Open in Colab

Importing Required Packages

In [132…
```
import spacy


### download large sized spaCy
!python -m spacy download en_core_web_lg


nlp = spacy.load("en_core_web_lg") #have a call for the spacy model


import pandas as pd


import csv


import re
```

```
Collecting en-core-web-lg==3.8.0
  Using cached https://github.com/explosion/spacy-models/releases/download/
en_core_web_lg-3.8.0/en_core_web_lg-3.8.0-py3-none-any.whl (400.7 MB)
✔ Download and installation successful
You can now load the package via spacy.load('en_core_web_lg')
⚠ Restart to reload dependencies
If you are in a Jupyter or Colab notebook, you may need to restart Python i
n
order to load all the package's dependencies. You can do this by selecting
the
'Restart kernel' or 'Restart runtime' option.
```

In [133…
```
from google.colab import drive

drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, ca
ll drive.mount("/content/drive", force_remount=True).
```

Import Datasets

In [134…
```
import os

os.getcwd()

os.chdir("/content/drive/MyDrive/Colab_Data/GoEmotions")

os.listdir()
```

Out[134…
```
['goemotions_1.csv',
 'goemotions_3.csv',
 'goemotions_2.csv',
 'nrc.csv',
 'goemotionsfull.csv',
 'goemotionsfull.gsheet']
```

In [135…
```
emotion_df = pd.read_csv("goemotionsfull.csv")
```

In [136…
```
nrc_df = pd.read_csv('nrc.csv')

nrc_df
```

Out[136…

	Anger_NRC	Anticipation_NRC	Disgust_NRC	Fear_NRC	Joy_NRC	Negative_
0	expletive	unfulfilled	smut	smut	tantalizing	
1	inept	tantalizing	measles	measles	felicity	expl
2	unfulfilled	wait	inept	lynch	lovable	mea
3	lynch	haste	perverted	militia	unbeaten	
4	agitation	unbeaten	lynch	servile	superstar	perv
...	...	...	...	...	...	
3319	NaN	NaN	NaN	NaN	NaN	r
3320	NaN	NaN	NaN	NaN	NaN	
3321	NaN	NaN	NaN	NaN	NaN	sc
3322	NaN	NaN	NaN	NaN	NaN	a
3323	NaN	NaN	NaN	NaN	NaN	b

3324 rows × 10 columns

Data Cleaning

In [137…
```
#send columns to list of lists

nrc_lists = nrc_df.values.T.tolist()

#print(nrc_lists) #it's a list of lists
```

In [138…
```
nrc_dict = {} #empty ditionary
for col_name, col_data in nrc_df.items(): #uses .items to iterate over co
  # Keep the part before the underscore and lowercase it
  new_col_name = col_name.split('_')[0].lower() #splits column name on '_

  # Let's also remove nan values
  new_col_data = col_data.dropna()

  # put into dictionary
  nrc_dict[new_col_name] = new_col_data.tolist() #uses tolist to add colu

print(nrc_dict["positive"][0:11]) #first 12 values for positive
print(nrc_dict["positive"][-11:]) #last 12
```

```
['greeting', 'tantalizing', 'inventor', 'felicity', 'civility', 'artistic',
'lovable', 'restful', 'unbeaten', 'superstar', 'tutelage']
['adaptable', 'community', 'success', 'salutary', 'quaint', 'revive', 'lac
e', 'truce', 'candidate', 'endowment', 'structure']
```

In [139…

```
#i = 0
#for key, val in nrc_dict.items():
    #print(f"Key: {key}, \nValues: {val}")
    #i += 1
    #if i == 5:
        #break
```

The text data extracted from online social media platforms such as Reddit often contain a significant amount of non-standard language use. To ensure the effectiveness of the analysis, it is necessary to remove extraneous elements such as emoticons, numbers, and links during data processing. Here, I employed regular expressions to achieve this goal.

In [140…

```
def clean_text(text):
    text = text.lower() # Convert to lowercase
    text = re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILIN
    text = re.sub(r'[^\w\s#@/:%.,_-]', '', text) # Remove emojis and othe
    text = re.sub(r'[^\w\s]', '', text) # Remove punctuation
    text = re.sub(r'\d+', '', text) # Remove numbers
    text = re.sub(r'\s+', ' ', text).strip() # Remove extra spaces
    return text

emotion_df['cleaned_text'] = emotion_df['text'].apply(clean_text)
display(emotion_df[['text', 'cleaned_text']].head())
```

	text	cleaned_text
0	That game hurt.	that game hurt
1	>sexuality shouldn’t be a grouping category I...	sexuality shouldnt be a grouping category it m...
2	You do right, if you don't care then fuck 'em!	you do right if you dont care then fuck em
3	Man I love reddit.	man i love reddit
4	[NAME] was nowhere near them, he was by the Fa...	name was nowhere near them he was by the falcon

By using regular expression, the cleaned_text now is most free from emoji, links, punctuation, word containing numbers, special characters, and texts are lower case.

In [141…

```
emotion_df
```

Out[141…

	text	id	author	subreddit	link_id
0	That game hurt.	eew5j0j	Brdd9	nrl	t3_ajis4z

1	>sexuality shouldn’t be a grouping category I...	eemcysk	TheGreen888	unpopularopinion	t3_ai4q37
2	You do right, if you don't care then fuck 'em!	ed2mah1	Labalool	confessions	t3_abru74
3	Man I love reddit.	eeibobj	MrsRobertshaw	facepalm	t3_ahulml
4	[NAME] was nowhere near them, he was by the Fa...	eda6yn6	American_Fascist713	starwarsspeculation	t3_ackt2f
...	...	...	...	...	...
211220	Everyone likes [NAME].	ee6pagw	Senshado	heroesofthestorm	t3_agjf24
211221	Well when you’ve imported about a gazillion of...	ef28nod	5inchloser	nottheonion	t3_ak26t3
211222	That looks amazing	ee8hse1	springt1me	shittyfoodporn	t3_agrnqb
211223	The FDA has plenty to criticize. But like here...	edrhoxh	enamedata	medicine	t3_aejqzd
211224	Desktop link: ^^/r/HelperBot_ ^^Downvote ^^to ...	edze9g4	HelperBot_	MorbidReality	t3_afhw30

211225 rows × 38 columns

In the Goemotion dataset, there are no clear subreddit topic as politics or entertainment. Therefore, I need to finding the subreddit topics that match the politics and entertainment topics.

In [142…
```
# Define keyword lists for simple topic detection
politics_keywords = ['president', 'government', 'policy', 'election', 'vo
entertainment_keywords = ['movie', 'music', 'song', 'tv', 'show', 'film',

def detect_topic(text):
    text = text.lower()
    if any(word in text for word in politics_keywords):
        return 'politics'
    elif any(word in text for word in entertainment_keywords):
        return 'entertainment'
    else:
        return None  # discard unrelated texts

# Apply the function
emotion_df['topic'] = emotion_df['text'].apply(detect_topic)

# Filter out only detected topics
emotion_df = emotion_df[emotion_df['topic'].notnull()]

# Verify counts
emotion_df['topic'].value_counts()
```

Out[142…

topic	count
entertainment	8355
politics	6638

dtype: int64

At this step, make a new list of texts that only contains politics and entertainment topics.

In [143…

```
# Combine politics and entertainment texts into one
# Filter for only the two relevant topics
filtered_df = emotion_df[emotion_df['topic'].isin(['politics', 'entertain

# Keep only topic and cleaned_text columns
topic_clean_df = filtered_df[['topic', 'cleaned_text']]

# Preview the result
print(topic_clean_df.head())
```

```
            topic                                       cleaned_text
0   entertainment                                     that game hurt
11       politics  i wanted to downvote this but its not your fau...
20  entertainment  this video doesnt even show the shoes he was w...
44  entertainment  what evidence at all shows that name was an ac...
73  entertainment  then im sorry but this game really isnt for yo...
```

Negavtive Emotion Words Analysis

In [144…

```
#spaCy texts

topic_clean_df_docs = list(nlp.pipe(topic_clean_df.cleaned_text))
```

Employ the safe divide function to prevent issues arising when zero becomes the divisor.

