In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
import re
from sklearn.model_selection import train_test_split
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
stop_words=set(stopwords.words("english"))
from wordcloud import WordCloud
```

In [3]:
```
df=pd.read_excel(r"C:\Users\aryan\Downloads\Imdb.xlsx")
```

In [4]:
```
df.shape
```

Out[4]:
```
(50000, 2)
```

In [5]:
```
df.head()
```

Out[5]:

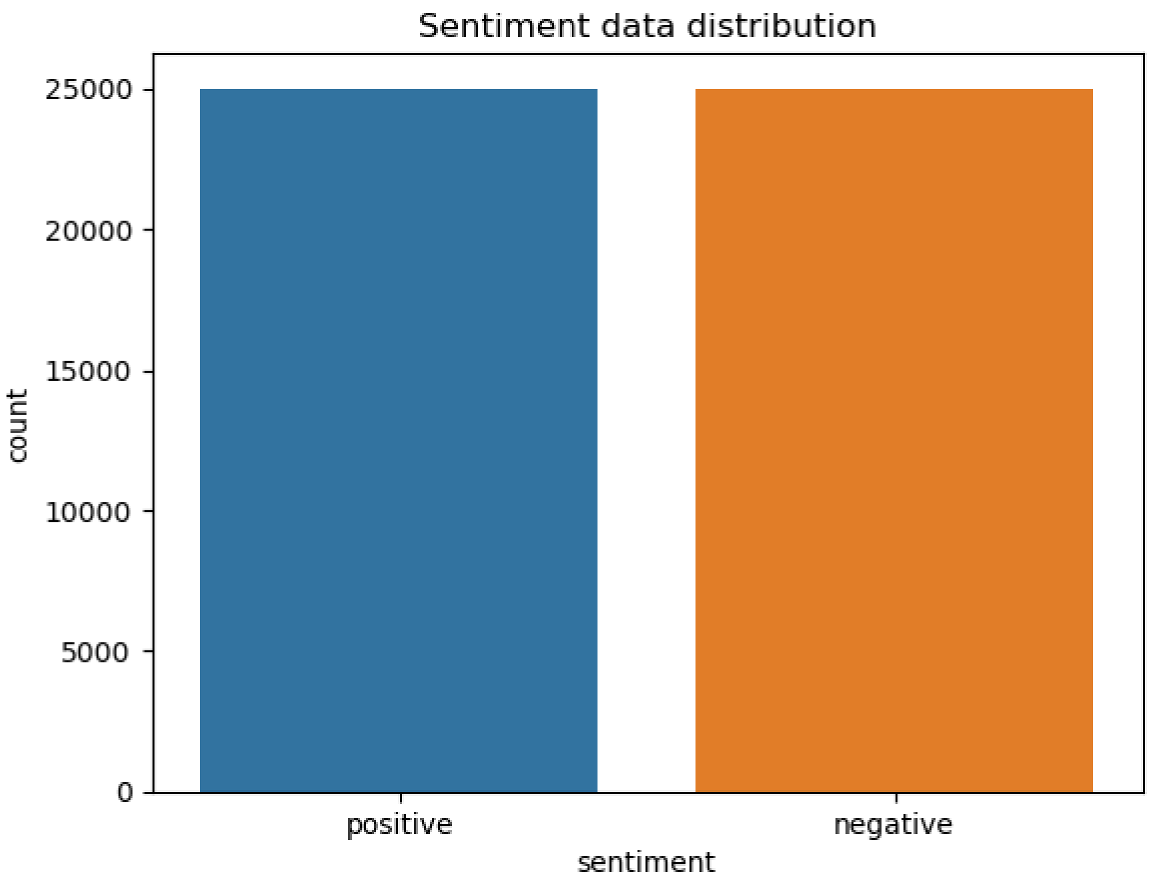
	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. <br /><br />The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   review     50000 non-null  object
 1   sentiment  50000 non-null  object
dtypes: object(2)
memory usage: 781.4+ KB
```

In [7]:
```
sns.countplot(x="sentiment",data=df)
plt.title("Sentiment data distribution")
```

