In [130…

```
# --- Core Libraries for Data Manipulation ---
import numpy as np
import pandas as pd
import re
# --- Libraries for Data Visualization ---
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
from collections import Counter

# --- Libraries for Natural Language Processing (NLP) ---
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer
```

In [131…

```
df = pd.read_csv('https://raw.githubusercontent.com/Himanshu-1703/reddit-sent
df.head()
```

Out[131…

	clean_comment	category
0	family mormon have never tried explain them t...	1
1	buddhism has very much lot compatible with chr...	1
2	seriously don say thing first all they won get...	-1
3	what you have learned yours and only yours wha...	0
4	for your own benefit you may want read living ...	1

In [132…

```
df.shape
```

Out[132… `(37249, 2)`

In [133…

```
df.sample()['clean_comment'].values
```

Out[133…

```
array([' important remember that the pakistan and the army operate separately
and history very good indicator that this means that often times there collis
ion interest between the two sides but the end the day the military establish
ment has always had the more power and the final say things this explains why
jem banned pakistan but not eradicated also explains why good relationship wi
th neighboring countries always suggested hinted pakistan but never materiali
zed would also like add that the oppressed minorities pakistan are aware the
army problem and are just sick and tired anyone else but their voices are sup
pressed '],
      dtype=object)
```

In [134…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 37249 entries, 0 to 37248
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   clean_comment  37149 non-null  object
 1   category       37249 non-null  int64
```

```
1   category        37249 non-null  int64
dtypes: int64(1), object(1)
memory usage: 582.1+ KB
```

In [135…
```
df.isnull().sum()
```

Out[135…
```
clean_comment    100
category           0
dtype: int64
```

In [136…
```
df[df['clean_comment'].isna()]
```

Out[136…

	clean_comment	category
413	NaN	0
605	NaN	0
2422	NaN	0
2877	NaN	0
3307	NaN	0
…	…	…
35975	NaN	0
36036	NaN	0
37043	NaN	0
37111	NaN	0
37238	NaN	0

100 rows × 2 columns

In [137…
```
df[df['clean_comment'].isna()]['category'].value_counts()
```

Out[137…
```
category
0    100
Name: count, dtype: int64
```

In [138…
```
df.dropna(inplace=True)
```

In [139…
```
df.duplicated().sum()
```

Out[139…
```
np.int64(350)
```

In [140…
```
df[df.duplicated()]
```

Out[140…

	clean_comment	category
375		0
392		0
617	aurum mom	0

651		0
1222		0
...	...	...
36915	who won	0
37044		0
37125	hari	0
37158	top kek	1
37234	this part series minute videos focusing each d...	1

350 rows × 2 columns

In [141…
```
df.drop_duplicates(inplace=True)
```

In [142…
```
df.duplicated().sum()
```

Out[142… `np.int64(0)`

In [143…
```
df[(df['clean_comment'].str.strip() == '')]
```

Out[143…

	clean_comment	category
181		0
4432	\n	0
10592		0
16173		0
32149	\n	0
34959		0

In [144…
```
df = df[~(df['clean_comment'].str.strip() == '')]
```

In [145…
```
df['clean_comment'] = df['clean_comment'].str.lower()
df.head()
```

Out[145…

	clean_comment	category
0	family mormon have never tried explain them t...	1
1	buddhism has very much lot compatible with chr...	1
2	seriously don say thing first all they won get...	-1
3	what you have learned yours and only yours wha...	0
4	for your own benefit you may want read living ...	1

In [146…
```
df[df['clean_comment'].apply(lambda x: x.endswith(' ') or x.startswith(' '))]
```

Out[146…

	clean_comment	category
0	family mormon have never tried explain them t...	1
1	buddhism has very much lot compatible with chr...	1
2	seriously don say thing first all they won get...	-1
3	what you have learned yours and only yours wha...	0
4	for your own benefit you may want read living ...	1
...	...	...
37241	let the janta decide not ulema clerics	0
37242	hona hai same with vaccination education insu...	0
37246	downvote karna tha par upvote hogaya	0
37247	haha nice	1
37248	facebook itself now working bjp' cell	0

32266 rows × 2 columns

In [147…
```
df['clean_comment'] = df['clean_comment'].str.strip()
df['clean_comment'].apply(lambda x: x.endswith(' ') or x.startswith(' ')).sum
```

Out[147… `np.int64(0)`

In [148…
```
url_pattern = r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\\(\\),]|(?:%[0-9a-
comments_with_urls = df[df['clean_comment'].str.contains(url_pattern, regex=T
comments_with_urls.head()
```

Out[148…

	clean_comment	category

In [149…
```
comments_with_newline = df[df['clean_comment'].str.contains('\n')]
comments_with_newline.head()
```

Out[149…

	clean_comment	category
448	what missing jpg\nand why this brilliant edit ...	1
781	india has been ruined congress and populist sc...	-1
847	like aap for its stand corruption and making p...	-1
871	reduced trade\ndeficit stronger rupee aren the...	0
1354	amsa press conference australian maritime safe...	1

In [150…

```
df['clean_comment'] = df['clean_comment'].str.replace('\n', ' ', regex=True)
comments_with_newline_remaining = df[df['clean_comment'].str.contains('\n')]
comments_with_newline_remaining
```

Out[150…

	clean_comment	category

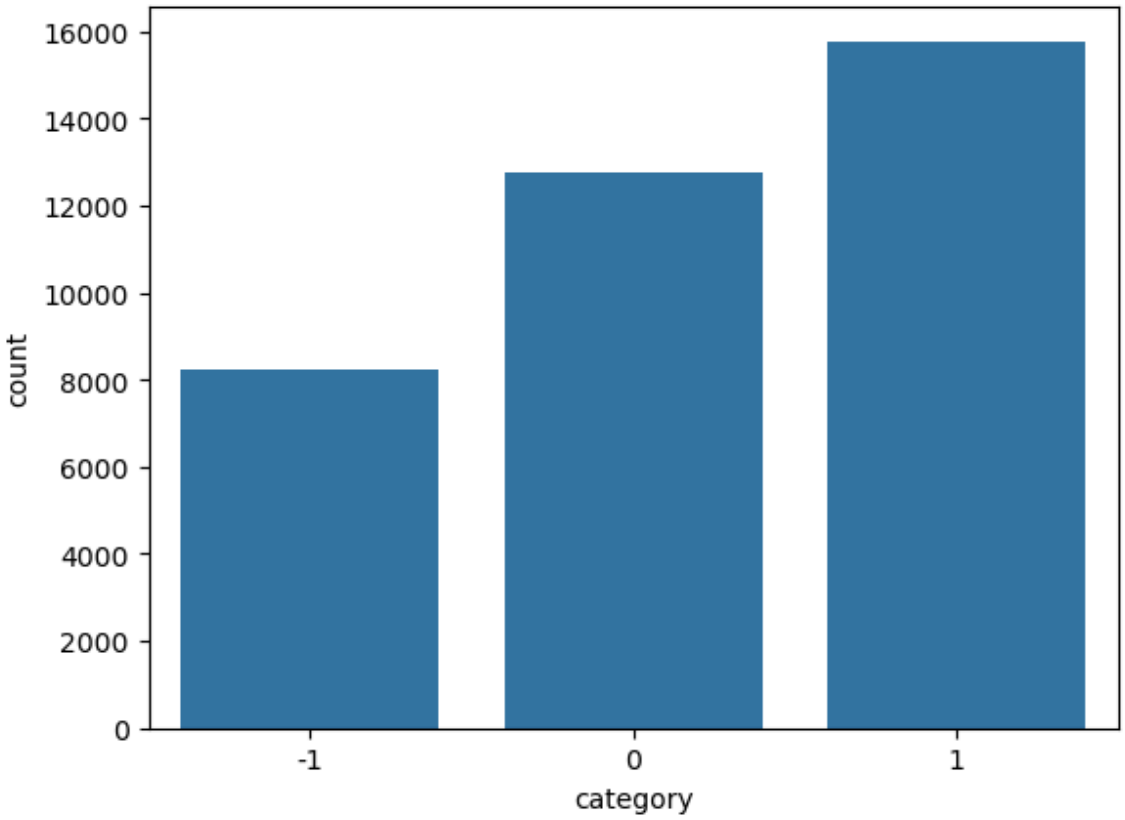
EDA

In [151…