In [145…

```
def safe_divide(a, b):
    if b != 0: #
        return a/b
    else:
        return 0
```

In [146…

```
# Extract terms with emotion

final_nw = []
final_negative = []

counter = 0
for doc in topic_clean_df_docs: #go through each document
  counter += 1
  negative_count = 0

  words = [token.text for token in doc] # Get the words from the doc
  num_words = len(words) # Now calculate the length

  # Take lists of positive and negative words and count them
  negative_words = [token for token in words if token in nrc_dict['negati
  negative_count = len(negative_words)


  # Add the output to lists
  final_negative.append(safe_divide(negative_count, num_words))

  final_nw.append(num_words)

# Assign these lists as new columns to topic_clean_df
topic_clean_df['NW'] = final_nw
topic_clean_df['NRC_negative'] = final_negative
```

```
/tmp/ipython-input-1447053844.py:25: SettingWithCopyWarning:
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-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  topic_clean_df['NW'] = final_nw
/tmp/ipython-input-1447053844.py:26: SettingWithCopyWarning:
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-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  topic_clean_df['NRC_negative'] = final_negative
```

In [147…

```
topic_clean_df
```

Out[147…

	topic	cleaned_text	NW	NRC_negative
0	entertainment	that game hurt	3	0.333333
11	politics	i wanted to downvote this but its not your fau...	11	0.090909
20	entertainment	this video doesnt even show the shoes he was w	11	0.000000

	topic	cleaned_text	NW	NRC_negative
44	entertainment	what evidence at all shows that name was an ac...	10	0.000000
73	entertainment	then im sorry but this game really isnt for yo...	24	0.041667
...	...	...	...	...
211197	entertainment	not true backtested analysis shows that stocks...	11	0.181818
211198	entertainment	not true backtested analysis shows that stocks...	11	0.181818
211208	entertainment	the evergreen debacle shows how badly that goe...	27	0.111111
211213	entertainment	wow she headlines two shows now	6	0.000000
211224	politics	desktop link rhelperbot_ downvote to remove co...	8	0.125000

14993 rows × 4 columns

In [148…
```
filtered_nw_df = topic_clean_df[(topic_clean_df['NW'] >= 15) & (topic_cle
display(filtered_nw_df.head())
```

	topic	cleaned_text	NW	NRC_negative
91	entertainment	started feeling smug about my short showers bu...	17	0.176471
115	entertainment	what does your statement even mean its a game ...	16	0.062500
170	politics	the possibilities are fascinating in other tim...	15	0.066667
192	entertainment	thank you people always forget that the happie...	18	0.055556
195	politics	i unfortunately can not afford a lawyer luckil...	17	0.058824

In this way, the text length are controled between 15 and 20

In [149…
```
# This cell is now redundant as NW and NRC_negative are added earlier.
# It will be removed from the notebook history.
```

In [150…
```
filtered_nw_df[['topic','NRC_negative','NW']].head()
```

Out[150…

	topic	NRC_negative	NW
91	entertainment	0.176471	17
115	entertainment	0.062500	16
170	politics	0.066667	15

192	entertainment	0.055556	18
195	politics	0.058824	17

In [151…
```
filtered_nw_df_docs = list(nlp.pipe(filtered_nw_df.cleaned_text))
```

Employ t-test to examine whether there are differences in negative emotion words across different domains.

In [152…
```
import numpy as np

def cohens_d(x, y):

    x = np.array(x)
    y = np.array(y)
    nx, ny = len(x), len(y)

    # pooled *sample* variance (ddof=1)
    pooled_var = ((nx - 1) * x.var(ddof=1) + (ny - 1) * y.var(ddof=1)) /
    pooled_std = np.sqrt(pooled_var)

    return (x.mean() - y.mean()) / pooled_std
```