Out[7]:
```
Text(0.5, 1.0, 'Sentiment data distribution')
```



In [8]:
```
for i in range(5):
    print(df["review"].iloc[i],"\n")
    print(df["sentiment"].iloc[i],"\n")
```

```
One of the other reviewers has mentioned that after watching just 1 Oz episode yo
u'll be hooked. They are right, as this is exactly what happened with me.<br /><b
r />The first thing that struck me about Oz was its brutality and unflinching sce
nes of violence, which set in right from the word GO. Trust me, this is not a sho
w for the faint hearted or timid. This show pulls no punches with regards to drug
s, sex or violence. Its is hardcore, in the classic use of the word.<br /><br />I
t is called OZ as that is the nickname given to the Oswald Maximum Security State
Penitentary. It focuses mainly on Emerald City, an experimental section of the pr
ison where all the cells have glass fronts and face inwards, so privacy is not hi
gh on the agenda. Em City is home to many..Aryans, Muslims, gangstas, Latinos, Ch
ristians, Italians, Irish and more....so scuffles, death stares, dodgy dealings a
nd shady agreements are never far away.<br /><br />I would say the main appeal of
the show is due to the fact that it goes where other shows wouldn't dare. Forget
pretty pictures painted for mainstream audiences, forget charm, forget romance...
OZ doesn't mess around. The first episode I ever saw struck me as so nasty it was
surreal, I couldn't say I was ready for it, but as I watched more, I developed a
taste for Oz, and got accustomed to the high levels of graphic violence. Not just
violence, but injustice (crooked guards who'll be sold out for a nickel, inmates
who'll kill on order and get away with it, well mannered, middle class inmates be
ing turned into prison bitches due to their lack of street skills or prison exper
ience) Watching Oz, you may become comfortable with what is uncomfortable viewin
g....thats if you can get in touch with your darker side.

positive

A wonderful little production. <br /><br />The filming technique is very unassumi
ng- very old-time-BBC fashion and gives a comforting, and sometimes discomfortin
g, sense of realism to the entire piece. <br /><br />The actors are extremely wel
l chosen- Michael Sheen not only "has got all the polari" but he has all the voic
es down pat too! You can truly see the seamless editing guided by the references
to Williams' diary entries, not only is it well worth the watching but it is a te
rrificly written and performed piece. A masterful production about one of the gre
at master's of comedy and his life. <br /><br />The realism really comes home wit
h the little things: the fantasy of the guard which, rather than use the traditio
nal 'dream' techniques remains solid then disappears. It plays on our knowledge a
nd our senses, particularly with the scenes concerning Orton and Halliwell and th
e sets (particularly of their flat with Halliwell's murals decorating every surfa
ce) are terribly well done.

positive

I thought this was a wonderful way to spend time on a too hot summer weekend, sit
ting in the air conditioned theater and watching a light-hearted comedy. The plot
is simplistic, but the dialogue is witty and the characters are likable (even the
well bread suspected serial killer). While some may be disappointed when they rea
lize this is not Match Point 2: Risk Addiction, I thought it was proof that Woody
Allen is still fully in control of the style many of us have grown to love.<br />
<br />This was the most I'd laughed at one of Woody's comedies in years (dare I s
ay a decade?). While I've never been impressed with Scarlet Johanson, in this she
managed to tone down her "sexy" image and jumped right into a average, but spirit
ed young woman.<br /><br />This may not be the crown jewel of his career, but it
was wittier than "Devil Wears Prada" and more interesting than "Superman" a great
comedy to go see with friends.

positive

Basically there's a family where a little boy (Jake) thinks there's a zombie in h
is closet & his parents are fighting all the time.<br /><br />This movie is slowe
r than a soap opera... and suddenly, Jake decides to become Rambo and kill the zo
mbie.<br /><br />OK, first of all when you're going to make a film you must Decid
```

```
e if its a thriller or a drama! As a drama the movie is watchable. Parents are di
vorcing & arguing like in real life. And then we have Jake with his closet which
totally ruins all the film! I expected to see a BOOGEYMAN similar movie, and inst
ead i watched a drama with some meaningless thriller spots.<br /><br />3 out of 1
0 just for the well playing parents & descent dialogs. As for the shots with Jak
e: just ignore them.

negative

Petter Mattei's "Love in the Time of Money" is a visually stunning film to watch.
Mr. Mattei offers us a vivid portrait about human relations. This is a movie that
seems to be telling us what money, power and success do to people in the differen
t situations we encounter. <br /><br />This being a variation on the Arthur Schni
tzler's play about the same theme, the director transfers the action to the prese
nt time New York where all these different characters meet and connect. Each one
is connected in one way, or another to the next person, but no one seems to know
the previous point of contact. Stylishly, the film has a sophisticated luxurious
look. We are taken to see how these people live and the world they live in their
own habitat.<br /><br />The only thing one gets out of all these souls in the pic
ture is the different stages of loneliness each one inhabits. A big city is not e
xactly the best place in which human relations find sincere fulfillment, as one d
iscerns is the case with most of the people we encounter.<br /><br />The acting i
s good under Mr. Mattei's direction. Steve Buscemi, Rosario Dawson, Carol Kane, M
ichael Imperioli, Adrian Grenier, and the rest of the talented cast, make these c
haracters come alive.<br /><br />We wish Mr. Mattei good luck and await anxiously
for his next work.

positive
```

