In [ ]:

```
!pip install pyspellchecker
```

```
Collecting pyspellchecker
  Downloading pyspellchecker-0.8.1-py3-none-any.whl (6.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.8/6.8 MB 11.3 MB/s eta 0:00:00
Installing collected packages: pyspellchecker
Successfully installed pyspellchecker-0.8.1
```

In [ ]:

```
import nltk
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
import re
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import mean_squared_error, r2_score, cohen_kappa_score, classification_report
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor
from spellchecker import SpellChecker
import string
import seaborn as sns
```

In [ ]:

```
from nltk.tokenize import word_tokenize
from sklearn.metrics import classification_report
from sklearn import svm
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report
from sklearn.preprocessing import MinMaxScaler
from matplotlib import rcParams
```

In [ ]:

```
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Out[ ]:

```
True
```

Loading Data

In [ ]:

```
# Load the dataset
dataframe = pd.read_csv('training.tsv', encoding='latin-1', sep='\t')
```

In [ ]:

```
dataframe.describe()
```

Out[ ]:

	essay_id	essay_set	rater1_domain1	rater2_domain1	rater3_domain1	domain1_score	rater1_domain2	rater2_domain2	domai
count	12976.000000	12976.000000	12976.000000	12976.000000	128.000000	12976.000000	1800.000000	1800.000000	180
mean	10295.395808	4.179485	4.127158	4.137408	37.828125	6.800247	3.333889	3.330556	
std	6309.074105	2.136913	4.212544	4.264330	5.240829	8.970705	0.729103	0.726807	
min	1.000000	1.000000	0.000000	0.000000	20.000000	0.000000	1.000000	1.000000	
25%	4438.750000	2.000000	2.000000	2.000000	36.000000	2.000000	3.000000	3.000000	
50%	10044.500000	4.000000	3.000000	3.000000	40.000000	3.000000	3.000000	3.000000	
75%	15681.250000	6.000000	4.000000	4.000000	40.000000	8.000000	4.000000	4.000000	
max	21633.000000	8.000000	30.000000	30.000000	50.000000	60.000000	4.000000	4.000000	

8 rows × 27 columns

In [ ]:

```
dataframe.head()
```

Out[ ]:

	essay_id	essay_set	essay	rater1_domain1	rater2_domain1	rater3_domain1	domain1_score	rater1_domain2	rater2_domain2	dom
0	1	1	Dear local newspaper, I think effects computer...	4	4	NaN	8	NaN	NaN	
1	2	1	Dear @CAPS1 @CAPS2, I believe that using compu...	5	4	NaN	9	NaN	NaN	
2	3	1	Dear, @CAPS1 @CAPS2 @CAPS3 More and more peopl...	4	3	NaN	7	NaN	NaN	
3	4	1	Dear Local Newspaper, @CAPS1 I have found that...	5	5	NaN	10	NaN	NaN	
4	5	1	Dear @LOCATION1, I know having computers has a...	4	4	NaN	8	NaN	NaN	

5 rows × 28 columns

Methods

In [ ]:

```python
# selecting which set to be used 1-8
# in order to combine them all assign set number to 9
def select_set(dataframe,setNumber):
    if setNumber == 9:
        dataframe2 = dataframe[dataframe.essay_set ==1]
        texts = dataframe2['essay']
        scores = dataframe2['domain1_score']
        scores = scores.apply(lambda x: (x*3)/scores.max())
        for i in range(1,9):
            dataframe2 = dataframe[dataframe.essay_set == i]
            texts = texts.append(dataframe2['essay'])
            s = dataframe2['domain1_score']
            s = s.apply(lambda x: (x*3)/s.max())
            scores = scores.append(s)
    else:
        dataframe2 = dataframe[dataframe.essay_set ==setNumber]
        texts = dataframe2['essay']
        scores = dataframe2['domain1_score']
        scores = scores.apply(lambda x: (x*3)/scores.max())
    return texts, scores
```

In [ ]:

```
# get histogram plot of scores and average score
def get_hist_avg(scores,bin_count):
    print(sum(scores)/len(scores))
    scores.hist(bins=bin_count)
```

In [ ]:

```
#average word length for a text
def avg_word_len(text):
    clean_essay = re.sub(r'\W', ' ', text)
    words = nltk.word_tokenize(clean_essay)
    total = 0
    for word in words:
        total = total + len(word)
    average = total / len(words)

    return average

# word count in a given text
def word_count(text):
    clean_essay = re.sub(r'\W', ' ', text)
    return len(nltk.word_tokenize(clean_essay))

# char count in a given text
def char_count(text):
    return len(re.sub(r'\s', '', str(text).lower()))

# sentence count in a given text
def sent_count(text):
    return len(nltk.sent_tokenize(text))
```

```python
#tokenization of texts to sentences
def sent_tokenize(text):
    stripped_essay = text.strip()

    tokenizer = nltk.data.load('tokenizers/punkt/english.pickle')
    raw_sentences = tokenizer.tokenize(stripped_essay)

    tokenized_sentences = []
    for raw_sentence in raw_sentences:
        if len(raw_sentence) > 0:
            clean_sentence = re.sub("[^a-zA-Z0-9]"," ", raw_sentence)
            tokens = nltk.word_tokenize(clean_sentence)
            tokenized_sentences.append(tokens)
    return tokenized_sentences


# lemma, noun, adjective, verb, adverb count for a given text

def count_lemmas(text):

    noun_count = 0
    adj_count = 0
    verb_count = 0
    adv_count = 0
    lemmas = []
    lemmatizer = WordNetLemmatizer()
    tokenized_sentences = sent_tokenize(text)

    for sentence in tokenized_sentences:
        tagged_tokens = nltk.pos_tag(sentence)

        for token_tuple in tagged_tokens:
            pos_tag = token_tuple[1]

            if pos_tag.startswith('N'):
                noun_count += 1
                pos = wordnet.NOUN
                lemmas.append(lemmatizer.lemmatize(token_tuple[0], pos))
            elif pos_tag.startswith('J'):
                adj_count += 1
                pos = wordnet.ADJ
                lemmas.append(lemmatizer.lemmatize(token_tuple[0], pos))
            elif pos_tag.startswith('V'):
                verb_count += 1
                pos = wordnet.VERB
                lemmas.append(lemmatizer.lemmatize(token_tuple[0], pos))
            elif pos_tag.startswith('R'):
                adv_count += 1
                pos = wordnet.ADV
                lemmas.append(lemmatizer.lemmatize(token_tuple[0], pos))
            else:
                pos = wordnet.NOUN
                lemmas.append(lemmatizer.lemmatize(token_tuple[0], pos))

    lemma_count = len(set(lemmas))

    return noun_count, adj_count, verb_count, adv_count, lemma_count
```

In [ ]:

```
def token_word(text):
    text = "".join([ch.lower() for ch in text if ch not in string.punctuation])
    tokens = nltk.word_tokenize(text)
    return tokens
```

In [ ]:

```
def misspell_count(text):
    spell = SpellChecker()
    # find those words that may be misspelled
    misspelled = spell.unknown(token_word(text))
    #print(misspelled)
    return len(misspelled)
```

In [ ]:

```
def create_features(texts):
    data = pd.DataFrame(columns=('Average_Word_Length','Sentence_Count','Word_Count',
                                'Character_Count', 'Noun_Count','Adjective_Count',
                                'Verb_Count', 'Adverb_Count', 'Lemma_Count' , 'Misspell_Count'
                                 ))

    data['Average_Word_Length'] = texts.apply(avg_word_len)
    data['Sentence_Count'] = texts.apply(sent_count)
    data['Word_Count'] = texts.apply(word_count)
    data['Character_Count'] = texts.apply(char_count)
    temp=texts.apply(count_lemmas)
    noun_count,adj_count,verb_count,adverb_count,lemma_count = zip(*temp)
    data['Noun_Count'] = noun_count
    data['Adjective_Count'] = adj_count
    data['Verb_Count'] = verb_count
    data['Adverb_Count'] = adverb_count
    data['Lemma_Count'] = lemma_count
    data['Misspell_Count'] = texts.apply(misspell_count)
    return data
```

In [ ]:

```
def data_prepare(texts,scores):
    #create features from the texts and clean non graded essays
    data = create_features(texts)
    data.describe()
    t1=np.where(np.asanyarray(np.isnan(scores)))
    scores=scores.drop(scores.index[t1])
    data=data.drop(scores.index[t1])