```
sns.countplot(data=df,x="category")
```

Out[151…

```
<Axes: xlabel='category', ylabel='count'>
```



In [152…

```
df['category'].value_counts(normalize=True).mul(100).round(2)
```

Out[152…

```
category
 1    42.86
 0    34.71
-1    22.42
Name: proportion, dtype: float64
```

In [153…

```
df['word_count'] = df['clean_comment'].apply(lambda x: len(x.split()))
df.sample(5)
```

Out[153…

	clean_comment	category	word_count
21504	how funny when upper caste guy does something ...	1	35
16383	next diwali will have completely turned life a...	-1	30
31791	these look forced	-1	3
31093	new starter pokemon for shield called sorry co...	-1	10

23554	what are the directives worshipping namo pbuh ...	-1	40

In [154…

```
df['word_count'].describe()
```

Out[154…

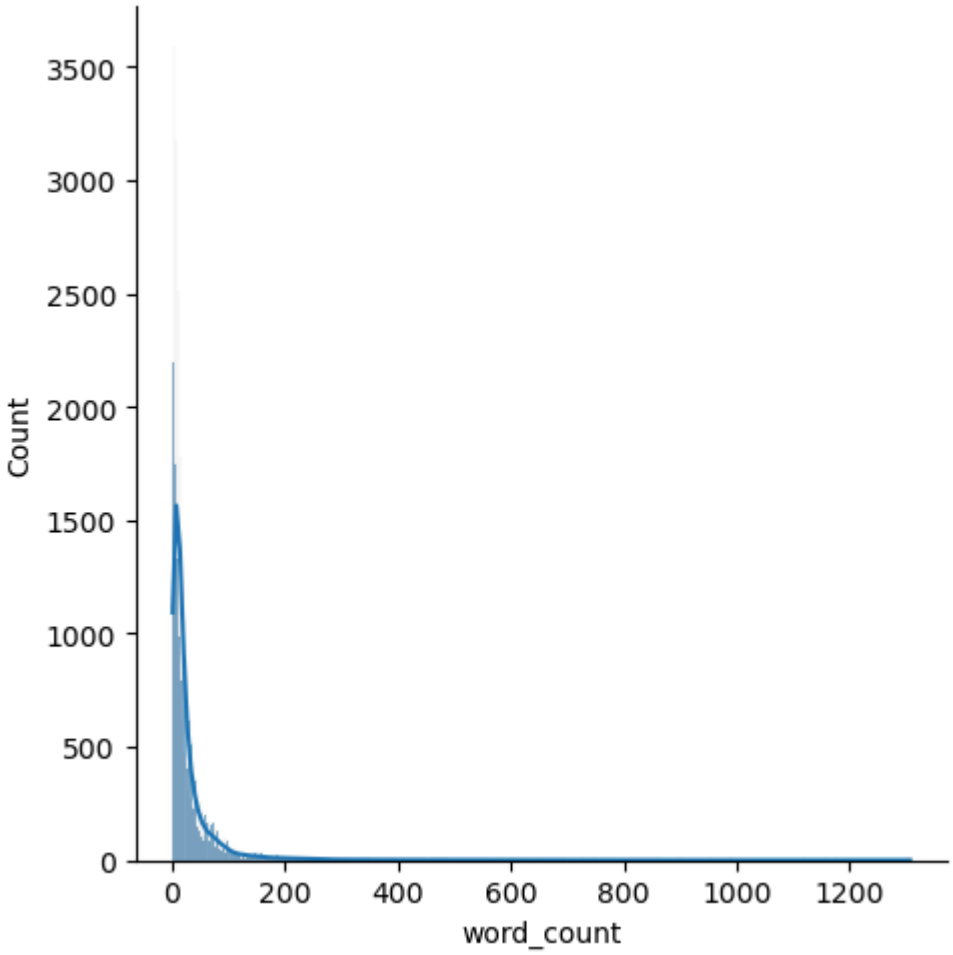
```
count    36793.000000
mean        29.667464
std         56.790738
min          1.000000
25%          6.000000
50%         13.000000
75%         30.000000
max       1307.000000
Name: word_count, dtype: float64
```

In [155…

```
sns.displot(df['word_count'], kde=True)
```

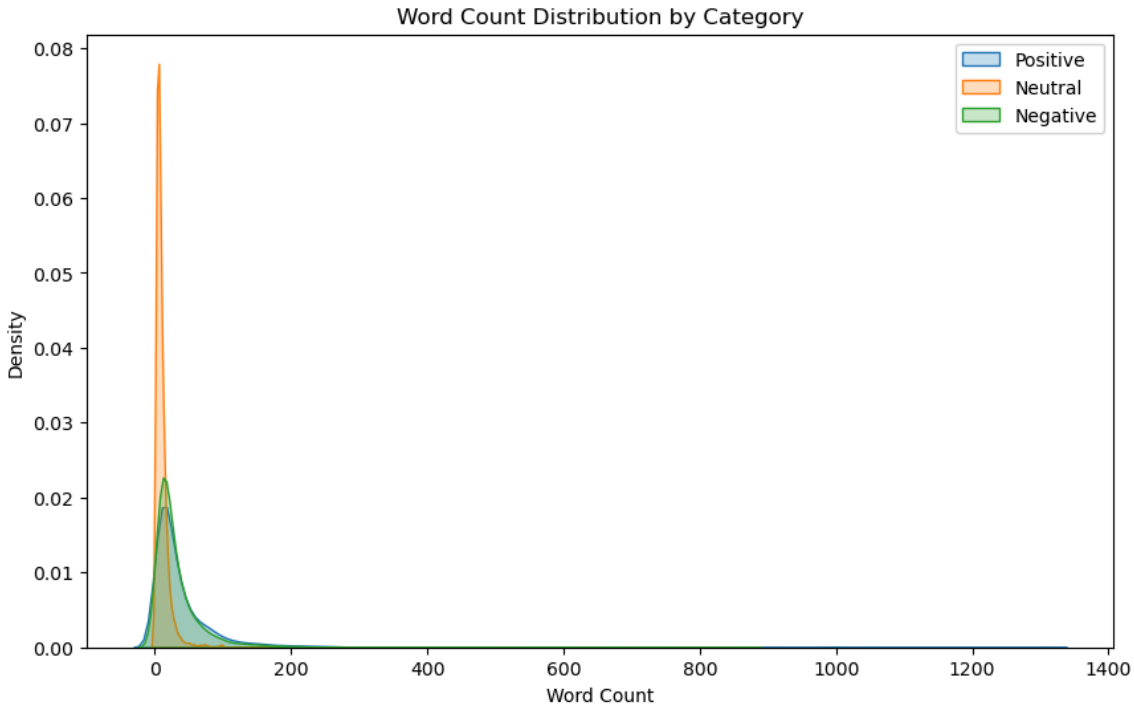
Out[155…

```
<seaborn.axisgrid.FacetGrid at 0x272a0d57250>
```



In [156…

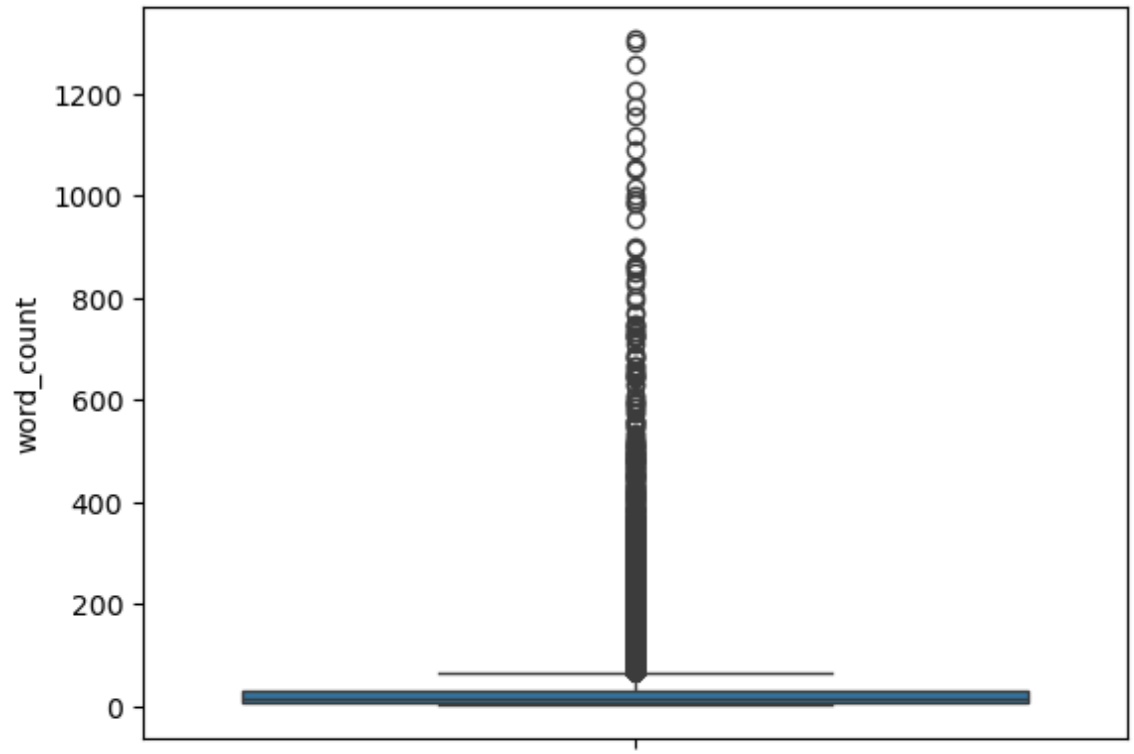
```
plt.figure(figsize=(10, 6))
sns.kdeplot(df[df['category'] == 1]['word_count'], label='Positive', fill=Tru
sns.kdeplot(df[df['category'] == 0]['word_count'], label='Neutral', fill=True
sns.kdeplot(df[df['category'] == -1]['word_count'], label='Negative', fill=Tr
plt.title('Word Count Distribution by Category')
plt.xlabel('Word Count')
plt.ylabel('Density')
plt.legend()
plt.show()
```



Positive comments (category 1): These tend to have a wider spread in word count, indicating that longer comments are more common in positive sentiments.
Neutral comments (category 0): The distribution shows a relatively lower frequency and is more concentrated around shorter comments compared to positive or negative ones.
Negative comments (category -1): These comments have a distribution somewhat similar to positive comments but with a smaller proportion of longer comments.

In [157…
```
sns.boxplot(df['word_count'])
```

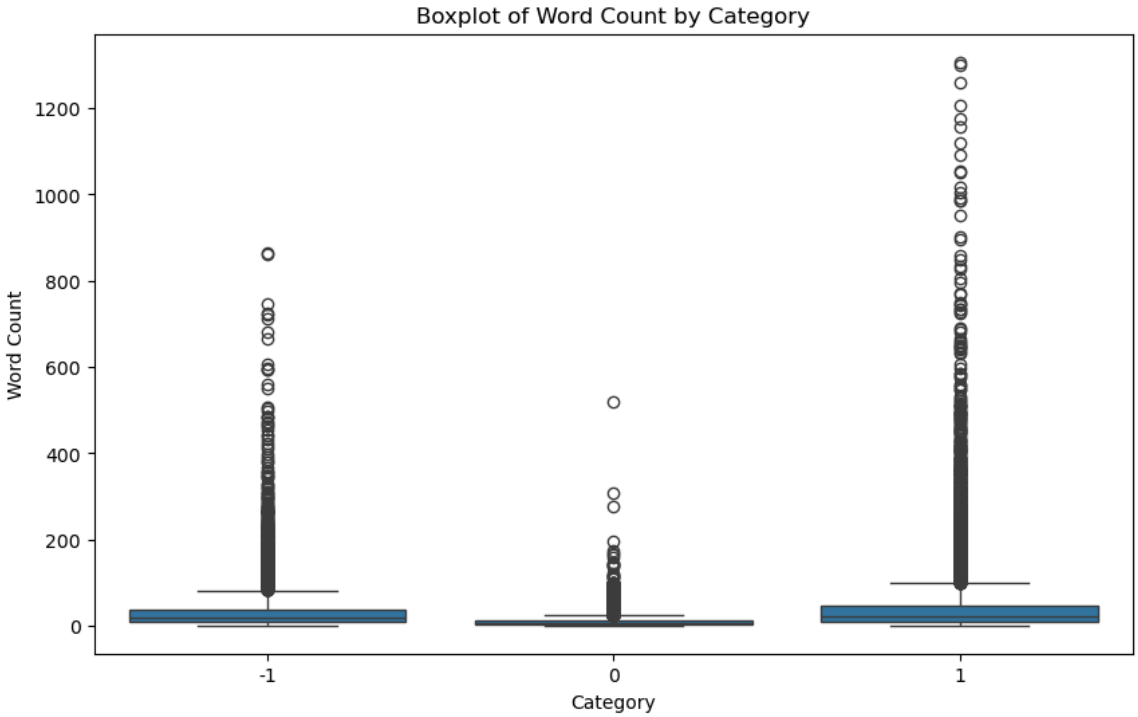
Out[157…
```
<Axes: ylabel='word_count'>
```



In [158…

In [158…

```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='category', y='word_count')
plt.title('Boxplot of Word Count by Category')
plt.xlabel('Category')
plt.ylabel('Word Count')
plt.show()
```



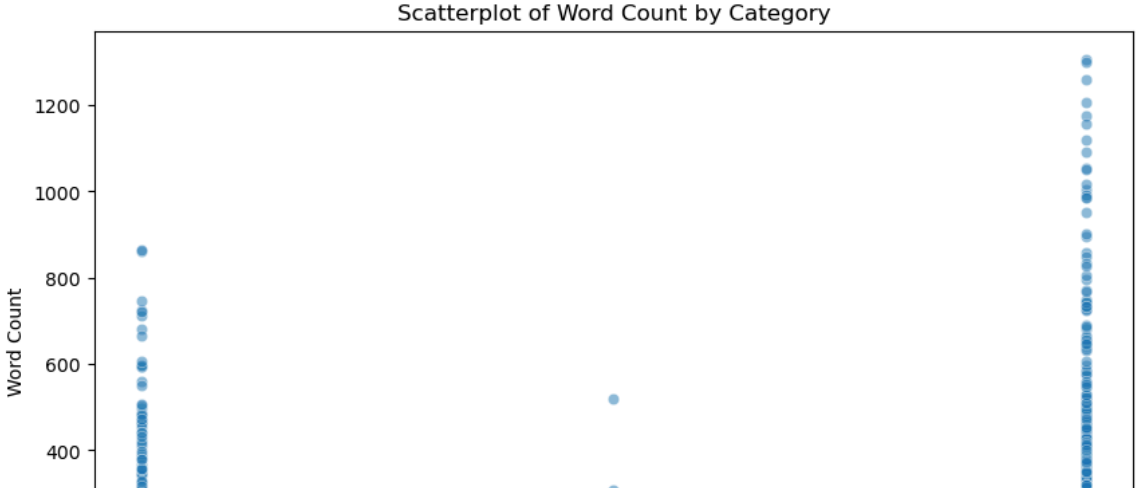
Positive comments (category 1): The median word count is relatively high, and there are several outliers with longer comments, indicating that positive comments tend to be more verbose.

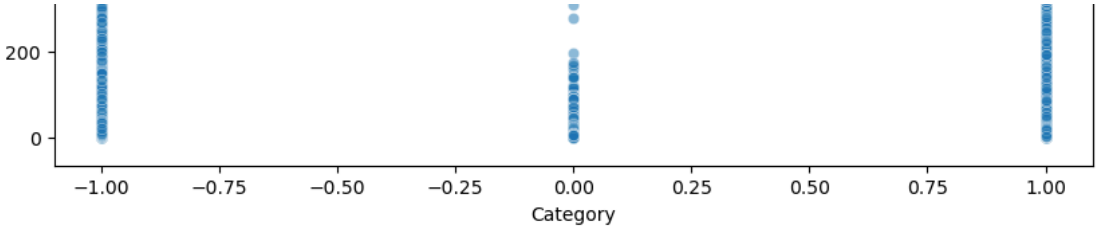
Neutral comments (category 0): The median word count is the lowest, with a tighter interquartile range (IQR), suggesting that neutral comments are generally shorter.

Negative comments (category -1): The word count distribution is similar to positive comments but with a slightly lower median and fewer extreme outliers.

In [159…

