M5010_LinguisticNotebook_22183822014

March 18, 2024

[5]:
```
import nltk
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
from nltk import stem
stemmer = stem.PorterStemmer()
from nltk import word_tokenize
nltk.download('stopwords')
from nltk.corpus import stopwords
stops = set(stopwords.words('english'))
nltk.download('punkt')
import string
punct = list(string.punctuation)
from collections import Counter
import requests
import pandas as pd
import seaborn as sns
sns.set()
from matplotlib import *
#!pip install PRAW
import numpy as np
import praw
import datetime
pd.options.mode.copy_on_write = True
```

```
[nltk_data] Downloading package wordnet to /home/svetlana/nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/svetlana/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /home/svetlana/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
```

First we must load the dataset collected in the previous assignment. I was interested in how people respond to news, but also to comments other people have left on these threads, and whether more people decide to upvote or downvote a particular comment - and for what reason. With the use of the VAD model and other methods, we can then extract non-obvious information from this data, and gather insights about the language people use when talking about news, and whether this has

an effect on the comment's 'Score' or upvote/downvote rate. For transparency, this dataset was taken from the top 10 posts of the past year on February 15th, 2024, on the subreddit r/worldnews, which was chosen as it was explicitly less US-focused than r/news or other equivalents.

[6]:
```
corpus = pd.read_pickle('newcorpus.pkl')
corpus
```

[6]:

	text	datetime	score	subreddit	redditor	type	title	words	valence	arousal
0	This invasion has done more to unite the Europ…	2023-03-27 11:38:10	21709	worldnews	greek_stallion	comment	Norway, Sweden, Finland, and Denmark struck a …			
1	Now you brought the Vikings back together, dam…	2023-03-27 12:37:39	20349	worldnews	Shotguns_x_559	comment	Norway, Sweden, Finland, and Denmark struck a …			
2	Unified Nordic Defense Force sounds like somet…	2023-03-27 13:02:47	4445	worldnews	BMCarbaugh	comment	Norway, Sweden, Finland, and Denmark struck a …			
3	The vikings have returned! But to the skies in…	2023-03-27 11:57:11	3710	worldnews	Ok_Imagination_7119	comment	Norway, Sweden, Finland, and Denmark struck a …			
4	This would make it one the largest air-forces …	2023-03-27 11:42:18	2884	worldnews	008Zulu	comment	Norway, Sweden, Finland, and Denmark struck a …			
…	…	…	…	…	…	…	…			
4469	Easily said when it isn't their money they are…	2023-02-24 16:14:58	-22	worldnews	AGitatedAG	comment	Lithuania's prime minister says Ukrainians sho…			
4470	Finally saying the quiet part out loud. Europe…	2023-02-24 16:38:29	-25	worldnews	RickyTicky5309	comment	Lithuania's prime minister says Ukrainians sho…			
4471	And the Ukrainian "make a wish" foundation con…	2023-02-24 15:06:17	-31	worldnews	dickchingy	comment	Lithuania's prime minister says Ukrainians sho…			
4472	get fucked warmonger	2023-02-24 16:30:54	-32	worldnews	tablefourtoo	comment	Lithuania's prime minister says Ukrainians sho…			
4473	Except Ukraine is well known for having a blac…	2023-02-24 15:10:47	-40	worldnews	AphexTwins903	comment	Lithuania's prime minister says Ukrainians sho…			