    #scaler = MinMaxScaler()
    #data = scaler.fit_transform(data)

    #train test split
    X_train, X_test, y_train, y_test = train_test_split(data, scores, test_size = 0.3)

    #checking is there any nan cells
    print(np.any(np.isnan(scores)))
    print(np.all(np.isfinite(scores)))
    return X_train, X_test, y_train, y_test, data
```

In [ ]:

```
def lin_regression(X_train,y_train,X_test,y_test):
    regr = LinearRegression()
    regr.fit(X_train, y_train)
    y_pred = regr.predict(X_test)
    # The mean squared error
    mse=mean_squared_error(y_test, y_pred)
    mse_per= 100*mse/3
    print("Mean squared error: {}".format(mse))
    print("Mean squared error in percentage: {}".format(mse_per))
    #explained variance score
    print('Variance score: {}'.format(regr.score(X_test, y_test)))
```

In [ ]:

```
def adaBoost_reg(X_train,y_train,X_test,y_test):
    #regr = RandomForestRegressor(max_depth=2, n_estimators=300)
    #regr = SVR(gamma='scale', C=1, kernel='linear')
    regr = AdaBoostRegressor()
    regr.fit(X_train, y_train)
    y_pred = regr.predict(X_test)
    # The mean squared error
    mse=mean_squared_error(y_test, y_pred)
    mse_per= 100*mse/3
    print("Mean squared error: {}".format(mse))
    print("Mean squared error in percentage: {}".format(mse_per))
    #explained variance score
    print('Variance score: {}'.format(regr.score(X_test, y_test)))

    feature_importance = regr.feature_importances_

    # make importances relative to max importance
    feature_importance = 100.0 * (feature_importance / feature_importance.max())
    feature_names = list(('Average_Word_Length','Sentence_Count','Word_Count',
                               'Character_Count', 'Noun_Count','Adjective_Count',
                               'Verb_Count', 'Adverb_Count', 'Lemma_Count' ,'Misspell_Count'
                                ))
    feature_names = np.asarray(feature_names)
    sorted_idx = np.argsort(feature_importance)
    pos = np.arange(sorted_idx.shape[0]) + .5
    plt.subplot(1, 2, 2)
    plt.barh(pos, feature_importance[sorted_idx], align='center')
    plt.yticks(pos, feature_names[sorted_idx])
    plt.xlabel('Relative Importance')
    plt.title('Variable Importance')
    plt.show()
```

In [ ]:

```
# convert numerical scores to labels
# (0-1.5) bad (1.5-2.3) average (2.3-3) good
# bad:    '0'
# average '1'
# good    '2'
def convert_scores(scores):
    def mapping(x):
        if x < np.percentile(scores,25):
            return 0
        elif x < np.percentile(scores,75):
            return 1
        else:
            return 2
    return scores.apply(mapping)
```

In [ ]:

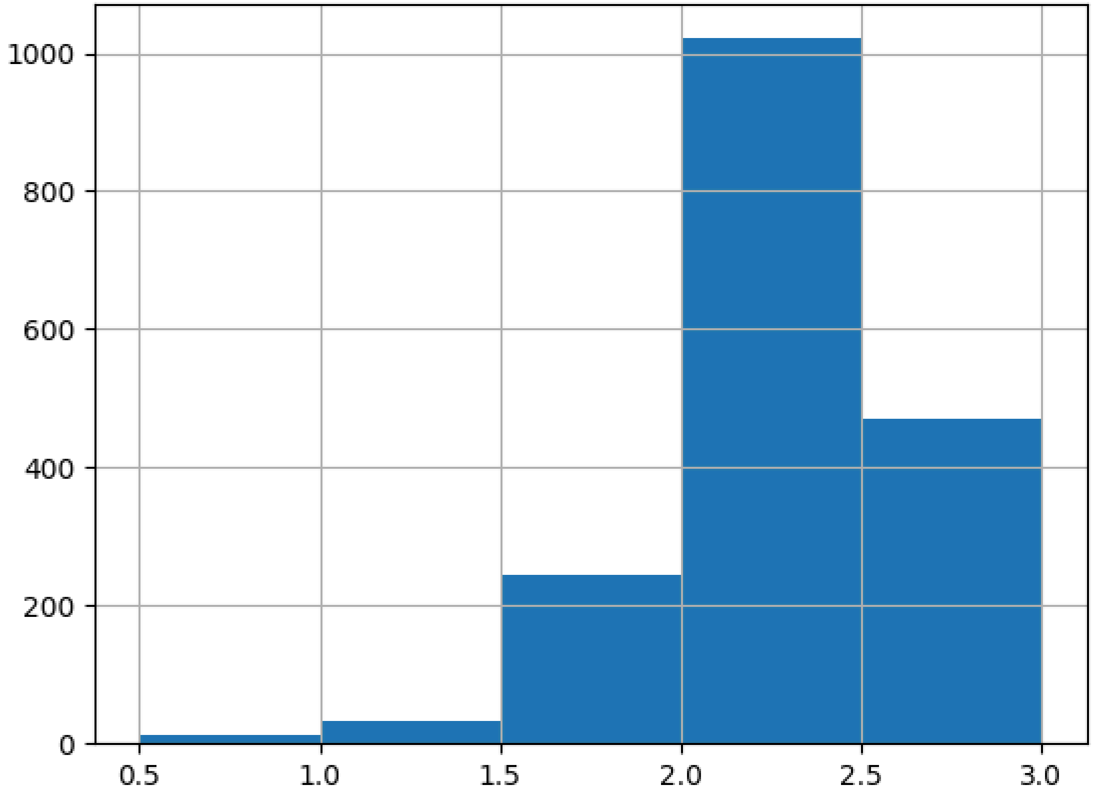
```
# selecting which set to be used 1-8
# in order to combine them all assign set number to 9
def select_set_classification(dataframe,setNumber):
    if setNumber == 9:
        dataframe2 = dataframe[dataframe.essay_set ==1]
        texts = dataframe2['essay']
        scores = dataframe2['domain1_score']
        scores = scores.apply(lambda x: (x*3)/scores.max())
        scores = convert_scores(scores)
        for i in range(1,9):
            dataframe2 = dataframe[dataframe.essay_set == i]
            texts = texts.append(dataframe2['essay'])
            s = dataframe2['domain1_score']
            s = s.apply(lambda x: (x*3)/s.max())
            s = convert_scores(s)
            scores = scores.append(s)
    else:
        dataframe2 = dataframe[dataframe.essay_set ==setNumber]
        texts = dataframe2['essay']
        scores = dataframe2['domain1_score']
        scores = scores.apply(lambda x: (x*3)/scores.max())
        scores = convert_scores(scores)
    return texts, scores
```

Dataset selection

In [19]:

```
# 1-8
# 9:all sets combined
texts, scores = select_set(dataframe,1)
get_hist_avg(scores,5)
X_train, X_test, y_train, y_test, data = data_prepare(texts,scores)
```

```
2.132080762759394
False
True
```

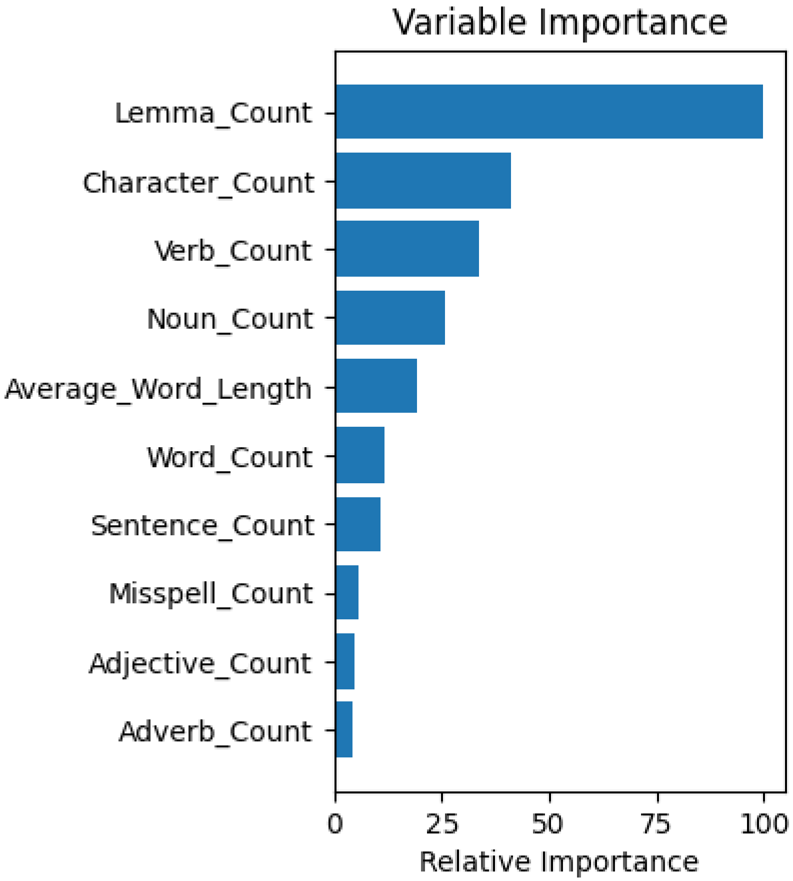


Regression Analysis

In [20]:

