
461: {'neg': 0.08, 'neu': 0.821, 'pos': 0.099, 'compound': 0.4883},
462: {'neg': 0.037, 'neu': 0.87, 'pos': 0.093, 'compound': 0.34},
463: {'neg': 0.099, 'neu': 0.794, 'pos': 0.108, 'compound': 0.5983},
464: {'neg': 0.019, 'neu': 0.868, 'pos': 0.113, 'compound': 0.8443},
465: {'neg': 0.0, 'neu': 0.838, 'pos': 0.162, 'compound': 0.7823},
466: {'neg': 0.0, 'neu': 0.772, 'pos': 0.228, 'compound': 0.9606},
467: {'neg': 0.009, 'neu': 0.845, 'pos': 0.147, 'compound': 0.9874},
468: {'neg': 0.008, 'neu': 0.818, 'pos': 0.174, 'compound': 0.9926},
469: {'neg': 0.049, 'neu': 0.951, 'pos': 0.0, 'compound': -0.3595},
470: {'neg': 0.0, 'neu': 0.957, 'pos': 0.043, 'compound': 0.25},
471: {'neg': 0.051, 'neu': 0.676, 'pos': 0.273, 'compound': 0.9749},
472: {'neg': 0.0, 'neu': 0.565, 'pos': 0.435, 'compound': 0.9649},
```

```
473: {'neg': 0.0, 'neu': 0.686, 'pos': 0.314, 'compound': 0.7506},
474: {'neg': 0.013, 'neu': 0.75, 'pos': 0.237, 'compound': 0.9828},
475: {'neg': 0.0, 'neu': 0.585, 'pos': 0.415, 'compound': 0.9095},
476: {'neg': 0.066, 'neu': 0.614, 'pos': 0.32, 'compound': 0.9684},
477: {'neg': 0.034, 'neu': 0.728, 'pos': 0.238, 'compound': 0.8555},
478: {'neg': 0.0, 'neu': 0.823, 'pos': 0.177, 'compound': 0.6239},
479: {'neg': 0.245, 'neu': 0.652, 'pos': 0.103, 'compound': -0.3855},
480: {'neg': 0.0, 'neu': 0.435, 'pos': 0.565, 'compound': 0.9935},
481: {'neg': 0.022, 'neu': 0.728, 'pos': 0.249, 'compound': 0.9451},
482: {'neg': 0.0, 'neu': 0.605, 'pos': 0.395, 'compound': 0.9079},
483: {'neg': 0.0, 'neu': 0.862, 'pos': 0.138, 'compound': 0.3384},
484: {'neg': 0.088, 'neu': 0.767, 'pos': 0.145, 'compound': 0.4516},
485: {'neg': 0.0, 'neu': 0.761, 'pos': 0.239, 'compound': 0.8547},
486: {'neg': 0.0, 'neu': 0.818, 'pos': 0.182, 'compound': 0.9224},
487: {'neg': 0.0, 'neu': 0.909, 'pos': 0.091, 'compound': 0.296},
488: {'neg': 0.179, 'neu': 0.707, 'pos': 0.114, 'compound': -0.3723},
489: {'neg': 0.0, 'neu': 0.861, 'pos': 0.139, 'compound': 0.9598},
490: {'neg': 0.0, 'neu': 0.763, 'pos': 0.237, 'compound': 0.9788},
491: {'neg': 0.055, 'neu': 0.704, 'pos': 0.241, 'compound': 0.9287},
492: {'neg': 0.0, 'neu': 0.717, 'pos': 0.283, 'compound': 0.9367},
493: {'neg': 0.056, 'neu': 0.855, 'pos': 0.089, 'compound': 0.5976},
494: {'neg': 0.1, 'neu': 0.645, 'pos': 0.254, 'compound': 0.6486},
495: {'neg': 0.0, 'neu': 0.788, 'pos': 0.212, 'compound': 0.9743},
496: {'neg': 0.0, 'neu': 0.554, 'pos': 0.446, 'compound': 0.9725},
497: {'neg': 0.059, 'neu': 0.799, 'pos': 0.142, 'compound': 0.7833},
498: {'neg': 0.025, 'neu': 0.762, 'pos': 0.212, 'compound': 0.9848},
499: {'neg': 0.041, 'neu': 0.904, 'pos': 0.055, 'compound': 0.128},
500: {'neg': 0.0, 'neu': 0.678, 'pos': 0.322, 'compound': 0.9811}}
```

[31]:

```python
pd.DataFrame(res)
```

[31]:

	1	2	3	4	5	6	7	8	9	\
neg	0.0000	0.1380	0.0910	0.0	0.0000	0.029	0.0340	0.0000	0.0000	
neu	0.6950	0.8620	0.7540	1.0	0.5520	0.809	0.6930	0.5200	0.8510	
pos	0.3050	0.0000	0.1550	0.0	0.4480	0.163	0.2730	0.4800	0.1490	
compound	0.9441	-0.5664	0.8265	0.0	0.9468	0.883	0.9346	0.9487	0.6369	

	10	…	491	492	493	494	495	496	497	\
neg	0.0000	…	0.0550	0.0000	0.0560	0.1000	0.0000	0.0000	0.0590	
neu	0.7050	…	0.7040	0.7170	0.8550	0.6450	0.7880	0.5540	0.7990	
pos	0.2950	…	0.2410	0.2830	0.0890	0.2540	0.2120	0.4460	0.1420	
compound	0.8313	…	0.9287	0.9367	0.5976	0.6486	0.9743	0.9725	0.7833	

	498	499	500
neg	0.0250	0.041	0.0000
neu	0.7620	0.904	0.6780
pos	0.2120	0.055	0.3220

```
compound  0.9848  0.128  0.9811

[4 rows x 500 columns]
```

[32]:
```python
pd.DataFrame(res).T #la transposé to flip the dataframe
```

[32]:
```
       neg    neu    pos  compound
1    0.000  0.695  0.305    0.9441
2    0.138  0.862  0.000   -0.5664
3    0.091  0.754  0.155    0.8265
4    0.000  1.000  0.000    0.0000
5    0.000  0.552  0.448    0.9468
..     …      …      …       …
496  0.000  0.554  0.446    0.9725
497  0.059  0.799  0.142    0.7833
498  0.025  0.762  0.212    0.9848
499  0.041  0.904  0.055    0.1280
500  0.000  0.678  0.322    0.9811

[500 rows x 4 columns]
```

[33]:
```python
vaders = pd.DataFrame(res).T
#reset_index() moves the current index into a new column and assigns another␣
↪new index column
vaders = vaders.reset_index().rename(columns={'index':'Id'})
vaders = vaders.merge(df, how='left')
```

[34]:
```python
vaders
```

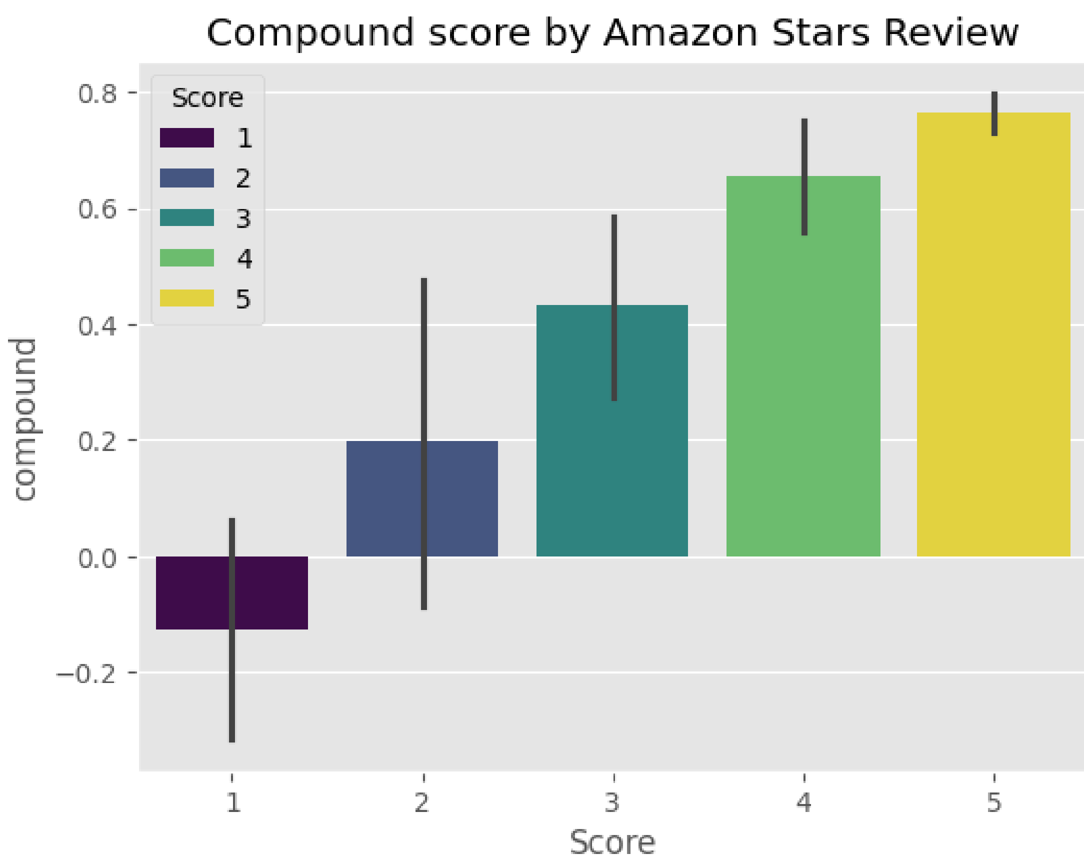
[34]:
```
      Id    neg    neu    pos  compound   ProductId          UserId  \
0      1  0.000  0.695  0.305    0.9441  B001E4KFG0  A3SGXH7AUHU8GW
1      2  0.138  0.862  0.000   -0.5664  B00813GRG4  A1D87F6ZCVE5NK
2      3  0.091  0.754  0.155    0.8265  B000LQOCH0   ABXLMWJIXXAIN
3      4  0.000  1.000  0.000    0.0000  B000UA0QIQ  A395BORC6FGVXV
4      5  0.000  0.552  0.448    0.9468  B006K2ZZ7K  A1UQRSCLF8GW1T
..   …     …      …      …       …           …             …
495  496  0.000  0.554  0.446    0.9725  B000G6RYNE   APGAA43E3WPN7
496  497  0.059  0.799  0.142    0.7833  B000G6RYNE    ABR7HU5H1KNE
497  498  0.025  0.762  0.212    0.9848  B000G6RYNE   AJQD2WWJYOYFQ
498  499  0.041  0.904  0.055    0.1280  B000G6RYNE  A16YH487W9ZYO0
499  500  0.000  0.678  0.322    0.9811  B000G6RYNE   A83YQC1XOU4CS

                         ProfileName  HelpfulnessNumerator  \
0                         delmartian                     1
1                             dll pa                     0
2    Natalia Corres "Natalia Corres"                     1
3                               Karl                     3
```

```
4      Michael D. Bigham "M. Wassir"                      0
..                               …                 …
495                         Darren                      0
496                          Keith                      0
497                        bubbles                      0
498               Bruce G. Lindsay                      0
499                       J. Baker                      0

     HelpfulnessDenominator  Score        Time  \
0                         1      5  1303862400
1                         0      1  1346976000
2                         1      4  1219017600
3                         3      2  1307923200
4                         0      5  1350777600
..                      …    …         …
495                       0      5  1201392000
496                       0      5  1196726400
497                       0      4  1186617600
498                       0      4  1184198400
499                       0      5  1183420800

                               Summary  \
0               Good Quality Dog Food
1                   Not as Advertised
2               "Delight" says it all
3                      Cough Medicine
4                         Great taffy
..                                 …
495                     amazing chips
496                    Best Chip Ever
497  Tangy, spicy, and sweet- oh my!
498         An indulgence with a bite
499                  The best I've had

                                                  Text
0    I have bought several of the Vitality canned d…
1    Product arrived labeled as Jumbo Salted Peanut…
2    This is a confection that has been around a fe…
3    If you are looking for the secret ingredient i…
4    Great taffy at a great price.  There was a wid…
..                                                 …
495  i rarely eat chips but i saw these and tried t…
496  This is easily the best potato chip that I hav…
497  Kettle Chips Spicy Thai potato chips have the …
498  Okay, I should not eat potato chips, nor shoul…
499  I don't write very many reviews but I have to …
```

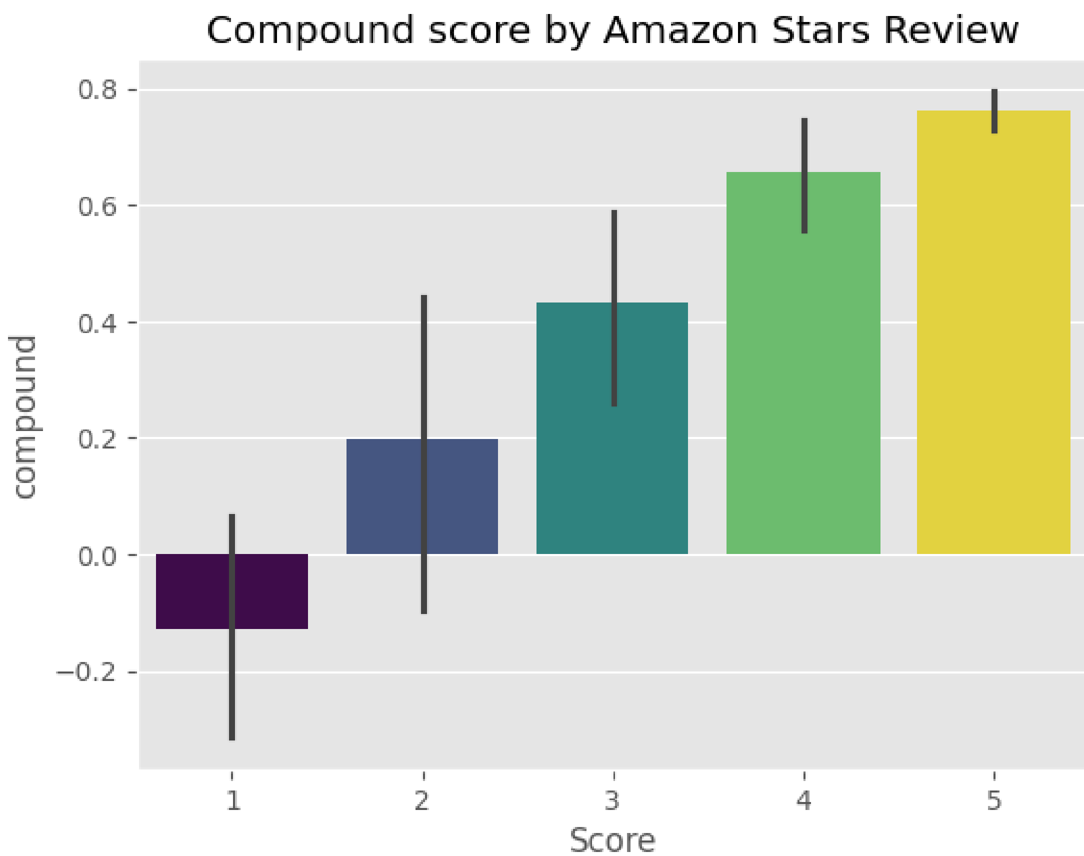
```
[500 rows x 14 columns]
```

[35]:
```
## plot VADERS result
```

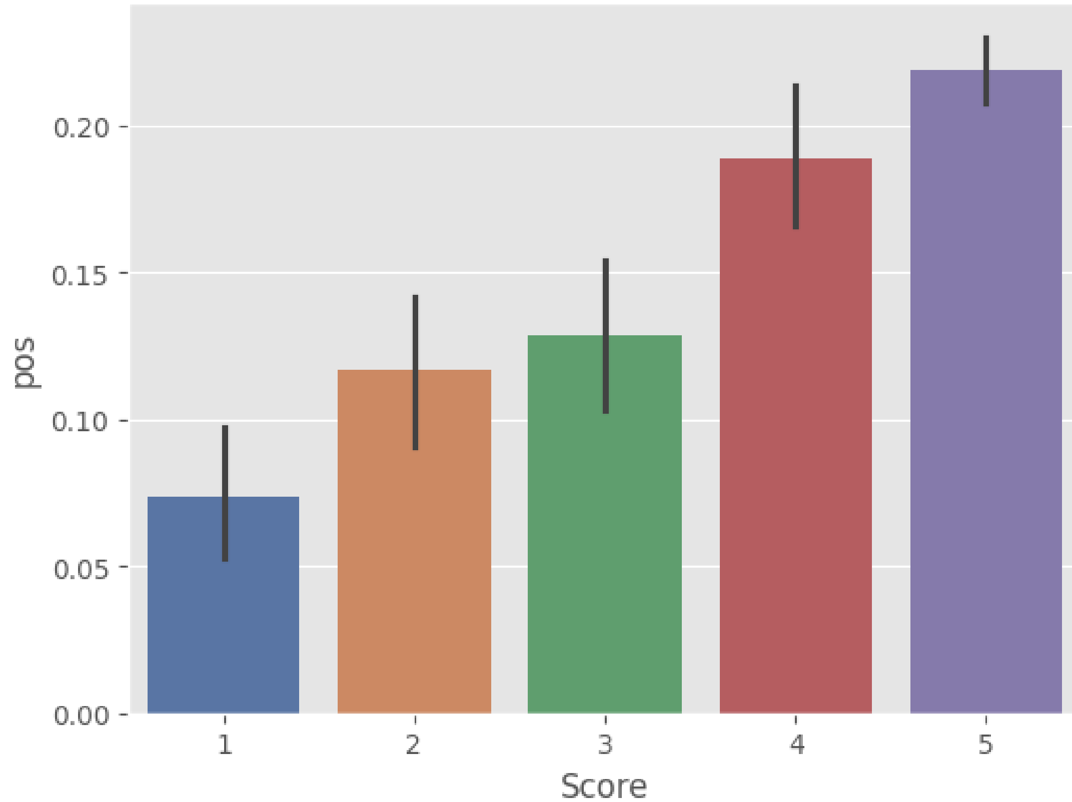
[36]:
```
#hue='Score': This tells Seaborn to color the bars based on the Score values.