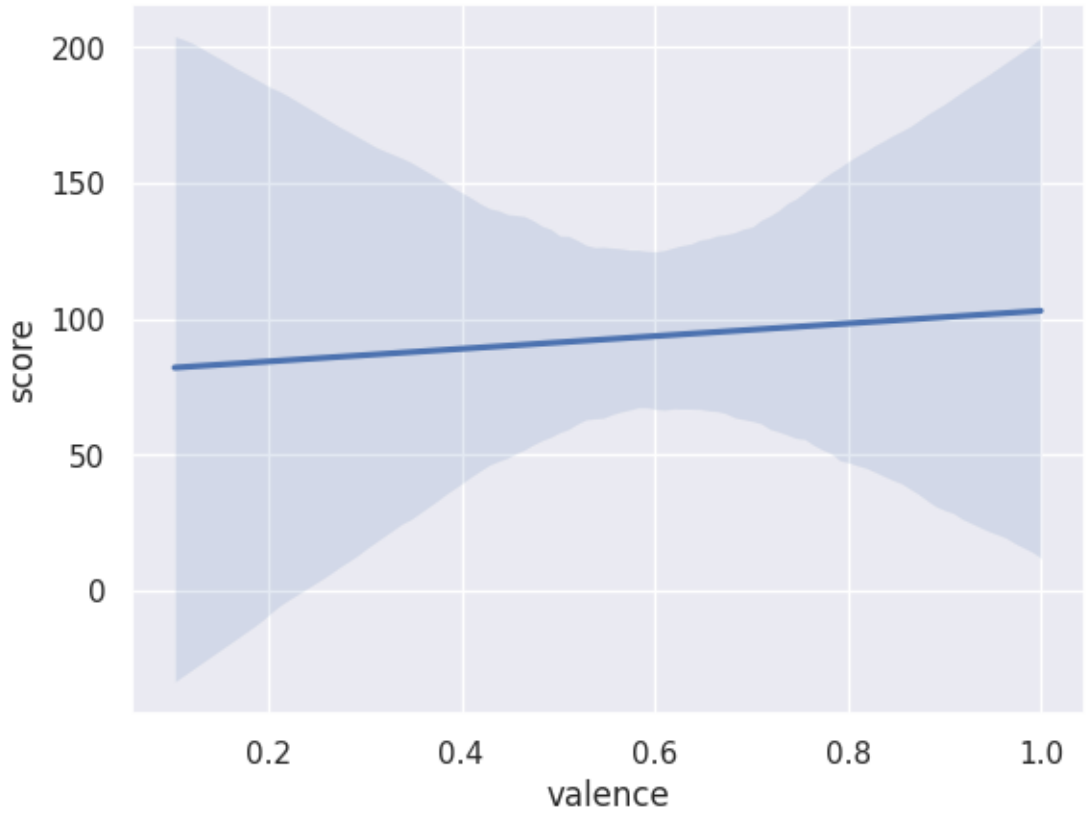
```
0     [invasion, ha, done, unite, european, continen…  0.556075  0.565259
1               [brought, viking, back, together, damn]  0.484521  0.559284
2     [unified, nordic, defense, force, sound, like,…  0.615946  0.598306
3                     [viking, returned, sky, instead]  0.645152  0.411664
4     [would, make, one, largest, air-forces, world,…  0.681533  0.489880
…                                                   …         …         …
4469  [easily, said, n't, money, giving, away, taxpa…  0.603680  0.482258
4470  [finally, saying, quiet, part, loud, europe, t…  0.586773  0.459753
4471  [ukrainian, ", make, wish, ", foundation, cont…  0.700545  0.490566
4472                          [get, fucked, warmonger]  0.626168  0.506861
4473  [except, ukraine, well, known, black, market, …  0.575701  0.444597

      dominance
0      0.521931
1      0.495146
2      0.609628
3      0.532160
4      0.654976
…         …
4469   0.599059
4470   0.539374
4471   0.676847
4472   0.805825
4473   0.620024

[4474 rows x 11 columns]
```

0.1 Method 1: VAD and Linear Regression

We can start with some simple graphs to see if there is any obvious correlation between the 'score' of each comment (i.e. the amount of upvotes it gets) and the Valence (how positive a word is), Arousal (how exciting a word is), and Dominance (how in control a word makes one feel) scores of the words contained within.

[7]:
```
sns.regplot(x = 'valence', y = 'score', scatter = False, data = corpus)
```

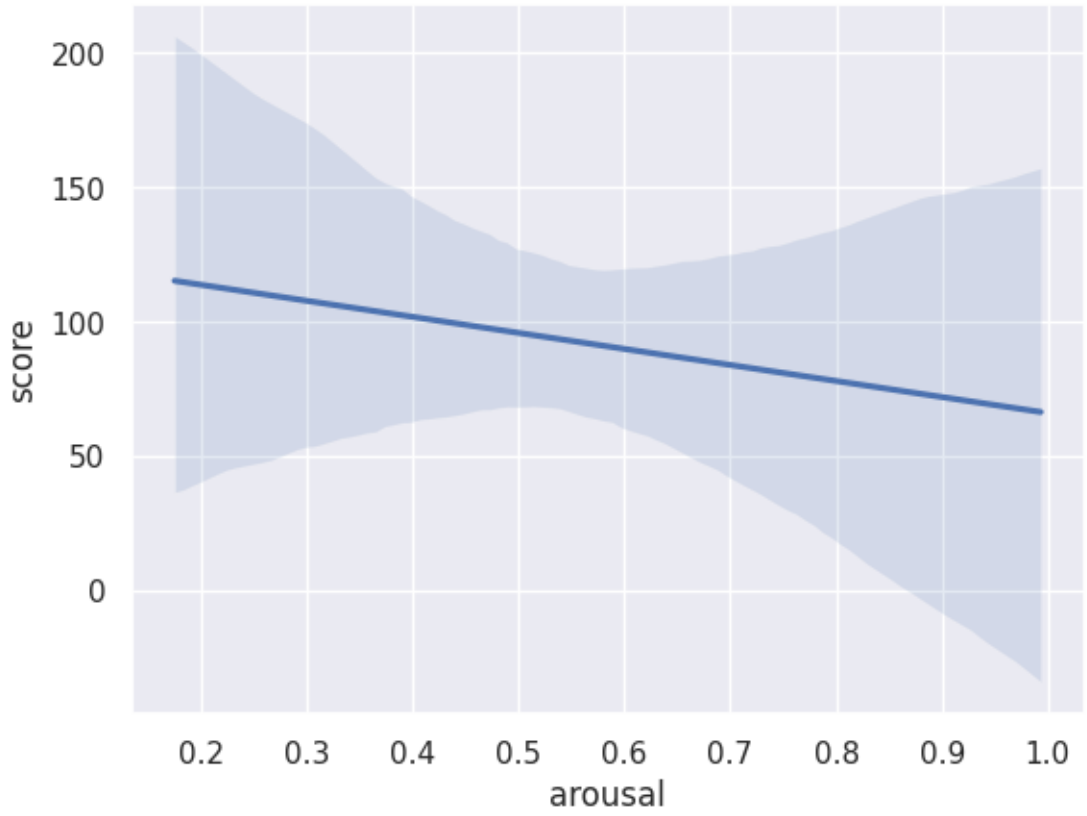
[7]:
```
<AxesSubplot: xlabel='valence', ylabel='score'>
```



This graph shows a small positive correlation between Valence and Score (or upvotes). This suggests that comments that are more positive&uplifting are upvoted more, which may speak as to what readers are more likely to upvote - hopeful and positive takes, rather than depressing and negative ones. However, this correlation is very weak, and this can only serve as a general indication of a directional trend, and a possible avenue for further research with a larger sample size.

[8]:
```
sns.regplot(x = 'arousal', y = 'score', scatter = False, data = corpus)
```

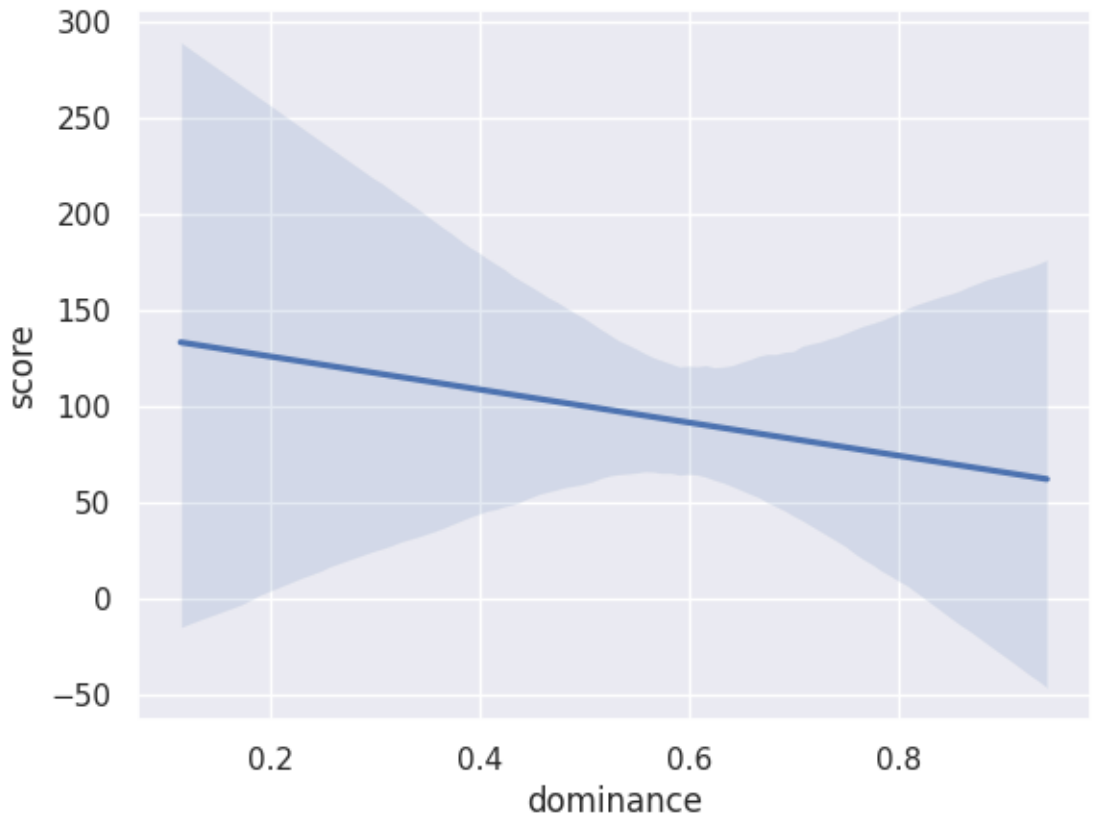
[8]:
```
<AxesSubplot: xlabel='arousal', ylabel='score'>
```



With Arousal we see the opposite, a negative correlation with the Score of a comment, and this time a slightly more significant one than with Valence. This suggests that the more exciting a comment is, the lower the Score it will get. It is important to know that 'exciting' here can cover negative emotions as well, and in the context of internet discussion, this will likely include 'inflammatory' statements that are meant to elicit a reaction, such as the use of swears or insults. This graph, however, shows that such comments tend to be punished/downvoted by users, implying that people generally do not appreciate inflammatory or bold statements, perhaps preffering more calm (or positive) discussion.

[9]:
```
sns.regplot(x = 'dominance', y = 'score', scatter = False, data = corpus)
```

[9]:
```
<AxesSubplot: xlabel='dominance', ylabel='score'>
```



This is probably the most interesting plot of the three, for the simple fact that Dominance usually correlates with Valence. Some words do exist with high Valence and low Dominance, such as 'silly'. This suggests that in regards to news, people may want comments about it to be positive but still aknowledge the lack of control the average person like themselves has about world events. On the other hand, this may be influenced by outlier words such as 'country', which has a Valence score of 0.73 and a Dominance score of 0.49, and is a word that appears frequently in this particular dataset. This highlights one problem of using VAD on its own - the difficulty of translating context between datasets, compared to the likely 'neutral' state the group of people originally scoring words like 'country' on the VAD scales were in.

As such, to see if there really is any correlation between Score and the three VAD scales, we can turn to Regression:

[10]:
```
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import r_regression
import gensim
import gensim.downloader as api
from sklearn.model_selection import train_test_split
```

We'll start off with Linear Regression on the whole dataset:

[11]:
```
linreg = LinearRegression() #Call the model

X = corpus[['valence', 'arousal', 'dominance']]
y = corpus['score']

reg = linreg.fit(X,y) #Fit the model
reg.score(X,y)
```

[11]:
```
0.0003282221521170303
```

This is a very low R-squared score - in DataScience, anything below 0.75 isn't considered strong enough to base theories on. When working with language, the standards are slightly lower due to the amount of noise any dataset will contain, but you generally want about 0.2. However, we can do some things to cut down on the noise, such as by only focusing on comments containing certain words (or tokens). For this, however, we will first need to find out what the most common tokens in this dataset are:

[14]:
```
#This creates a duplicate of our 'words' column in string rather than list␣
 ↪format to we can apply the following functions without them breaking
corpus['words_strings'] = corpus['words'].apply(lambda x: ', '.join(x))
```

[15]:
```
from collections import Counter
counts = Counter(" ".join(corpus['words_strings']).split()).most_common(25)
counts
```

[15]:
```
[('’,', 888),
 ("'s,", 779),
 ('russia,', 509),
 ("n't,", 494),
 ('ukraine,', 494),
 ('like,', 477),
 ('wa,', 404),
 ('twitter,', 372),
 ('would,', 338),
 ('people,', 337),
 ('nato,', 333),
 ('ha,', 331),
 ('war,', 317),
 ('putin,', 291),
 ('``,', 287),
 ('country,', 284),
 ('u,', 267),
 ('elon,', 258),
 ('one,', 236),
 ("'',", 227),
 ('free,', 223),
 ('speech,', 223),
```

```
 ('russian,', 216),
 ('get,', 214),
 ('musk,', 209)]
```

Here we have the top 25 most common tokens in our dataset. We can then re-run the Linear Regression on only those comments that contain a certain token. We'll start with 'russia' and 'russian' (both tokens appear in the top 25, and for NLP purposes are the same word):

[29]:
```
corpus_russia = corpus[corpus['words_strings'].str.contains('russia' or␣
 ↪'russian')]
X_2 = corpus_russia[['valence', 'arousal', 'dominance']]
y_2 = corpus_russia['score']
reg_2 = linreg.fit(X_2,y_2) #Fit the model
reg_2.score(X_2,y_2)
```

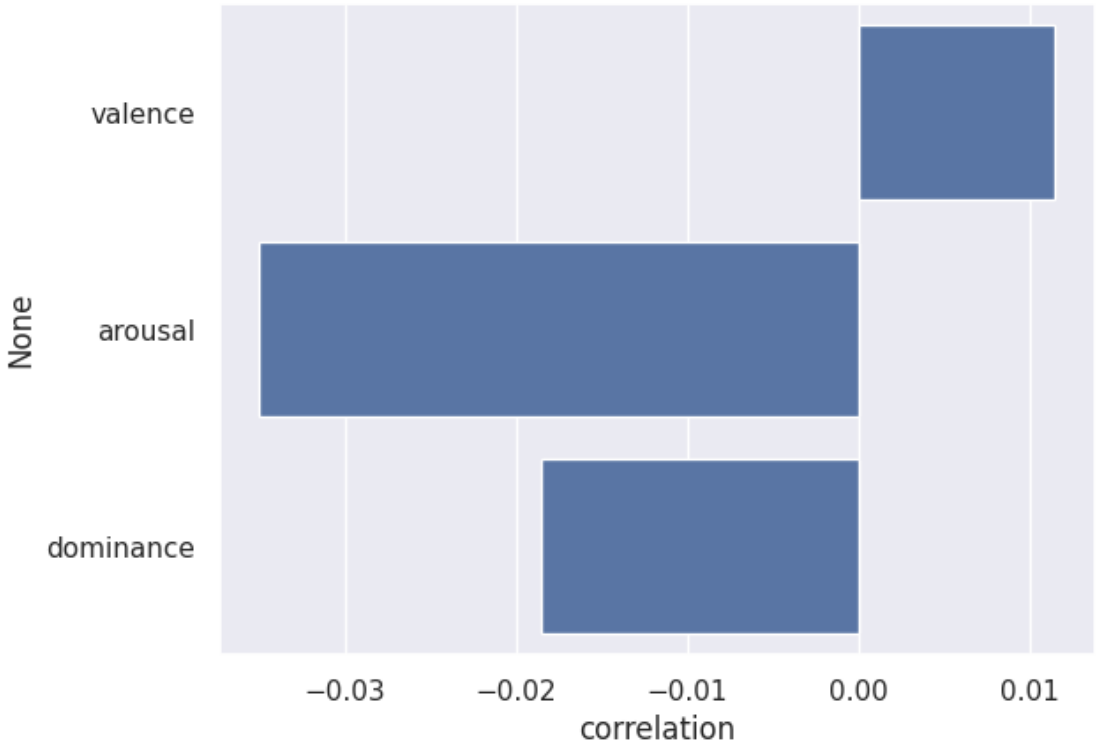
[29]:
```
0.0032942956649603117
```

This is a 10 times improvement, even if it is still below our 0.2 goal. However, at this point we can begin trying Pearson's correlation, with the idea to compare this graph to that of other words:

[26]:
```
correlation = r_regression(X_2, y_2)
corr_df = pd.DataFrame(correlation, index = X.columns, columns =␣
 ↪['correlation'])
```