```
print('Testing for Linear Regression \n')
lin_regression(X_train,y_train,X_test,y_test)
print('Testing for Adaboost Regression \n')
adaBoost_reg(X_train,y_train,X_test,y_test)
```

```
Testing for Linear Regression

Mean squared error: 0.04501437384270856
Mean squared error in percentage: 1.5004791280902854
Variance score: 0.7134352182307053
Testing for Adaboost Regression

Mean squared error: 0.04450911613048784
Mean squared error in percentage: 1.4836372043495947
Variance score: 0.7166517256188947
```



Dataset Selection 2

In [21]:

```
# 1-8
# 9:all sets combined
```

```
texts, scores = select_set_classification(dataframe,1)
X_train, X_test, y_train, y_test, data = data_prepare(texts,scores)
```

```
False
True
```

Classification Analysis

In [22]:

```
a=[0.1,1,10,100,500,1000]
for b in a:
    clf = svm.SVC(C=b, gamma=0.00001)
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    print (b)
    print (clf.score(X_test,y_test))
    print (np.mean(cross_val_score(clf, X_train, y_train, cv=3)))
```

```
0.1
0.7831775700934579
0.7764423076923078
1
0.788785046728972
0.78125
10
0.794392523364486
0.7948717948717948
100
0.7962616822429907
0.794871794871795
500
0.7925233644859813
0.7892628205128206
1000
0.7850467289719626
0.782852564102564
```

In [23]:

```
clf = svm.SVC(C=100, gamma=0.00001)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print('Cohen’s kappa score: {}'.format(cohen_kappa_score(y_test,y_pred)))
```

```
Cohen’s kappa score: 0.6327261963244278
```

In [24]:

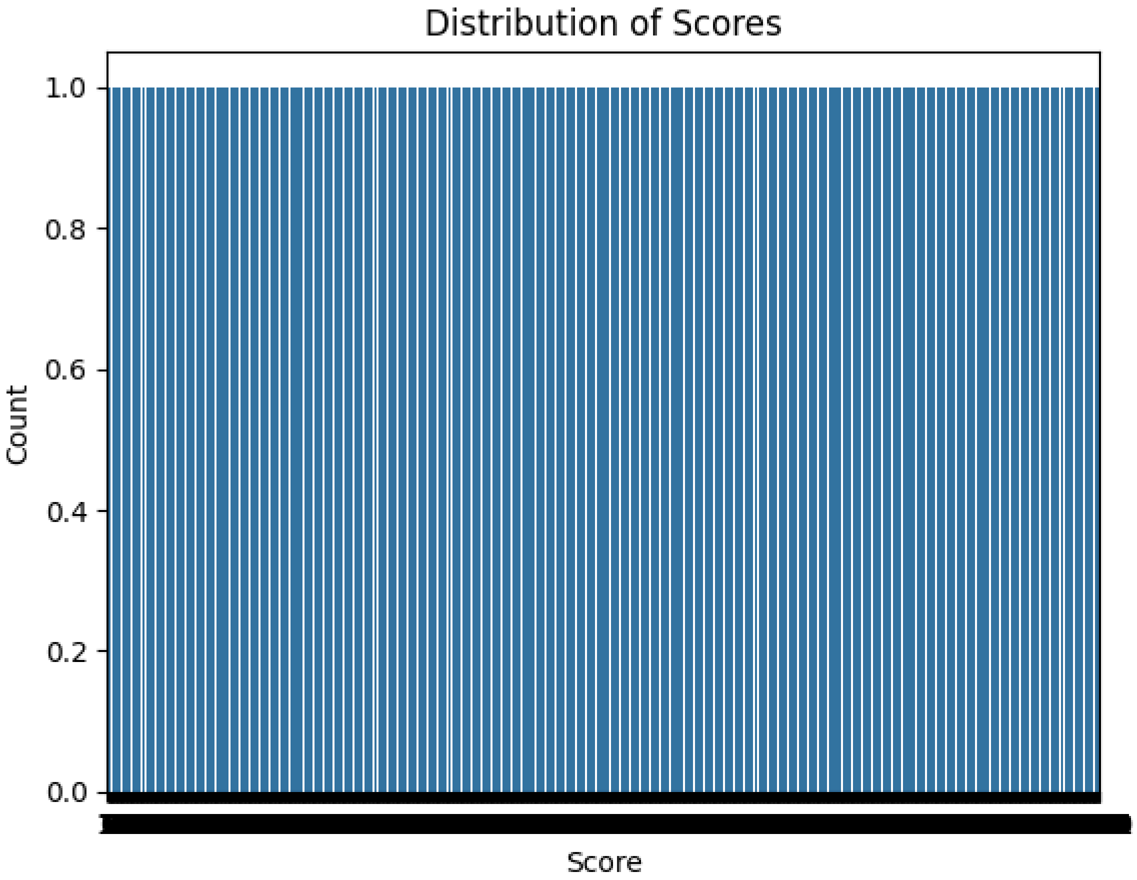
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.79      0.72      0.75        79
           1       0.80      0.87      0.83       307
           2       0.80      0.69      0.74       149

    accuracy                           0.80       535
   macro avg       0.80      0.76      0.78       535
weighted avg       0.80      0.80      0.79       535
```

Data Analysis

In [25]:

```
# Count plot for scores
sns.countplot(scores)
plt.title("Distribution of Scores")
plt.xlabel("Score")
plt.ylabel("Count")
plt.show()
```