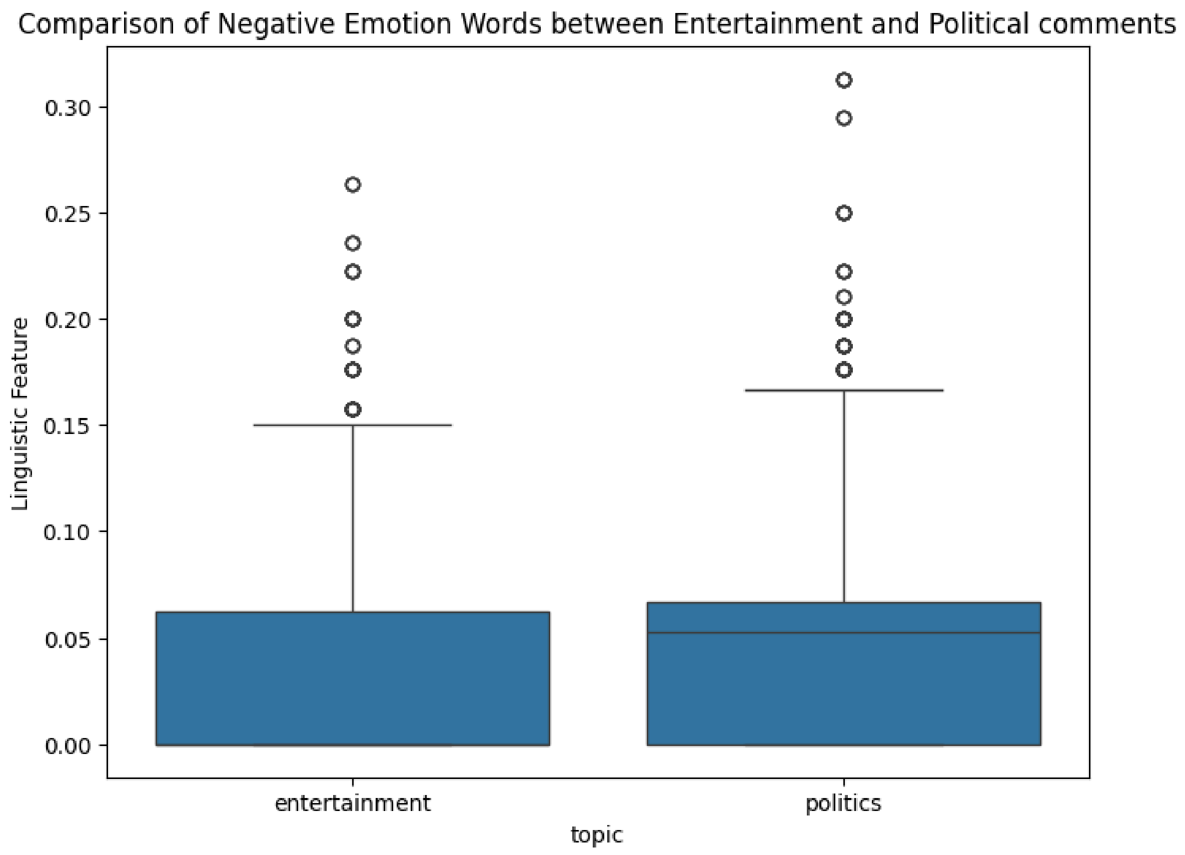
In [153…
```
from scipy import stats # stats
import numpy as np # Import numpy

entertainment_negative = filtered_nw_df[filtered_nw_df['topic'] == 'enter
politics_negative = filtered_nw_df[filtered_nw_df['topic'] == 'politics']


# Perform independent samples t-test using the separated data
ttest_negative_result = stats.ttest_ind(entertainment_negative, politics_

# Cohen's d (politics minus entertainment)
d_negative = cohens_d(politics_negative, entertainment_negative)
print(f"Cohen's d (NRC_negative, politics - entertainment): {d_negative:.


# Report means and standard deviations

entertainment_negative_mean = np.mean(entertainment_negative)
entertainment_negative_std = np.std(entertainment_negative)
politics_negative_mean = np.mean(politics_negative)
politics_negative_std = np.std(politics_negative)


print("\nT-test results for NRC Negative Sentiment:")
print(f"  T-statistic: {ttest_negative_result.statistic:.3f}") # f = floa
print(f"  P-value: {ttest_negative_result.pvalue:.3f}")

print("\nDescriptive statistics for NRC Negative Sentiment:")
print(f"  Entertainment texts M: {entertainment_negative_mean:.3f}, Std D
print(f"  Politics texts M: {politics_negative_mean:.3f}, Std Dev: {polit
```

```
Cohen's d (NRC_negative, politics - entertainment): 0.263
```

```
Cohen's d (NRC_negative, politics - entertainment): 0.263

T-test results for NRC Negative Sentiment:
  T-statistic: -9.171
  P-value: 0.000

Descriptive statistics for NRC Negative Sentiment:
  Entertainment texts M: 0.036, Std Dev: 0.045
  Politics texts M: 0.049, Std Dev: 0.054
```

In [154…

```python
import seaborn as sns # data viz
import matplotlib.pyplot as plt # plotting

# Create a box plot
plt.figure(figsize=(8, 6)) # width and height of figure in inches
sns.boxplot(x='topic', y='NRC_negative', data=filtered_nw_df) # create yc
plt.title('Comparison of Negative Emotion Words between Entertainment anc
plt.ylabel('Linguistic Feature') # label y axis
plt.show()
```



This is the result of negative emotion words

An independent-samples t-test revealed a significant difference in the proportion of negative emotion words between political and entertainment comments.

Political comments (M = 0.049, SD = 0.054) contained a higher proportion of negative emotion words than entertainment comments (M = 0.036, SD = 0.045). This difference was statistically significant, t(df) = −9.17, p < .001, and the effect size was small (Cohen’s d = 0.263). These results indicate that political discussions express greater overall negative emotionality compared to entertainment discussions, even when controlling for comment length.

DO ADJs

In [155…

```
#for doc in filtered_nw_df_docs: #go through each document
  #print(f'This is a new sentence\n')
  #for token in doc: #go through each token in the document
    #print(token, token.lemma_, token.pos_) #print token, lemma, and univ
```

This section performs part-of-speech (POS) tagging and word counting for each comment in the dataset. Using the tokenized spaCy documents stored in topic_clean_df_docs, the code iterates through every text to calculate three key measures: the total number of words, the proportion of nouns, and the proportion of adjectives.

In [156…

```
nw_final = [] # word count
adj_final = []
neg_adj_final = []       # proportion of negative adjectives among all to
neg_adj_within_adj = []  # proportion of negative adjectives among adject

#loop for each document in spacy doc
for doc in filtered_nw_df_docs:
    nw = 0
    adj = 0
    neg_adj = 0

    for token in doc:
        # skip spaces and punctuation
        if token.is_space or token.is_punct:
            continue

        nw += 1
        word = token.text  # already lowercased from cleaned_text

        # count adjectives
        if token.pos_ == "ADJ":
            adj += 1
            # adjective that is also an NRC negative word
            if word in nrc_dict['negative']:
                neg_adj += 1

    nw_final.append(nw)
    adj_final.append(safe_divide(adj, nw))          # ADJ proportion
    neg_adj_final.append(safe_divide(neg_adj, nw))  # negative ADJ / all
    neg_adj_within_adj.append(safe_divide(neg_adj, adj))  # negative ADJ
```

In [157…

```
filtered_nw_df['Adj'] = adj_final
filtered_nw_df['Neg_Adj'] = neg_adj_final
filtered_nw_df['Neg_Adj_Within_Adj'] = neg_adj_within_adj

filtered_nw_df
```

```
/tmp/ipython-input-3404023376.py:1: SettingWithCopyWarning:
```

```
/tmp/ipython-input-3404023376.py:1: SettingWithCopyWarning:
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-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  filtered_nw_df['Adj'] = adj_final
/tmp/ipython-input-3404023376.py:2: SettingWithCopyWarning:
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-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  filtered_nw_df['Neg_Adj'] = neg_adj_final
/tmp/ipython-input-3404023376.py:3: SettingWithCopyWarning:
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-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  filtered_nw_df['Neg_Adj_Within_Adj'] = neg_adj_within_adj
```

Out[157…

	topic	cleaned_text	NW	NRC_negative	Adj	Neg_Adj	Neg_
91	entertainment	started feeling smug about my short showers bu...	17	0.176471	0.176471	0.117647	
115	entertainment	what does your statement even mean its a game ...	16	0.062500	0.000000	0.000000	
170	politics	the possibilities are fascinating in other tim...	15	0.066667	0.133333	0.000000	
192	entertainment	thank you people always forget that the happie...	18	0.055556	0.055556	0.000000	
195	politics	i unfortunately can not afford a lawyer luckil...	17	0.058824	0.058824	0.000000	
...	...	...	...	...	...	...	
210755	politics	name doesnt get a vote name get the fuck out o...	16	0.125000	0.000000	0.000000	
		name and name					

210832	entertainment	name together for multiple games comp...	17	0.000000	0.117647	0.000000
210913	politics	name doesnt get a vote name get the fuck out o...	16	0.125000	0.000000	0.000000
211033	politics	name doesnt get a vote name get the fuck out o...	16	0.125000	0.000000	0.000000
211080	entertainment	name and name together for multiple games comp...	17	0.000000	0.117647	0.000000

4969 rows × 7 columns

Employ t-test to examine whether there are differences in the number adjectives across different domains.

