Chapter 6 - Ex4: Movie Review Sentiment Analysis

Cho dữ liệu train.tsv, test.tsv nằm trong thư mục movies - Rotten Tomatoes movie review dataset . Bộ dữ liệu này được sử dụng để dự đoán tình cảm của người dùng dành cho các phim.

Có các nhãn "sentiment" sau:

- 0 - negative
- 1 - somewhat negative
- 2 - neutral
- 3 - somewhat positive
- 4 - positive

Yêu cầu:

1. Đọc dữ liệu
2. Phân tích sơ bộ dữ liệu
3. Tạo wordcloud của Positive và Negative review. In danh sách 20 từ có trọng số lớn (chữ to) trong word cloud. Trực quan hóa dữ liệu
4. Lọc lại dữ liệu, chỉ giữ lại những mẫu có phần Phrase nhận xét từ 50 ký tự trở lên. Chia dữ liệu thành 2 bộ source và target
5. Chọn phương pháp để chuẩn hóa dữ liệu và thực hiện việc chuẩn hóa (với dữ liệu đã lọc ở phần 4)

Đọc dữ liệu

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

- Dữ liệu được download từ: link (https://www.kaggle.com/c/movie-review-sentiment-analysis-kernels-only/data)

In [3]:

```
# Đọc dữ liệu
df_train = pd.read_csv("movie_review/train.tsv", sep='\t')
df_train.head()
```

Out[3]:

	PhraseId	SentenceId	Phrase	Sentiment
0	1	1	A series of escapades demonstrating the adage ...	1
1	2	1	A series of escapades demonstrating the adage ...	2
2	3	1	A series	2
3	4	1	A	2
4	5	1	series	2

In [4]:

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 156060 entries, 0 to 156059
Data columns (total 4 columns):
PhraseId      156060 non-null int64
SentenceId    156060 non-null int64
Phrase        156060 non-null object
Sentiment     156060 non-null int64
dtypes: int64(3), object(1)
memory usage: 4.8+ MB
```

In [5]:

```
df_test = pd.read_csv("movie_review/test.tsv", sep='\t')
df_test.head()
```

Out[5]:

	PhraseId	SentenceId	Phrase
0	156061	8545	An intermittently pleasing but mostly routine ...
1	156062	8545	An intermittently pleasing but mostly routine ...
2	156063	8545	An
3	156064	8545	intermittently pleasing but mostly routine effort
4	156065	8545	intermittently pleasing but mostly routine

In [6]:

```
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 66292 entries, 0 to 66291
Data columns (total 3 columns):
PhraseId      66292 non-null int64
SentenceId    66292 non-null int64
Phrase        66292 non-null object
dtypes: int64(2), object(1)
memory usage: 1.5+ MB
```

Phân tích sơ bộ dữ liệu

In [7]:

```
group_count = df_train['Sentiment'].value_counts()
```

In [8]:

```
group_count
```

Out[8]:

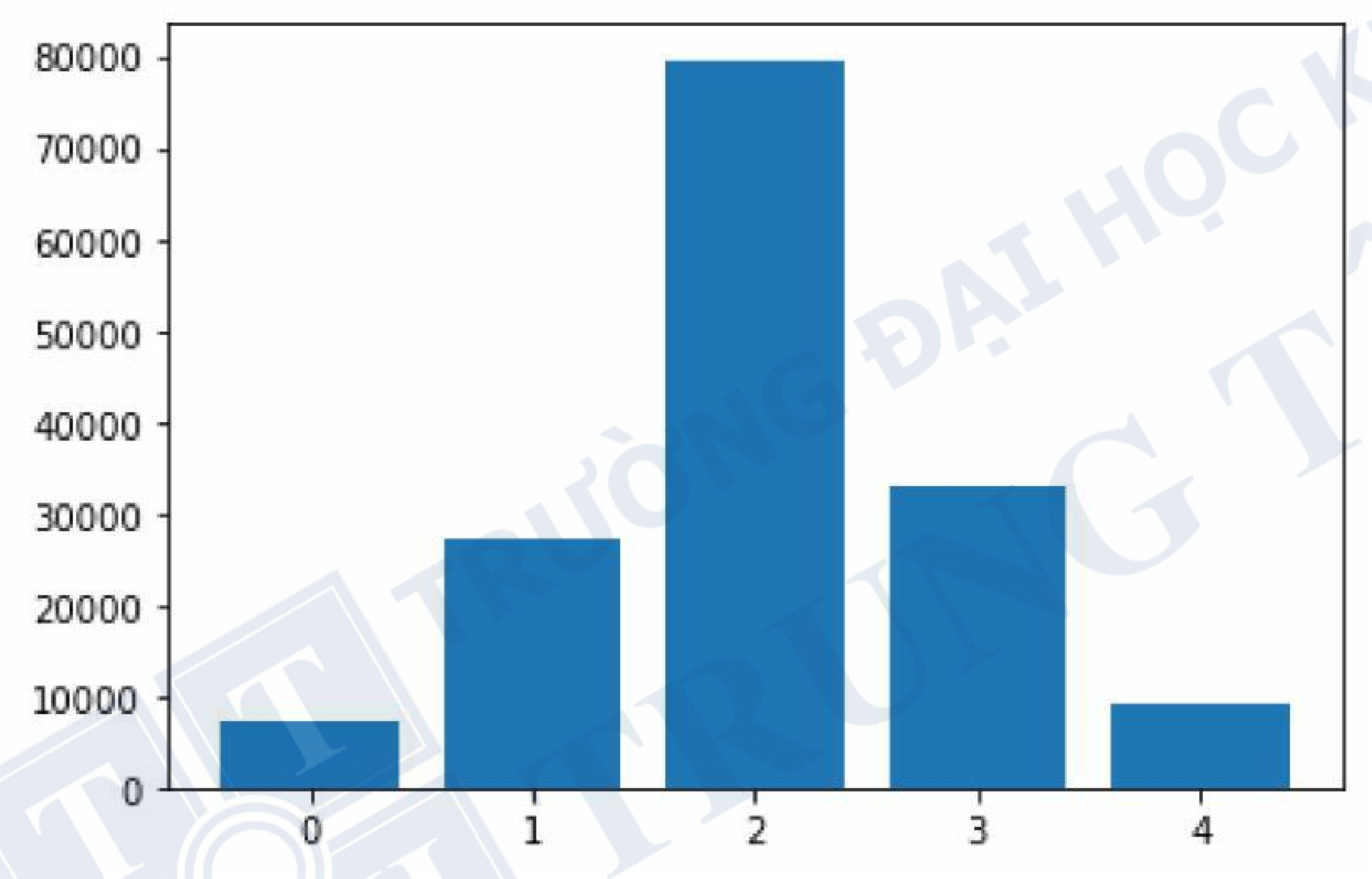
```
2    79582
3    32927
1    27273
4     9206
0     7072
Name: Sentiment, dtype: int64
```

In [9]:

```
plt.bar(group_count.index, group_count)
```

Out[9]:

```
<BarContainer object of 5 artists>
```



In [10]:

```
data_train = df_train.drop(['PhraseId','SentenceId'],axis=1)
data_train.head()
```

Out[10]:

	Phrase	Sentiment
0	A series of escapades demonstrating the adage ...	1
1	A series of escapades demonstrating the adage ...	2
2	A series	2
3	A	2
4	series	2

In [11]:

```
data_train['phrase_len'] = [len(t) for t in data_train.Phrase]
data_train.head(4)
```

Out[11]:

	Phrase	Sentiment	phrase_len
0	A series of escapades demonstrating the adage ...	1	188
1	A series of escapades demonstrating the adage ...	2	77
2	A series	2	8
3	A	2	1

Tạo wordcloud của Positive và Negative review. In danh sách 20 từ có trọng số lớn (chữ to) trong word cloud. Trực quan hóa dữ liệu

In [12]:

```
# Negative
neg_phrases = data_train[data_train.Sentiment == 0]
```

In [13]:

```
from wordcloud import WordCloud, STOPWORDS
```

In [14]:

```
stopwords = set(STOPWORDS)
#stopwords
```

In [15]:

```
neg_words = []
for t in neg_phrases.Phrase:
    neg_words.append(t)
neg_words[:4]
```

Out[15]:

```
['would have a hard time sitting through this one',
 'have a hard time sitting through this one',
 'Aggressive self-glorification and a manipulative whitewash',
 'self-glorification and a manipulative whitewash']
```

In [16]:

```
neg_text = pd.Series(neg_words).str.cat(sep=' ')
neg_text[:100]
```

Out[16]:

```
'would have a hard time sitting through this one have a hard time sitting th
rough this one Aggressive'
```

In [17]:

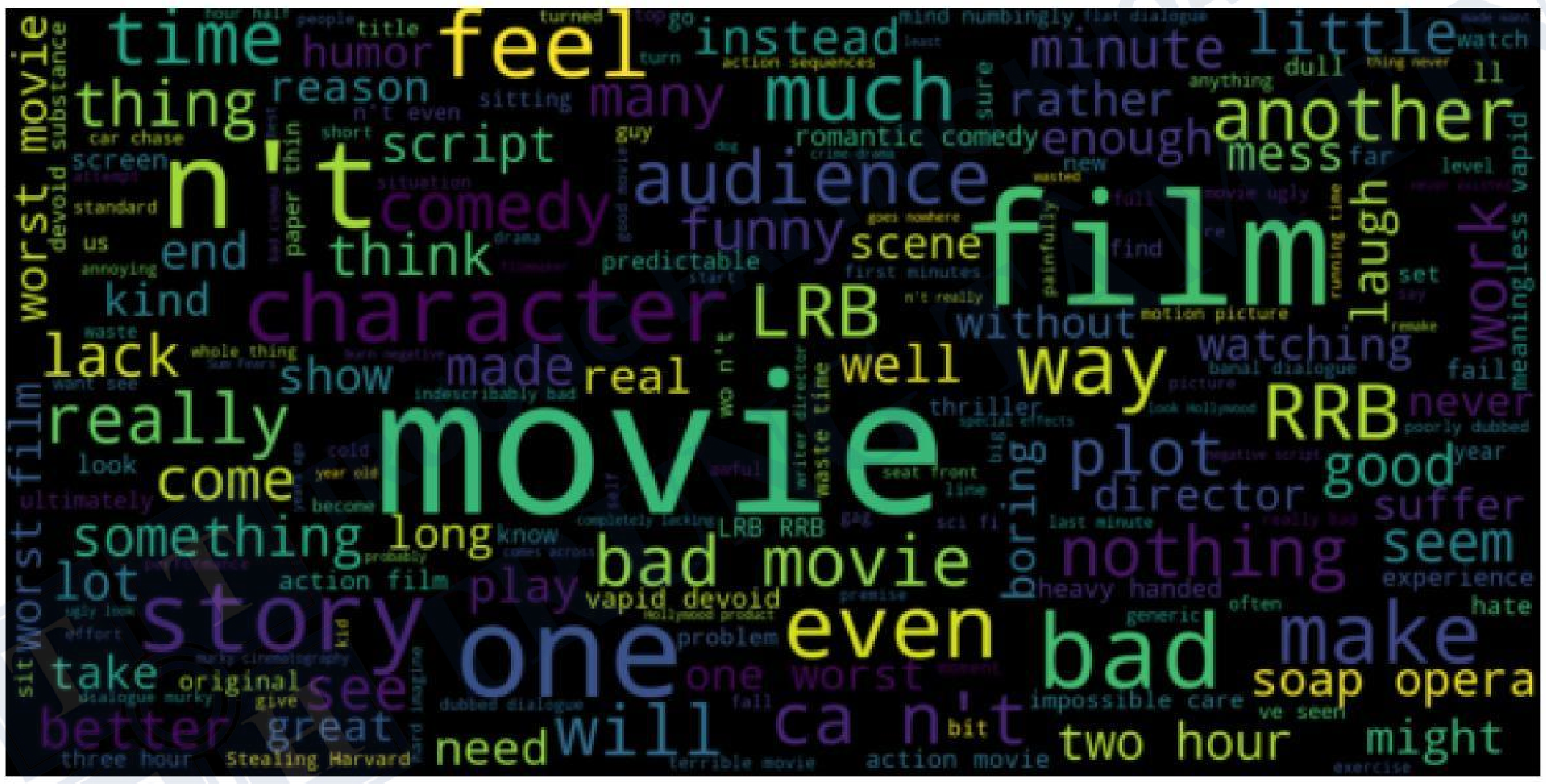
```
# instantiate a word cloud object
wc = WordCloud(
    background_color='black',
    max_words=200,
    stopwords=stopwords,
    width=1600, height=800,
    max_font_size=200
)

# generate the word cloud
wc.generate(neg_text)
```

Out[17]:

```
<wordcloud.wordcloud.WordCloud at 0x131058b4470>
```

In [18]:

```
# display the word clouds
plt.figure(figsize=(10, 12))
plt.imshow(wc, interpolation='bilinear')
plt.axis('off')
plt.show()
```



In [19]:

```
wc.words_
```

Out[19]:

```
{'movie': 1.0,
 'film': 0.6113207547169811,
 'one': 0.4018867924528302,
 "n't": 0.37358490566037733,
 'bad': 0.24528301886792453,
 'story': 0.22641509433962265,
 'feel': 0.19622641509433963,
 'character': 0.19622641509433963,
 'even': 0.18867924528301888,
 'way': 0.16792452830188678,
 'make': 0.16792452830188678,
 'much': 0.15283018867924528,
 'RRB': 0.13773584905660377,
 'time': 0.1339622641509434,
 'audience': 0.13018867924528302,
 'LRB': 0.12452830188679245,
 'another': 0.12452830188679245,
 "ca n't": 0.12264150943396226,
```

In [20]:

```
first_20_big_words = {k: wc.words_[k] for k in list(wc.words_)[:20]}
```

In [21]:

```
first_20_big_words
```

Out[21]:

```
{'movie': 1.0,
 'film': 0.6113207547169811,
 'one': 0.4018867924528302,
 "n't": 0.37358490566037733,
 'bad': 0.24528301886792453,
 'story': 0.22641509433962265,
 'feel': 0.19622641509433963,
 'character': 0.19622641509433963,
 'even': 0.18867924528301888,
 'way': 0.16792452830188678,
 'make': 0.16792452830188678,
 'much': 0.15283018867924528,
 'RRB': 0.13773584905660377,
 'time': 0.1339622641509434,
 'audience': 0.13018867924528302,
 'LRB': 0.12452830188679245,