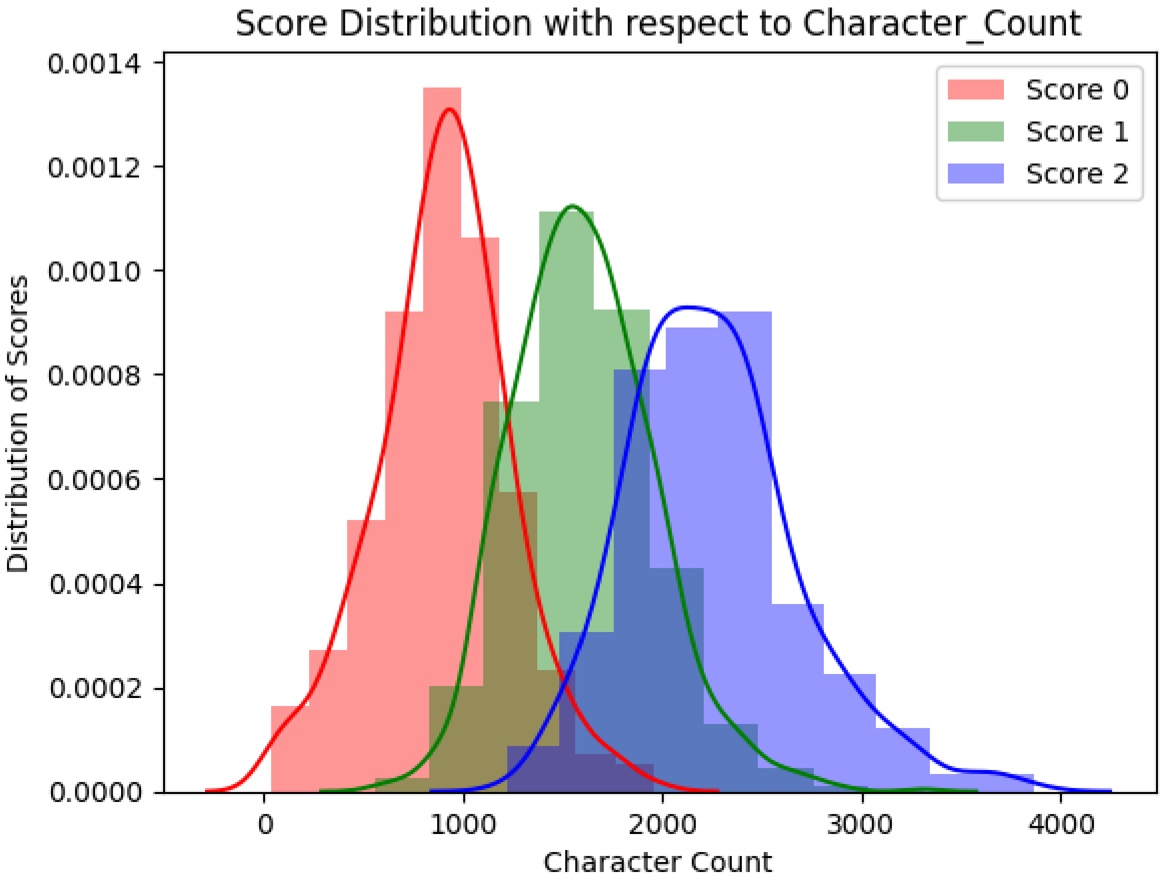
In [26]:

```python
# Plotting score distributions with respect to different features
def plot_score_distribution_by_feature(data, scores, feature_name, xlabel):
    zero = data[(data[feature_name] > 0) & (scores == 0)]
    one = data[(data[feature_name] > 0) & (scores == 1)]
    two = data[(data[feature_name] > 0) & (scores == 2)]

    sns.distplot(zero[feature_name], bins=10, color='r', label='Score 0')
    sns.distplot(one[feature_name], bins=10, color='g', label='Score 1')
    sns.distplot(two[feature_name], bins=10, color='b', label='Score 2')

    plt.title(f"Score Distribution with respect to {feature_name}")
    plt.xlabel(xlabel)
    plt.ylabel("Distribution of Scores")
    plt.legend()
    plt.show()
```

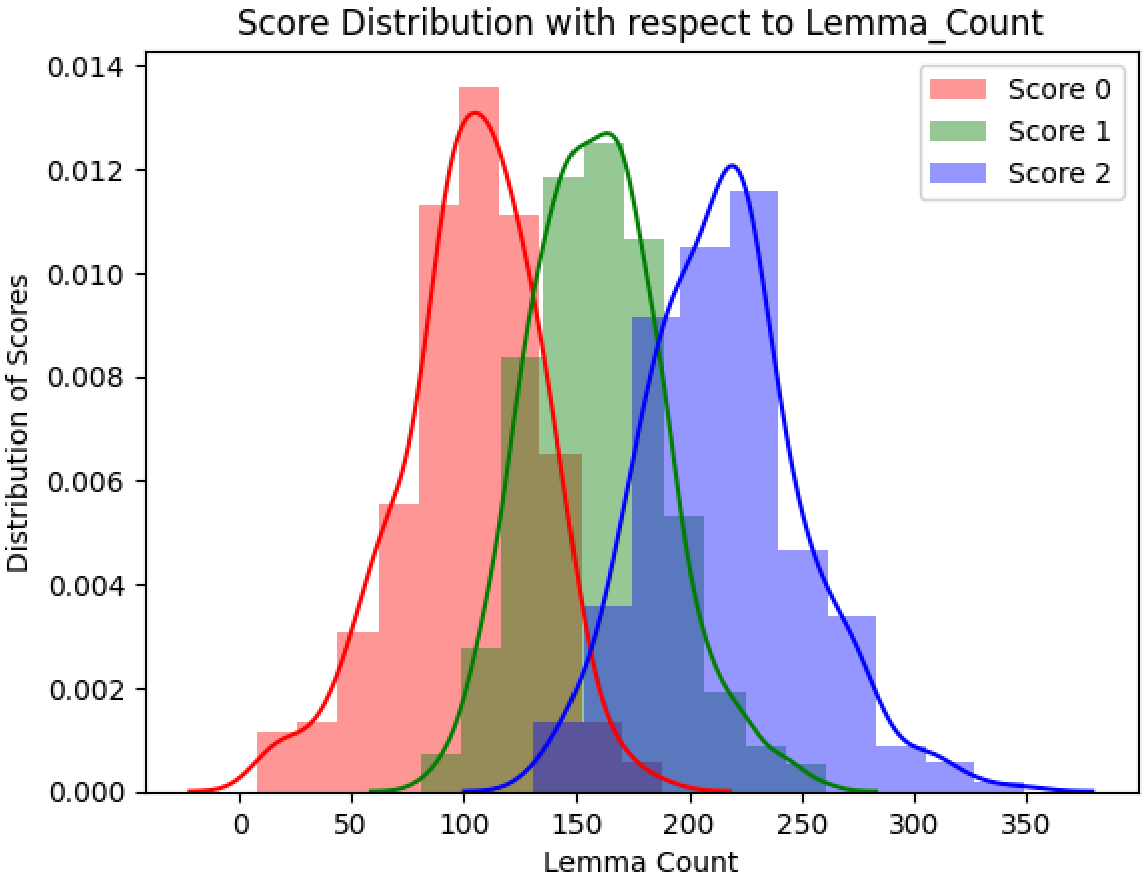
In [27]:

```
# Plot score distributions for different features
plot_score_distribution_by_feature(data, scores, "Character_Count", "Character Count")
plot_score_distribution_by_feature(data, scores, "Lemma_Count", "Lemma Count")
```

```
<ipython-input-26-457e4d892a7d>:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(zero[feature_name], bins=10, color='r', label='Score 0')
<ipython-input-26-457e4d892a7d>:8: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(one[feature_name], bins=10, color='g', label='Score 1')
<ipython-input-26-457e4d892a7d>:9: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(two[feature_name], bins=10, color='b', label='Score 2')
```


Score Distribution with respect to Character_Count
Score 0
Score 1
Score 2
Distribution of Scores
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
Character Count
0
1000
2000
3000
4000

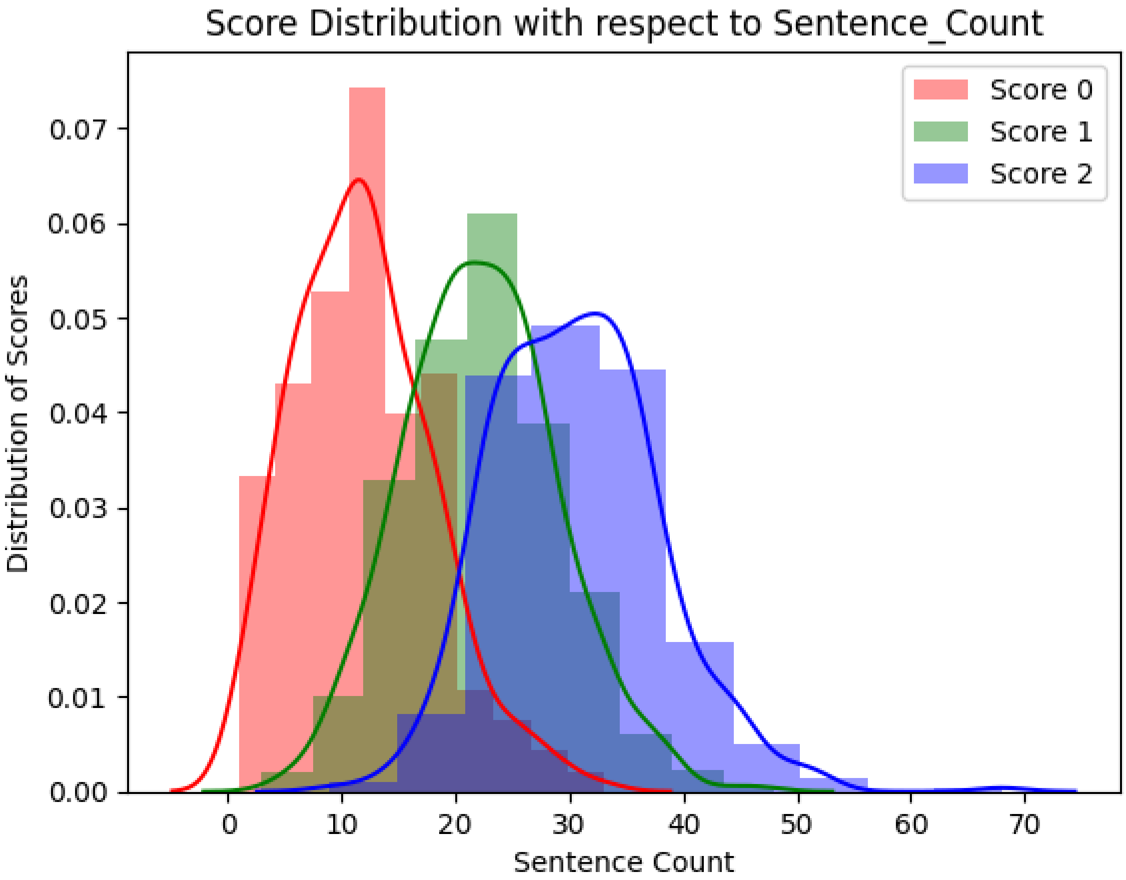
```
<ipython-input-26-457e4d892a7d>:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(zero[feature_name], bins=10, color='r', label='Score 0')
<ipython-input-26-457e4d892a7d>:8: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(one[feature_name], bins=10, color='g', label='Score 1')
<ipython-input-26-457e4d892a7d>:9: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(two[feature_name], bins=10, color='b', label='Score 2')
```