```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='category', y='word_count', alpha=0.5)
plt.title('Scatterplot of Word Count by Category')
plt.xlabel('Category')
plt.ylabel('Word Count')
plt.show()
```

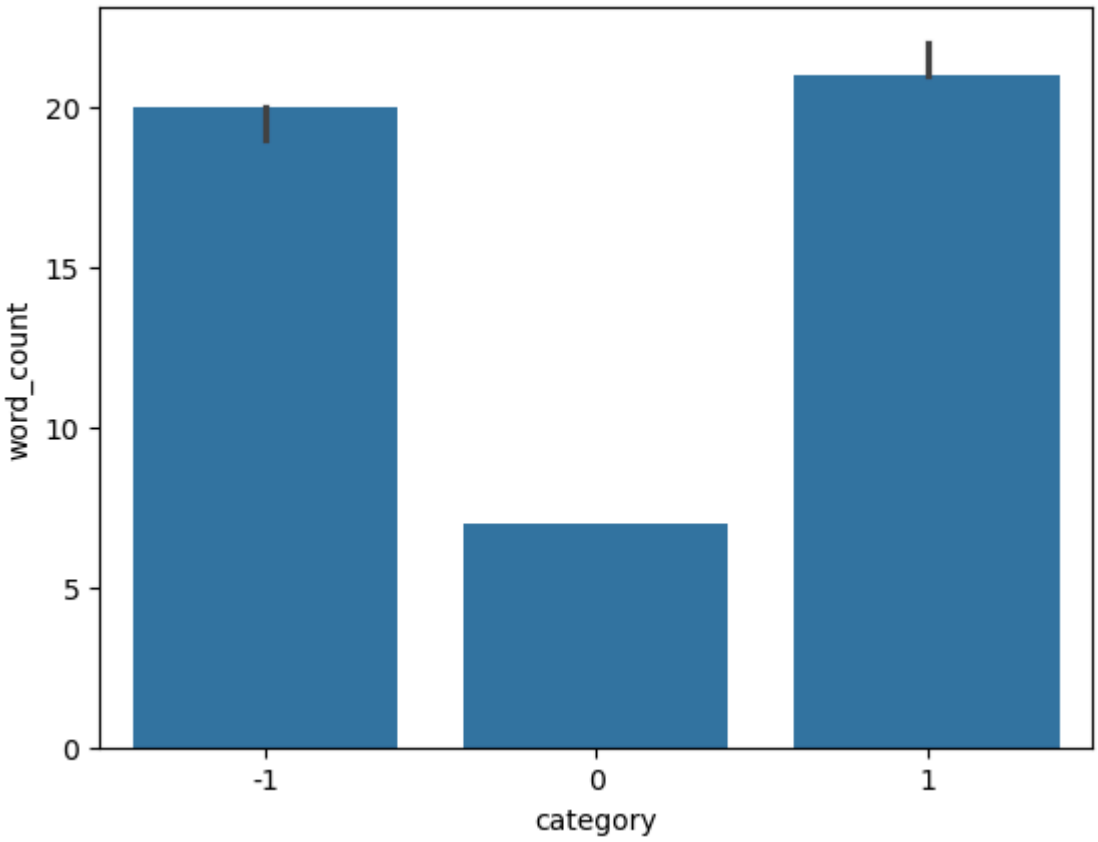




In [160…

```
sns.barplot(df,x='category',y='word_count',estimator='median')
```

Out[160… <Axes: xlabel='category', ylabel='word_count'>



In [161…

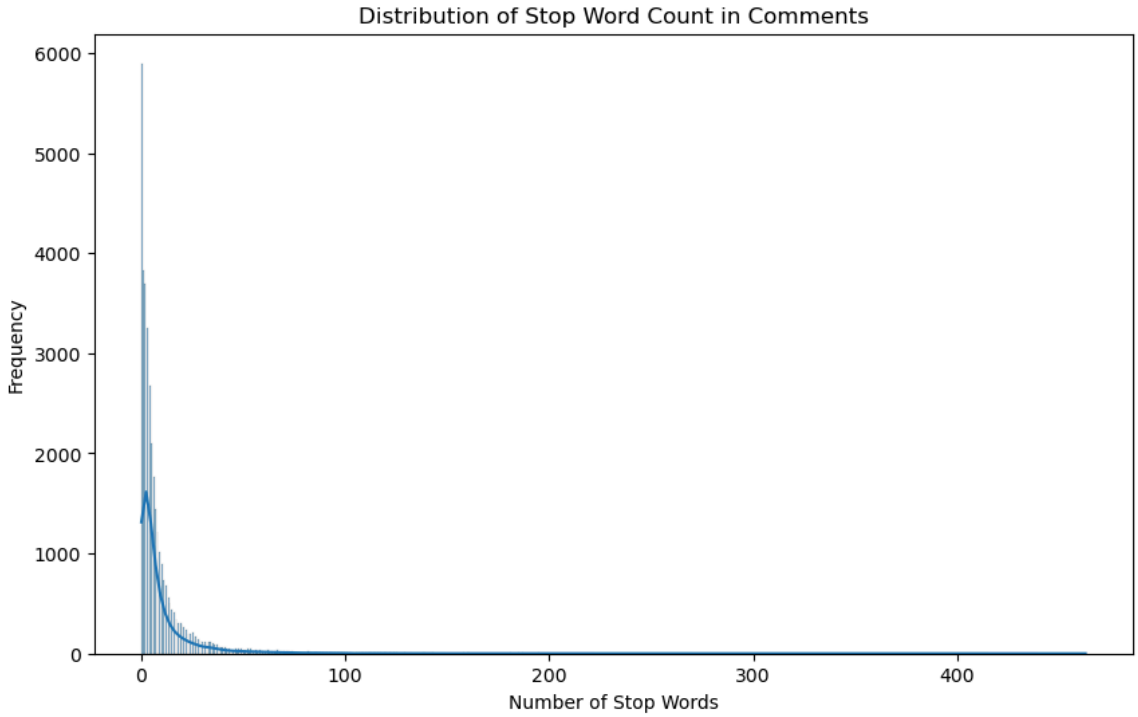
```
stop_words = set(stopwords.words('english'))
df['num_stop_words'] = df['clean_comment'].apply(lambda x: len([word for word
df.sample(5)
```

Out[161…

	clean_comment	category	word_count	num_stop_words
8116	there any difference between maratha and mahar...	0	7	4
6154	times now showing mgt 100 nda	0	6	1
18137	for hindi with karan karan johar asked for her...	1	37	9
6585	russia are india allies try hit india one side...	-1	18	7
13697	too would say that when the end that receiving...	0	10	5

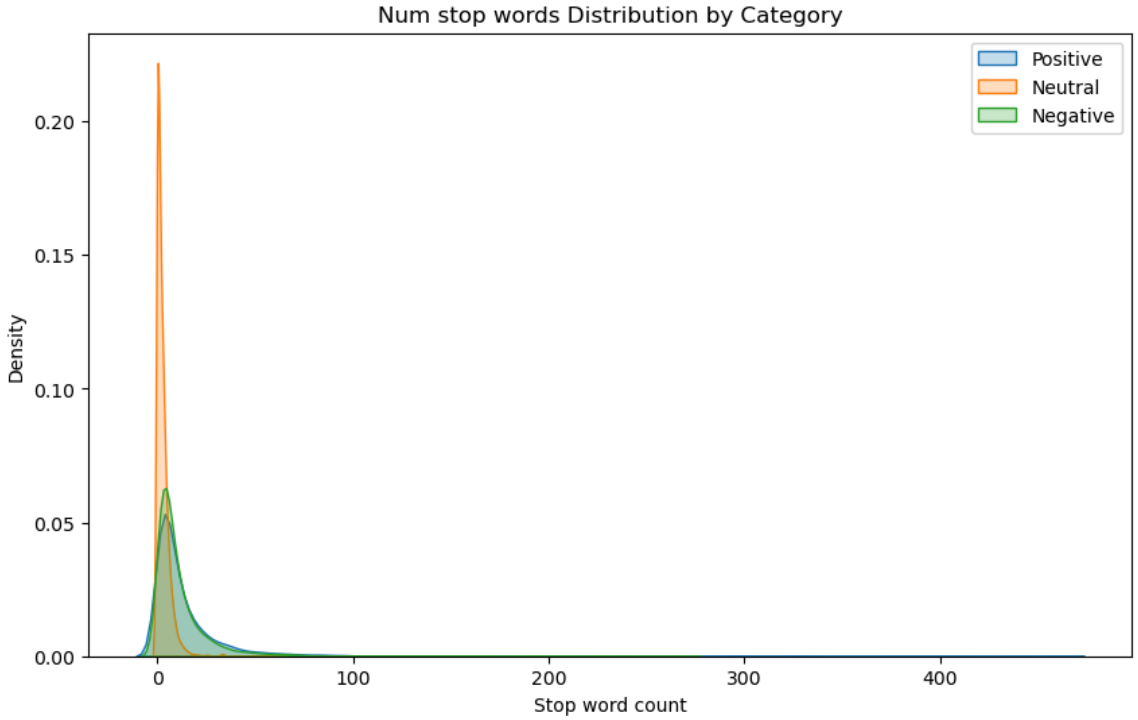
In [162…