In [9]:
```python
def no_of_words(text):
    words=text.split()
    wor_count=len(words)
    return wor_count
```

In [10]:
```python
df["word_count"]=df["review"].apply(no_of_words)
```

In [11]:
```python
df.head()
```

Out[11]:

	review	sentiment	word_count
0	One of the other reviewers has mentioned that ...	positive	307
1	A wonderful little production. <br /><br />The...	positive	162
2	I thought this was a wonderful way to spend ti...	positive	166
3	Basically there's a family where a little boy ...	negative	138
4	Petter Mattei's "Love in the Time of Money" is...	positive	230

In [12]:
```python
df["sentiment"].replace("positive",0,inplace=True)
```

In [13]:
```python
df["sentiment"].replace("negative",1,inplace=True)
```

In [14]:
```python
df.head()
```

Out[14]:

	review	sentiment	word_count
0	One of the other reviewers has mentioned that ...	0	307
1	A wonderful little production. <br /><br />The...	0	162
2	I thought this was a wonderful way to spend ti...	0	166
3	Basically there's a family where a little boy ...	1	138
4	Petter Mattei's "Love in the Time of Money" is...	0	230

In [15]:
```
def data_processing(text):
    text=text.lower()
    text = re.sub(r"<br\s*/?>", " ", text)
    text=re.sub(r"https\S+|www\S+|http\S+","",text)
    text=re.sub(r"@w+|/#","",text)
    text=re.sub(r"[^\w\s]"," ",text)
    tokens = word_tokenize(text)
    filtered_tokens=(w for w in tokens if w not in stop_words)
    return " ".join(filtered_tokens)
```

In [16]:
```
df["review"]=df["review"].apply(data_processing)
```

In [17]:
```
df.duplicated().sum()
```

Out[17]:
```
422
```

In [18]:
```
df=df.drop_duplicates("review")
```

In [19]:
```
df.duplicated().sum()
```

Out[19]:
```
0
```

In [20]:
```
stemmer=PorterStemmer()
def stemming(data):
    text=[stemmer.stem(word) for word in data]
    return data
```

In [21]:
```
df.review =df["review"].apply(lambda x: stemming (x))
```

In [22]:
```
df["word_count"]=df["review"].apply(no_of_words)
```

In [23]:
```
df.head()
```

Out[23]:

	review	sentiment	word_count
0	one reviewers mentioned watching 1 oz episode ...	0	163
1	wonderful little production filming technique ...	0	86
2	thought wonderful way spend time hot summer we...	0	85
3	basically family little boy jake thinks zombie...	1	66
4	petter mattei love time money visually stunnin...	0	125

In [24]:
```
pos_re = df[df.sentiment==0]
pos_re.head()
```

Out[24]:

	review	sentiment	word_count
0	one reviewers mentioned watching 1 oz episode ...	0	163
1	wonderful little production filming technique ...	0	86
2	thought wonderful way spend time hot summer we...	0	85
4	petter mattei love time money visually stunnin...	0	125
5	probably time favorite movie story selflessnes...	0	56