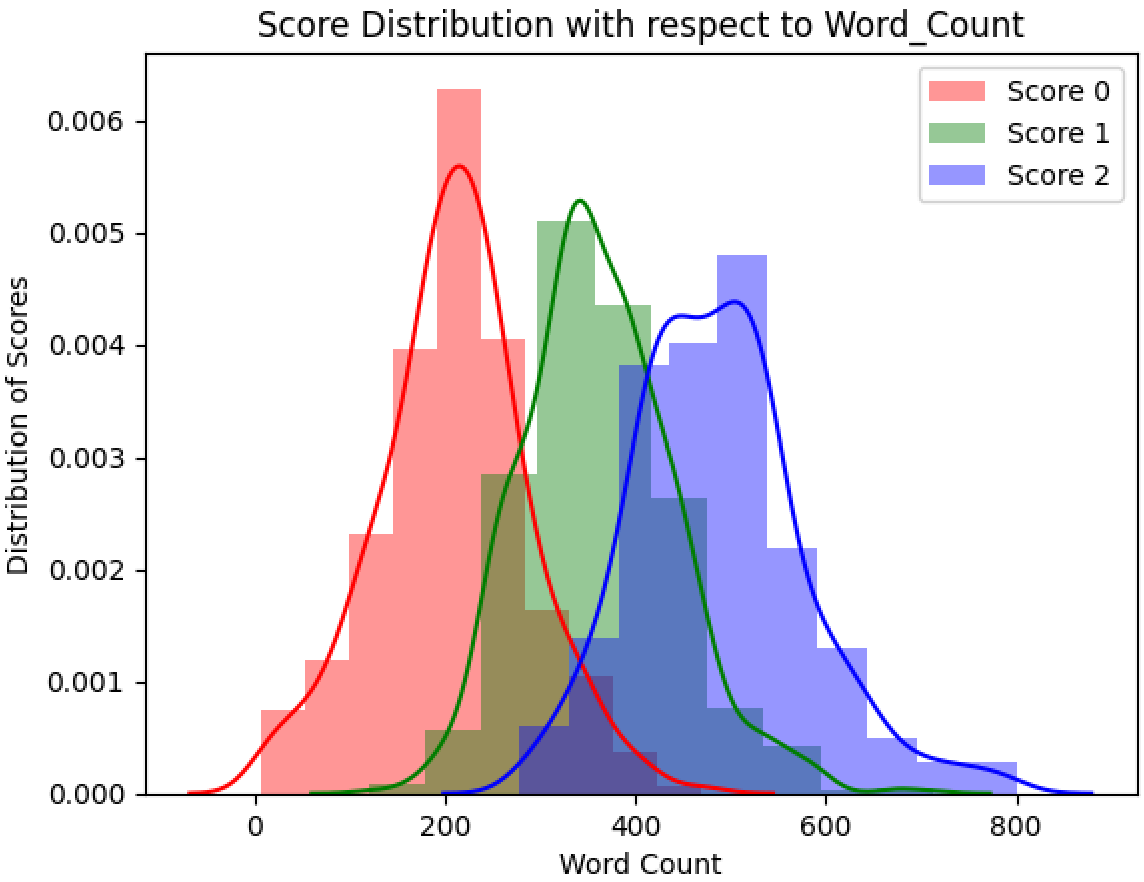
In [28]:

```
plot_score_distribution_by_feature(data, scores, "Sentence_Count", "Sentence Count")
plot_score_distribution_by_feature(data, scores, "Word_Count", "Word Count")
```

```
<ipython-input-26-457e4d892a7d>:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(zero[feature_name], bins=10, color='r', label='Score 0')
<ipython-input-26-457e4d892a7d>:8: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(one[feature_name], bins=10, color='g', label='Score 1')
<ipython-input-26-457e4d892a7d>:9: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(two[feature_name], bins=10, color='b', label='Score 2')
```

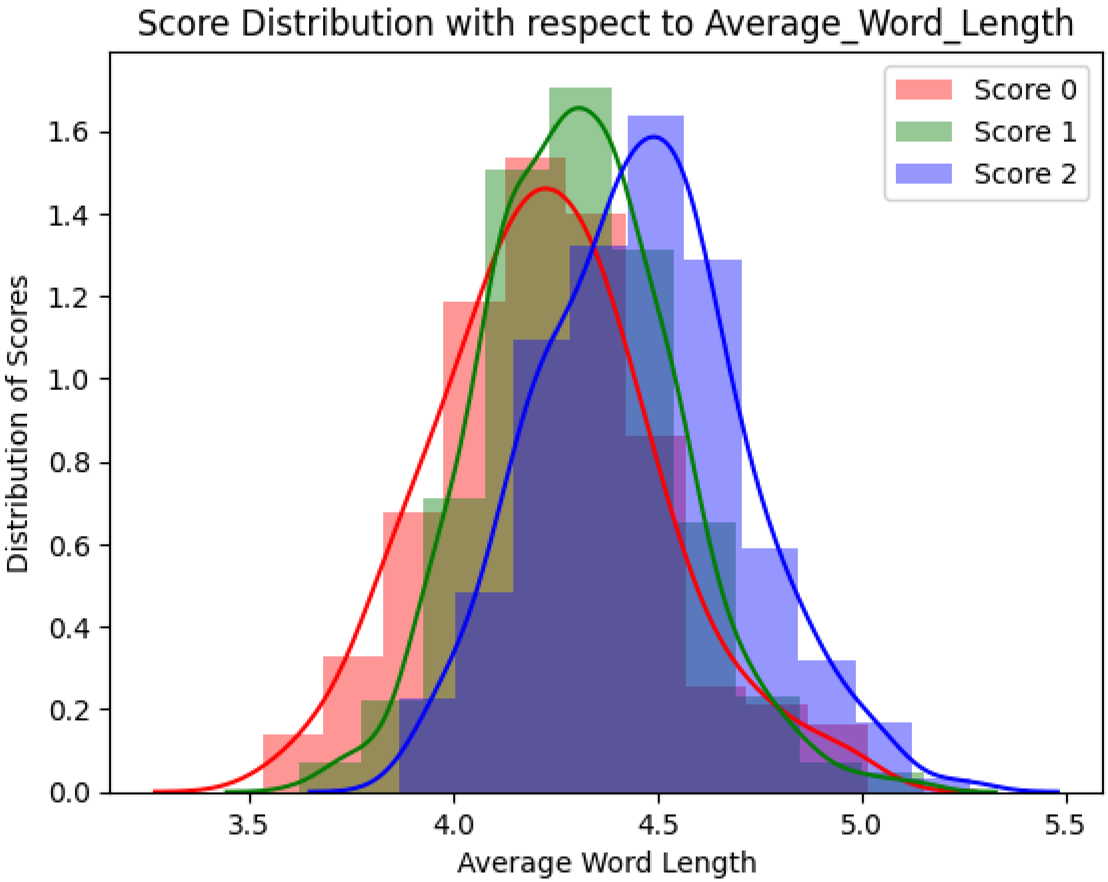

Score Distribution with respect to Sentence_Count
Score 0
Score 1
Score 2
Distribution of Scores
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0
10
20
30
40
50
60
70
Sentence Count

```
<ipython-input-26-457e4d892a7d>:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(zero[feature_name], bins=10, color='r', label='Score 0')
<ipython-input-26-457e4d892a7d>:8: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(one[feature_name], bins=10, color='g', label='Score 1')
<ipython-input-26-457e4d892a7d>:9: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(two[feature_name], bins=10, color='b', label='Score 2')
```



```
In [29]: plot_score_distribution_by_feature(data, scores, "Average_Word_Length", "Average Word Length")
```

```
<ipython-input-26-457e4d892a7d>:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(zero[feature_name], bins=10, color='r', label='Score 0')
<ipython-input-26-457e4d892a7d>:8: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(one[feature_name], bins=10, color='g', label='Score 1')
<ipython-input-26-457e4d892a7d>:9: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(two[feature_name], bins=10, color='b', label='Score 2')
```



Kappa Score Reliability

In Cohen's initial publication, he categorized Cohen's kappa values differently. Values equal to or less than 0 indicate no agreement, while those between 0.01 and 0.20 suggest none to slight agreement, 0.21–0.40 indicate fair agreement, 0.41–0.60 denote moderate agreement, 0.61–0.80 represent substantial agreement, and 0.81–1.00 signify almost perfect agreement. McHugh notes that several sources advise 80% agreement as the minimum acceptable level of interrater agreement.

Latent Semantic Analysis

In [30]:

```
# Import necessary libraries for LSA
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import TruncatedSVD
```

In [31]:

```
# Function to preprocess the text data
def preprocess_text(text):
    # Remove punctuation and numbers
    text = re.sub(r'[^\w\s]', '', text)
    text = re.sub(r'\d+', '', text)
    # Convert to lowercase
    text = text.lower()
    # Tokenize
    tokens = word_tokenize(text)
    # Lemmatize
    lemmatizer = WordNetLemmatizer()
    lemmas = [lemmatizer.lemmatize(token) for token in tokens]
    return ' '.join(lemmas)
```

In [32]:

```
# Preprocess essays
dataframe['processed_essay'] = dataframe['essay'].apply(preprocess_text)

# Create a TfidfVectorizer object
tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_features=1000)

# Fit and transform the processed essays
tfidf_matrix = tfidf_vectorizer.fit_transform(dataframe['processed_essay'])

# Number of topics/components to extract via SVD
num_topics = 100

# Create a TruncatedSVD object
svd_model = TruncatedSVD(n_components=num_topics)

# Fit and transform the TF-IDF matrix
lsa_topics = svd_model.fit_transform(tfidf_matrix)

# Display the shape of the resulting matrix
print('Shape of LSA topic matrix:', lsa_topics.shape)
```

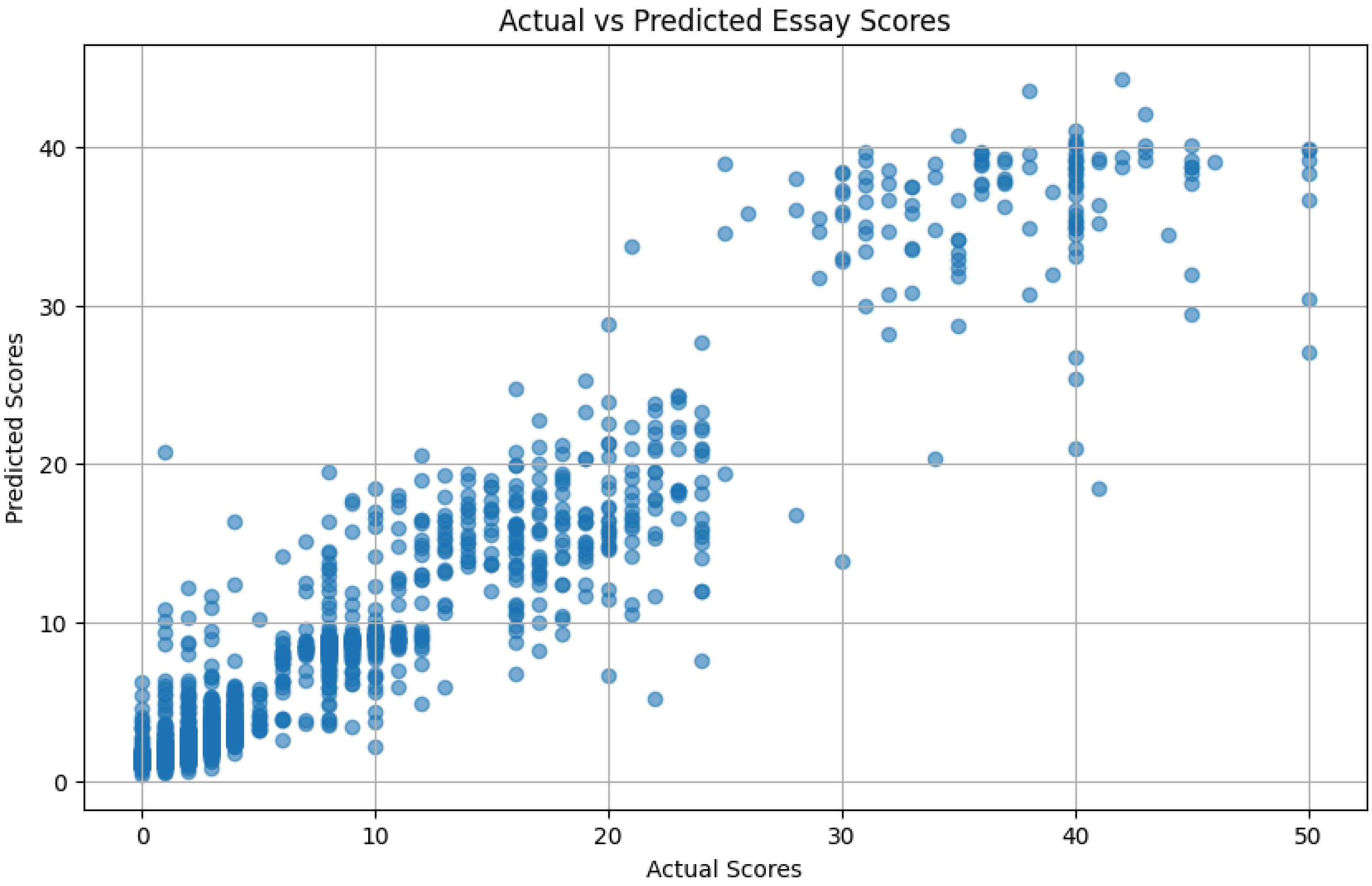
```
Shape of LSA topic matrix: (12976, 100)
```

In [33]:

```
# Import additional necessary libraries
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

# Create a predictive model using Gradient Boosting
```

```python
gbr = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(lsa_topics, dataframe['domain1_score'], test_size=0.2, random_state

# Fit the model
gbr.fit(X_train, y_train)

# Predict on the testing set
y_pred = gbr.predict(X_test)

# Calculate and print the performance metrics
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("R^2 Score:", r2_score(y_test, y_pred))

# Plotting actual vs predicted scores
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.6)
plt.xlabel('Actual Scores')
plt.ylabel('Predicted Scores')
plt.title('Actual vs Predicted Essay Scores')
plt.grid(True)
plt.show()
```

```
Mean Squared Error: 6.6727342190247025
R^2 Score: 0.9142959704413365
```



The MSE of approximately 6.67 and the R^2 Score of approximately 0.91 indicate that the model performs well in predicting essay scores based on the LSA topics.

The Automated Essay Scoring system can be made more accurate and robust by including new features and methods.This can be achieved by highlighting key features or parts of the essay that contributed significantly to its score.