ax = sns.barplot(data=vaders, x='Score', y='compound',hue='Score',␣
 ↪palette='viridis')
ax.set_title('Compound score by Amazon Stars Review')
plt.show()
```



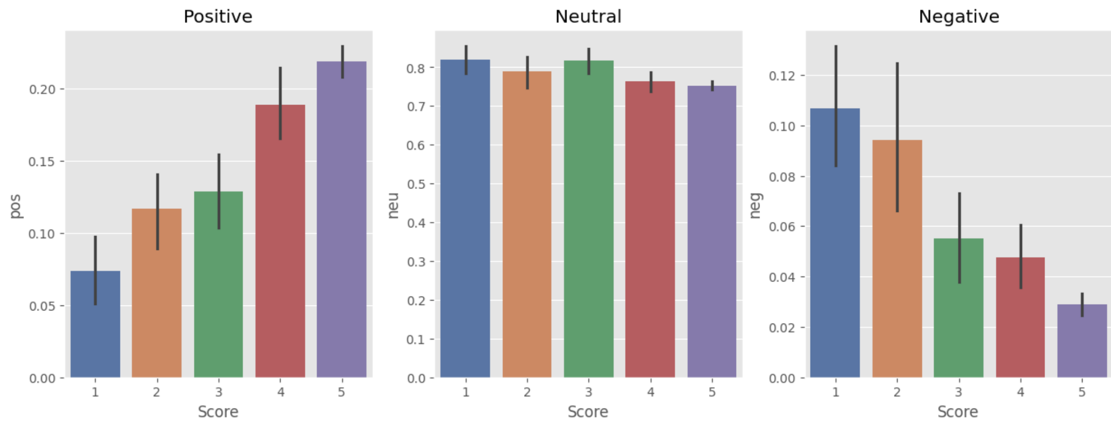
[37]:
```
ax = sns.barplot(data=vaders, x='Score', y='compound',hue='Score',␣
 ↪palette='viridis', legend=False)
ax.set_title('Compound score by Amazon Stars Review')
plt.show()
```



[38]:
```
ax = sns.barplot(data=vaders, x='Score', y='pos',hue='Score',palette='deep',␣
    ↪legend=False)
```



[39]:
```
fig, axs = plt.subplots(1,3, figsize=(15,5))
sns.barplot(data=vaders, x='Score', y='pos', ax = axs[0],␣
↪hue='Score',palette='deep', legend=False)
sns.barplot(data=vaders, x='Score', y='neu', ax = axs[1],␣
↪hue='Score',palette='deep', legend=False)
sns.barplot(data=vaders, x='Score', y='neg', ax = axs[2],␣
↪hue='Score',palette='deep', legend=False)
axs[0].set_title('Positive')
axs[1].set_title('Neutral')
axs[2].set_title('Negative')
plt.show()
```



[40]:
```
from transformers import AutoTokenizer #Converts raw text into tokens that a␣
↪model can understand.
from transformers import AutoModelForSequenceClassification #Uses a pre-trained␣
↪model to classify sequences of text into categories.
from scipy.special import softmax #Turns raw model scores into probabilities␣
↪that sum up to 1.
```

[41]:
```
MODEL = f"cardiffnlp/twitter-roberta-base-sentiment" #pre-trained sentiment␣
↪analysis model based on RoBERTa, trained on Twitter data.
tokenizer = AutoTokenizer.from_pretrained(MODEL, force_download=True) # Loads␣
↪the tokenizer associated with the specified model
model = AutoModelForSequenceClassification.from_pretrained(MODEL,␣
↪force_download=True) #Loads the pre-trained sequence classification model␣
↪for sentiment analysis
```

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

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

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

special_tokens_map.json:   0%|          | 0.00/150 [00:00<?, ?B/s]

c:\users\bendh\appdata\local\programs\python\python39\lib\site-
packages\transformers\tokenization_utils_base.py:1601: FutureWarning:
`clean_up_tokenization_spaces` was not set. It will be set to `True` by default.
This behavior will be depracted in transformers v4.45, and will be then set to
`False` by default. For more details check this issue:
https://github.com/huggingface/transformers/issues/31884
  warnings.warn(

config.json:   0%|          | 0.00/747 [00:00<?, ?B/s]

config.json:   0%|          | 0.00/747 [00:00<?, ?B/s]

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

[57]:
```
#VADER results on example
print(example)
sia.polarity_scores(example)
```

