

Homework1:VSM and KNN

姓名：刘敏

学号：201814820

- **实验要求：**

预处理文本数据集，并且得到每个文本的 VSM 表示

实现 KNN 分类器，测试其在 20news-18828 数据集上的分类效果

- **实验过程：**

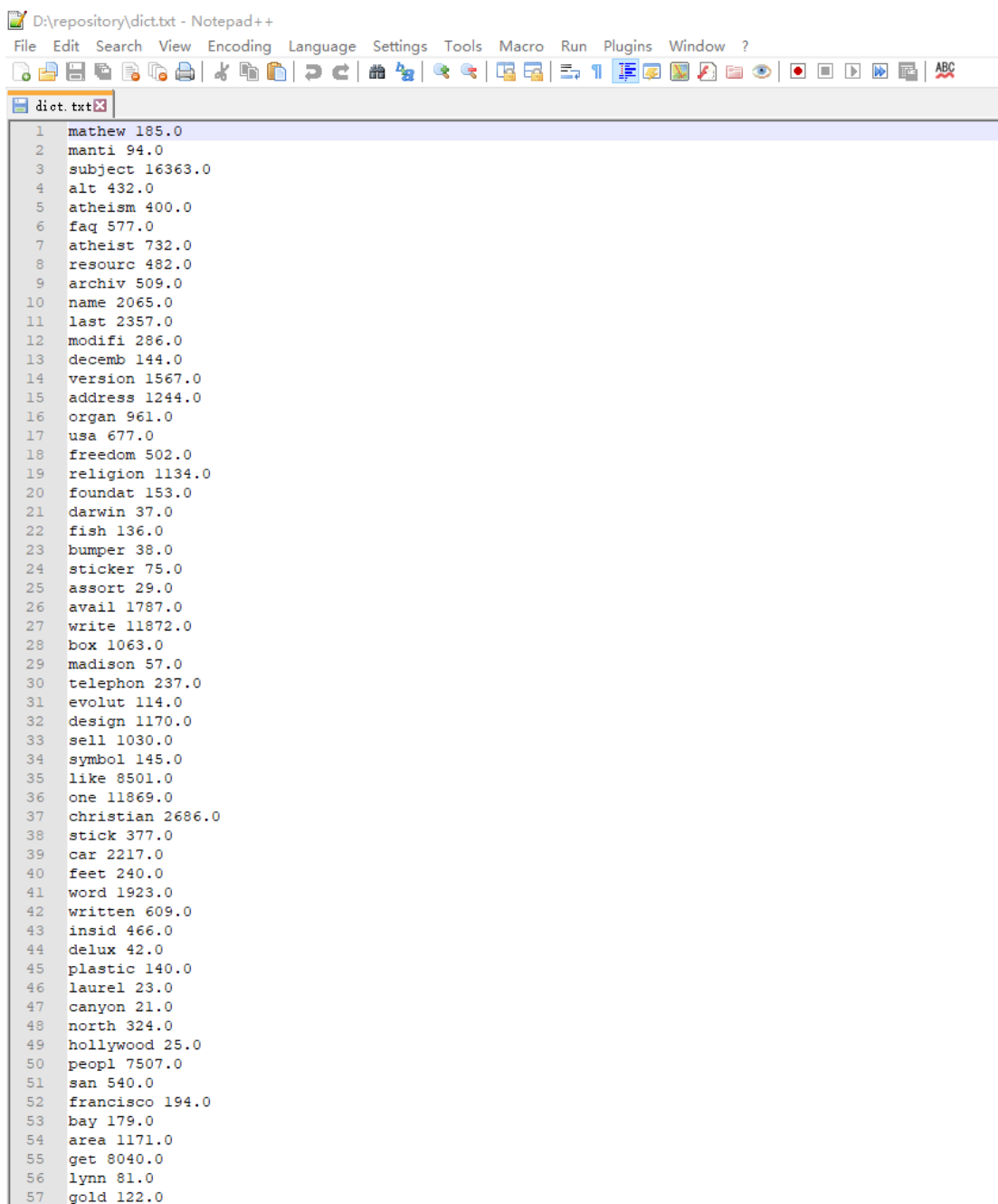
1. 数据预处理：基于 nltk 库对数据依次进行去除数字、分词、词干抽取、去停用词以及长度小于 3 的 word 等基本的预处理操作，预处理之后的数据集为 Processed20news-18828. 代码详见 Preprocess.py
2. 划分数据集：使用 splitData()方法将预处理之后的数据集按 8:2 的比例划分为五组不同的训练集和测试集，如图所示。代码详见 Preprocess.py

此电脑 > DATA (D:) > repository > TrainData		
名称	修改日期	类型
0	2018/11/2 20:44	文件夹
1	2018/11/2 20:45	文件夹
2	2018/11/2 20:45	文件夹
3	2018/11/2 20:45	文件夹
4	2018/11/2 20:45	文件夹

此电脑 > DATA (D:) > repository > TestData		
名称	修改日期	类型
0	2018/11/2 20:44	文件夹
1	2018/11/2 20:45	文件夹
2	2018/11/2 20:45	文件夹
3	2018/11/2 20:45	文件夹
4	2018/11/2 20:45	文件夹

3. 构建字典：仅使用其中一组即第 0 组训练集构建字典（本意是在 5 组训练集上分别构建字典，最后再在对应的测试集上训练 KNN 分类器，但是时间有限，最终只

使用了一组训练集与测试集), 由 creatWordMap()方法实现, 这里 <key,value> 的 key 是单词, value 是单词在整个训练集中出现的次数, 然后将词频小于等于 7 的单词过滤掉, 得到新的字典, 即 dict.txt, 如图所示; 再将训练集和测试集中没有出现在字典中的单词过滤掉, 得到新的训练集和测试集, 如图所示。代码详见 createDict.py (最后准确率只有 0.7 多, 应该是字典构建地不太好, 应该尝试从全局角度考虑, 剔除 df 值比较小的单词)



```
D:\repository\dict.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
dict.txt
1 mathew 185.0
2 manti 94.0
3 subject 16363.0
4 alt 432.0
5 atheism 400.0
6 faq 577.0
7 atheist 732.0
8 resourc 482.0
9 archiv 509.0
10 name 2065.0
11 last 2357.0
12 modifi 286.0
13 decemb 144.0
14 version 1567.0
15 address 1244.0
16 organ 961.0
17 usa 677.0
18 freedom 502.0
19 religion 1134.0
20 foundat 153.0
21 darwin 37.0
22 fish 136.0
23 bumper 38.0
24 sticker 75.0
25 assort 29.0
26 avail 1787.0
27 write 11872.0
28 box 1063.0
29 madison 57.0
30 telephon 237.0
31 evolut 114.0
32 design 1170.0
33 sell 1030.0
34 symbol 145.0
35 like 8501.0
36 one 11869.0
37 christian 2686.0
38 stick 377.0
39 car 2217.0
40 feet 240.0
41 word 1923.0
42 written 609.0
43 insid 466.0
44 delux 42.0
45 plastic 140.0
46 laurel 23.0
47 canyon 21.0
48 north 324.0
49 hollywood 25.0
50 peopl 7507.0
51 san 540.0
52 francisco 194.0
53 bay 179.0
54 area 1171.0
55 get 8040.0
56 lynn 81.0
57 gold 122.0
```

此电脑 > DATA (D:) > repository > filteredTrainData			
名称	修改日期	类型	
0	2018/11/2 23:00	文件夹	

此电脑 > DATA (D:) > repository > filteredTestData			
名称	修改日期	类型	
0	2018/11/3 13:07	文件夹	

4. VSM: 首先计算训练集和测试集文档中单词的 idf 值, 将 idf 值写入文件保存下来, 再使用 computeTFIDF()方法计算各自的 tfidf 值, 分别写入 TrainTFIDF.txt 和 TestTFIDF.txt, 实现文本文档的 VSM 表示。代码详见 VSM.py

5. KNN: 计算每个测试集文档与训练集中所有文档的距离 (余弦相似度), 选取 k 个最近邻做类别预测, 计算准确率。代码详见 KNN.py

● 实验结果:

最开始将词频小于等于 19 的单词过滤掉, 字典大小为 9356, 最后分类效果不佳, 又尝试将阈值设置为 13, 字典大小为 11879, 分类准确率仅提升 0.01, 最后将阈值设置为 7, 字典大小为 17217, 准确率可以提高 0.03

```

Run: KNN x
This is 3751 round!
RightCount:2686
This is 3752 round!
RightCount:2686
KNN classifier's accuracy based 15 is: 0.715885

Process finished with exit code 0
  
```

过滤词频为 19, K 为 15 的准确率

```
Run: KNN x
This is 3752 round!
RightCount:2725
KNN classifier's accuracy based 15 is: 0.726279
Runtime is :1712.0767579626254

Process finished with exit code 0
```

Python Console Terminal 4: Run 6: TODO

过滤词频为 13, K 为 15 的准确率

```
Run: KNN x
RightCount:2804
This is 3751 round!
RightCount:2804
This is 3752 round!
RightCount:2804
KNN classifier's accuracy based 20 is: 0.747335
Runtime is :1717.6828651335754

Process finished with exit code 0
```

Python Console Terminal 4: Run 6: TODO

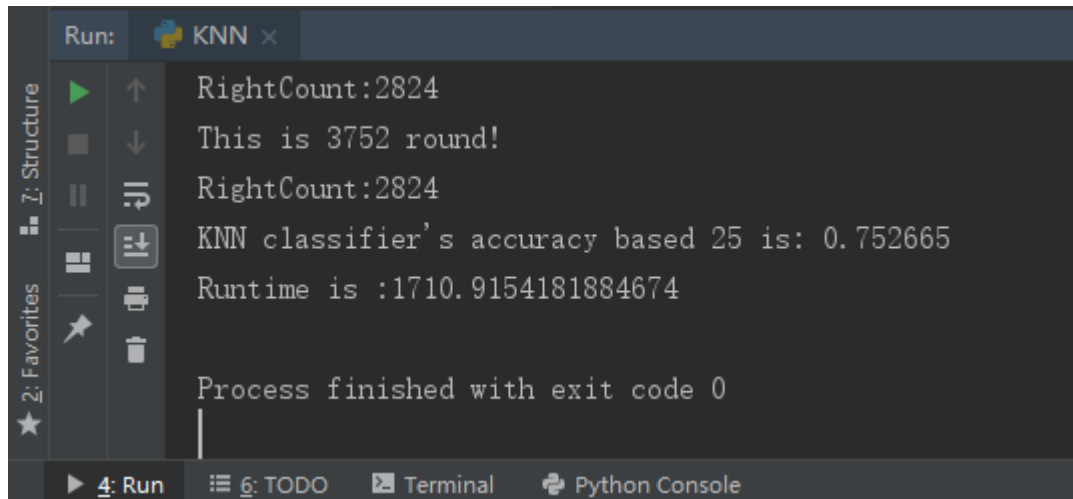
过滤词频为 7, K 为 15 的准确率

```
Run: KNN x
RightCount:2804
This is 3752 round!
RightCount:2804
KNN classifier's accuracy based 20 is: 0.747335
Runtime is :1738.4032352804363

Process finished with exit code 0
```

4: Run 6: TODO Terminal Python Console

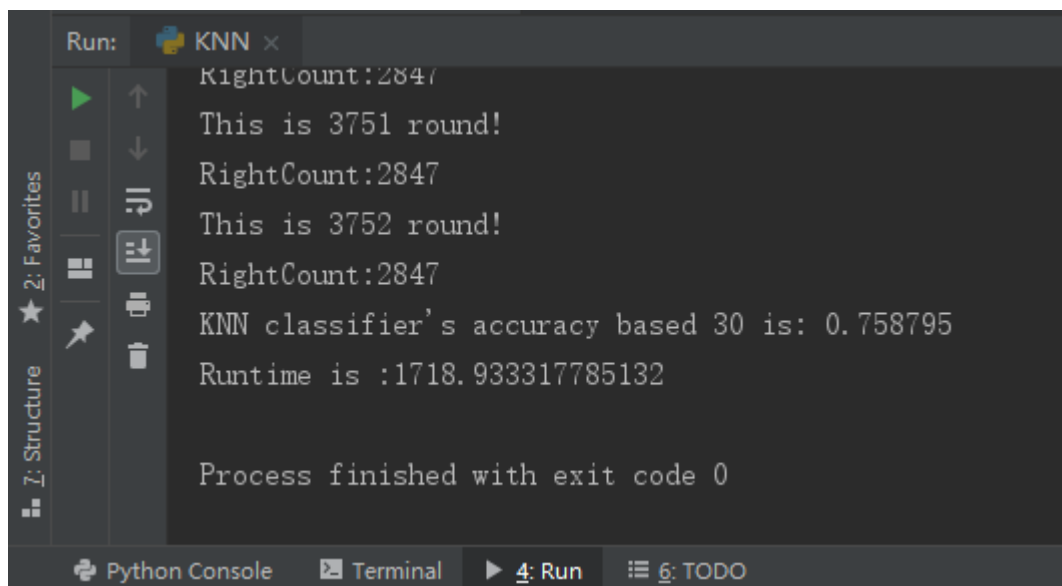
过滤词频为 7, K 为 20 的准确率



```
Run: KNN x
RightCount:2824
This is 3752 round!
RightCount:2824
KNN classifier's accuracy based 25 is: 0.752665
Runtime is :1710.9154181884674

Process finished with exit code 0
```

过滤词频为 7，K 为 25 的准确率



```
Run: KNN x
RightCount:2847
This is 3751 round!
RightCount:2847
This is 3752 round!
RightCount:2847
KNN classifier's accuracy based 30 is: 0.758795
Runtime is :1718.933317785132

Process finished with exit code 0
```

过滤词频为 7，K 为 30 的准确率

总结：KNN 算法比较简单，但是处理 20 万个文档时运行时间稍微有点长，通过几次调参，我的代码在过滤词频为 7，K 为 30 的参数下取得了较好的准确率，为 0.758795，但是仍然不是很理想，构建字典的过程中应该统计具有全局意义的 df，我想在此基础上得到的最终的字典应该比较优，之后会继续改进。