CSE6060

Statistical Natural Language Processing

Sentiment Analysis

Name : Kavianand G

Reg. No. : 19MAI0050

Date : 27 – June – 2020

Loading Dataset

In [2]:

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer, TfidfVec
```

In [5]:

```
df = pd.read_csv("IMDB Dataset.csv")
```

In [6]:

```
# After loading, lets introspect this dataset
df.head(10)
```

Out[6]:

	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
5	Probably my all-time favorite movie, a story o...	positive
6	I sure would like to see a resurrection of a u...	positive
7	This show was an amazing, fresh & innovative i...	negative
8	Encouraged by the positive comments about this...	negative
9	If you like original gut wrenching laughter yo...	positive

In [7]:

```python
df['review'][0]
```

Out[7]:

```
"One of the other reviewers has mentioned that after watching just 1 Oz ep
isode you'll be hooked. They are right, as this is exactly what happened w
ith me.<br /><br />The first thing that struck me about Oz was its brutali
ty and unflinching scenes of violence, which set in right from the word G
O. Trust me, this is not a show for the faint hearted or timid. This show
pulls no punches with regards to drugs, sex or violence. Its is hardcore,
in the classic use of the word.<br /><br />It is called OZ as that is the
nickname given to the Oswald Maximum Security State Penitentary. It focuse
s mainly on Emerald City, an experimental section of the prison where all
the cells have glass fronts and face inwards, so privacy is not high on th
e agenda. Em City is home to many..Aryans, Muslims, gangstas, Latinos, Chr
istians, Italians, Irish and more....so scuffles, death stares, dodgy deal
ings and shady agreements are never far away.<br /><br />I would say the m
ain appeal of the show is due to the fact that it goes where other shows w
ouldn't dare. Forget pretty pictures painted for mainstream audiences, for
get charm, forget romance...OZ doesn't mess around. The first episode I ev
er saw struck me as so nasty it was surreal, I couldn't say I was ready fo
r it, but as I watched more, I developed a taste for Oz, and got accustome
```

Transforming Documents into Feature Vectors

In [8]:

```python
count = CountVectorizer() # initialize Count Vectorizer
docs = np.array(['The sun is shining',
                 'The weather is sweet',
                 'The sun is shining, the weather is sweet, and one and one is two'])
bag = count.fit_transform(docs)
print(count.vocabulary_)
print("\n")
print(bag.toarray())
```

```
{'the': 6, 'sun': 4, 'is': 1, 'shining': 3, 'weather': 8, 'sweet': 5, 'and':
0, 'one': 2, 'two': 7}


[[0 1 0 1 1 0 1 0 0]
 [0 1 0 0 0 1 1 0 1]
 [2 3 2 1 1 1 2 1 1]]
```

Term Frequency and Inverse Document

In [9]:

```
tfidf = TfidfTransformer(use_idf = True, norm='l2', smooth_idf=True)
np.set_printoptions(precision=2)
print(tfidf.fit_transform(count.fit_transform(docs)).toarray())
```

```
[[0.   0.43 0.   0.56 0.56 0.   0.43 0.   0.  ]
 [0.   0.43 0.   0.   0.   0.56 0.43 0.   0.56]
 [0.5  0.45 0.5  0.19 0.19 0.19 0.3  0.25 0.19]]
```

```
# Tokenization of Documents
```

In [10]:

```
from nltk.stem.porter import PorterStemmer
porter = PorterStemmer()

def stemmer_tokenize(text):
    return [porter.stem(word) for word in text.split()]

stemmer_tokenize('coders like coding and thus they code')
```

Out[10]:

```
['coder', 'like', 'code', 'and', 'thu', 'they', 'code']
```

Vectorising the Dataset

In [11]:

```
tfidf = TfidfVectorizer(strip_accents = None,
                        lowercase = False,
                        tokenizer = stemmer_tokenize,
                        use_idf = True,
                        norm = 'l2',
                        smooth_idf = True)

Y = df.sentiment.values
X = tfidf.fit_transform(df.review)
```

Document Classification Using Logistic regression

In [12]:

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegressionCV
import pickle

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state = 1, test_size =

# Model
clf = LogisticRegressionCV(cv = 5,
                           scoring = 'accuracy',
                           random_state=0,
                           n_jobs=-3,
                           verbose=3,
                           max_iter=300).fit(X_train, Y_train)

# saving the model
saved_model = open('saved_model.sav','wb')

# using the pickle library's dump function to write the trained classifier to the file
pickle.dump(clf, saved_model)
saved_model.close()

```

```
[Parallel(n_jobs=-3)]: Using backend LokyBackend with 10 concurrent workers.
[Parallel(n_jobs=-3)]: Done   2 out of   5 | elapsed:  2.8min remaining:  4.
2min
[Parallel(n_jobs=-3)]: Done   5 out of   5 | elapsed:  3.0min finished
```

Model Evaluation

In [13]:

```python
filename = 'saved_model.sav'
saved_clf = pickle.load(open(filename, 'rb'))

# test the saved model on the test data
saved_clf.score(X_test, Y_test)
#saved_clf.close()
```

Out[13]:

```
0.8898
```

In [ ]:

```python

```