[27]:
```
sns.barplot(y = corr_df.index, x = 'correlation', data = corr_df)
```

[27]:
```
<AxesSubplot: xlabel='correlation', ylabel='None'>
```



This graph is interesting again as the Dominance and Valence scales are 'pointed' in opposite directions. This means that comments containing the token 'russia' are most likely to gain upvotes if they are, in order of significance, low arousal, low dominance, and high valence. However, this doesn't tell us much yet - let's compare to the token 'ukraine'. Due to current events, we can expect comments containing these tokens to appear in similar contexts in regard to the same news, which also helps narrow down on extraneous variables, and get to what people tend to feel about the countries as they are:

[30]:
```
corpus_ukraine = corpus[corpus['words_strings'].str.contains('ukraine')]
X_2 = corpus_ukraine[['valence', 'arousal', 'dominance']]
y_2 = corpus_ukraine['score']
reg_2 = linreg.fit(X_2,y_2) #Fit the model
reg_2.score(X_2,y_2)
```

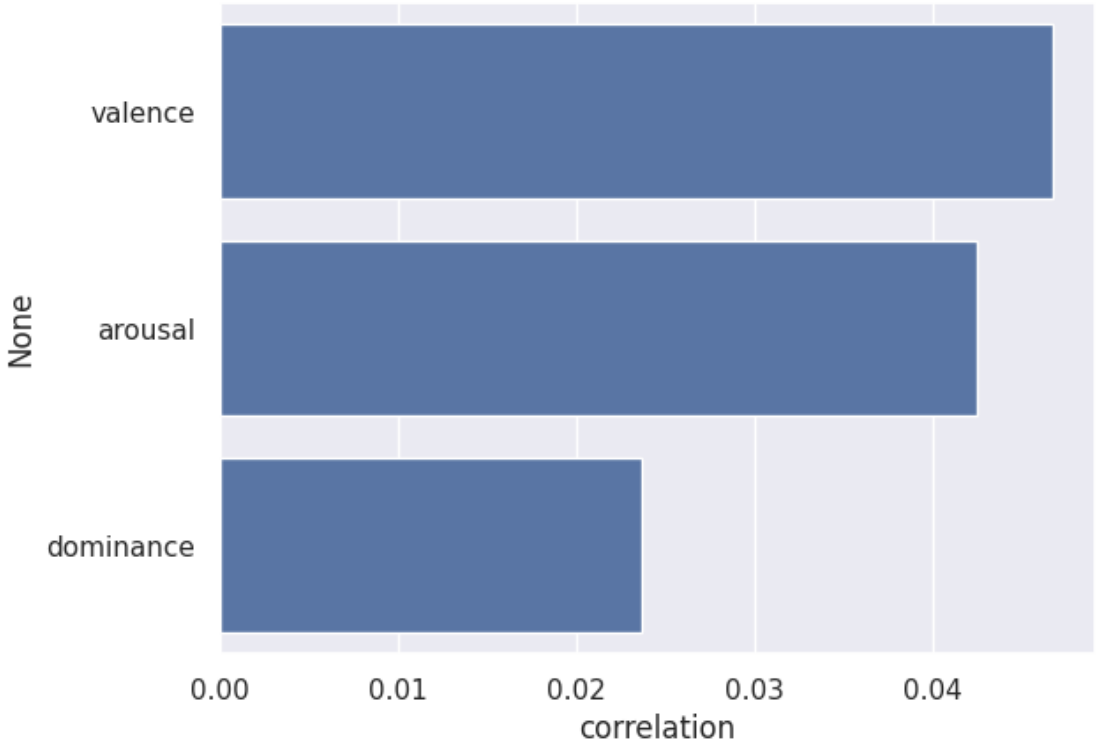
[30]:
```
0.006445591130335049
```

While not perfect, this is a twofold improvement on our previous attempt. Let's see what the Pearson's correlation tells us:

[31]:
```
correlation = r_regression(X_2, y_2)
corr_df = pd.DataFrame(correlation, index = X.columns, columns =␣
↪['correlation'])
```

[32]:
```
sns.barplot(y = corr_df.index, x = 'correlation', data = corr_df)
```

[32]:
```
<AxesSubplot: xlabel='correlation', ylabel='None'>
```



We can immediately see the difference of what comments are being upvoted on whether they contain 'russia' or 'ukraine' - high valence, high arousal, and high dominance words in comments all correlate with more upvotes for the latter. The biggest difference between the two countries in total axis & direction then is the scale of Arousal, suggesting that 'exciting' comments about Russia are downvoted, while 'exciting' comments about Ukraine are upvoted, suggesting that people have different tolerances for comments regarding these countries, which may reflect the average political stance of users on r/worldnews.

