Applied Machine Learning Homework 5

Due 12 Dec,2022 (Monday) 11:59PM EST

Your Name: Liwen Zhu

Your UNI: lz2512

Natural Language Processing

We will train a supervised model to predict if a movie has a positive or a negative review.

Dataset loading & dev/test splits

1.0) Load the movie reviews dataset from NLTK library

In [1]:
```
import nltk
nltk.download("movie_reviews")
import pandas as pd
from nltk.corpus import twitter_samples
from sklearn.model_selection import train_test_split
from nltk.corpus import stopwords
nltk.download('stopwords')
nltk.download('punkt')
stop = stopwords.words('english')
import string
import re
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
[nltk_data] Downloading package movie_reviews to
[nltk_data]     /Users/alanzhu/nltk_data...
[nltk_data]   Package movie_reviews is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/alanzhu/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /Users/alanzhu/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

In [2]:
```
from nltk.corpus import movie_reviews
```

In [3]:
```
negative_fileids = movie_reviews.fileids('neg')
positive_fileids = movie_reviews.fileids('pos')

pos_document = [(' '.join(movie_reviews.words(file_id)),category) for file_i
neg_document = [(' '.join(movie_reviews.words(file_id)),category) for file_i

# List of postive and negative reviews
pos_list = [pos[0] for pos in pos_document]
neg_list = [neg[0] for neg in neg_document]
```

1.1) Make a data frame that has reviews and its label

In [4]:
```
# code here
movie = pd.DataFrame(pos_document+neg_document,columns=["Review","Label"])
```

1.2 look at the class distribution of the movie reviews

In [5]:
```
# code here
movie["Label"].value_counts()
```

Out[5]:
```
pos    1000
neg    1000
Name: Label, dtype: int64
```

1.3) Create a development & test split (80/20 ratio):

In [6]:
```
# code here
X_dev,X_test,y_dev,y_test = train_test_split(movie["Review"],movie["Label"],
```

Data preprocessing

We will do some data preprocessing before we tokenize the data. We will remove `#` symbol, hyperlinks, stop words & punctuations from the data. You may use `re` package for this.

1.4) Replace the `#` symbol with '' in every review

In [7]:
```
# code here
X_dev = X_dev.str.replace('#','')
X_test = X_test.str.replace('#','')
```

1.5) Replace hyperlinks with '' in every review

In [8]:
```
# code here
```

1.6) Remove all stop words

In [9]:
```
# code here
for word in stop:
    X_dev = X_dev.str.replace(' '+word+' ',' ').replace(' '+word,' ').replac
    X_test = X_test.str.replace(' '+word+' ',' ').replace(' '+word,' ').repl
```

1.7) Remove all punctuations

In [10]:
```
# code here
for punctuation in string.punctuation:
    X_dev = X_dev.str.replace(punctuation,' ')
    X_test = X_test.str.replace(punctuation,' ')
```

```
/var/folders/b0/kn5sds75613b8_2rx88jfxf80000gn/T/ipykernel_81963/3922321045.
py:3: FutureWarning: The default value of regex will change from True to Fal
se in a future version. In addition, single character regular expressions wi
ll *not* be treated as literal strings when regex=True.
  X_dev = X_dev.str.replace(punctuation,' ')
/var/folders/b0/kn5sds75613b8_2rx88jfxf80000gn/T/ipykernel_81963/3922321045.
py:4: FutureWarning: The default value of regex will change from True to Fal
se in a future version. In addition, single character regular expressions wi
ll *not* be treated as literal strings when regex=True.
  X_test = X_test.str.replace(punctuation,' ')
```

1.8) Apply stemming on the development & test datasets using Porter algorithm

In [11]:
```
#code here
def stemSentence(sentece):
    porter = PorterStemmer()
    token_words = word_tokenize(sentece)
    stem_sentence = [porter.stem(word) for word in token_words]
    return " ".join(stem_sentence)

for index,sentence in X_dev.iteritems():
    X_dev[index] = stemSentence(sentence)
for index,sentence in X_test.iteritems():
    X_test[index] = stemSentence(sentence)
```

Model training

1.9) Create bag of words features for each review in the development dataset

In [12]:
```
#code here
vector = CountVectorizer(stop_words = 'english')
X_dev_bow = vector.fit_transform(X_dev)
feature_names = vector.get_feature_names()
print(feature_names[:10])
print(feature_names[10000:10020])
print(feature_names[::5000])
```

```
['00', '000', '0009f', '007', '03', '04', '05', '05425', '10', '100']
['hype', 'hyper', 'hyperact', 'hyperbol', 'hyperdr', 'hyperjump', 'hyperkine
t', 'hypernatur', 'hyperr', 'hypersleep', 'hyperspe', 'hyperviol', 'hypnos
i', 'hypnot', 'hypnotherapist', 'hypnotis', 'hypnotist', 'hypochondriac', 'h
ypocrisi', 'hypocrit']
['00', 'dah', 'hype', 'painter', 'sturm']
/Users/alanzhu/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/depre
cation.py:87: FutureWarning: Function get_feature_names is deprecated; get_f
eature_names is deprecated in 1.0 and will be removed in 1.2. Please use get
_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

1.10) Train a Logistic Regression model on the development dataset

In [13]:
```
#code here
lr = LogisticRegression().fit(X_dev_bow,y_dev)
```

```
/Users/alanzhu/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_mode
l/_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion
  n_iter_i = _check_optimize_result(
```

1.11) Create TF-IDF features for each review in the development dataset

In [14]:
```
#code here
vector_tf = TfidfVectorizer()
X_dev_tf = vector_tf.fit_transform(X_dev)
feature_names_tf = vector_tf.get_feature_names()
print(feature_names_tf[:10])
```

```
print(feature_names_tf[10000:10020])
print(feature_names_tf[::5000])
```

```
['00', '000', '0009f', '007', '03', '04', '05', '05425', '10', '100']
['humil', 'humili', 'hummabl', 'hummana', 'hummer', 'hummingbird', 'hummm',
'humor', 'humorist', 'humorless', 'humour', 'humourless', 'hump', 'humpalo
t', 'humphrey', 'humphri', 'humve', 'hun', 'hunch', 'hunchback']
['00', 'cyborsuit', 'humil', 'overfil', 'straddl']
```

```
/Users/alanzhu/opt/anaconda3/lib/python3.9/site-packages/sklearn/utils/depre
cation.py:87: FutureWarning: Function get_feature_names is deprecated; get_f
eature_names is deprecated in 1.0 and will be removed in 1.2. Please use get
_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

1.12) Train the Logistic Regression model on the development dataset with TF-IDF features

In [15]:
```
#code here
lr_tf = LogisticRegression().fit(X_dev_tf,y_dev)
```

1.13) Compare the performance of the two models on the test dataset. Explain the difference in results obtained?

In [16]:
```
#code here
X_test_lr = vector.transform(X_test)
X_test_tf = vector_tf.transform(X_test)
print(f"The performance of bag of words is {lr.score(X_test_lr,y_test)}")
print(f"The performance of tf-idf is {lr_tf.score(X_test_tf,y_test)}")
print("The bag of words weights more on the high frequency word. \
On the other hand, the TF-IDF model weights more on the unique word. \
The two models weight words differetly, which causes the difference in resul
```

```
The performance of bag of words is 0.835
The performance of tf-idf is 0.8225
The bag of words weights more on the high frequency word. On the other hand,
the TF-IDF model weights more on the unique word. The two models weight word
s differetly, which causes the difference in results.
```

In [ ]: