Assignment5.pyvb

파일 편집 보기 실행 커널 도구 설정 도움말 2025-05-19

Information

Writer : Jaehyeok Woo
Std_ID : 3244337
Assignment1 : Text classification using neural networks
Deadline : June 16, 2025

- Library

```
import matplotlib.pyplot as plt
import numpy as np
import os
import glob
import time
from keras.datasets import load_files
nltk.download('stopwords')
nltk.download('wordnet')
from nltk import *
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.model_selection import train_test_split

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

- Data

```
review_data = load_files('/content/drive/My Drive/Spring2025/Machine_Learning/Dev-Machinelearning/Assignment1/train_review')
X, y = review_data.data, review_data.target

documents = []

stemmer = WordNetLemmatizer()

for sen in range(0, len(X)):
    # Remove all the special characters
    document = re.sub(r'\W', ' ', str(X[sen]))

    # remove all single characters
    document = re.sub(r'\s+[a-zA-Z]\s+', ' ', document)

    # Remove single characters from the start
    document = re.sub(r'\^[a-zA-Z]\s+', ' ', document)

    # Substituting multiple spaces with single space
    document = re.sub(r'\s+', ' ', document, flags=re.I)

    # Removing prefixed 'b'
    document = re.sub(r'^b\s+', ' ', document)

    # Converting to lowercase
    document = document.lower()

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

    documents.append(document)

vectorizer = CountVectorizer(max_features=2000, min_df=5, max_df=0.7, stop_words=stopwords.words('english'))
X = vectorizer.fit_transform(documents).toarray()

tfidfconverter = TfidfTransformer()
X = tfidfconverter.fit_transform(X).toarray()

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, shuffle=False)
```

```
# Jae Hyeok
train_data = X_train.T
test_data = X_test.T
train_label = y_train
test_label = y_test

for sen in range(0, len(X)):
    # Remove all the special characters
    document = re.sub(r'\W', ' ', str(X[sen]))

    # remove all single characters
    document = re.sub(r'\s+[a-zA-Z]\s+', ' ', document)

    # Remove single characters from the start
    document = re.sub(r'\^[a-zA-Z]\s+', ' ', document)

    # Substituting multiple spaces with single space
    document = re.sub(r'\s+', ' ', document, flags=re.I)

    # Removing prefixed 'b'
    document = re.sub(r'^b\s+', ' ', document)

    # Converting to lowercase
    document = document.lower()

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

    documents.append(document)

vectorizer = CountVectorizer(max_features=2000, min_df=5, max_df=0.7, stop_words=stopwords.words('english'))
X = vectorizer.fit_transform(documents).toarray()

tfidfconverter = TfidfTransformer()
X = tfidfconverter.fit_transform(X).toarray()

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, shuffle=False)
```

```
# Jae Hyeok
train_data = X_train.T
test_data = X_test.T
train_label = y_train
test_label = y_test

for sen in range(0, len(X)):
    # Remove all the special characters
    document = re.sub(r'\W', ' ', str(X[sen]))

    # remove all single characters
    document = re.sub(r'\s+[a-zA-Z]\s+', ' ', document)

    # Remove single characters from the start
    document = re.sub(r'\^[a-zA-Z]\s+', ' ', document)

    # Substituting multiple spaces with single space
    document = re.sub(r'\s+', ' ', document, flags=re.I)

    # Removing prefixed 'b'
    document = re.sub(r'^b\s+', ' ', document)

    # Converting to lowercase
    document = document.lower()

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

    documents.append(document)

vectorizer = CountVectorizer(max_features=2000, min_df=5, max_df=0.7, stop_words=stopwords.words('english'))
X = vectorizer.fit_transform(documents).toarray()

tfidfconverter = TfidfTransformer()
X = tfidfconverter.fit_transform(X).toarray()

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, shuffle=False)
```

```
# Jae Hyeok
train_data = X_train.T
test_data = X_test.T
train_label = y_train
test_label = y_test
```

- Check the convergence

```
print("Iteration : %d%d%d")
print("Loss : %f%f%f", % (loss, ", ", second))
print("Loss diff : %f", % (loss[1] - loss[0]))
```

```
Iteration : 0200
Loss : 0.0000000000000000
Loss diff : 0.0000000000000000
```

- Predict

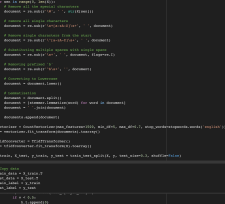
```
y_pred_test = predict(network, node, train_test, X_test, layer_num)
y_pred_train = predict(network, node, X_train, X_train, layer_num)
```

Result

- 1. Loss Curve

```
plt.figure(figsize=(8,6))
plt.plot(l_history[0], color='blue', label='Train Loss')
plt.plot(l_history[1], color='red', label='Test Loss')
plt.grid()
plt.legend()
plt.title('Loss Curve', fontsize=15)
plt.xlabel('Iteration', fontsize=12)
plt.ylabel('Error', fontsize=12)
```

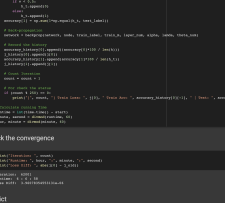
```
Text(0, 0.5, 'Error')
```



- 2. Accuracy Curve

```
plt.figure(figsize=(8,6))
plt.plot(accuracy_history[0], color='blue', label='Train Accuracy (%)')
plt.plot(accuracy_history[1], color='red', label='Test Accuracy (%)')
plt.grid()
plt.legend()
plt.title('Accuracy Curve', fontsize=15)
plt.xlabel('Iteration', fontsize=12)
plt.ylabel('Accuracy', fontsize=12)
```

```
Text(0, 0.5, 'Accuracy')
```



- 3. Quantitative results

1. Training results

```
print(confusion_matrix(y_train,y_pred_train))
print(classification_report(y_train,y_pred_train))
print(accuracy_score(y_train, y_pred_train))
```

```
[[479 28]
 [ 19 483]]
              precision    recall  f1-score   support

           0       0.97      0.94      0.97       507
           1       0.94      0.97      0.97       502

    accuracy                           0.97      1009
   macro avg       0.97      0.97      0.97      1009
weighted avg       0.97      0.97      0.97      1009

0.9693548387096774
```

2. Testing results

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

```
[[236 26]
 [ 12 226]]
              precision    recall  f1-score   support

           0       0.95      0.90      0.94       262
           1       0.90      0.95      0.93       238

    accuracy                           0.93       500
   macro avg       0.93      0.93      0.93       500
weighted avg       0.93      0.93      0.93       500

0.9320000000000000
```

```
[ ]
```