I then re-run this several times to see what was the highest R Squared value I could find for a token that appeared in the top 25. I eventually ended up with 'nato'. While still not a great R Squared value, it can still tell us something:

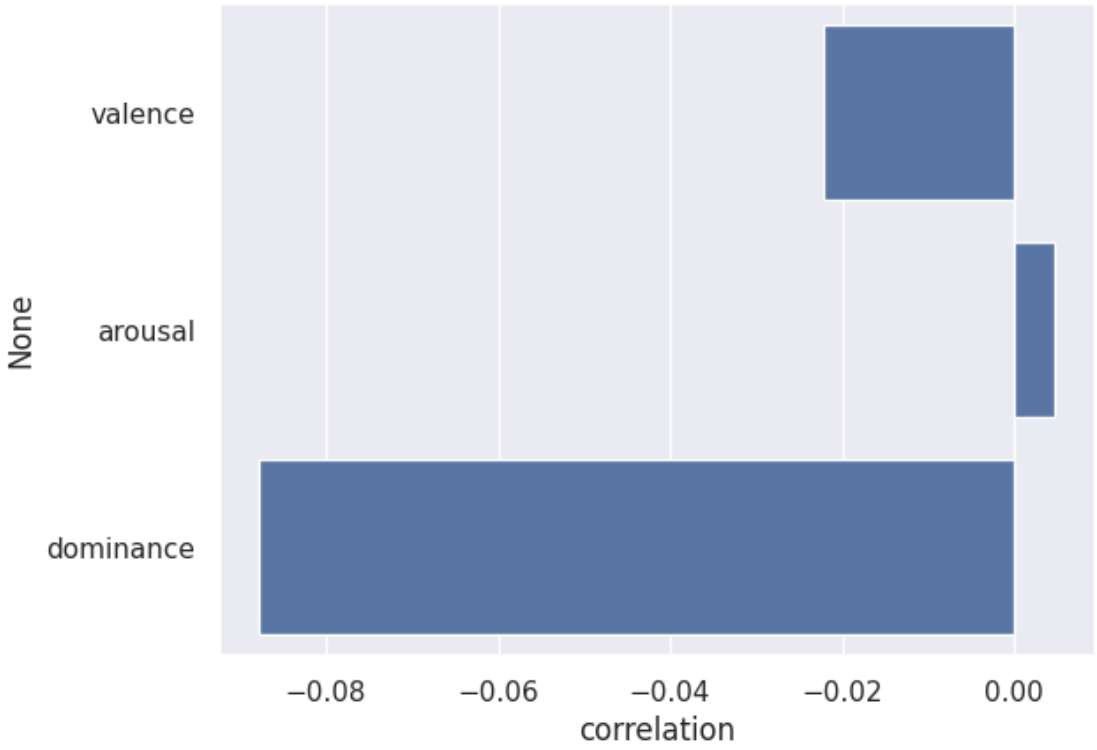
[52]:
```
corpus_nato = corpus[corpus['words_strings'].str.contains('nato')]
X_2 = corpus_nato[['valence', 'arousal', 'dominance']]
y_2 = corpus_nato['score']
reg_2 = linreg.fit(X_2,y_2) #Fit the model
reg_2.score(X_2,y_2)
```

[52]:
```
0.011126749322338148
```

[53]:
```
correlation = r_regression(X_2, y_2)
corr_df = pd.DataFrame(correlation, index = X.columns, columns =␣
↪['correlation'])
```

[54]:
```
sns.barplot(y = corr_df.index, x = 'correlation', data = corr_df)
```

[54]:
```
<AxesSubplot: xlabel='correlation', ylabel='None'>
```



Again, it is interesting just how much Dominance stands out, even if this time its 'direction' is the same as that of Valence. The negative correlation here is very strong, and an unexpected find, as it suggests people have really strong feelings about NATO and feeling in control of situations, even more so than they do with nation states like Russia and Ukraine, highlighting an interesting possible avenue for future research in regards to average people's feelings on geopolitical conflicts and entities.

0.2 Method 2: TF-IDF

The first method focused heavily on individual words, within individual comments, and how that correlated with Score. So for my second method, I decided to go in the opposite direction, and go as wide as possible, in an attempt to really make use of the large dataset of 4000 comments that I had. For this I chose TF-IDF analysis, which essentially rewards words that are unique to some comments, but not all, and punishes words that appear in a lot of comments. We can then use this data to visualise 'clusters' of comments connected by certain words or concepts that may otherwise

not be obvious to a human eye.

[55]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import PCA
from sklearn.cluster import AgglomerativeClustering
from IPython.display import IFrame
import plotly.express as px
from sklearn.cluster import KMeans
from sklearn.cluster import AffinityPropagation
from scipy.spatial import distance
```

[56]:
```
#all code below here for this method originally by James Carney
vectorizer = TfidfVectorizer(input = 'content', strip_accents = 'ascii',␣
 ↪stop_words = 'english')
text = [i for i in corpus['words_strings']]
vectors = vectorizer.fit_transform(text)
vectors = vectors.todense().tolist()
```

[57]:
```
TFIDF = pd.DataFrame(
    vectors,columns=vectorizer.get_feature_names_out())
```

[58]:
```
TFIDF
```

[58]:

	00	000	02	039	04	05	09	10	100	1000	…	zero	zest	\
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	
…	…	…	…	…	…	…	…	…	…	…	…	…	…	
4469	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	
4470	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	
4471	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	
4472	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	
4473	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	0.0	0.0	

	zimbabwe	zip	zl	zombied	zone	zoom	zoomin	zuckerberg
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
…	…	…	…	…	…	…	…	…
4469	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4470	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4471	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4472	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```
4473       0.0  0.0  0.0       0.0    0.0    0.0      0.0          0.0

[4474 rows x 7963 columns]
```