```
This oatmeal is not good. Its mushy, soft, I don't like it. Quaker Oats is the
way to go.
```

[57]:
```
{'neg': 0.22, 'neu': 0.78, 'pos': 0.0, 'compound': -0.5448}
```

[43]:
```
#run for Roberta model
encoded_text = tokenizer(example, return_tensors='pt') #return the tokenized␣
↪text as PyTorch tensors (PyTorch format)
encoded_text
```

[43]:
```
{'input_ids': tensor([[    0,   713,  1021, 38615,    16,    45,   205,     4,
3139, 39589,
           219,     6,  3793,     6,    38,   218,    75,   101,    24,     4,
          3232,  4218,   384,  2923,    16,     5,   169,     7,   213,     4,
             2]]), 'attention_mask': tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1]])}
```

[44]:
```
 #The ** syntax is a way in Python to unpack a dictionary into keyword␣
↪arguments.
output = model(**encoded_text) #the keys in the dictionary encoded_text are␣
↪passed as named arguments to the model.
output
```

[44]:
```
SequenceClassifierOutput(loss=None, logits=tensor([[ 3.1436, -0.7107, -2.6559]],
grad_fn=<AddmmBackward0>), hidden_states=None, attentions=None)
```

[45]:
```
scores = output[0][0].detach().numpy() #extracts and converts the logits from␣
↪the model's output to a NumPy array for further processing.
scores = softmax(scores) #convert the logits into a probability distribution
scores
```

[45]:
```
array([0.97635514, 0.02068745, 0.00295737], dtype=float32)
```

[46]:
```
scores_dict = {
    'roberta_neg' : scores[0],
    'roberta_neu' : scores[1],
    'roberta_pos' : scores[2]
}
scores_dict
```

[46]:
```
{'roberta_neg': 0.97635514,
 'roberta_neu': 0.020687453,
 'roberta_pos': 0.0029573692}
```

[58]:
```
def polarity_scores_roberta(example):
    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]
    }
    return scores_dict
```

[60]:
```
res = {}
for i, row in tqdm (df.iterrows(), total =len(df)):
    try:
        text = row['Text']
        myid = row['Id']
        vader_result = sia.polarity_scores(text)
        vader_result_rename = {}
        for key, value in vader_result.items():
            vader_result_rename[f"vader_{key}"]= value
        roberta_result = polarity_scores_roberta(text)
        both = {**vader_result_rename, **roberta_result}
        res[myid] = both
    except RuntimeError:
        print(f'Broke for ID {myid}')
```