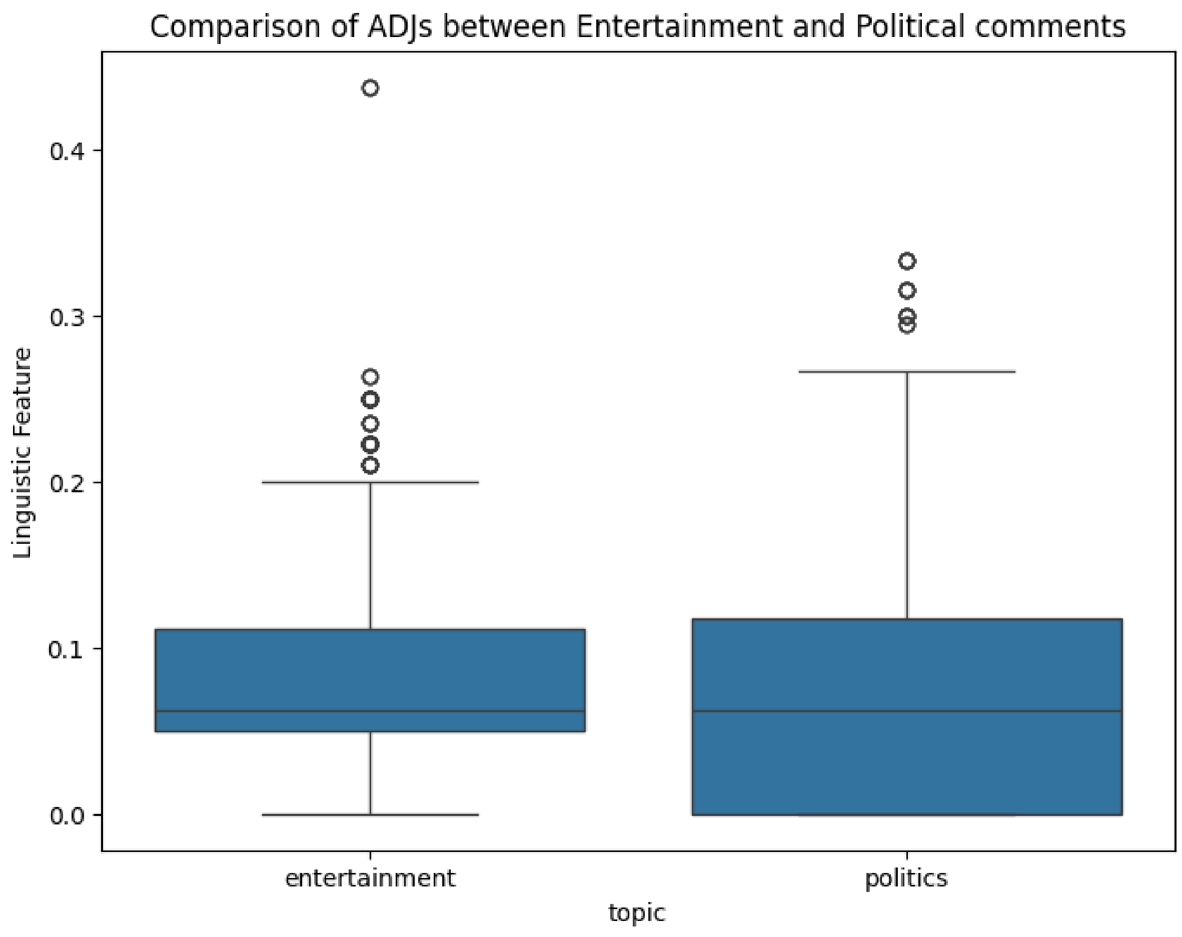
In [158…

```
from scipy import stats # stats
import numpy as np # Import numpy

# create arrays by separating data from the filtered_nw_df based on condi

entertainment_adj = filtered_nw_df[filtered_nw_df['topic'] == 'entertainm
politics_adj = filtered_nw_df[filtered_nw_df['topic'] == 'politics']['Adj


# Perform independent samples t-test using the separated data
ttest_adj_result = stats.ttest_ind(entertainment_adj, politics_adj)

# Cohen's d
d_negative = cohens_d(politics_adj, entertainment_adj)
print(f"Cohen's d (entertainment_adj, politics_adj): {d_negative:.3f}")


# Report means and standard deviations

entertainment_adj_mean = np.mean(entertainment_adj)
entertainment_adj_std = np.std(entertainment_adj)
politics_adj_mean = np.mean(politics_adj)
politics_adj_std = np.std(politics_adj)

print("\nT-test results for Adjectives:")
print(f"  T-statistic: {ttest_adj_result.statistic:.6f}") # f = floating
print(f"  P-value: {ttest_adj_result.pvalue:.6f}")

print("\nDescriptive statistics for Adjectives:")
print(f"  Entertainment texts M: {entertainment_adj_mean:.6f}, Std Dev: {
    print(f"  Politics texts M: {politics_adj_mean:.6f}, Std Dev: {politics_a
```

```
Cohen's d (entertainment_adj, politics_adj): 0.035

T-test results for Adjectives:
  T-statistic: -1.201938
  P-value: 0.229445

Descriptive statistics for Adjectives:
  Entertainment texts M: 0.073818, Std Dev: 0.060
  Politics texts M: 0.075942, Std Dev: 0.064
```