In [25]:
```
neg_re = df[df.sentiment==1]
neg_re.head()
```

Out[25]:

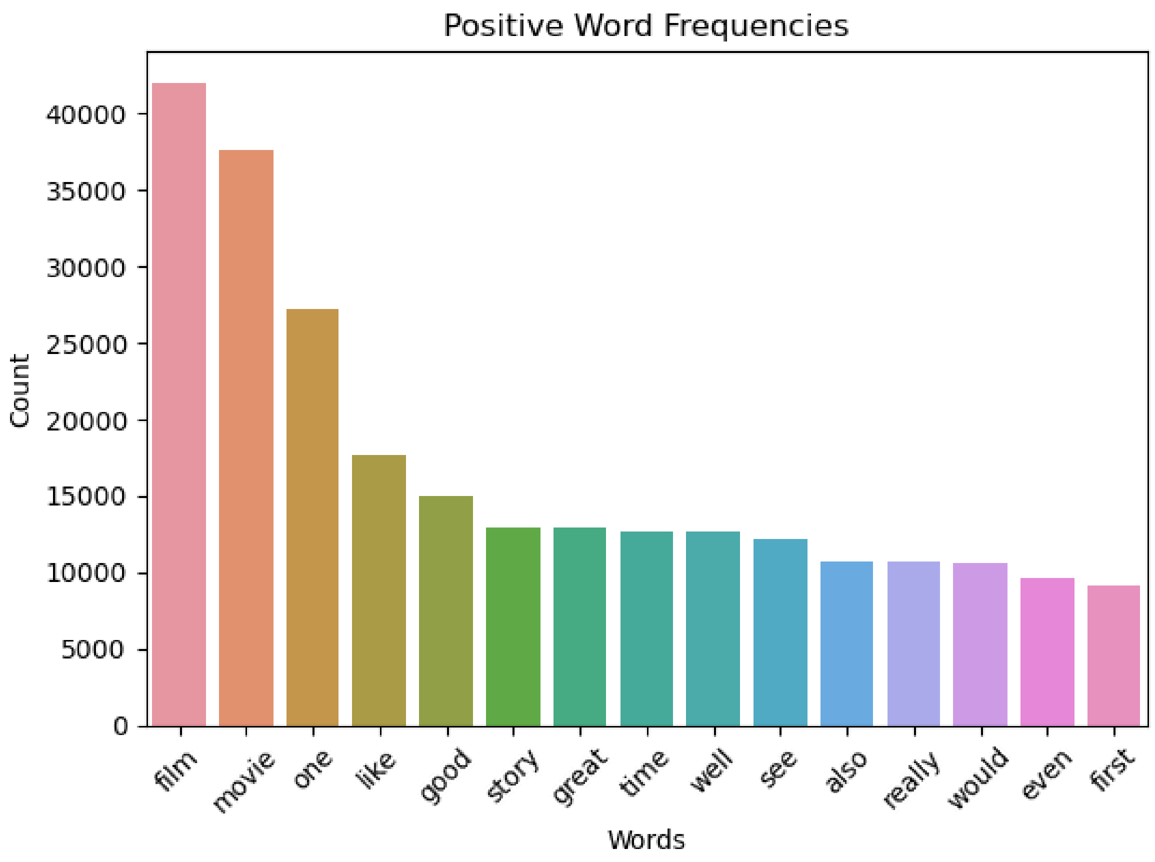
	review	sentiment	word_count
3	basically family little boy jake thinks zombie...	1	66
7	show amazing fresh innovative idea 70 first ai...	1	83
8	encouraged positive comments film looking forw...	1	64
10	phil alien one quirky films humour based aroun...	1	50
11	saw movie 12 came recall scariest scene big bi...	1	84

In [26]:
```
from collections import Counter
count=Counter()
```

In [27]:
```
for text in pos_re["review"].values:
    for word in text.split():
        count[word] += 1
```

In [28]:
```
pos_words=pd.DataFrame(count.most_common(15))
pos_words.columns = ["words","count"]
```

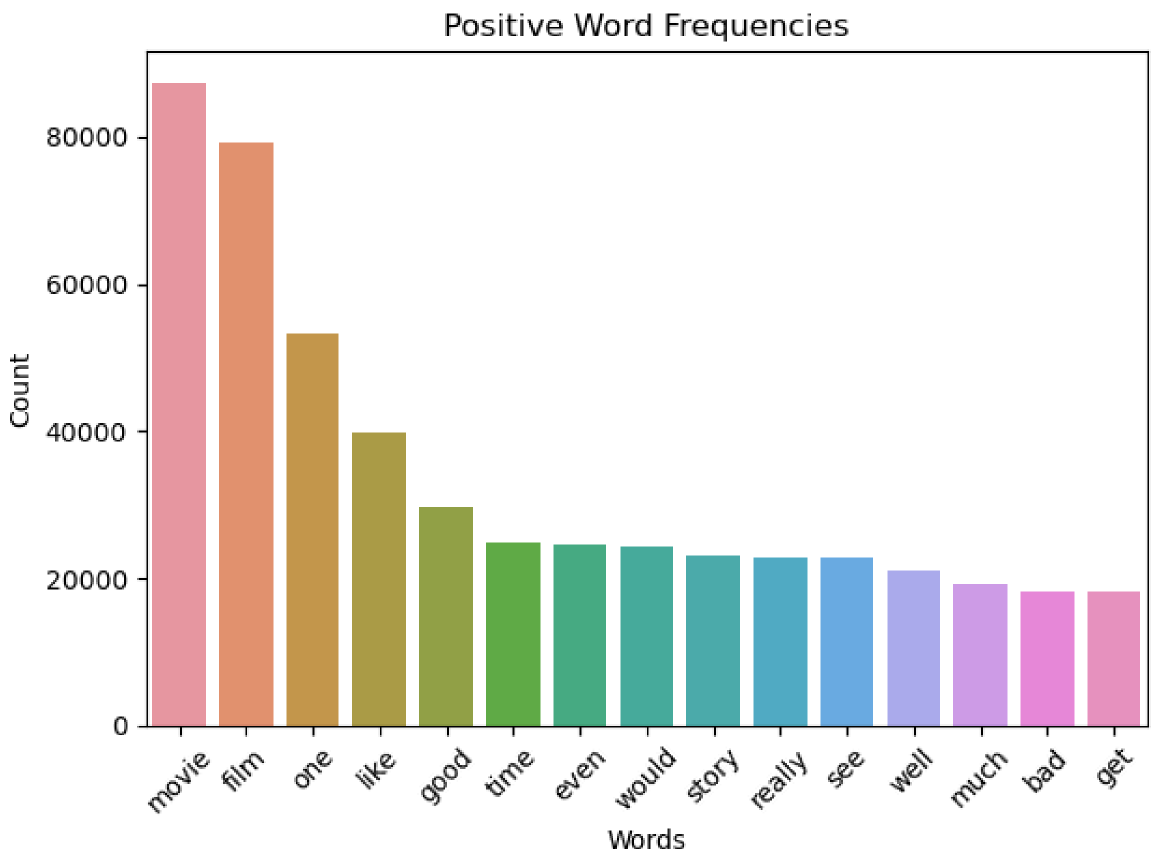
In [29]:
```
sns.barplot(x="words",y="count",data=pos_words)
plt.title("Positive Word Frequencies")
plt.xlabel("Words")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [30]:
```
for text in neg_re["review"].values:
    for word in text.split():
        count[word] += 1
```

In [31]:
```
neg_words=pd.DataFrame(count.most_common(15))
neg_words.columns = ["words","count"]
```

In [32]:
```
sns.barplot(x="words",y="count",data=neg_words)
plt.title("Positive Word Frequencies")
plt.xlabel("Words")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [33]:
```
x=df["review"]
y=df["sentiment"]
```

In [34]:
```
vect=TfidfVectorizer()
```

In [35]:
```
x=vect.fit_transform(df["review"])
```

In [36]:
```
x_train, x_test, y_train, y_split = train_test_split(x,y,test_size=0.2, random_s
```

In [37]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, f1_score, classification_report, con
import warnings
warnings.filterwarnings("ignore")
```

In [38]:
```
modelLR=LogisticRegression()
modelLR.fit(x_train,y_train)
y_predLR=modelLR.predict(x_test)
LRacc=accuracy_score(y_predLR,y_split)
LRacc
```

Out[38]:
```
0.8966212808875441
```

In [39]:
```
print(confusion_matrix(y_split,y_predLR))
print(classification_report(y_split,y_predLR))
```

```
[[4559  435]
 [ 590 4331]]
              precision    recall  f1-score   support

           0       0.89      0.91      0.90      4994
           1       0.91      0.88      0.89      4921

    accuracy                           0.90      9915
   macro avg       0.90      0.90      0.90      9915
weighted avg       0.90      0.90      0.90      9915
```

In [40]:
```
modelNB=MultinomialNB()
modelNB.fit(x_train,y_train)
y_predNB=modelNB.predict(x_test)
NBacc=accuracy_score(y_predNB,y_split)
NBacc
```

Out[40]:
```
0.8658598083711548
```

In [41]:
```
print(confusion_matrix(y_split,y_predNB))
print(classification_report(y_split,y_predNB))
```

```
[[4310  684]
 [ 646 4275]]
              precision    recall  f1-score   support

           0       0.87      0.86      0.87      4994
           1       0.86      0.87      0.87      4921

    accuracy                           0.87      9915
   macro avg       0.87      0.87      0.87      9915
weighted avg       0.87      0.87      0.87      9915
```

In [42]:
```
modelSVC=LinearSVC()
modelSVC.fit(x_train,y_train)
y_predSVC=modelSVC.predict(x_test)
SVCacc=accuracy_score(y_predSVC,y_split)
SVCacc
```

Out[42]:
```
0.897327281896117
```

In [43]:
```
print(confusion_matrix(y_split,y_predSVC))
print(classification_report(y_split,y_predSVC))
```

```
[[4542  452]
 [ 566 4355]]
              precision    recall  f1-score   support

           0       0.89      0.91      0.90      4994
           1       0.91      0.88      0.90      4921

    accuracy                           0.90      9915
   macro avg       0.90      0.90      0.90      9915
weighted avg       0.90      0.90      0.90      9915
```

In [44]:
```
from sklearn.model_selection import GridSearchCV
param_grid = {
    'C': [0.1, 1, 10, 100],
    'loss': ['hinge', 'squared_hinge']
}
grid = GridSearchCV(LinearSVC(), param_grid, refit=True, verbose=3)
grid.fit(x_train, y_train)
```

```
Fitting 5 folds for each of 8 candidates, totalling 40 fits
[CV 1/5] END ................C=0.1, loss=hinge;, score=0.886 total time=   0.0s
[CV 2/5] END ................C=0.1, loss=hinge;, score=0.885 total time=   0.0s
[CV 3/5] END ................C=0.1, loss=hinge;, score=0.881 total time=   0.0s
[CV 4/5] END ................C=0.1, loss=hinge;, score=0.882 total time=   0.0s
[CV 5/5] END ................C=0.1, loss=hinge;, score=0.877 total time=   0.0s
[CV 1/5] END ........C=0.1, loss=squared_hinge;, score=0.896 total time=   0.1s
[CV 2/5] END ........C=0.1, loss=squared_hinge;, score=0.898 total time=   0.1s
[CV 3/5] END ........C=0.1, loss=squared_hinge;, score=0.894 total time=   0.1s
[CV 4/5] END ........C=0.1, loss=squared_hinge;, score=0.894 total time=   0.1s
[CV 5/5] END ........C=0.1, loss=squared_hinge;, score=0.890 total time=   0.1s
[CV 1/5] END ..................C=1, loss=hinge;, score=0.894 total time=   0.3s
[CV 2/5] END ..................C=1, loss=hinge;, score=0.901 total time=   0.7s
[CV 3/5] END ..................C=1, loss=hinge;, score=0.895 total time=   0.6s
[CV 4/5] END ..................C=1, loss=hinge;, score=0.894 total time=   0.3s
[CV 5/5] END ..................C=1, loss=hinge;, score=0.890 total time=   0.3s
[CV 1/5] END ..........C=1, loss=squared_hinge;, score=0.893 total time=   0.3s
[CV 2/5] END ..........C=1, loss=squared_hinge;, score=0.896 total time=   0.3s
[CV 3/5] END ..........C=1, loss=squared_hinge;, score=0.891 total time=   0.3s
[CV 4/5] END ..........C=1, loss=squared_hinge;, score=0.889 total time=   0.3s
[CV 5/5] END ..........C=1, loss=squared_hinge;, score=0.891 total time=   0.3s
[CV 1/5] END .................C=10, loss=hinge;, score=0.874 total time=   1.4s
[CV 2/5] END .................C=10, loss=hinge;, score=0.876 total time=   2.2s
[CV 3/5] END .................C=10, loss=hinge;, score=0.872 total time=   1.3s
[CV 4/5] END .................C=10, loss=hinge;, score=0.870 total time=   4.7s
[CV 5/5] END .................C=10, loss=hinge;, score=0.870 total time=   1.6s
[CV 1/5] END .........C=10, loss=squared_hinge;, score=0.877 total time=   1.3s
[CV 2/5] END .........C=10, loss=squared_hinge;, score=0.878 total time=   1.0s
[CV 3/5] END .........C=10, loss=squared_hinge;, score=0.873 total time=   1.1s
[CV 4/5] END .........C=10, loss=squared_hinge;, score=0.875 total time=   1.1s
[CV 5/5] END .........C=10, loss=squared_hinge;, score=0.874 total time=   1.0s
[CV 1/5] END ................C=100, loss=hinge;, score=0.870 total time=   1.9s
[CV 2/5] END ................C=100, loss=hinge;, score=0.871 total time=   1.6s
[CV 3/5] END ................C=100, loss=hinge;, score=0.867 total time=   1.8s
[CV 4/5] END ................C=100, loss=hinge;, score=0.868 total time=   4.8s
[CV 5/5] END ................C=100, loss=hinge;, score=0.867 total time=   1.8s
[CV 1/5] END ........C=100, loss=squared_hinge;, score=0.870 total time=   1.5s
[CV 2/5] END ........C=100, loss=squared_hinge;, score=0.873 total time=   2.4s
[CV 3/5] END ........C=100, loss=squared_hinge;, score=0.868 total time=   1.4s
[CV 4/5] END ........C=100, loss=squared_hinge;, score=0.869 total time=   4.2s
[CV 5/5] END ........C=100, loss=squared_hinge;, score=0.868 total time=   1.8s
```

Out[44]:

```
▸     GridSearchCV
▸ estimator: LinearSVC
      ▸ LinearSVC
```

In [45]:

```
print("Best Parameters:", grid.best_params_)
print("Best Score:", grid.best_score_)
```

```
Best Parameters: {'C': 1, 'loss': 'hinge'}
Best Score: 0.894825605339243
```

In [46]:
```
modelSVC=LinearSVC(C=1, loss = "hinge")
modelSVC.fit(x_train,y_train)
y_predSVC=modelSVC.predict(x_test)
SVCacc=accuracy_score(y_predSVC,y_split)
SVCacc
```

Out[46]:
```
0.8994452849218356
```

In [47]:
```
print(confusion_matrix(y_split,y_predSVC))
print(classification_report(y_split,y_predSVC))
```

```
[[4550  444]
 [ 553 4368]]
              precision    recall  f1-score   support

           0       0.89      0.91      0.90      4994
           1       0.91      0.89      0.90      4921

    accuracy                           0.90      9915
   macro avg       0.90      0.90      0.90      9915
weighted avg       0.90      0.90      0.90      9915
```

In [ ]:
```
new_review = input("Enter Movie Review: ")
processed_review = data_processing(new_review)
new_review_vector = vect.transform([processed_review])
prediction = modelSVC.predict(new_review_vector)
if prediction == 0:
    print("Positive")
else:
    print("negative")
```

In [ ]:
```

```