```
  0%|          | 0/500 [00:00<?, ?it/s]

Broke for ID 83
Broke for ID 187
```

[61]:
```
vader_result_rename
```

[61]:
```
{'vader_neg': 0.0,
 'vader_neu': 0.678,
 'vader_pos': 0.322,
 'vader_compound': 0.9811}
```

[62]:
```
roberta_result
```

[62]:
```
{'roberta_neg': 0.0024397583,
 'roberta_neu': 0.01132722,
 'roberta_pos': 0.986233}
```

[63]:
```
both
```

[63]: {'vader_neg': 0.0,
 'vader_neu': 0.678,
 'vader_pos': 0.322,
 'vader_compound': 0.9811,
 'roberta_neg': 0.0024397583,
 'roberta_neu': 0.01132722,
 'roberta_pos': 0.986233}

[64]:
```
results_df = pd.DataFrame(res).T
#reset_index() moves the current index into a new column and assigns another␣
↪new index column
results_df = results_df.reset_index().rename(columns={'index':'Id'})
results_df = results_df.merge(df, how='left')
```

[65]:
```
results_df.head()
```

[65]:
```
   Id  vader_neg  vader_neu  vader_pos  vader_compound  roberta_neg  \
0   1      0.000      0.695      0.305          0.9441     0.009624
1   2      0.138      0.862      0.000         -0.5664     0.508986
2   3      0.091      0.754      0.155          0.8265     0.003229
3   4      0.000      1.000      0.000          0.0000     0.002295
4   5      0.000      0.552      0.448          0.9468     0.001635

   roberta_neu  roberta_pos   ProductId          UserId  \
0     0.049980     0.940395  B001E4KFG0  A3SGXH7AUHU8GW
1     0.452413     0.038600  B00813GRG4  A1D87F6ZCVE5NK
2     0.098067     0.898704  B000LQOCH0   ABXLMWJIXXAIN
3     0.090219     0.907486  B000UA0QIQ  A395BORC6FGVXV
4     0.010302     0.988063  B006K2ZZ7K  A1UQRSCLF8GW1T

                       ProfileName  HelpfulnessNumerator  \
0                       delmartian                     1
1                           dll pa                     0
2  Natalia Corres "Natalia Corres"                     1
3                             Karl                     3
4    Michael D. Bigham "M. Wassir"                     0

   HelpfulnessDenominator  Score        Time                Summary  \
0                       1      5  1303862400  Good Quality Dog Food
1                       0      1  1346976000      Not as Advertised
2                       1      4  1219017600  "Delight" says it all
3                       3      2  1307923200         Cough Medicine
4                       0      5  1350777600            Great taffy