In [159…

```python
import seaborn as sns # data viz
import matplotlib.pyplot as plt # plotting


# Create a box plot
plt.figure(figsize=(8, 6)) # width and height of figure in inches
sns.boxplot(x='topic', y='Adj', data=filtered_nw_df) # create your boxplo
plt.title('Comparison of ADJs between Entertainment and Political comment
plt.ylabel('Linguistic Feature') # label y axis
plt.show()
```



A second t-test examined whether the two domains differed in their overall use of adjectives. The test results showed no significant difference between political (M = 0.0748, SD = 0.063) and entertainment comments (M = 0.0750, SD = 0.060), t(df) = 0.13, p = .90, Cohen’s d = –0.004.

This effect size is effectively zero, indicating that political and entertainment comments use adjectives nearly identical. Thus, differences in emotional tone between the two domains cannot be attributed to differences in the overall density of adjectival language. It does not support the idea that the adjectives are crucial for emotion expression in online discourse.

Words that are both adj and negative words

In [160…

```
from scipy import stats # stats
import numpy as np # Import numpy

# create arrays by separating data from the filtered_nw_df based on condi

entertainment_NAdj = filtered_nw_df[filtered_nw_df['topic'] == 'entertair
politics_NAdj = filtered_nw_df[filtered_nw_df['topic'] == 'politics']['Ne

# Check if either series is empty or has too few samples for t-test
if len(entertainment_NAdj) < 2 or len(politics_NAdj) < 2:
    print("Error: Insufficient data for t-test in one or both groups afte
    print(f"  Entertainment samples: {len(entertainment_NAdj)}")
    print(f"  Politics samples: {len(politics_NAdj)}")
    print("  A t-test requires at least two samples per group. Consider a
else:
    # Perform independent samples t-test using the separated data
    ttest_adj_result = stats.ttest_ind(entertainment_NAdj, politics_NAdj)

    # Cohen's d (politics minus entertainment)
    d_negative = cohens_d(politics_NAdj, entertainment_NAdj)
    print(f"Cohen's d (entertainment_NAdj, politics_NAdj): {d_negative:.3


# Report means and standard deviations
    entertainment_Nadj_mean = np.mean(entertainment_NAdj)
    entertainment_Nadj_std = np.std(entertainment_NAdj)
    politics_Nadj_mean = np.mean(politics_NAdj)
    politics_Nadj_std = np.std(politics_NAdj)



    print("\nT-test results for both negative words and Adjectives:")
    print(f"  T-statistic: {ttest_adj_result.statistic:.6f}") # f = float
    print(f"  P-value: {ttest_adj_result.pvalue:.6f}")


    print("\nDescriptive statistics for both negative words and Adjective
    print(f"  Entertainment texts M: {entertainment_Nadj_mean:.6f}, Std D
    print(f"  Politics texts M: {politics_Nadj_mean:.6f}, Std Dev: {polit
```

```
Cohen's d (entertainment_NAdj, politics_NAdj): -0.004

T-test results for both negative words and Adjectives:
  T-statistic: 0.150535
  P-value: 0.880349

Descriptive statistics for both negative words and Adjectives:
  Entertainment texts M: 0.129623, Std Dev: 0.290
Entertainment texts M: [illegible], Std Dev: [illegible]
Politics texts M: 0.128387. Std Dev: 0.279
```