“智慧政务”中的文本挖掘应用

一、问题背景

近年来，随着微信、微博、市长信箱、阳光热线等网络问政平台逐步成为政府了解民意、汇聚民智、凝聚民气的重要渠道，各类社情民意相关的文本数据量不断攀升，给以往主要依靠人工来进行留言划分和热点整理的相关部门的工作带来了极大挑战。同时，随着大数据、云计算、人工智能等技术的发展，建立基于自然语言处理技术的智慧政务系统已经是社会治理创新发展的新趋势，对提升政府的管理水平和施政效率具有极大的推动作用。

二、解决问题 1——群众留言分类

针对在处理网络问政平台的群众留言时，存在工作量大、效率低且差错率高等问题下建立关于留言内容的一级标签分类模型。

步骤：程序首先读入附件二获取数据，若想要分类则需按照一定的词句进行归类，我们采用 jieba 分词对留言详情进行分词；

```
import jieba, time

train_df.columns = ['留言编号', '留言用户', '留言主题', '留言时间', '留言详情', '一级标签']
stopword_list = [k.strip() for k in open('stopwords.txt', encoding='utf-8') if k.strip() != '']
cutWords_list = []

i = 0
startTime = time.time()
for article in train_df['留言详情']:
    cutWords = [k for k in jieba.cut(article) if k not in stopword_list]
    i += 1
    # if i != 0:
    #     print('前%d篇留言分词共花费%.2f秒' % (i, time.time() - startTime))
    cutWords_list.append(cutWords)
```

分词后继续提取关键词（这是比较重要的一步），利用删除停止词来筛选出对我们有用的信息，提出关键词才可继续分类；

```
with open('cutWords_list.txt', 'w',encoding="utf-8") as file:
    for cutWords in cutWords_list:
        file.write(' '.join(cutWords) + '\n')

import warnings

warnings.filterwarnings('ignore')
from gensim.models import Word2Vec

word2vec_model = Word2Vec(cutWords_list, size=100, iter=10, min_count=20)
```

接着，对于每一条留言，获取留言的每一个分词在 word2vec 模型的相关性向量。然后把一条留言的所有分词在 word2vec 模型中的相关性向量求和取平均数，即该条留言在 word2vec 模型中的相关性向量（**用一条留言分词向量的平均数作为该留言在模型中的相关性向量**）；

```
import numpy as np


def getVector_v1(cutWords, word2vec_model):
    count = 0
    article_vector = np.zeros(word2vec_model.layer1_size)
    for cutWord in cutWords:
        if cutWord in word2vec_model:
            article_vector += word2vec_model[cutWord]
            count += 1

    return article_vector / count


startTime = time.time()
vector_list = []
i = 0
for cutWords in cutWords_list[:5000]:
    i += 1
    # if i % 1000 == 0:
    #     print('前%d篇文章形成词向量花费%.2f秒' % (i, time.time() - startTime))
    vector_list.append(getVector_v1(cutWords, word2vec_model))
X = np.array(vector_list)
print('Total Time You Need To Get X:%.2f秒' % (time.time() - startTime))
X.dump('articles_vector.txt')
#加载数据可以用下面的代码
X = np.load('articles_vector.txt')
```

最后，我们进行模型测试

调用 ***sklearn.externals*** 库的 ***joblib*** 对象的 ***load*** 方法加载模型赋值给变量 ***logistic_model***

调用 ***DataFrame*** 对象的 ***groupby*** 方法对每个分类分组，从而每种文章类别的分类准确性

调用自定义的 ***getVector*** 方法将文章转换为相关性向量

自定义 ***getVectorMatrix*** 方法获得测试集的特征矩阵

调用 ***StandardScaler*** 对象的 ***transform*** 方法将预测标签做标签编码，从而获得预测目标值

```
import pandas as pd
import numpy as np
from sklearn.externals import joblib
import jieba
def getVectorMatrix(article_series):
        return np.array([getVector_v4(jieba.cut(k), word2vec_model) for k in article_series])

logistic_model = joblib.load('logistic.model')

test_df = pd.read_excel('附件2.xlsx',sep='\t', header=None)
test_df.columns = ['一级标签', '留言详情']
for name, group in test_df.groupby('一级标签'):
    featureMatrix = getVectorMatrix(group['留言详情'])
    target = labelEncoder.transform(group['一级标签'])
    print(name, logistic_model.score(featureMatrix, target))
```

我们来看看各个分类的精确率和召回率：

```
from sklearn.metrics import classification_report
test_df = pd.read_csv('附件2.xlsx', sep='\t', header=None)
test_df.columns = ['一级标签', '留言详情']
test_label = labelEncoder.transform(test_df['一级标签'])
y_pred = logistic_model.predict( getVectorMatrix(test_df['留言详情']) )
print(labelEncoder.inverse_transform([[x] for x in range(12)]))
print(classification_report(test_label, y_pred))
```