demo_news-class

September 25, 2023

[1]:
```
import os
import random
```

[10]:
```
import jieba
```

0.1 1.

[6]:
```
"""
  :
Parameters:
    folder_path -
    test_size -          20
Returns:
    all_words_list -
    train_data_list -
    test_data_list -
    train_class_list -
    test_class_list -
"""
def TextProcessing(folder_path, test_size=0.2):
    folder_list = os.listdir(folder_path)  #  folder_path
    data_list = []  #
    class_list = []  #

    #
    for folder in folder_list:
        new_folder_path = os.path.join(folder_path, folder)  #
        files = os.listdir(new_folder_path)  #     txt

        j = 1
        #   txt
        for file in files:
            if j > 100:  #  txt   100
                break
            with open(os.path.join(new_folder_path, file), 'r',␣
 ↪encoding='utf-8') as f:  #  txt
                raw = f.read()
```

```
            word_cut = jieba.cut(raw, cut_all=False)  #        generator
            word_list = list(word_cut)  # generator  list

            data_list.append(word_list)  #
            class_list.append(folder)  #
            j += 1

    data_class_list = list(zip(data_list, class_list))  # zip
    random.shuffle(data_class_list)  #  data_class_list
    index = int(len(data_class_list) * test_size) + 1  #
    train_list = data_class_list[index:]  #
    test_list = data_class_list[:index]  #
    train_data_list, train_class_list = zip(*train_list)  #
    test_data_list, test_class_list = zip(*test_list)  #

    all_words_dict = {}  #
    for word_list in train_data_list:
        for word in word_list:
            if word in all_words_dict.keys():
                all_words_dict[word] += 1
            else:
                all_words_dict[word] = 1

    #
    all_words_tuple_list = sorted(all_words_dict.items(), key=lambda f: f[1],␣
 ↪reverse=True)
    all_words_list, all_words_nums = zip(*all_words_tuple_list)  #
    all_words_list = list(all_words_list)  #
    return all_words_list, train_data_list, test_data_list, train_class_list,␣
 ↪test_class_list
```

[9]:
```
folder_path = './news'  #
all_words_list, train_data_list, test_data_list, train_class_list,␣
 ↪test_class_list = TextProcessing(folder_path,test_size=0.2)
```

[3]:
```
"""
  :
Parameters:
    words_file -
Returns:
    words_set -    set
"""
def MakeWordsSet(words_file):
    words_set = set()  #  set
    with open(words_file, 'r', encoding='utf-8') as f:  #
        for line in f.readlines():  #
```

```
            word = line.strip()  #
            if len(word) > 0:  #     words_set
                words_set.add(word)
    return words_set  #
```

[12]:
```
#  stopwords_set
stopwords_file = './stopwords_cn.txt'
stopwords_set = MakeWordsSet(stopwords_file)
```

0.2 2.

[ ]:
```
```

[4]:
```
"""
  :
Parameters:
    all_words_list -
    deleteN -     deleteN
    stopwords_set -
Returns:
    feature_words -
"""
def words_dict(all_words_list, deleteN, stopwords_set=set()):
    feature_words = []  #
    n = 1
    for t in range(deleteN, len(all_words_list), 1):
        if n > 1000:  # feature_words  1000
            break
            #                 1 5
        if not all_words_list[t].isdigit() and all_words_list[t] not in␣
 ↪stopwords_set and 1 < len(all_words_list[t]) < 5:
            feature_words.append(all_words_list[t])
        n += 1
    return feature_words
```

[13]:
```
feature_words = words_dict(all_words_list, 450, stopwords_set)
```

[15]:
```
"""
  : feature_words
Parameters:
    train_data_list -
    test_data_list -
    feature_words -
Returns:
    train_feature_list -
    test_feature_list -
"""
```

```
def TextFeatures(train_data_list, test_data_list, feature_words):
    def text_features(text, feature_words):  #      1
        text_words = set(text)
        features = [1 if word in text_words else 0 for word in feature_words]
        return features

    train_feature_list = [text_features(text, feature_words) for text in␣
 ↪train_data_list]
    test_feature_list = [text_features(text, feature_words) for text in␣
 ↪test_data_list]
    return train_feature_list, test_feature_list  #
```

[16]:
```
train_feature_list, test_feature_list = TextFeatures(train_data_list,␣
 ↪test_data_list, feature_words)
```

[ ]:

[ ]:

0.3 3. - scikit-learn

[ ]:
```
from sklearn.naive_bayes import MultinomialNB
```

[18]:
```
"""
  :
Parameters:
    train_feature_list -
    test_feature_list -
    train_class_list -
    test_class_list -
Returns:
    test_accuracy -
"""
def TextClassifier(train_feature_list, test_feature_list, train_class_list,␣
 ↪test_class_list):
    classifier = MultinomialNB().fit(train_feature_list, train_class_list)
    test_accuracy = classifier.score(test_feature_list, test_class_list)
    return test_accuracy
```

0.4 4.

[22]:
```
test_accuracy_list = []
test_accuracy = TextClassifier(train_feature_list, test_feature_list,␣
 ↪train_class_list, test_class_list)
test_accuracy_list.append(test_accuracy)
```

[23]:
```
ave = lambda c: sum(c) / len(c)
```

[24]:
```
print(ave(test_accuracy_list))
```

```
0.6842105263157895
```

[ ]:
```

```