In [34]:
```
# Code to integrate interactive feedback based on feature importance
def generate_feedback(essay, model, vectorizer, top_k=5):
    # Get feature importance from the model
```

```python
    importances = model.feature_importances_
    # Get feature names from the vectorizer
    feature_names = vectorizer.get_feature_names_out()
    # Sort features by importance
    important_indices = importances.argsort()[::-1][:top_k]
    important_words = [feature_names[idx] for idx in important_indices]

    # Generate feedback message
    feedback = "Key points affecting the essay score: " + ", ".join(important_words)

    # Highlight important words in the essay
    highlighted_essay = essay
    for word in important_words:
        highlighted_essay = highlighted_essay.replace(word, f"**{word}**")

    return highlighted_essay, feedback

# Example usage in workflow
example_essay = dataframe.iloc[0]['essay']
highlighted_essay, feedback = generate_feedback(example_essay, gbr, tfidf_vectorizer)

print("Highlighted Essay:\n", highlighted_essay)
print("Feedback:\n", feedback)
```

```
Highlighted Essay:
 Dear local newspaper, I think effects computers have on people are great learning skills/affects because they give us
time to chat with friends/new people, helps us learn about the globe(astronomy) and keeps us out of troble! Thing abou
t! Dont you think so? How would you feel if your teenager is always on the phone with friends! Do you ever time to chat
with your friends or buisness partner about things. Well now - there's a new way to chat the computer, theirs plenty of
sites on the internet to do so: @ORGANIZATION1, @ORGANIZATION2, @CAPS1, facebook, myspace ect. Just think now while you
r setting up meeting with your boss on the computer, your teenager is having fun on the phone not rushing to get off ca
use you want to use it. How did you learn about other countrys/states outside of yours? Well I have by computer/interne
t, it's a new way to learn about what going on in our time! You might think your child spends a lot of time on the comp
uter, but ask them so question about the economy, sea floor spreading or even about the @DATE1's you'll be surprise at
how much he/she knows. Believe it or not the computer is much interesting then in class all day reading out of books. I
f your child is home on your computer or at a local library, it's better than being out with friends being fresh, or be
ing perpressured to doing something they know isnt right. You might not know where your child is, @CAPS2 forbidde in a
hospital bed because of a drive-by. Rather than your child on the computer learning, chatting or just playing games, sa
fe and sound in your home or community place. Now I hope you have reached a point to understand and agree with me, beca
use computers can have great effects on you or child because it gives us time to chat with friends/new people, helps us
learn about the globe and believe or not keeps us out of troble. Thank you for listening.
Feedback:
 Key points affecting the essay score: actually, action, activity, accident, abandoned
```

The output includes the highlighted essay text with important words emphasized and the feedback message indicating the key points affecting the essay score based on feature importance.