 'another': 0.12452830188679245,
 "ca n't": 0.12264150943396226,
 'will': 0.1169811320754717,
 'bad movie': 0.1169811320754717}
```

In [22]:

```
first_20_big_words.items()
```

Out[22]:

```
dict_items([('movie', 1.0), ('film', 0.6113207547169811), ('one', 0.40188679
24528302), ("n't", 0.37358490566037733), ('bad', 0.24528301886792453), ('sto
ry', 0.22641509433962265), ('feel', 0.19622641509433963), ('character', 0.19
622641509433963), ('even', 0.18867924528301888), ('way', 0.1679245283018867
8), ('make', 0.16792452830188678), ('much', 0.15283018867924528), ('RRB', 0.
13773584905660377), ('time', 0.1339622641509434), ('audience', 0.13018867924
528302), ('LRB', 0.12452830188679245), ('another', 0.12452830188679245), ("c
a n't", 0.12264150943396226), ('will', 0.1169811320754717), ('bad movie', 0.
1169811320754717)])
```

In [23]:

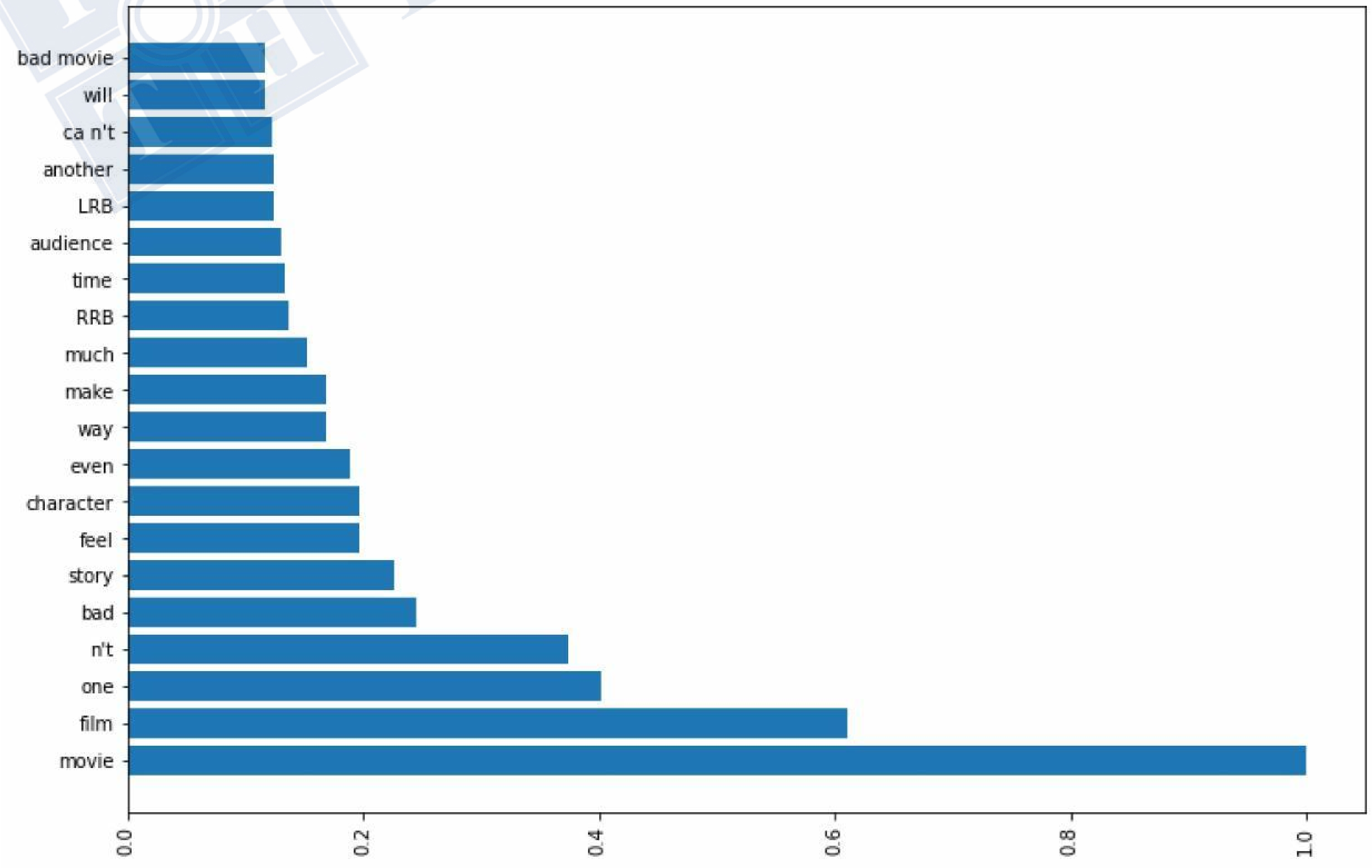
```
df_first_20_big_words = pd.DataFrame(list(first_20_big_words.items()),
                                     columns=['word', 'freq'])
df_first_20_big_words.head()
```

Out[23]:

	word	freq
0	movie	1.000000
1	film	0.611321
2	one	0.401887
3	n't	0.373585
4	bad	0.245283

In [24]:

```
plt.figure(figsize=(12,8))
plt.barh(df_first_20_big_words.word, df_first_20_big_words.freq)
plt.xticks(rotation='90')
plt.show()
```



In [25]:

```
# Posvitive
pos_phrases = data_train[data_train.Sentiment == 4]
```

In [26]:

```
pos_words = []
for t in pos_phrases.Phrase:
    pos_words.append(t)
pos_words[:4]
```

Out[26]:

```
['This quiet , introspective and entertaining independent is worth seeking
.',
 'quiet , introspective and entertaining independent',
 'entertaining',
 'is worth seeking']
```

In [27]:

```
pos_text = pd.Series(pos_words).str.cat(sep=' ')
pos_text[:100]
```

Out[27]:

```
'This quiet , introspective and entertaining independent is worth seeking .
quiet , introspective and'
```

In [28]:

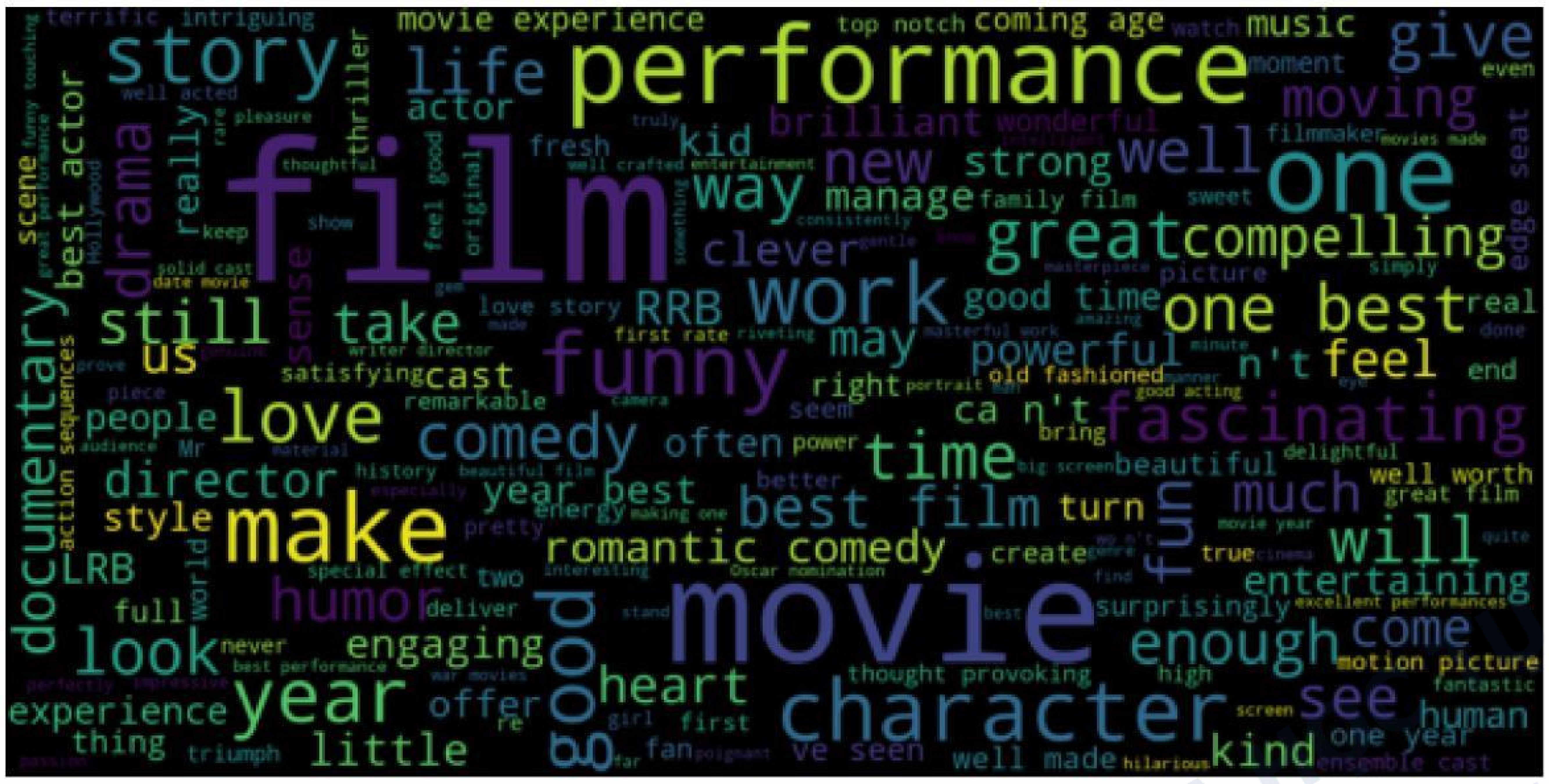
```
# instantiate a word cloud object
wc_pos = WordCloud(
    background_color='black',
    max_words=200,
    stopwords=stopwords,
    width=1600, height=800,
    max_font_size=200
)

# generate the word cloud
wc_pos.generate(pos_text)
```

Out[28]:

```
<wordcloud.wordcloud.WordCloud at 0x13105cd3240>
```

In [29]:

```python
# display the word clouds
plt.figure(figsize=(10, 12))
plt.imshow(wc_pos, interpolation='bilinear')
plt.axis('off')
plt.show()
```



In [30]:

```python
first_20_pos_big_words = {k: wc_pos.words_[k] for k in list(wc_pos.words_)[:20]}
```

In [31]:

```python
first_20_pos_big_words
```

Out[31]:

```
{'film': 1.0,
 'movie': 0.5105633802816901,
 'one': 0.25704225352112675,
 'performance': 0.2323943661971831,
 'make': 0.18779342723004694,
 'funny': 0.1467136150234742,
 'work': 0.1467136150234742,
 'character': 0.14553990610328638,
 'story': 0.13028169014084506,
 'good': 0.1267605633802817,
 'year': 0.11502347417840375,
 'love': 0.10328638497652583,
 'great': 0.09272300469483569,
 'fascinating': 0.0880281690140845,
 'time': 0.0880281690140845,
 'one best': 0.08450704225352113,
 'will': 0.08215962441314555,
 'way': 0.07863849765258216,
 'comedy': 0.07746478873239436,
 'look': 0.07746478873239436}
```

In [32]:

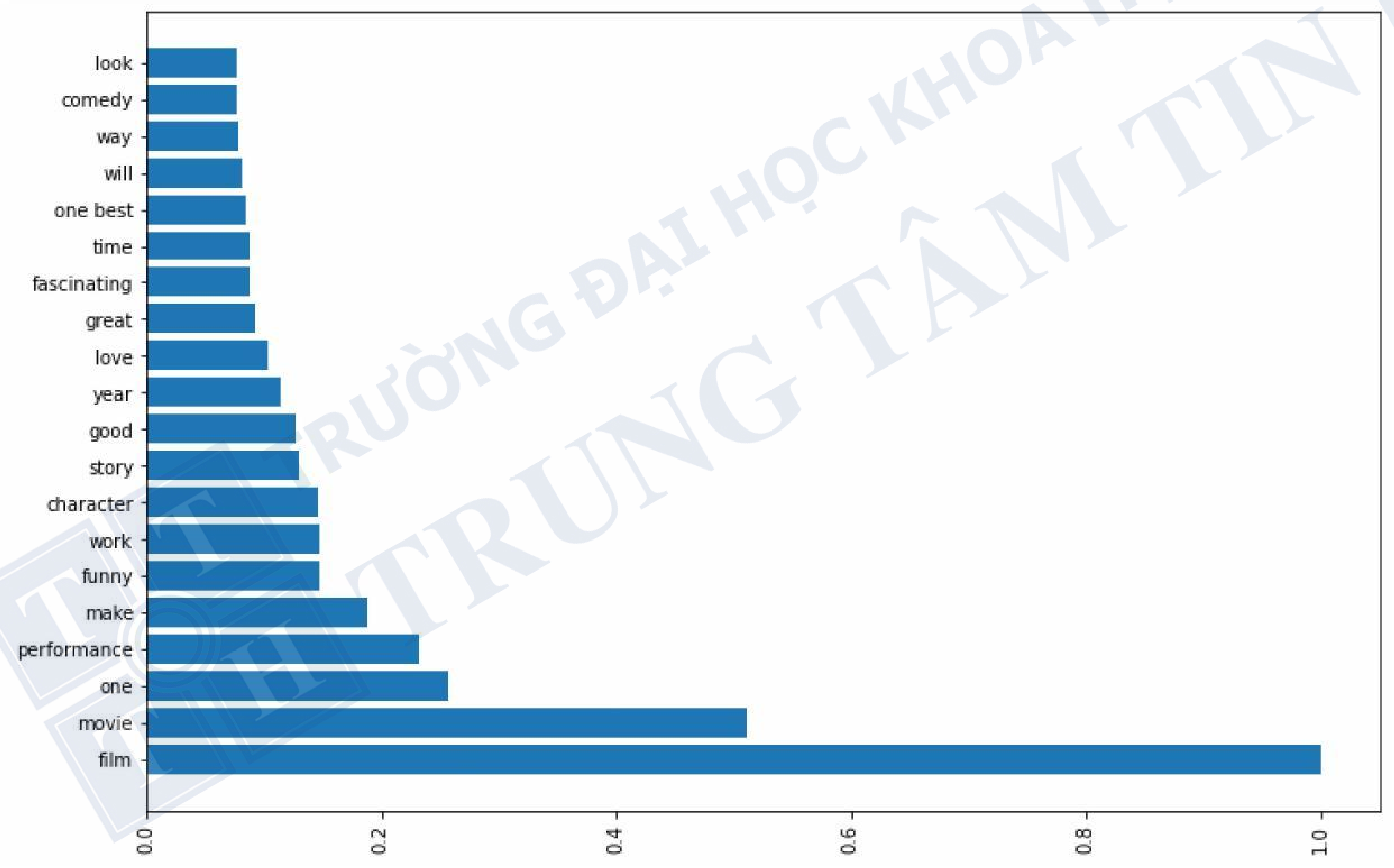
```
df_first_20_pos_big_words = pd.DataFrame(list(first_20_pos_big_words.items()),
                                         columns=['word', 'freq'])
df_first_20_pos_big_words.head()
```

Out[32]:

	word	freq
0	film	1.000000
1	movie	0.510563
2	one	0.257042
3	performance	0.232394
4	make	0.187793

In [33]:

```
plt.figure(figsize=(12,8))
plt.barh(df_first_20_pos_big_words.word, df_first_20_pos_big_words.freq)
plt.xticks(rotation='90')
plt.show()
```



Lọc lại dữ liệu, chỉ giữ lại những mẫu có phần Phrase nhận xét từ 50 ký tự trở lên. Chia dữ liệu thành 2 bộ source và target

In [34]:

```
data_train.head()
```

Out[34]:

	Phrase	Sentiment	phrase_len
0	A series of escapades demonstrating the adage ...	1	188
1	A series of escapades demonstrating the adage ...	2	77
2	A series	2	8
3	A	2	1
4	series	2	6

In [35]:

```
data_filter_len_50 = data_train[data_train.phrase_len>=50]
```

In [36]:

```
data_filter_len_50.head()
```

Out[36]:

	Phrase	Sentiment	phrase_len
0	A series of escapades demonstrating the adage ...	1	188
1	A series of escapades demonstrating the adage ...	2	77
5	of escapades demonstrating the adage that what...	2	68
7	escapades demonstrating the adage that what is...	2	65
9	demonstrating the adage that what is good for ...	2	55

In [37]:

```
data_filter_len_50.shape
```

Out[37]:

```
(42249, 3)
```

Một số nhận xét:

Trên thực tế, chúng ta quan tâm đến các ý kiến khen/chê nhiều hơn là ý kiến trung lập.

Vì vậy, trong bài này chúng ta cũng sẽ bỏ qua ý kiến trung lập và chỉ tập trung vào khen chê (xóa bỏ các ý kiến trung lập)

Khi tập trung vào khen/chê, chúng ta cũng có 2 giải pháp:

1. Phân theo 4 loại: 0 - negative, 1 - somewhat negative, 3 - somewhat positive, 4 - positive
2. Phân theo 2 loại: gộp negative và somewhat negative thành 0 (chê), gộp somewhat positive và positive thành 1 (khen)

In [38]:

```
# Giải pháp 1
data_filter_len_50_new = data_filter_len_50[data_filter_len_50.Sentiment !=2]
```

In [39]:

```
data_filter_len_50_new.shape
```

Out[39]:

```
(30025, 3)
```

In [40]:

```
# Giải pháp
# Tạo 1 cột mới chỉ có 2 Type: 0: negatve, 1: positive
data_filter_len_50_new['Type'] = \
        np.where(data_filter_len_50_new['Sentiment'] >= 3, 1, 0)
```

```
c:\program files\python36\lib\site-packages\ipykernel_launcher.py:3: Setting
WithCopyWarning:
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: http://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (http://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy)
  This is separate from the ipykernel package so we can avoid doing imports
 until
```

In [41]:

```
data_filter_len_50_new['Type']
```

Out[41]:

```
0         0
33        0
63        1
64        1
66        1
         ..
156014    1
156021    0
156026    0
156031    0
156047    0
Name: Type, Length: 30025, dtype: int32
```

In [42]:

```
X = data_filter_len_50_new.Phrase
X.head()
```

Out[42]:

```
0     A series of escapades demonstrating the adage ...
33    the gander , some of which occasionally amuses...
63    This quiet , introspective and entertaining in...
64    This quiet , introspective and entertaining in...
66    quiet , introspective and entertaining indepen...
Name: Phrase, dtype: object
```

In [43]:

```
y = data_filter_len_50_new.Sentiment
y.head()
```

Out[43]:

```
0     1
33    1
63    4
64    3
66    4
Name: Sentiment, dtype: int64
```

In [44]:

```
y_new = data_filter_len_50_new.Type
y_new.head()
```

Out[44]:

```
0     0
33    0
63    1
64    1
66    1
Name: Type, dtype: int32
```

Chọn phương pháp để chuẩn hóa dữ liệu và thực hiện việc chuẩn hóa (với dữ liệu đã lọc ở phần 4)

CountVectorizer

In [45]:

```
from sklearn.feature_extraction.text import CountVectorizer
```

In [46]:

```
cvector = CountVectorizer(stop_words='english')
cvector.fit(X)
```

Out[46]:

```
CountVectorizer(stop_words='english')
```

In [47]:

```
len(cvector.get_feature_names())
```

Out[47]:

```
14208
```

In [48]:

```
# Apply the vectorizer
cv_transformed = cvector.transform(X)
```

In [49]:

```
# Print the full array
cv_array = cv_transformed.toarray()
```

In [50]:

```
cv_array.shape
```

Out[50]:

```
(30025, 14208)
```