```
plt.figure(figsize=(10, 6))
sns.histplot(df['num_stop_words'], kde=True)
plt.title('Distribution of Stop Word Count in Comments')
plt.xlabel('Number of Stop Words')
plt.ylabel('Frequency')
plt.show()
```



In [163…

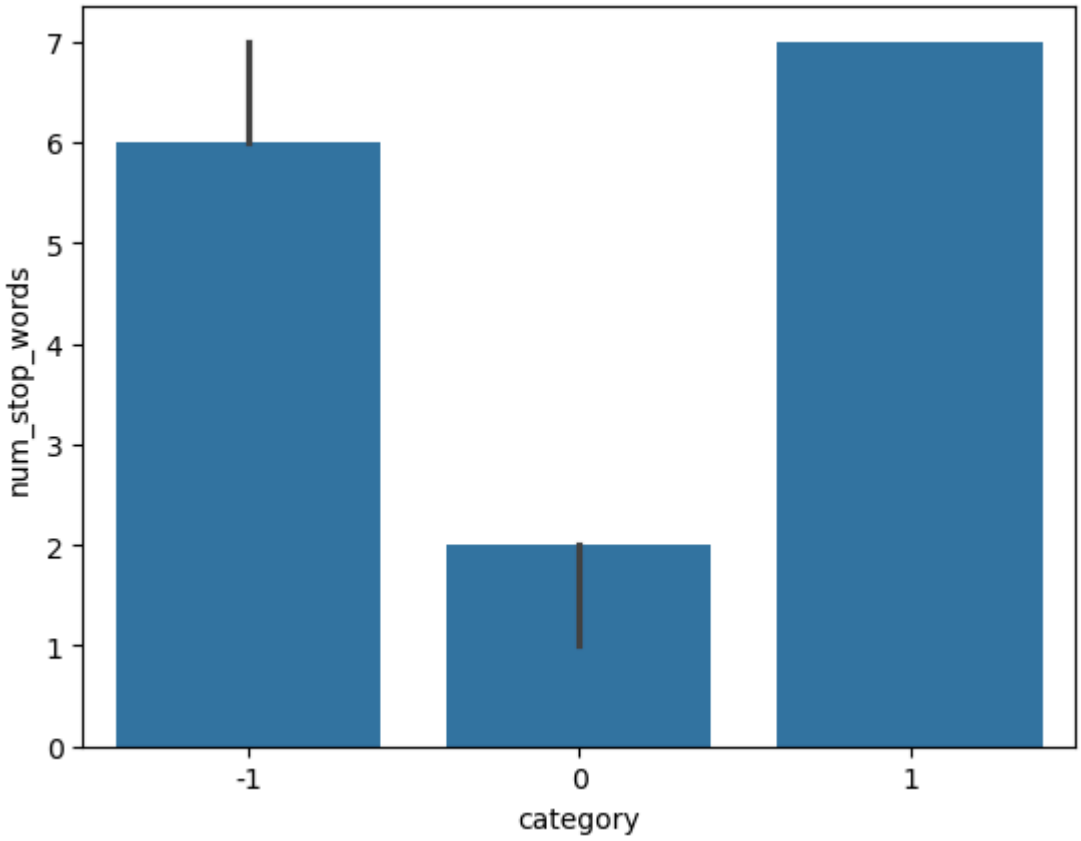
```
plt.figure(figsize=(10, 6))
sns.kdeplot(df[df['category'] == 1]['num_stop_words'], label='Positive', fill
sns.kdeplot(df[df['category'] == 0]['num_stop_words'], label='Neutral', fill=
sns.kdeplot(df[df['category'] == -1]['num_stop_words'], label='Negative', fil
plt.title('Num stop words Distribution by Category')
plt.xlabel('Stop word count')
plt.ylabel('Density')
plt.legend()
plt.show()
```



In [164…

```
sns.barplot(df,x='category',y='num_stop_words',estimator='median')
```

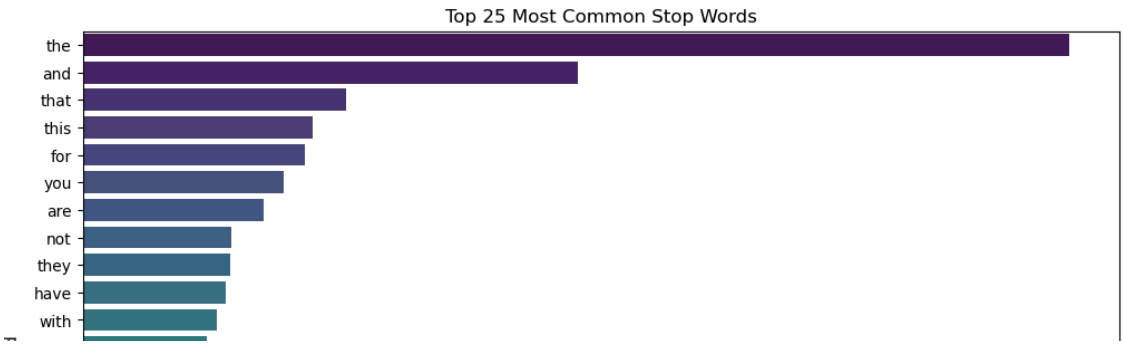
Out[164… <Axes: xlabel='category', ylabel='num_stop_words'>

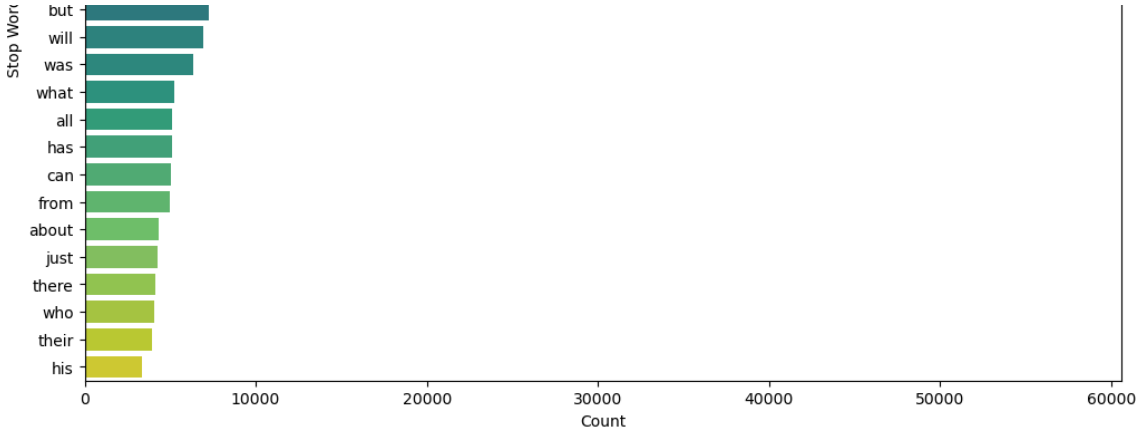


In [165…

```
all_stop_words = [word for comment in df['clean_comment'] for word in comment
most_common_stop_words = Counter(all_stop_words).most_common(25)
top_25_df = pd.DataFrame(most_common_stop_words, columns=['stop_word', 'count
plt.figure(figsize=(12, 8))
sns.barplot(data=top_25_df, x='count', y='stop_word', palette='viridis')
plt.title('Top 25 Most Common Stop Words')
plt.xlabel('Count')
plt.ylabel('Stop Word')
plt.show()
```

```
C:\Users\Suraj\AppData\Local\Temp\ipykernel_21180\3230028471.py:5: FutureWarnin
g:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same e
ffect.

  sns.barplot(data=top_25_df, x='count', y='stop_word', palette='viridis')
```





In [166…

```
df['num_chars'] = df['clean_comment'].apply(len)

df.head()
```

Out[166…

	clean_comment	category	word_count	num_stop_words	num_chars
0	family mormon have never tried explain them th...	1	39	13	259
1	buddhism has very much lot compatible with chr...	1	196	59	1268
2	seriously don say thing first all they won get...	-1	86	40	459
3	what you have learned yours and only yours wha...	0	29	15	167
4	for your own benefit you may want read living ...	1	112	45	690

In [167…

```
df['num_chars'].describe()
```

Out[167…

```
count    36793.000000
mean       181.852798
std        359.702163
min          1.000000
25%         38.000000
50%         80.000000
75%        184.000000
max       8664.000000
Name: num_chars, dtype: float64
```

In [168…

```
all_text = ' '.join(df['clean_comment'])
char_frequency = Counter(all_text)
char_frequency_df = pd.DataFrame(char_frequency.items(),
                                 columns=['character', 'frequency']).sort_val
char_frequency_df['character'].values
```

Out[168…

```
array([' ', 'e', 't', ..., '段', '她', '谁'], dtype=object)
```

In [169…

```
char_frequency_df.tail(10)
```

Out[169…

	character	frequency
1340	遥	1
1341	则	1
1342	豹	1
1343	皿	1
1344	煮	1
1345	唯	1
1346	统	1
1330	段	1
1331	她	1
1332	谁	1

In [170…

```
df['num_punctuation_chars'] = df['clean_comment'].apply(
    lambda x: sum([1 for char in x if char in '.,!?;:"\'()[]{}-'])
)
df.sample(5)
```

Out[170…

	clean_comment	category	word_count	num_stop_words	num_chars	num_pur
17317	nope wrong congress the one who played and sti...	1	73	19	439	
23760	notice trend people ordering cai png without p...	1	23	4	124	
10419	politics there are never complete victories fa...	-1	25	6	179	
15592	missus understands love gaming just just makes...	1	54	21	307	
27711	this should plastered the headlines tomorrow t...	1	12	6	76	

In [171…

```
df['num_punctuation_chars'].describe()
```

Out[171…

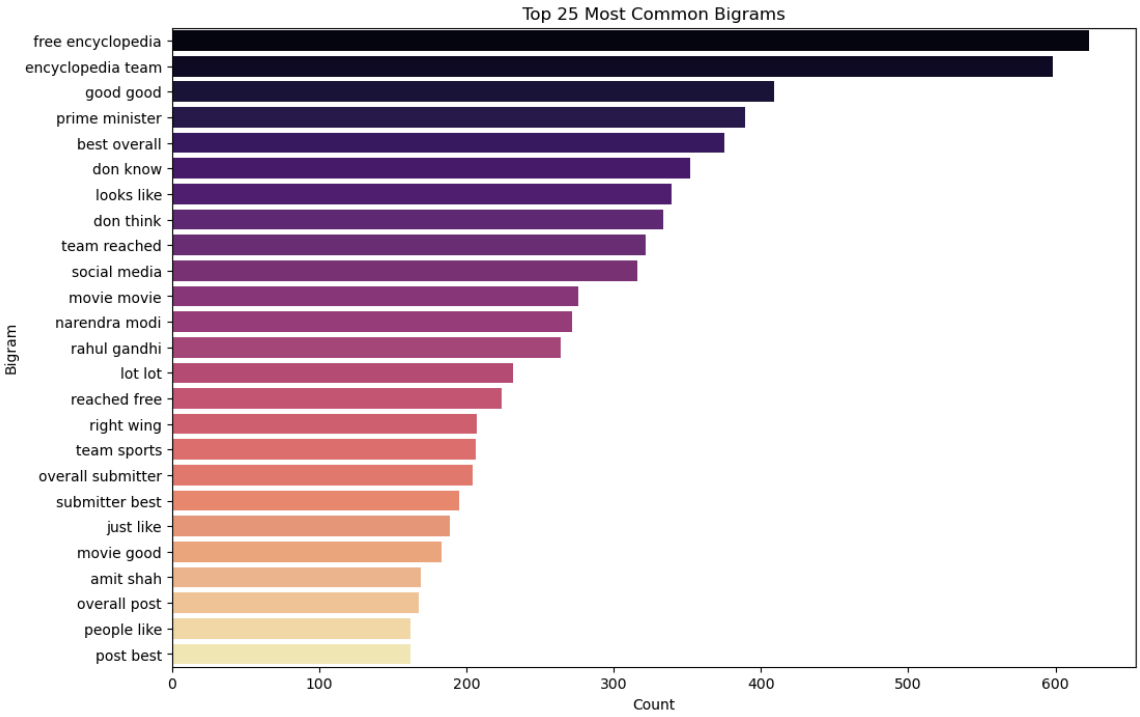
```
count    36793.0
mean         0.0
std          0.0
min          0.0
25%          0.0
```

```
50%          0.0
75%          0.0
max          0.0
Name: num_punctuation_chars, dtype: float64
```

In [172…

```
def get_top_ngrams(corpus, n=None):
    vec = CountVectorizer(ngram_range=(2, 2), stop_words='english').fit(corpu
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.
    words_freq = sorted(words_freq, key=lambda x: x[1], reverse=True)
    return words_freq[:n]
top_25_bigrams = get_top_ngrams(df['clean_comment'], 25)

top_25_bigrams_df = pd.DataFrame(top_25_bigrams, columns=['bigram', 'count'])

plt.figure(figsize=(12, 8))
sns.barplot(data=top_25_bigrams_df, x='count', y='bigram', palette='magma')
plt.title('Top 25 Most Common Bigrams')
plt.xlabel('Count')
plt.ylabel('Bigram')
plt.show()
```

```
C:\Users\Suraj\AppData\Local\Temp\ipykernel_21180\3143955522.py:13: FutureWarni
ng:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same e
ffect.

  sns.barplot(data=top_25_bigrams_df, x='count', y='bigram', palette='magma')
```

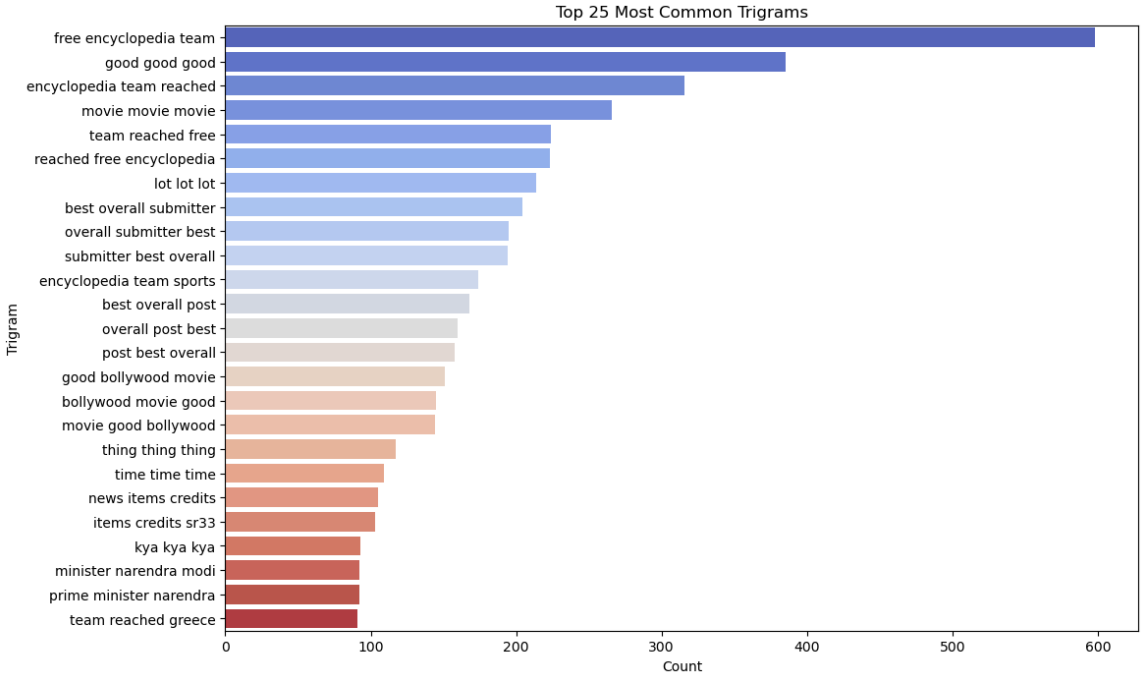


In [173…

```
def get_top_trigrams(corpus, n=None):
    vec = CountVectorizer(ngram_range=(3, 3), stop_words='english').fit(corpu
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.
    words_freq = sorted(words_freq, key=lambda x: x[1], reverse=True)
    return words_freq[:n]
```

```
top_25_trigrams = get_top_trigrams(df['clean_comment'], 25)
top_25_trigrams_df = pd.DataFrame(top_25_trigrams, columns=['trigram', 'count
plt.figure(figsize=(12, 8))
sns.barplot(data=top_25_trigrams_df, x='count', y='trigram', palette='coolwar
plt.title('Top 25 Most Common Trigrams')
plt.xlabel('Count')
plt.ylabel('Trigram')
plt.show()
```

```
C:\Users\Suraj\AppData\Local\Temp\ipykernel_21180\3703536328.py:12: FutureWarni
ng:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same e
ffect.

  sns.barplot(data=top_25_trigrams_df, x='count', y='trigram', palette='coolwar
m')
```



In [174…

```
df['clean_comment'] = df['clean_comment'].apply(lambda x: re.sub(r'[^A-Za-z0-
```

In [175…

```
all_text = ' '.join(df['clean_comment'])
char_frequency = Counter(all_text)
char_frequency_df = pd.DataFrame(char_frequency.items(),
                                 columns=['character', 'frequency']).sort_val
                                ascending=False)
char_frequency_df.head()
```

Out[175…

	character	frequency
6		1091592
12	e	666610
13	t	491287

1	a	481134
3	i	401388

In [176…

```
df.head()
```

Out[176…

	clean_comment	category	word_count	num_stop_words	num_chars	num_punctua
0	family mormon have never tried explain them th...	1	39	13	259	
1	buddhism has very much lot compatible with chr...	1	196	59	1268	
2	seriously don say thing first all they won get...	-1	86	40	459	
3	what you have learned yours and only yours wha...	0	29	15	167	
4	for your own benefit you may want read living ...	1	112	45	690	

In [177…

```
stop_words = set(stopwords.words('english')) - {'not', 'but', 'however', 'no'
df['clean_comment'] = df['clean_comment'].apply(
    lambda x: ' '.join([word for word in x.split() if word.lower() not in sto
)
```

In [178…

```
df.head()
```

Out[178…

	clean_comment	category	word_count	num_stop_words	num_chars	num_punctua
0	family mormon never tried explain still stare ...	1	39	13	259	
1	buddhism much lot compatible christianity espe...	1	196	59	1268	
2	seriously say thing first get complex explain ...	-1	86	40	459	
	learned want					

3	teach different focus goal not wr...	0	29	15	167
4	benefit may want read living buddha living chr...	1	112	45	690

In [179…

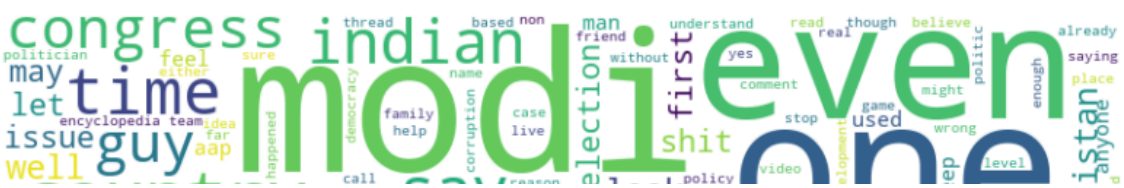
```
lemmatizer = WordNetLemmatizer()
df['clean_comment'] = df['clean_comment'].apply(
    lambda x: ' '.join([lemmatizer.lemmatize(word) for word in x.split()])
)
df.head()
```

Out[179…

	clean_comment	category	word_count	num_stop_words	num_chars	num_punctua
0	family mormon never tried explain still stare ...	1	39	13	259	
1	buddhism much lot compatible christianity espe...	1	196	59	1268	
2	seriously say thing first get complex explain ...	-1	86	40	459	
3	learned want teach different focus goal not wr...	0	29	15	167	
4	benefit may want read living buddha living chr...	1	112	45	690	

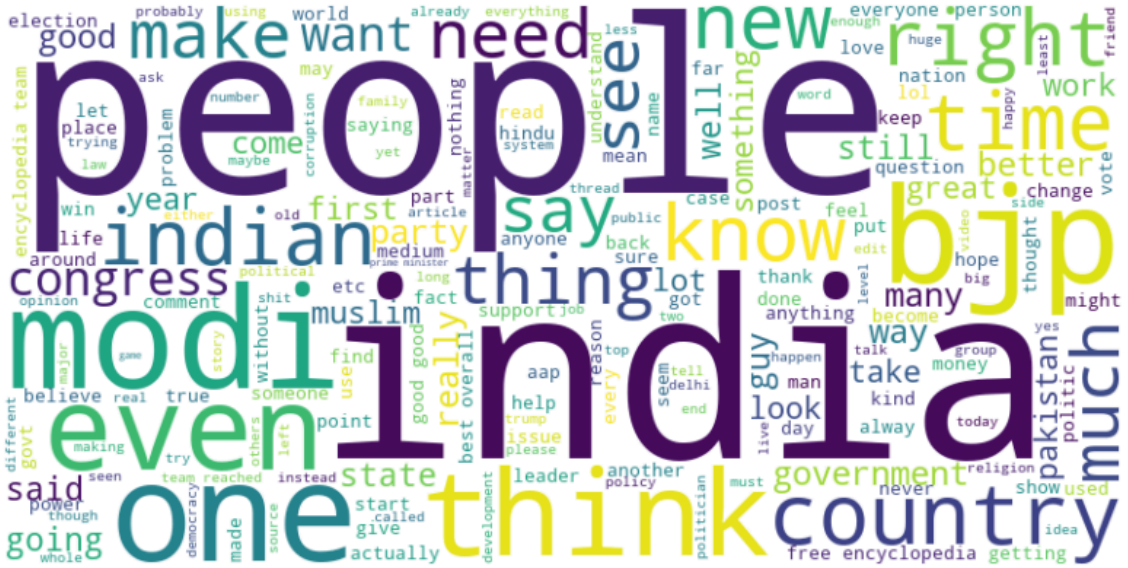
In [180…

```
def plot_word_cloud(text):
    wordcloud = WordCloud(width=800, height=400, background_color='white').ge
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis("off")
    plt.show()

plot_word_cloud(df['clean_comment'])
```



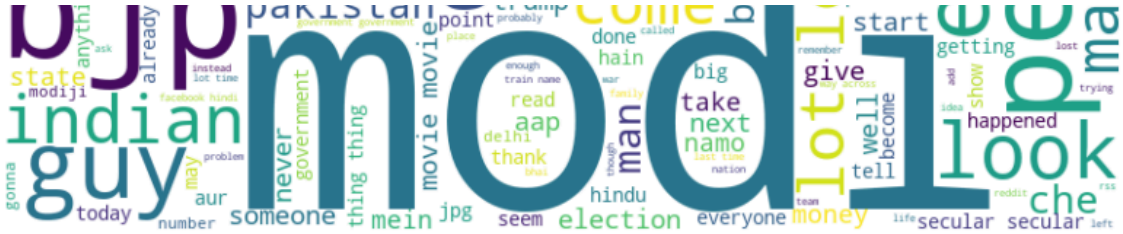
In [181…

```
def plot_word_cloud(text):
    wordcloud = WordCloud(width=800, height=400, background_color='white').ge
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis("off")
    plt.show()

plot_word_cloud(df[df['category'] == 1]['clean_comment'])
```



In [182…

```
def plot_word_cloud(text):
    wordcloud = WordCloud(width=800, height=400, background_color='white').ge
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis("off")
    plt.show()
plot_word_cloud(df[df['category'] == 0]['clean_comment'])
```

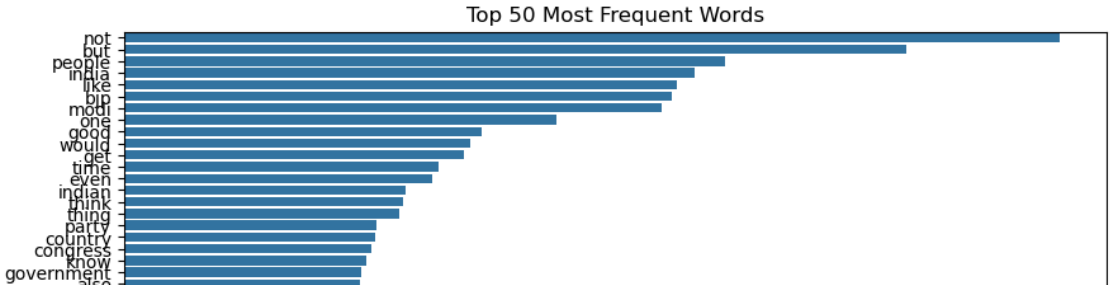


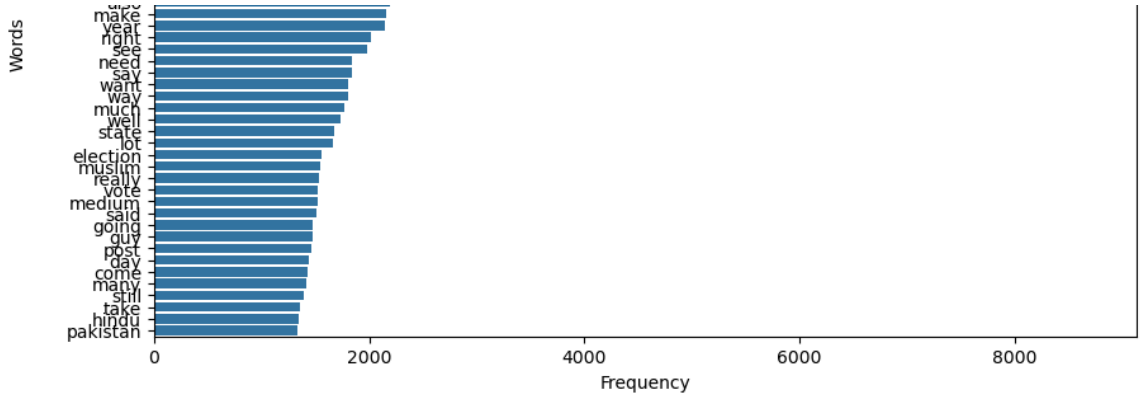
In [183…

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt

def plot_word_cloud(text):
    wordcloud = WordCloud(width=800, height=400, background_color='white').ge
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis("off")
    plt.show()

plot_word_cloud(df[df['category'] == -1]['clean_comment'])
```

In [184…

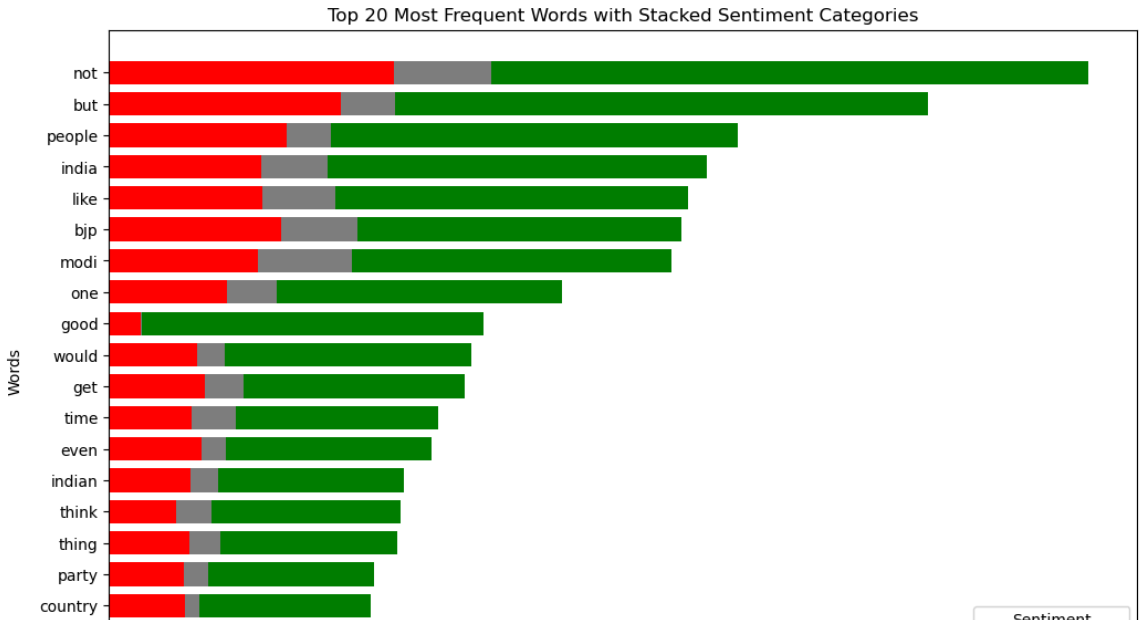
```
def plot_top_n_words(df, n=20):
    words = ' '.join(df['clean_comment']).split()
    counter = Counter(words)
    most_common_words = counter.most_common(n)
    words, counts = zip(*most_common_words)
    plt.figure(figsize=(10, 6))
    sns.barplot(x=list(counts), y=list(words))
    plt.title(f'Top {n} Most Frequent Words')
    plt.xlabel('Frequency')
    plt.ylabel('Words')
    plt.show()
plot_top_n_words(df, n=50)
```

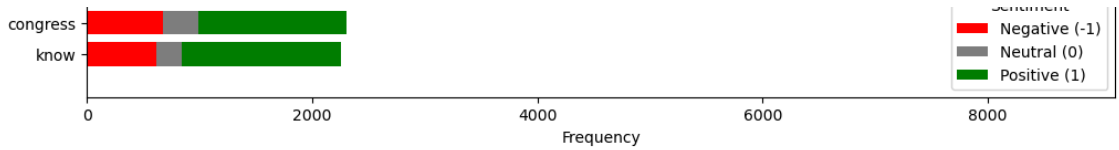




In [185…

```
def plot_top_n_words_by_category(df, n=20, start=0):
    word_category_counts = {}
    for idx, row in df.iterrows():
        words = row['clean_comment'].split()
        category = row['category']
        for word in words:
            if word not in word_category_counts:
                word_category_counts[word] = { -1: 0, 0: 0, 1: 0 }
            word_category_counts[word][category] += 1
    total_word_counts = {word: sum(counts.values()) for word, counts in word_
    most_common_words = sorted(total_word_counts.items(), key=lambda x: x[1],
    top_words = [word for word, _ in most_common_words]
    word_labels = top_words
    negative_counts = [word_category_counts[word][-1] for word in top_words]
    neutral_counts = [word_category_counts[word][0] for word in top_words]
    positive_counts = [word_category_counts[word][1] for word in top_words]
    plt.figure(figsize=(12, 8))
    bar_width = 0.75
    plt.barh(word_labels, negative_counts, color='red', label='Negative (-1)'
    plt.barh(word_labels, neutral_counts, left=negative_counts, color='gray',
    plt.barh(word_labels, positive_counts, left=[i+j for i,j in zip(negative_

    plt.xlabel('Frequency')
    plt.ylabel('Words')
    plt.title(f'Top {n} Most Frequent Words with Stacked Sentiment Categories
    plt.legend(title='Sentiment', loc='lower right')
    plt.gca().invert_yaxis()
    plt.show()
plot_top_n_words_by_category(df, n=20)
```




congress
know
0
2000
4000
6000
8000
Frequency
Negative (-1)
Neutral (0)
Positive (1)