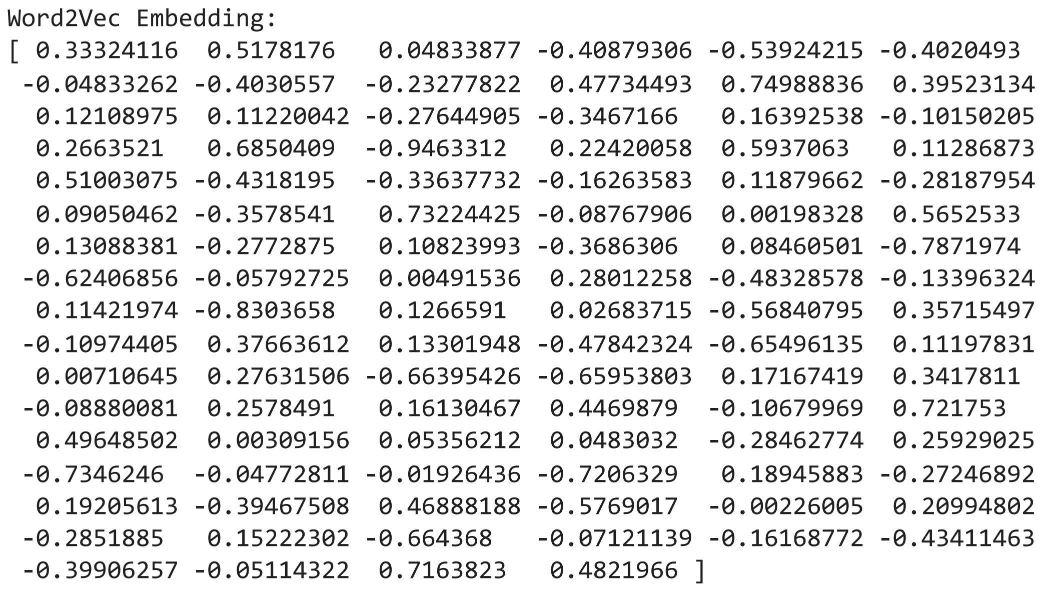
Text Embeddings

In [35]:

```python
# Import necessary libraries
from gensim.models import Word2Vec
import numpy as np
import nltk

# Function to train Word2Vec embeddings
def train_word2vec_embeddings(texts):
    # Tokenize essays into sentences
    tokenized_sentences = [nltk.word_tokenize(essay) for essay in texts]

    # Train Word2Vec model
    w2v_model = Word2Vec(sentences=tokenized_sentences, vector_size=100, window=5, min_count=1, workers=4)

    return w2v_model

# Train Word2Vec embeddings on the essay texts
word2vec_model = train_word2vec_embeddings(dataframe['essay'])

# Function to convert essays into averaged Word2Vec embeddings
def essays_to_word2vec_embeddings(texts, word2vec_model):
    # Tokenize essays into sentences
    tokenized_sentences = [nltk.word_tokenize(essay) for essay in texts]

    # Initialize an empty array to store essay embeddings
    essay_embeddings = []

    # Convert each essay into an averaged Word2Vec embedding
    for sentence in tokenized_sentences:
        # Initialize an empty array to store word embeddings for the current sentence
        sentence_embeddings = []
        # Calculate Word2Vec embedding for each word in the sentence
        for word in sentence:
            try:
                word_embedding = word2vec_model.wv[word]
                sentence_embeddings.append(word_embedding)
            except KeyError:
                # If the word is not in the vocabulary, skip it
                pass
        # Calculate average embedding for the sentence
        if sentence_embeddings:
            sentence_avg_embedding = np.mean(sentence_embeddings, axis=0)
            essay_embeddings.append(sentence_avg_embedding)

    # Convert list of embeddings into numpy array
    essay_embeddings = np.array(essay_embeddings)

    return essay_embeddings

# Convert a subset of essays into Word2Vec embeddings and print the results
subset_texts = dataframe['essay'][:3]  # Selecting the first 3 essays as a subset

# Convert essays into averaged Word2Vec embeddings
subset_embeddings = essays_to_word2vec_embeddings(subset_texts, word2vec_model)

# Print the essay texts and their corresponding Word2Vec embeddings
for i, essay in enumerate(subset_texts):
    print("Essay Text:")
    print(essay)
    print("Word2Vec Embedding:")
    print(subset_embeddings[i])
    print()
```

```
Essay Text:
Dear local newspaper, I think effects computers have on people are great learning skills/affects because they give us t
ime to chat with friends/new people, helps us learn about the globe(astronomy) and keeps us out of troble! Thing about!
Dont you think so? How would you feel if your teenager is always on the phone with friends! Do you ever time to chat wi
th your friends or buisness partner about things. Well now - there's a new way to chat the computer, theirs plenty of s
ites on the internet to do so: @ORGANIZATION1, @ORGANIZATION2, @CAPS1, facebook, myspace ect. Just think now while your
setting up meeting with your boss on the computer, your teenager is having fun on the phone not rushing to get off caus
e you want to use it. How did you learn about other countrys/states outside of yours? Well I have by computer/internet,
it's a new way to learn about what going on in our time! You might think your child spends a lot of time on the compute
r, but ask them so question about the economy, sea floor spreading or even about the @DATE1's you'll be surprise at how
much he/she knows. Believe it or not the computer is much interesting then in class all day reading out of books. If yo
ur child is home on your computer or at a local library, it's better than being out with friends being fresh, or being
perpressured to doing something they know isnt right. You might not know where your child is, @CAPS2 forbidde in a hosp
ital bed because of a drive-by. Rather than your child on the computer learning, chatting or just playing games, safe a
nd sound in your home or community place. Now I hope you have reached a point to understand and agree with me, because
computers can have great effects on you or child because it gives us time to chat with friends/new people, helps us lea
rn about the globe and believe or not keeps us out of troble. Thank you for listening.
Word2Vec Embedding:
[ 0.5468641   0.5580616   0.05306306 -0.30091447 -0.29165122 -0.8424343
 -0.04453002 -0.39928073 -0.1756803   0.22311333  0.69735265  0.41565186
  0.09168867  0.17281201 -0.30275348 -0.3776755   0.2518385  -0.13177207
  0.49817282  0.2104999  -0.746111    0.38394803  0.70875484  0.05454785
  1.0243285  -0.5975635  -0.24935113 -0.1902369   0.12174203 -0.09563191
 -0.2853429  -0.42767107  0.75147325  0.01495011 -0.00212888  0.14489006
  0.18312426 -0.40141463  0.3887339  -0.0879816   0.06464011 -0.5552891
 -0.68087137  0.02025217  0.09032232  0.04473187 -0.46471506 -0.23975326
  0.11547776 -0.91515815  0.1604746   0.22682013 -0.39651105  0.37271744
 -0.1973211   0.34564272  0.03591789 -0.372764   -0.40764302  0.32106608
  0.17902052  0.21859032 -0.42176855 -0.64879215  0.2546602   0.4183305
 -0.25295454  0.28676063  0.17887254  0.5063924  -0.00493101  0.9245875
  0.5548363  -0.35993543  0.11114766 -0.02551652 -0.31557828  0.00157454
 -0.96309096 -0.25376692  0.17934225 -0.6633482   0.16948344 -0.2602437
  0.42086032 -0.39021698  0.20300817 -0.40678075 -0.1738447  -0.01464855
 -0.15693164  0.29024038 -0.7472578  -0.33686927  0.02217986 -0.56634784
 -0.125523    0.02185498  0.40212163  0.53153366]

Essay Text:
Dear @CAPS1 @CAPS2, I believe that using computers will benefit us in many ways like talking and becoming friends will
others through websites like facebook and mysace. Using computers can help us find coordibates, locations, and able our
selfs to millions of information. Also computers will benefit us by helping with jobs as in planning a house plan and t
yping a @NUM1 page report for one of our jobs in less than writing it. Now lets go into the wonder world of technology.
Using a computer will help us in life by talking or making friends on line. Many people have myspace, facebooks, aim, t
hese all benefit us by having conversations with one another. Many people believe computers are bad but how can you mak
e friends if you can never talk to them? I am very fortunate for having a computer that can help with not only school w
ork but my social life and how I make friends. Computers help us with finding our locations, coordibates and millions o
```

```
f information online. If we didn't go on the internet a lot we wouldn't know how to go onto websites that @MONTH1 help
us with locations and coordinates like @LOCATION1. Would you rather use a computer or be in @LOCATION3. When your suppo
sed to be vacationing in @LOCATION2. Million of information is found on the internet. You can as almost every question
and a computer will have it. Would you rather easily draw up a house plan on the computers or take @NUM1 hours doing on
e by hand with ugly erazer marks all over it, you are garrenteed that to find a job with a drawing like that. Also when
appling for a job many workers must write very long papers like a @NUM3 word essay on why this job fits you the most, a
nd many people I know don't like writing @NUM3 words non-stopp for hours when it could take them I hav an a computer. T
hat is why computers we needed a lot now adays. I hope this essay has impacted your descion on computers because they a
re great machines to work with. The other day I showed my mom how to use a computer and she said it was the greatest in
vention sense sliced bread! Now go out and buy a computer to help you chat online with friends, find locations and mill
ions of information on one click of the button and help your self with getting a job with neat, prepared, printed work
that your boss will love.
Word2Vec Embedding:
[ 0.45997354  0.65415186  0.08450483 -0.37064767 -0.47165874 -0.5338241
  0.13817884 -0.44245097 -0.02813374  0.19159473  0.7059995   0.3361897
  0.05629054  0.04586164 -0.2851749  -0.49733913  0.17948537 -0.08388498
  0.42971352  0.4783707  -0.69170713  0.3591774   0.6220524   0.18268491
  0.77393496 -0.4454806  -0.32081687 -0.13599886  0.05843005 -0.23747274
 -0.2037592  -0.18936987  0.60010475 -0.07811461 -0.06236929  0.29866397
  0.16932696 -0.38721144  0.17622645 -0.21164738 -0.08114216 -0.5181029
 -0.6311683  -0.33759868  0.07183433  0.07402414 -0.5561383  -0.2756738
  0.1524565  -0.6606833  -0.01290823 -0.03112466 -0.4970862   0.5330502
 -0.02885071  0.3255732   0.11442916 -0.6358405  -0.5519764   0.3424997
  0.18173753  0.09737751 -0.46917078 -0.70732576  0.07420357  0.4079008
 -0.19037776  0.2658575  -0.12212857  0.5330517  -0.08389772  0.7055903
  0.5379835  -0.11197814  0.09011327  0.06347732 -0.35219738  0.1681923
 -0.77982944 -0.04856224  0.21472819 -0.5348745   0.08402745 -0.3157298
  0.24143732 -0.39069185  0.40834394 -0.40815932 -0.1731285   0.09329446
 -0.17796026  0.26187918 -0.58083147 -0.1646311   0.07103706 -0.5423428
 -0.1260275  -0.12754315  0.6120681   0.52600276]

Essay Text:
Dear, @CAPS1 @CAPS2 @CAPS3 More and more people use computers, but not everyone agrees that this benefits society. Thos
e who support advances in technology believe that computers have a positive effect on people. Others have different ide
as. A great amount in the world today are using computers, some for work and spme for the fun of it. Computers is one o
f mans greatest accomplishments. Computers are helpful in so many ways, @CAPS4, news, and live streams. Don't get me wr
ong way to much people spend time on the computer and they should be out interacting with others but who are we to tell
them what to do. When I grow up I want to be a author or a journalist and I know for a fact that both of those jobs inv
olve lots of time on time on the computer, one @MONTH1 spend more time then the other but you know exactly what @CAPS5
getting at. So what if some expert think people are spending to much time on the computer and not exercising, enjoying
natures and interacting with family and friends. For all the expert knows that its how must people make a living and we
don't know why people choose to use the computer for a great amount of time and to be honest it's non of my concern and
it shouldn't be the so called experts concern. People interact a thousand times a day on the computers. Computers keep
lots of kids of the streets instead of being out and causing trouble. Computers helps the @ORGANIZATION1 locate most wa
nted criminals. As you can see computers are more useful to society then you think, computers benefit society.
```



```
Word2Vec Embedding:
[ 0.33324116  0.5178176   0.04833877 -0.40879306 -0.53924215 -0.4020493
 -0.04833262 -0.4030557  -0.23277822  0.47734493  0.74988836  0.39523134
  0.12108975  0.11220042 -0.27644905 -0.3467166   0.16392538 -0.10150205
  0.2663521   0.6850409  -0.9463312   0.22420058  0.5937063   0.11286873
  0.51003075 -0.4318195  -0.33637732 -0.16263583  0.11879662 -0.28187954
  0.09050462 -0.3578541   0.73224425 -0.08767906  0.00198328  0.5652533
  0.13088381 -0.2772875   0.10823993 -0.3686306   0.08460501 -0.7871974
 -0.62406856 -0.05792725  0.00491536  0.28012258 -0.48328578 -0.13396324
  0.11421974 -0.8303658   0.1266591   0.02683715 -0.56840795  0.35715497
 -0.10974405  0.37663612  0.13301948 -0.47842324 -0.65496135  0.11197831
  0.00710645  0.27631506 -0.66395426 -0.65953803  0.17167419  0.3417811
 -0.08880081  0.2578491   0.16130467  0.4469879  -0.10679969  0.721753
  0.49648502  0.00309156  0.05356212  0.0483032  -0.28462774  0.25929025
 -0.7346246  -0.04772811 -0.01926436 -0.7206329   0.18945883 -0.27246892
  0.19205613 -0.39467508  0.46888188 -0.5769017  -0.00226005  0.20994802
 -0.2851885   0.15222302 -0.664368   -0.07121139 -0.16168772 -0.43411463
 -0.39906257 -0.05114322  0.7163823   0.4821966 ]
```

The output of the code consists of the essay texts and their corresponding Word2Vec embeddings.The output for each essay includes the essay text followed by the Word2Vec embedding vector representing the semantic content of the essay.