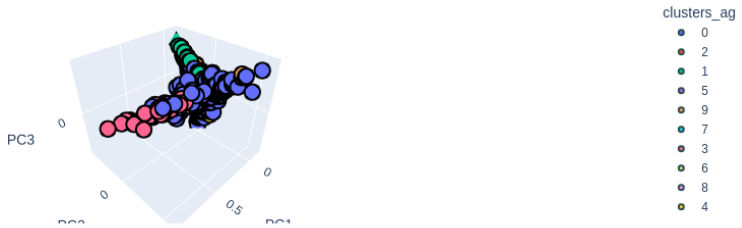
[59]:
```
pca_1 = PCA(n_components = 3)
comps_1 = pca_1.fit_transform(TFIDF)
pc_df_1 = pd.DataFrame(data = comps_1, columns = ['PC'+str(i) for i in range(1,␣
 ↪comps_1.shape[1]+1)])
```

[60]:
```
#after several trials, settled on 10 clusters as the optimal - discussed more␣
 ↪in 700 word Reflection
clustering = AgglomerativeClustering(n_clusters = 10, metric = 'euclidean',␣
 ↪linkage = 'ward').fit(TFIDF)
```

[61]:
```
kmeans = KMeans(n_clusters=5, random_state=0, n_init="auto").fit(TFIDF)
```

[62]:
```
df_all = pd.concat([TFIDF, pc_df_1], axis = 1)
df_all['clusters_ag'] = [str(i) for i in clustering.labels_]
df_all['clusters_knn'] = [str(i) for i in kmeans.labels_]
df_all['comment_text'] = corpus['words_strings']
```

[63]:
```
fig = px.scatter_3d(df_all, x='PC1', y='PC2', z='PC3',
              color='clusters_ag', hover_data = ['comment_text'])

fig.update_traces(marker=dict(size = 5, line=dict(width=2,
                                    color='DarkSlateGrey')),
                  selector=dict(mode='markers'))

fig.show()
#fig.write_html('10clustersworldnews.html')
```



The results are, from my best understanding (as there is a reason these were done by machine learning - humans just can't work well on such granular levels!), these 10 clusters of comments:

0 - Miscellaneous, largest cluster.

1 - About the Russian invasion of Ukraine, NATO, Trump.

2 - Containing the token 'viking', or otherwise to do with Scandinavia.

3 - Containing the tokens 'free' and/or 'speech', but usually occuring together as 'free speech'.

4 - To do with numbers.

5 - Very small cluster of about 10 comments, all containg the tokens 'kalmar' and 'union'.

6 - Small cluster containing tokens 'stop' and 'using'.

7 - About half contain the tokens 'palestine' *and* 'ohio', likely reffering to the village of East Palestine in the US where a train derailment occured last year.

8 - Containting the token 'welcome'.

9 - Small cluster containting the token 'fuck'.

Overall the most interesting cluster in relation to news here for me is that of 3, as freedom of speech an inherently political topic that nontheless is not directly linked to any current crisis or individual country. The second largest cluster, it suggests that future avenues of exploration of people's thoughts about freedom of speech in the context of news may be interesting.

0.3 Coda: Combinging Methods

After sitting on these results for some time, I kept coming back to cluster 2, as the token 'viking' did not appear in the top 100 words by frequency, much less the top 25. Thus I knews had missed doing any VAD analysis on comments containing it in my first method, and I wanted to try one more regression, even knowing the sample size was likely going to be small, to see if I could gain any last insights:

[66]:
```
corpus_viking = corpus[corpus['words_strings'].str.contains('viking')]
X_2 = corpus_viking[['valence', 'arousal', 'dominance']]
y_2 = corpus_viking['score']
reg_2 = linreg.fit(X_2,y_2) #Fit the model
reg_2.score(X_2,y_2)
```

