

Co544-Machine Learning and Data Mining
Introduction to text classification

TODO 1: Find data preprocessing steps other than mentioned above.

- Remove HTML tags.
- Remove extra whitespaces.
- Convert accented characters to ASCII characters.
- Expand contractions.
- Lowercase all texts.
- Convert number words to numeric form.
- Remove numbers.
- Tokenization — convert sentences to words
- Removing unnecessary punctuation, tags
- Removing stop words — frequent words such as "the", "is", etc. that do not have specific semantic
- Stemming — words are reduced to a root by removing inflection through dropping unnecessary characters, usually a suffix.
- Lemmatization — Another approach to remove inflection by determining the part of speech and utilizing a detailed database of the language.

TODO 2: Discuss advantages and disadvantages of 'Bag of Words model'

Advantages:

- Very simple to understand and implement.

Disadvantages:

- This method discards word order thereby ignoring the context and in turn meaning of words in the document. To solve this problem we have to use another approach called Word Embedding.
- Bag of words leads to a high dimensional feature vector due to the large size of Vocabulary, V.
- Bag of words doesn't leverage co-occurrence statistics between words. In other words, it assumes all words are independent of each other.
- It leads to a highly sparse vector as there is nonzero value in dimensions corresponding to words that occur in the sentence.

TODO 3: Train a Random Forest model, a Support Vector Machine model and a Naive Bayesian classifier. Compare the accuracies of four models including the Logistic Regression model. What is the best model? Justify your answer.

Importing libraries

```
import numpy as np
import re
import nltk
from sklearn.datasets import load_files
nltk.download('stopwords')
import pickle
from nltk.corpus import stopwords
nltk.download('wordnet')
```

importing the dataset

```
movie_data = load_files(r"txt_sentoken")
X, y = movie_data.data, movie_data.target
```

text preprocessing

```
documents = []

from nltk.stem import WordNetLemmatizer

stemmer = WordNetLemmatizer()

for sen in range(0, len(X)):

    document = re.sub(r'\W', ' ', str(X[sen])) # Remove all the special characters
    document = re.sub(r'\s+[a-zA-Z]\s+', ' ', document) #remove all single characters
    document = re.sub(r'\^[a-zA-Z]\s+', ' ', document) #Remove single characters
                                                from the start
    document = re.sub(r'\s+', ' ', document, flags=re.I) # Substituting multiple
                                                         spaces with single space
    document = re.sub(r'^\b\s+', '', document) # Removing prefixed 'b'
    document = document.lower() # Converting to Lowercase

    # Lemmatization
    document = document.split()

    document = [stemmer.lemmatize(word) for word in document]
    document = ' '.join(document)

    documents.append(document)
```

converting text to numbers(BoW)

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(max_features=1500, min_df=5, max_df=0.7, stop_words =
stopwords.words('english'))
X = vectorizer.fit_transform(documents).toarray()
```

Finding TFIDF

```
from sklearn.feature_extraction.text import TfidfTransformer
tfidfconverter = TfidfTransformer()
X = tfidfconverter.fit_transform(X).toarray()
```

split into train and test data sets

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0)
```

Training Text Classification Model and predicting

- Random forest model

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators=1000, random_state=0)
classifier.fit(X_train, y_train)

# predict the sentiment for the documents in the test set
y_pred = classifier.predict(X_test)

# Evaluating the Model
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

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

```
[[180 28]
 [ 30 162]]
      precision    recall  f1-score   support

     0       0.86      0.87      0.86       208
     1       0.85      0.84      0.85       192

 accuracy          0.85          400
 macro avg       0.85      0.85      0.85          400
weighted avg       0.85      0.85      0.85          400

0.855
```

- Support vector machine model

```
from sklearn import svm #import svm model

clf = svm.SVC(kernel='linear') # create a svm classifier
clf.fit(X_train, y_train) # train the model using the train data set
y_pred = clf.predict(X_test) #predict the response for test data set
print(accuracy_score(y_test,y_pred))
```

Accuracy:-

```
0.8375

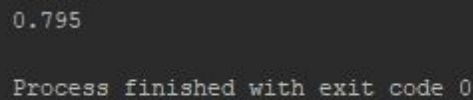
Process finished with exit code 0
```

- **Naïve Bayesian Classifier**

```
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
nb = MultinomialNB()
nb.fit(X_train, y_train)

# Now we can use the model to predict classifications for our test features.
y_pred = nb.predict(X_test)
print(metrics.accuracy_score(y_test, y_pred))
```

Accuracy:-



```
0.795

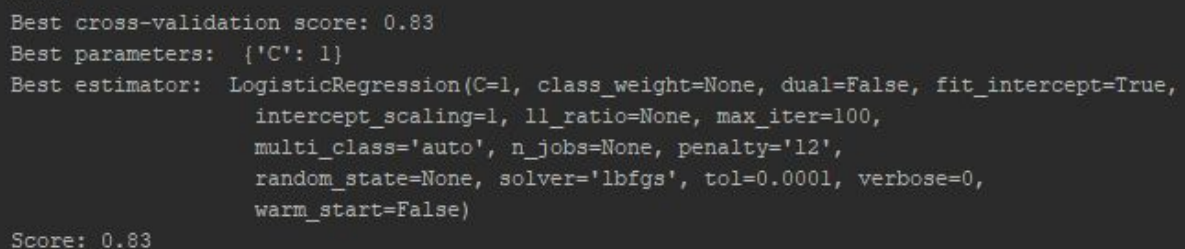
Process finished with exit code 0
```

- **Logistic Regression Model**

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
param_grid = {'C': [0.001, 0.01, 0.1, 1, 10]}
grid = GridSearchCV(LogisticRegression(), param_grid, cv=5)
grid.fit(X_train, y_train)

print("Best cross-validation score: {:.2f}".format(grid.best_score_))
print("Best parameters: ", grid.best_params_)
print("Best estimator: ", grid.best_estimator_)

lr = grid.best_estimator_
lr.fit(X_train, y_train)
lr.predict(X_test)
print("Score: {:.2f}".format(lr.score(X_test, y_test)))
```



```
Best cross-validation score: 0.83
Best parameters: {'C': 1}
Best estimator: LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, l1_ratio=None, max_iter=100,
multi_class='auto', n_jobs=None, penalty='l2',
random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
warm_start=False)
Score: 0.83
```

By considering all of the above text classification models following are the accuracy of each model:

- Random forest model = 0.855
- Support vector machine model = 0.8375
- Naïve Bayesian Classifier = 0.795
- Logistic Regression Model = 0.83

Therefore, the best model is the “Random Forest Model” because it has achieved 0.855 of Accuracy level. It is the highest accuracy level among other models. It builds multiple decision trees and merges them together to get a more accurate and stable prediction. Random Forest adds additional randomness to the model while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. It means random forest replaces the data used to construct the tree and also the explanatory variables are bootstrapped so that partition is not done on the same important variable. This results in a wide diversity that generally results in a better model. It increases predictive power of the algorithm and also helps prevent overfitting. Random forest is the most simple and widely used algorithm. Used for both classification and regression. That is why random forest becomes the best.