                                                Text
0  I have bought several of the Vitality canned d…
1  Product arrived labeled as Jumbo Salted Peanut…
```

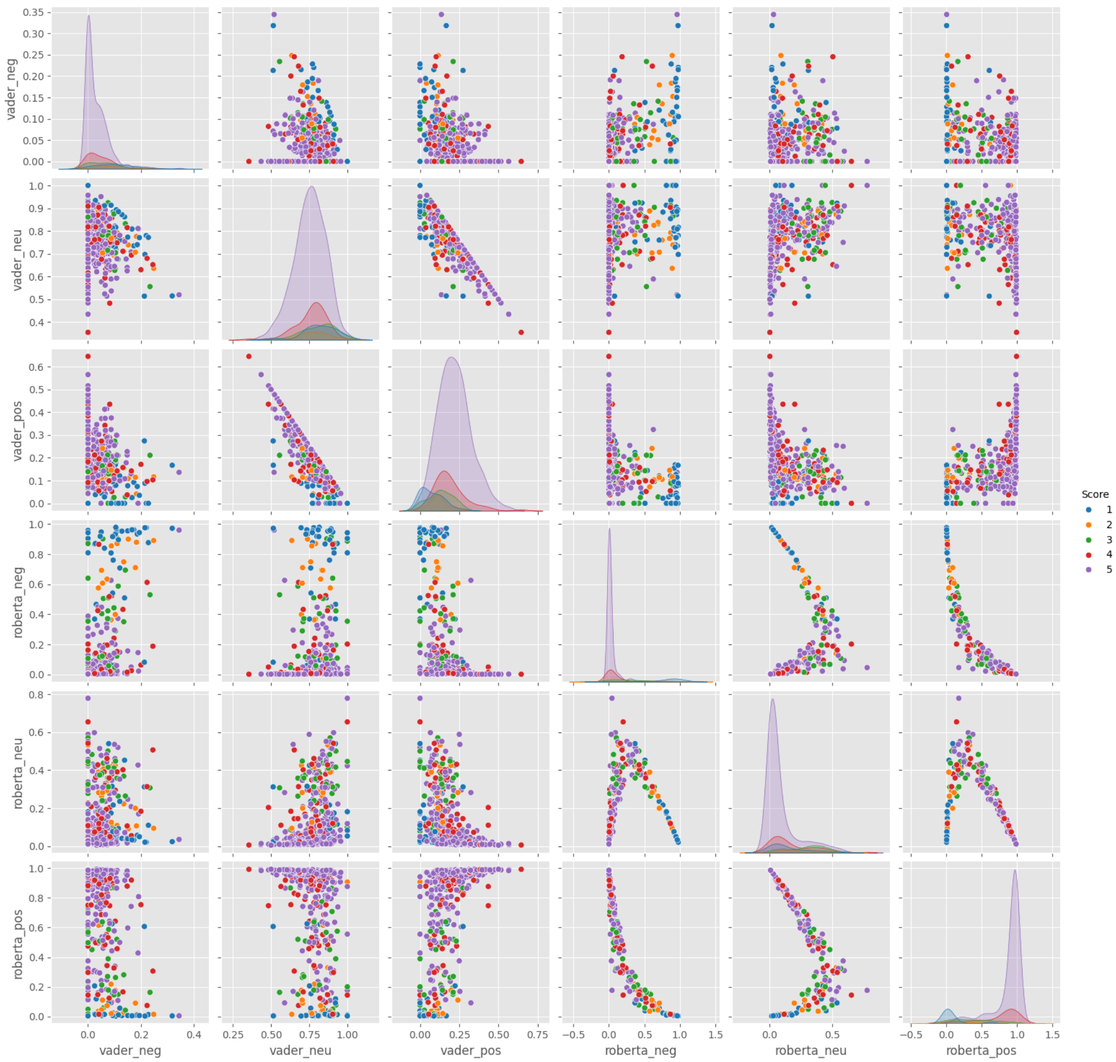
```
2  This is a confection that has been around a fe…
3  If you are looking for the secret ingredient i…
4  Great taffy at a great price.  There was a wid…
```

[66]:
```
## Compare Scores between Models
```

[67]:
```
results_df.columns
```

[67]:
```
Index(['Id', 'vader_neg', 'vader_neu', 'vader_pos', 'vader_compound',
       'roberta_neg', 'roberta_neu', 'roberta_pos', 'ProductId', 'UserId',
       'ProfileName', 'HelpfulnessNumerator', 'HelpfulnessDenominator',
       'Score', 'Time', 'Summary', 'Text'],
      dtype='object')
```

[68]:
```
sns.pairplot(data=results_df,
             vars=['vader_neg', 'vader_neu', 'vader_pos',
                   'roberta_neg', 'roberta_neu', 'roberta_pos'],
            hue='Score',
            palette='tab10')
plt.show()
```



[69]:
```
#Review examples (pos 1 star and neg 5 stars)
```

[80]:
```
#Positive sentiment : 1star Review
```

[77]:
```
results_df.query('Score==1') \
          .sort_values('roberta_pos', ascending = False)['Text']
```

[77]:
```
252    I felt energized within five minutes, but it l…
206    To me, these are nothing like the regular Alto…
322    So we cancelled the order.  It was cancelled w…
163    Seriously this product was as tasteless as the…
73     Buyer Beware Please! This sweetener is not for…
12     My cats have been happily eating Felidae Plati…
214    Hey, the description says 360 grams - that is …
```

```
152    These singles sell for $2.50 - $3.36 at the st…
1      Product arrived labeled as Jumbo Salted Peanut…
211    As with canidae, Felidae has also changed thei…
309    The package came with the label torn off and n…
227    This candy is not as described. The middle is …
265    I used to buy this sugar for years. I do not e…
359    A very bitter tasting coffee even when enhance…
280    I paid $1.79 for a 2 ounce pkg of these at lun…
62     Arrived in 6 days and were so stale i could no…
333    Serveice delivery with the seller was excellen…
329    I received the items in a timely manner. Upon …
310    I wouldn't even think of buying this product u…
167    Besides being smaller than runts, they look th…
231    Terrible! Artificial lemon taste, like Pledge …
399    This mix is very poorly packaged and breaks op…
303    This is the first time I've really been misled…
401    Perhaps the worst bottle of wine I've ever had…
26     The candy is just red , No flavor . Just  plan…
75     No tea flavor at all. Just whole brunch of art…
98     I fed this to my Golden Retriever and he hated…
332    Taste like it is stale.  Will not order this a…
166    I don't know how long these sat on the back of…
415    A vile, miserable pancake. I put these in fron…
379    These condiments are overpriced and terrible. …
391    I haven't used the ham base. It is loaded with…
400    Just awful!  I thought food was supposed to ta…
250    Five minutes in, one tentacle was bitten off, …
50     This oatmeal is not good. Its mushy, soft, I d…
255    I so wish I would have read this review before…
Name: Text, dtype: object
```

[78]:
```python
results_df.query('Score==1') \
          .sort_values('roberta_pos', ascending = False)['Text'].values[0]
```

[78]:
```
'I felt energized within five minutes, but it lasted for about 45 minutes. I
paid $3.99 for this drink. I could have just drunk a cup of coffee and saved my
money.'
```

[79]:
```python
results_df.query('Score==1') \
          .sort_values('vader_pos', ascending = False)['Text'].values[0]
```

[79]:
```
'So we cancelled the order.  It was cancelled without any problem.  That is a
positive note…'
```

[81]:
```python
#Negative sentiment : 5 star Review
```

[82]:
```
results_df.query('Score==5') \
          .sort_values('roberta_neg', ascending = False)['Text'].values[0]
```

[82]: 'this was sooooo deliscious but too bad i ate em too fast and gained 2 pds! my fault'

[83]:
```
results_df.query('Score==5') \
          .sort_values('vader_neg', ascending = False)['Text'].values[0]
```

[83]: 'this was sooooo deliscious but too bad i ate em too fast and gained 2 pds! my fault'

[84]:
```
#Transformers pipeline
```

[86]:
```
from transformers import pipeline
```

[88]:
```
sent_pipeline = pipeline("sentiment-analysis")
```

```
No model was supplied, defaulted to distilbert/distilbert-base-uncased-
finetuned-sst-2-english and revision af0f99b
(https://huggingface.co/distilbert/distilbert-base-uncased-finetuned-
sst-2-english).
Using a pipeline without specifying a model name and revision in production is
not recommended.
```

```
config.json:   0%|          | 0.00/629 [00:00<?, ?B/s]
```

```
model.safetensors:   0%|          | 0.00/268M [00:00<?, ?B/s]
```

```
tokenizer_config.json:   0%|          | 0.00/48.0 [00:00<?, ?B/s]
```

```
vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]
```

```
c:\users\bendh\appdata\local\programs\python\python39\lib\site-
packages\transformers\tokenization_utils_base.py:1601: FutureWarning:
`clean_up_tokenization_spaces` was not set. It will be set to `True` by default.
This behavior will be depracted in transformers v4.45, and will be then set to
`False` by default. For more details check this issue:
https://github.com/huggingface/transformers/issues/31884
  warnings.warn(
```

[92]:
```
sent_pipeline('What a day!')
```

[92]: [{'label': 'POSITIVE', 'score': 0.9973642230033875}]

[ ]:
```

```