黑客派

全网新闻数据类别整理与新闻文本分类

作者：lichao0x7cc

原文链接：https://hacpai.com/article/1493815899897

来源网站：黑客派

一、 数据获取与整理

1. 获取

搜狗实验室 http://www.sogou.com/labs/resource/ca.php

我下载的是精简版(一个月数据, 437MB) 与 完整版(711MB)

注意：因为我的机器硬件所限，我把我整理出的类别文件删除一些，防止引起python的内存错误。

2. 整理

利用Python 脚本 自动处理文本编码。

```
import os

file_list = os.listdir('.')
file_prefix = 'dd/'
for file_name in file_list:
    #os.system("cat " + file_name)
    os.system("cat " + file_name + " | " + "iconv -f gbk -t utf-8 -c" + " > " + "1" + file_name)
```

二、文本内容分类

1. 内容分类

文本格式

注意：content字段去除了HTML标签，保存的是新闻正文文本

类别标记说明(URL到类别的映射关系，3KB)

思路是利用URL 到类别的映射提取出对应类别的正文文本内容写入到对应类别的文件夹中，文件标题为新闻标题内容为正文文本。

2.代码

```
<!--more-->


#coding:utf-8
import os

# 获取文件类别与URL之间的映射关系
file_content = None

with open('categories_2012.txt') as f:
    file_content = f.readlines()

categories = {}
```

原文链接：全网新闻数据类别整理与新闻文本分类

```
f_item = None
for item in file_content:
    #print item
    if 'http' not in item[:-4] and ':  ' in item:
        if item[:-4] not in categories:
            categories[item[:-4]] = []
        f_item = item[:-4]
    elif 'http' in item and '前缀' not in item:
        #print item
        categories[f_item].append(item)

for k, v in categories.items():
    print k, v

#获取文本内容 并存储到相应的类别文件夹中
for label in categories.keys():
    try:
        os.mkdir(label.decode('utf-8'))
    except OSError:
        for list_file in os.listdir(label.decode('utf-8')):
            os.remove(label.decode('utf-8') + '/' + list_file)
        os.rmdir(label.decode('utf-8'))
        os.mkdir(label.decode('utf-8'))


for file_name in os.listdir('Sogou/'):
    with open('Sogou/' + file_name) as file:
        title = None
        for content in file:
            if 'contenttitle' in content:
                title = content[14:-16]
                #print title
            if 'url' in content:
                url_content = content[5:-7]

                #print url_content[: 8 + url_content[7:].find('/') + url_content[8 + url_content[7:].find('/'):].find('/
) + 1]
                #print url_content
                flag = False
                label = None
                for k, v in categories.items():
                    for url_list in v:
                        #print url_content, url_list
                        if url_list.strip() in url_content.strip():
                            #print 'ok'
                            #print url_content
                            #print url_content, url_list
                            #print 'ok'
                            flag = True
                            label = k
                            break
                    if flag:
                        break
                #print '-------------------------------------------'
            if '<content>' in content:
                x_content = content[9:-11]
                #print x_content
                #print label
                if not label:
```

```
            continue
        file_path = label + '/' + title + '.txt'
        print file_path.decode('utf-8')
        try:
            with open(file_path.decode('utf-8'),'w+') as label_f:
                label_f.write(x_content)
        except Exception:
            pass
```

三、分类、聚类算法处理

利用准确率与召回率 计算F1-SCORE
存疑，这样写正确么？ 希望有前辈看到给我发封电子邮件lichao0x7cc@gmail.com)

结果：

KNN算法处理

KMeans算法处理

代码

```
#coding:utf-8
from sklearn.cross_validation import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.cluster import KMeans
from sklearn.linear_model import SGDClassifier
from sklearn.datasets import load_files
from sklearn.feature_extraction.text import TfidfVectorizer
import jieba
import os


def jieba_cut(x):
    return jieba.cut(x)

sogou_Data = load_files('data/', encoding='utf-8')
data_train, data_test, label_train, label_test = train_test_split(sogou_Data.data, sogou_Data.target, test_si
e=0.2)

tfidf_model = TfidfVectorizer(binary=False, tokenizer=jieba_cut)
train_Data = tfidf_model.fit_transform(data_train)

clf = KNeighborsClassifier(n_neighbors=23).fit(train_Data, label_train)

test_Data = tfidf_model.transform(data_test)
pred = clf.predict(test_Data)
label_list_test = list(label_test)
#print label_test
correct = 0
#KNN
R = 0.0
R_score = {}

#print label_test
for label_pred, label_correct in zip(pred, label_list_test):
    print '预测结果: ',sogou_Data.target_names[label_pred]
```

```
    print '正确结果: ', sogou_Data.target_names[label_correct]
    print '\n'
    if sogou_Data.target_names[label_pred] == sogou_Data.target_names[label_correct]:
        correct += 1
        if sogou_Data.target_names[label_pred] not in R_score:
            R_score[sogou_Data.target_names[label_pred]] = 1
        else:
            R_score[sogou_Data.target_names[label_pred]] += 1

file_list = os.listdir('data/')
for dir_name in file_list:
    try:
        name_dir = dir_name.decode('gbk').encode('utf-8')
    except BaseException:
        pass
    for name in R_score.keys():
        if name_dir == name:
            R_score[sogou_Data.target_names[label_pred]] = R_score[sogou_Data.target_names[label_pred]]
 float(len(os.listdir('data/' + name_dir +'/')))


for v in R_score.values():
    R += v

P = correct / float(len(label_list_test))

F1 = (2*R*P) / (P + R)

print 'F1-Score: ', F1


#KMeans
clf = KMeans(n_clusters=23).fit(train_Data, None)
pred = clf.predict(test_Data)

correct = 0
R = 0.0
R_score = {}

#print label_test
for label_pred, label_correct in zip(pred, label_list_test):
    print '预测结果: ',sogou_Data.target_names[label_pred]
    print '正确结果: ', sogou_Data.target_names[label_correct]
    print '\n'
    if sogou_Data.target_names[label_pred] == sogou_Data.target_names[label_correct]:
        correct += 1
        if sogou_Data.target_names[label_pred] not in R_score:
            R_score[sogou_Data.target_names[label_pred]] = 1
        else:
            R_score[sogou_Data.target_names[label_pred]] += 1

file_list = os.listdir('data/')
for dir_name in file_list:
    try:
        name_dir = dir_name.decode('gbk').encode('utf-8')
    except BaseException:
        pass
    for name in R_score.keys():
        if name_dir == name:
```

```
            R_score[sogou_Data.target_names[label_pred]] = R_score[sogou_Data.target_names[label_pred]]
 float(len(os.listdir('data/' + name_dir +'/')))


for v in R_score.values():
    R += v

P = correct / float(len(label_list_test))

F1 = (2*R*P) / (P + R)

print 'F1-Score: ', F1
```

我的博客地址为： http://keysub.me