Python 实现微博中文文本聚类

刘浚哲

北京大学物理学院 *1500011370*

May 31, 2018

1. 数据预处理 [1]

本次我们处理的仍然是微博文本数据集, 由上一次文本处理经验, 我们在统计过所有文本的词频分布数据后, 选择在所有文本中出现频数大于 100 次的词语作为特征词. 确定特征词之后, 我们需要重新统计各文本在这些特征词下的词频, 实际上也是将文本转化为向量的过程. 统计得到了文本向量矩阵之后, 可以进一步计算 TF-IDF 文本向量, 计算这个统计量的原因在于: 它更能够表示特征词在文本中的重要特性, 便于我们后边计算两个文本向量之间的距离. 以上操作可以由 sklearn 库中的文本特征提取函数 feature extraction 所包含的 TfidfVectorizer 来直接得到. 如下所示:

```
data = text_data.load_data(type="str")
countVectorizer = text.TfidfVectorizer(input='content', min_df=MIN_Freq, use_idf=IDF)
term_freq = countVectorizer.fit_transform(data)
```

若选择最少出现次数 MIN_Freq 为 100 次, 则得到 463 个特征词, 它能够解释 63% 的数据方差. 同时我们对结果矩阵也进行了 SVD(奇异值分解) 操作. 得到该矩阵之后, 我们调用 scipy 库的 spatial.distance.pdist 函数计算矩阵中两两向量之间的距离, 便于聚类算法直接调用, 具体如下:

```
svd = TruncatedSVD(n_components=Components_N)
svd_result = svd.fit_transform(term_freq)
marked_svd = mark_allzeros(svd_result)
Dist = spatial.distance.pdist(marked_svd, metric="cosine")
```

其中以余弦为度量来计算两向量之间的距离. Dist 结果是一个一维数组, spatial.distance.pdist 函数通过计算矩阵上三角部分向量两两之间的距离, 按照行优先的顺序保存在 Dist 数组当中.

2. 文本聚类算法实现

采用比较基础的层次聚类方法进行聚类. 算法中包含四个函数:

```
def matrix_to_array(n, i, j):
    '''从上三角矩阵的index(i,j)转成condensed array的下标'''

def index_matrix(d, i):
    '''Dist为一维数组 (长度为 K(K-1)/2) (scipy.spatial.distance.pdist的返回值)
    n_clusters是长度为K的一维数组, 其元素为每个聚类里的元素个数, 至少为1'''

def My_Cluster(Dist, linkage="average"):
```

[1] *Email Address:1500011370@pku.edu.cn*

```
    '''层次聚类算法'''

def My_Predict(record, n_clusters, re="predict"):
    '''从聚类函数的返回值中推断聚类结果'''
```

并由 *calc_score* 当中的函数计算每个聚类的熵, 和聚类结果的总熵:

```
def count_table(classes, cluster_labels, Clusters_N):
    '''
    计算每个聚类的熵
    '''

def total_entropy(count_table):
    '''
    计算聚类结果的总熵
    '''
```

得到最终结果并输出:

```
my_cluster = My_Cluster.My_Cluster(Dist)
my_pred = My_Cluster.My_Predict(np.array(my_cluster, dtype=np.int32)[:, :2], n_clusters=9)
print(my_cluster)
print(my_pred)

for labels_pred in [my_pred]:
    count_table = calc_score.count_table(text_data.init_num_by_cls, labels_pred, Clusters_N)
    print(count_table)
    total_entropy = calc_score.total_entropy(count_table)
    print("Total Entropy:", total_entropy)
    print("homogeneity_score", metrics.homogeneity_completeness_v_measure(text_data.labels_true(),
        labels_pred))

plt.show()
```

其中我们已经预设了聚成 9 个聚类. 得到最终结果显示如下:

```
/Users/macbookair/anaconda3/envs/python3/bin/python "/Users/macbookair/iCloud/Desk
100%|██████████| 9/9 [00:05<00:00,  1.63it/s]
词典总词数: 463
Explained variance of the SVD step: 65%
[[2104   18   12   84   71   28    1   16   41]
 [ 954   10   13   28   87   69    4   21   25]
 [ 200   24    7   36  337    9    1   27   29]
 [ 146  101    2   21  108    9    0   35   23]
 [ 579   20   11   42   10   11    2   13  103]
 [1193   17   31   58   37   11    2    6   42]
 [ 783   26   31 2113   96   48   18   40  170]
 [ 520   31  114  399  115  170   27  213  666]
 [1083    5    3   21   27    9    1   11    7]]
Total Entropy: 1.66535290096
homogeneity_score (0.17956475422536491, 0.26140458469977501, 0.21289031169111367)

Process finished with exit code 0
```

图 1: 训练结果, 混淆矩阵与最终结果的熵

3. 附录: 源代码

见附件, 其中包括: KMeans 实现代码, 层次聚类实现代码.

References:

[1]