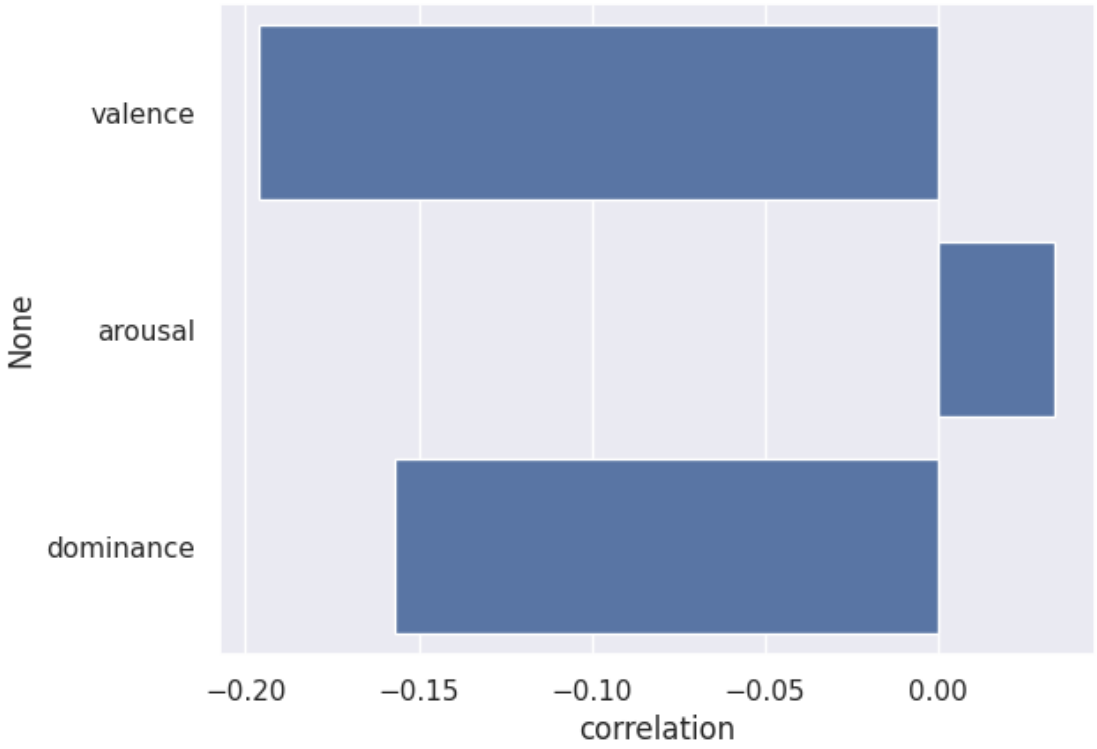
[66]:
```
0.03942098739070843
```

Best regression score yet! Again, nowhere close to 0.2, but still much better than what we started with.

[67]:
```
correlation = r_regression(X_2, y_2)
corr_df = pd.DataFrame(correlation, index = X.columns, columns =␣
 ↪['correlation'])
```

[68]:
```
sns.barplot(y = corr_df.index, x = 'correlation', data = corr_df)
```

[68]:
```
<AxesSubplot: xlabel='correlation', ylabel='None'>
```



On first glance, this seemed strange - why did it seem like people were downvoting comments that were positively speaking about vikings so much? However, one has to remember that words on their own mean little without context, particularly when we are dealing with political topics. For the past few years, Scandinavian mythos and Vikings have become increasingly associated with and coopted by the neo-nazi movement, meaning users who are aware of this connection and anti neo-nazis may be more likely to downvote comments that contained any positive mention of them. Of course, as mentioned in the article linked above, it is important to remember that this is approriation, and at the end of the day, vikings and Scandinativan symbols belong to the people of those cultures, rather than a political stance, and it makes me wonder if any actual comments about vikings in this dataset or elsewhere on reddit, disproportionately punished by users who mistake them for dogwhistles.

0.4 700 Word Reflection

As mentioned in the results analysis, I chose these two methods in order to analyse my corpus at both the word level, and as a whole, as I felt this way I could get the most interesting results. While the VAD and Linear Regression method did not go to entirely to plan (I definitely expected higher R squared numbers) I still feel the general trends of whether high/low Valence, Arousal, and Dominance impacted a comment's score in comparison with each other to be insightful. And of course, no result is still a result - why is it that the R Squared value for 'ukraine' was that much higher than the R Squared value for 'russia'? And is that linked to the different ways the two countries are currently perceived by the general population? Answering these questions is beyong the scope of this NLP-focused assignment, but it is the NLP that helps guide this potential future

analysis.

Originally I also planned to use the Word2Vec method, where you train a model using the entire dataset, resulting in being able to compare words to each other. For example, you could give the model multiple words in your data set and ask which one it thinks doesn't match. However, after running the code and playing around with the exact word inputs, I realised I wasn't getting anything useful. Even after removing stop words in my dataset, most of the 'closest words' for the most frequent and interesting tokens ('russia', 'ukraine', 'war', etc) were ones like 'also', 'since', or 'country'. All it really told me was that people discussing news will be discussing news - there were no particular standouts which could result in insightful analysis. Now, no result is still a result, and perhaps this speaks to how Redditors tend to discuss different topics in similar terms, or that the moderation on the subreddit is good enough to get rid of enough outliers, or simply that my dataset was too small. But for this assignment, and my original goal of wanting to find out how news impacts people's responses, I found it more useful to stick to the VAD modelling and TF-IDF with its larger-scale clusters.

On the topic of TFIDF, the most difficult part was getting the number of clusters right. I originally started with 5, found that too low to be useful, and went up to 10. This was more interesting (with 'cluster 1' appearing) but I was still unsatisfied with there being one single big cluster 0 for most words in the middle. However, increasing the cluster size to 20 and even 30 did not break up the one big cluster, but rather made the previous already-small clusters even smaller, to the point where some contained only a handful of comments. I'm not entirely sure why the model was doing that - maybe it speaks to the homogeneity of most Reddit comments, although after a cursory look at this big cluster I could not find any real similarities between the comments there, it just seemed like a 'miscellaneous' & 'neutral' bucket all rolled into one. For this reason I went back to 10 clusters, as this seemed like the optimal balance for the interesting themes that were brought out, while still keeping the sample sizes of each cluster big enough to feel like they statistically mattered.

On the whole, I think this analysis is a really good directional starting point for more research, even if by itself it is admittedly a bit weak. On one hand you have the aforementioned low R Squared scores, which make it difficult to give any even semi-conclusive results. On the other there's the sample - the population validity seems fairly low, as it's unclear how well we could transfer these Reddit-based results to the population as a whole. Yet it definitely seems that language in some instances does influence how people view news, and may help confirm certain cultural trends, such as how aware Reddit users seem to be of different neo-nazi symbols - here the cluster analysis really shines, though if I were to do this assignment again I'd aim for a larger overall corpus to give the TF-IDF more to work with.

Word count: 689. No AI tools have been used in the preparation for this assignment.

[ ]: