文本分类作业报告

人工智能82班 刘志成 2183511589

相关代码已上传至我的github仓库：
https://github.com/zchliu/2020-Fall

问题引入

使用朴素贝叶斯的方法进行文本分类，数据集自选，需要提供模型评估的结果以及讨论出现这种结果的可能原因

数学模型

首先对下面公式中可能出现的数学符号进行说明

- 假设训练集一共有K个类别，每一个类别用字母C_k表示，$k = 0, 1, ..., K-1$
- 假设训练集一共有L个词，每一个词用字母W_l表示，$l = 0, 1, ..., L-1$，每一个词的出现次数用字母M_l表示，$M_l = 0, 1, 2, ...$
- 假设训练集一共有N篇文档，每一篇文档用字母X_n表示，$n = 0, 1, ..., N-1$
- 假设有一个待预测文本T，T中出现的单词用字母t_j表示，这个词出现的次数用字母m_j表示

tf-idf权重简介

tf-idf由两部分组成，tf即词频，表征这个词在这个类中的出现频率，tf越高说明这个词的重要性越大，idf即逆文档频率，表征这个词在其他文档中出现的频率，idf越小说明这个词越重要。tfidf的主要思想是：如果某个词或短语在一篇文章中出现的频率TF高，并且在其他文章中很少出现，则认为此词或者短语具有很好的类别区分能力，适合用来分类。

如果由不同的类中的不同的词的tf-idf权重组合在一起，就构成了一个tf-idf权重矩阵

则一个K个类别，L个词的$K \times L$的权重矩阵M元素$M[k, l]$计算公式如下：

$$M[k, l] = f_{kl} \times log(\frac{N+1}{N_l+1})$$

f_{kl}是类别k中第l个词的词频

$$f_k = \frac{M_l}{\sum_{\forall l', W_{l'} \in C_k} M_{l'}}$$

N_l是在所有文档中第l个词出现的文档数

$$N_l = \sum_{\forall n, W_l \in X_n} 1$$

朴素贝叶斯的推导

对于一个给定文档，我们在每一个类中分别计算这个文档在这个类当中的概率值，取最大的就是这篇文档的类别

$$\begin{aligned}
&\arg\max_k P(C_k|T) \\
=&\arg\max_k P(C_k)P(T|C_k) \\
=&\arg\max_k P(C_k)\Pi_j P(t_j|C_k)^{m_j} \\
=&\arg\max_k (\ln(P(C_k)) + m_j \sum_j \ln(P(t_j|C_k)))
\end{aligned}$$

其中$P(t_j|C_k)$为tfidf权重矩阵中的单词t_j在类别C_k下的tfidf值

编程实现

仅展示主要函数

在预处理中生成词典，统计每一个词的f_{kl}和N_l，最后计算tfidf，生成tfidf矩阵用pickle保存

```
def build_matrix(corpus_dir):

    clas_list = os.listdir(corpus_dir)

    dictionary = defaultdict(item)

    Frequency_list = []
    N_list = []

    N = 0
    for i in range(len(clas_list)):

        F = 0
        begin_time = time.time()
        corpus_clas_dir = corpus_dir + "//" + clas_list[i] + "//"
        doc_list = os.listdir(corpus_clas_dir)
        N_list.append(len(doc_list))

        for j in range(len(doc_list)):

            N += 1
            content = readfile(corpus_clas_dir + doc_list[j])
            content = re.sub(pattern, ' ', content)
            content_seg = jieba.cut(content)

            seen = []
            for word in content_seg:
                if (word in stopwords):
                    continue
                if (word not in seen):
                    dictionary[word].nk += 1
                    seen.append(word)
                F += 1
                dictionary[word].fk[i] += 1

        Frequency_list.append(F)
        end_time = time.time()
        print(str(clas_list[i]) + " has words：" + str(F) + ".has docs：" + str(len(doc_list)) + ".consuming time

    # 计算tfidf
    TFIDF_list = []
    for i in range(len(clas_list)):
        TFIDF = 0
        for word in dictionary.keys():
            dictionary[word].tfidf[i] = dictionary[word].fk[i] / Frequency_list[i] * np.log((N + 1) / (dictionary
            TFIDF += dictionary[word].tfidf[i]
        TFIDF_list.append(TFIDF)

    # 归一化
    for i in range(len(clas_list)):
        for word in dictionary.keys():
```

```
            dictionary[word].tfidf[i] = dictionary[word].tfidf[i] / TFIDF_list[i]

    dictionary_file = open("dictionary.pickle", "wb")
    pickle.dump((dictionary, N_list, clas_list, Frequency_list), dictionary_file)
    dictionary_file.close()

    print("dictionary.pickle has been created!")
```

对给定文本T，用朴素贝叶斯算法预测

```
def predict(path):
    if (not os.path.exists("dictionary.pickle")):
        print("dictionaty.pickle doesn't exist!please first build dictionary!")
    else:
        dictionary_file = open("dictionary.pickle", "rb")
        dictionary, N_list, clas_list, frequency_list = pickle.load(dictionary_file)
        dictionary_file.close()

        clas_len = len(clas_list)
        input_path = path
        words_dict = preprocessing.build_eigen(input_path)
        # print(words_dict)

        predicted_clas = ""
        Pmax = -100000000000
        for i in range(clas_len):
            P = 0
            P += np.log(N_list[i] / np.sum(N_list))
            for word in words_dict:
                # print(words_dict[word] * np.log((dictionary[word].tfidf[i] + 1e-6) / (weight_sum + 1e-6)))
                P += words_dict[word] * np.log(dictionary[word].tfidf[i] + 1e-10)

            # print(i, P)
            if (P > Pmax):
                predicted_clas = clas_list[i]
                Pmax = P

        return predicted_clas
```

模型评估

数据集介绍

数据集选自《复旦大学中文语料库》，网站在
https://www.kesci.com/home/dataset/5d3a9c86cf76a600360edd04

每一个类别的文件是一个编码为GBK的.txt文件，为了方便后续的处理，首先要把文档转换成utf-8的编码格式，具体处理过程可以参见代码 `ToUtf8.py`

这是一个数量不均衡的数据集，例如Space类有1282个文档109万个词，而Communication类只有52篇文档9千个词，由于数据量过小的类分类时候的泛化能力比较差，因此删除文档数在200以下的类别，最后得到的语料库的信息为：

类别名	文档数	总词数
C11-Space	1282	1093603
C19-Computer	2715	2658597
C3-Art	1482	3126295
C31-Environment	2435	2370912
C32-Agriculture	2043	2929601
C34-Economy	3201	5810974
C38-Politics	2050	2834429
C39-Sports	2507	3076325
C7-History	934	2257097

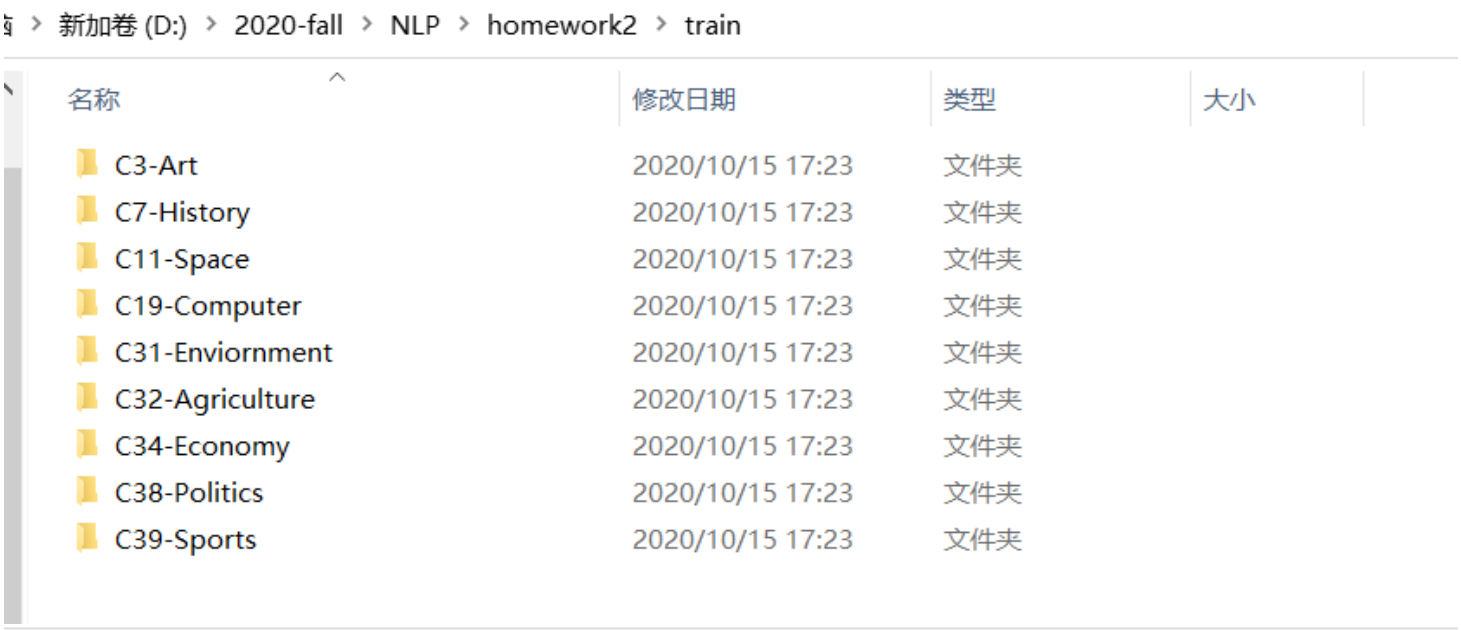
我们随意打开一个文本 `C32-Agriculture0011.txt` 查看他的内容（部分）如下：

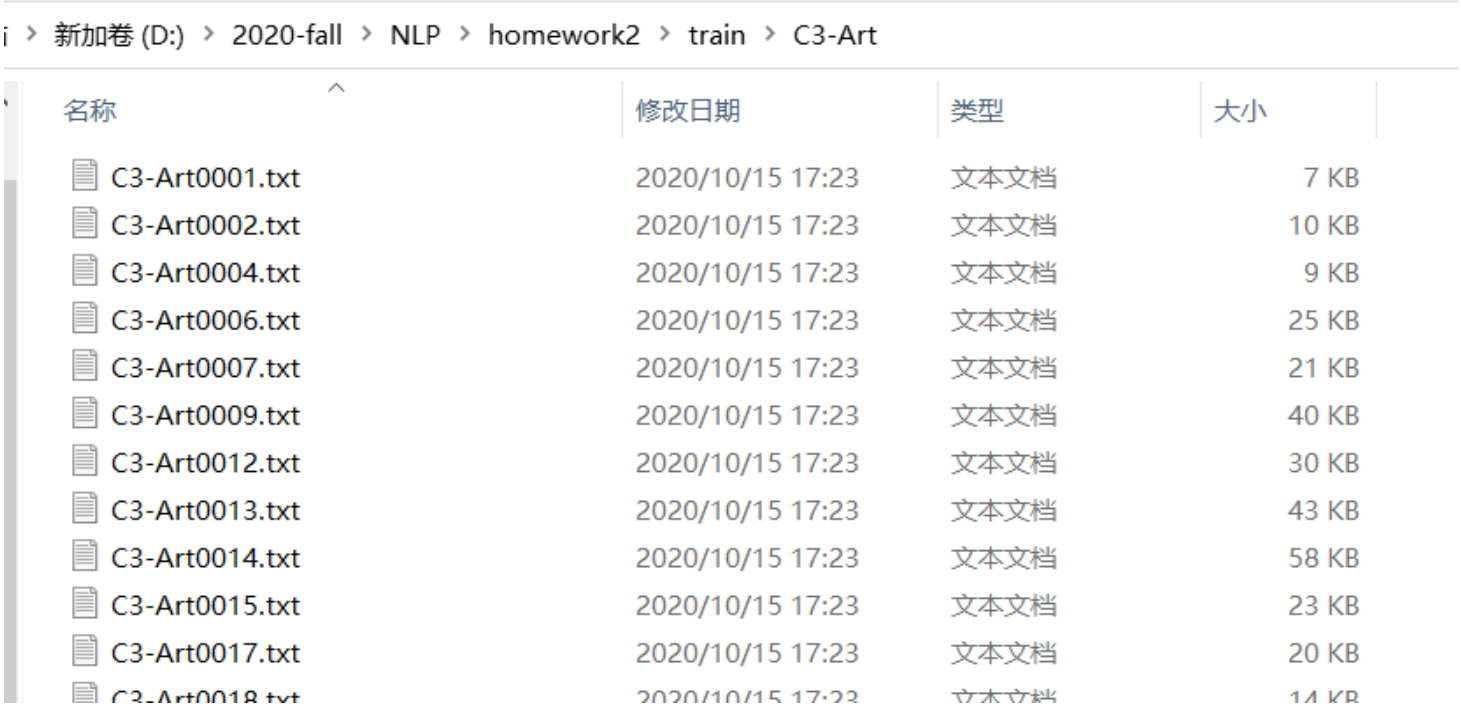
【 文献号 】1-3028
【原文出处】南方农村
【原刊地名】广州
【原刊期号】199702
【原刊页号】1-4
【分 类 号】F2
【分 类 名】农业经济
【 作　者 】高启杰
【复印期号】199707
【 标　题 】对农业高新技术产业化几个理论问题的思考
【 正　文 】
一、农业高新技术产业及其特征
高新技术一词国外最早出现于70年代，它主要是指那些知识、技术和资金密集的新兴技术，例如信息技术、新能源与新材料技术、海洋技术、电子技术、生物技术等。在农业领域前景最大的高新技术要数农业生物技术和电子技术。近10多年来，我国农业高新技术特别是生物技术发展较快，并形成了一支从事农业高新技术研究与开发的队伍。据不完全统计，到1993年底，我国从事农业高新技术研究、开发、中试的机构有300多家，科技人员达3万余人。在生物技术、核技术农业应用研究、计算机农业应用、遥感技术农业应用及生物农药方面已取得了一大批具有国际先进水平的研究成果。
高新技术的发展和高新技术产业的建立是紧密地联系在一起的。农业高新技术只有形成产业才能充分显示出它的作用。正因为如此，近年来世界各国都在调整高新技术发展战略，形成了一股高新

数据集切分

将数据集按8：2的比例随机的分成数据集train和测试集test，具体处理过程参见 `split_corpus.py`

这样，我们在当前目录下可以得到一个train和test的文件夹，里面的每一个子文件夹装着表示这个类的文档，如下图：





生成词典

利用训练集里面的文档，用tfidf权重可以生成模型的字典，具体过程参见 `preprocessing.py`

处理的中间过程和训练集的一些基本信息如下：

```
PS D:\2020-fall\NLP\homework2> python -u "d:\2020-fall\NLP\homework2\preprocessing.py"
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\ASUS\AppData\Local\Temp\jieba.cache
Loading model cost 0.742 seconds.
C11-Space has words：874310.has docs：1022.consuming time：50.45410180091858
C19-Computer has words：2147392.has docs：2190.consuming time：153.58699750900269
C3-Art has words：2509816.has docs：1186.consuming time：115.28371405601501
C31-Enviornment has words：1902377.has docs：1956.consuming time：122.72820568084717
C32-Agriculture has words：2344587.has docs：1646.consuming time：102.78697228431702
C34-Economy has words：4679666.has docs：2587.consuming time：210.50154376029968
C38-Politics has words：2309289.has docs：1671.consuming time：99.68630456924438
C39-Sports has words：2437595.has docs：2004.consuming time：108.01152014732361
C7-History has words：1833656.has docs：761.consuming time：86.74753952026367
dictionary.pickle has been created!
```

我们可以将字典里面的tfidf权重进行排序，看一下每一个类都有哪一些关键词，将tfidf按降序排列，结果如下，参见 `text_classify.py`：

C11-Space	tfidf
发动机	0.0039411628246573405
叶片	0.0035742686415050568
测量	0.003555303628765195
试验	0.002302707815509278
温度	0.002273906800634483
振动	0.0022463192828663186
转子	0.0022364374935663874
图	0.0021951465887993247
复合材料	0.0020296156931578534
模型	0.0019635917692025395
系统	0.0019404141762068188
故障	0.001927943341087899
转速	0.0018915209599119367
计算	0.0018579465994946775
换热	0.0017552134470854124

C11-Space	tfidf
应力	0.001726495559432222
飞机	0.001715768242676238
合金	0.0016936910083226126
飞行	0.0016934847503302518
涂层	0.0016791425564285998
参数	0.0016385851820591585

C19-Computer	tfidf
算法	0.006048611723383723
系统	0.004088801044830149
模型	0.004055223663284258
用户	0.003306409812006816
数据库	0.002839203599099568
数据	0.002838432009377388
服务器	0.002781831470867803
网络	0.0026464345952257953
函数	0.002608549900410681
对象	0.002375372124762043
结点	0.002322388492619274
控制器	0.002227634016221695
定义	0.002221869259598123
矩阵	0.0021696764950053325
图像	0.002093465617682595
调度	0.0019912950477496433
参数	0.001984414667083757

C19-Computer	tfidf
控制	0.0019644334176758638
基于	0.001938097598546011
定理	0.0019103115239577704
输出	0.0018857195341967514

C3-Art	tfidf
艺术	0.008376916039905391
文艺	0.004269891432107159
文学	0.003298573751566718
作品	0.0021140182348950536
文化	0.0021124714056314367
文艺学	0.001976658311269081
审美	0.0019705204510220113
小说	0.0019547940244792443
创作	0.0018808189203929806
电影	0.0017315640973008112
美学	0.0016575845128088368
文论	0.001544445185822235
作家	0.0013024273684837909
诗	0.001270502145150608
精神	0.001268664593724987
艺术家	0.0012343865559303455
批评	0.001229552631571279
人物	0.0011854856251811061
历史	0.001180915266152882

C3-Art	tfidf
叙事	0.0011173461061188177
文艺理论	0.0010951692421403445

C31-Enviornment	tfidf
浓度	0.0044289885828845636
土壤	0.0034270316500568 44
吸附	0.0030054434053889 03
环境	0.0024743254317375 21
含量	0.0022940596073948 76
降解	0.0021534425271691805
污染	0.0021203550641147 04
试验	0.0019423473267495 8
废水	0.0018822894633668723
污泥	0.0018121104492621 29
排放	0.0016547960749523512
污染物	0.0016508334923242243
氧化	0.0015162711973878348
生物	0.0015101850223720684
汞	0.0014583143472608813
实验	0.0014221350280988784
温度	0.0013611689930500522
大气	0.0013106972134903 67
测定	0.0013083947580293071
活性	0.0012882280367153 44
反应器	0.0012808106970798544

C32-Agriculture	tfidf
农业	0.0202086153171935
农产品	0.004408929630472706
农民	0.004305892836196153
农村	0.0040348569196122135
粮食	0.0036198739216955125
生产	0.003313199385642945
农户	0.00305213789421527
产业化	0.0028956444926027613
土地	0.002607059521147943
经营	0.002324535795779779
市场	0.002295632921220766
经济	0.002057303547708474
发展	0.0018783610464861754
我国	0.0017919631951113081
价格	0.001741393959531016
劳动力	0.0017382062244203434
小麦	0.0016095962184657776
玉米	0.0015828413482295656
耕地	0.0015243829311322816
投入	0.0014748938563951797
增长	0.0014494551812178334

C34-Economy	tfidf
经济	0.007094473137198449
企业	0.004283843018151159

C34-Economy	tfidf
市场	0.0027163996317422614
增长	0.002713960749422232
投资	0.0025212294015892326
澳门	0.0022015550705065537
发展	0.0019943789836064612
社会主义	0.001985460401790099
政府	0.0019608105693681146
资本	0.0018241615258893667
产业	0.001792703337797147
经济学	0.0017251725697044115
我国	0.0017102629098566294
消费	0.001705247808425932
改革	0.0016575580976277056
市场经济	0.0016033584004215935
知识经济	0.0015964816816518058
制度	0.0015858408124686094
社会	0.00152038625150168
政策	0.0014765036959038342
贸易	0.0013827129704904947

C38-Politics	tfidf
政治	0.014912746722783666
民主	0.005141389953629428
社会主义	0.00352332519898321
社会	0.0030012287820835055

C38-Politics	tfidf
政治学	0.0027352509216058505
权力	0.002661568583969531
党	0.0023769932974854247
制度	0.0022589453668825604
国家	0.0022045523594388795
经济	0.0021794055712548656
人民	0.002058441829424776
邓小平	0.0019171309647036752
建设	0.0018743481842219947
领导	0.0017273222641521128
思想	0.0016882651018483073
文化	0.0016795412265703889
干部	0.0016241223901299257
发展	0.0015659231916352607
群众	0.0015040554121683596
利益	0.001490640662220919
体制改革	0.0014784121081090772

C39-Sports	tfidf
体育	0.010192487461317283
教育	0.006577768256284938
学生	0.006286185909694767
学校	0.003761376192617652
运动员	0.0031909110650469273
课程	0.0031647777947702706

C39-Sports	tfidf
教学	0.0030683863831966044
教师	0.0027056443906827827
训练	0.002319484225829313
运动	0.0020902228966813933
比赛	0.0017315957107993133
幼儿	0.001644505421146043
学习	0.001575370588144632
培养	0.0015543612011306158
健身	0.0015421941733231124
竞技	0.001428172492703399
活动	0.0013604975884489202
学科	0.0013597511698097437
社会	0.0013444590510721481
文化	0.001312461633743476
发展	0.0012052075532482148

C7-History	tfidf
历史	0.0040721636836853195
小说	0.0020045000862919126
史学	0.001987276093238599
文学	0.0019489176842015195
文化	0.001549971653626454
中国	0.0014084774812034887
民族	0.0010313226573472243
政治	0.0010090338343435314

C7-History	tfidf
革命	0.0009621935016723979
创作	0.0009591525722454604
作品	0.0009480241501369966
艺术	0.0009349117266910528
社会	0.0009172099857895499
人物	0.0008756698468915009
作家	0.0007922293159569333
史	0.0007913560783447926
鲁迅	0.0007906953350687796
人	0.0007902965213626904
说	0.0007289438496171922
西方	0.0007281826756708023
历史学	0.0007245360731517233

模型在测试集上分类正确率

最后，我们在测试集上运行，参见 `evaluate.py` ，结果如下：

```
PS D:\2020-fall\NLP\homework2> python -u "d:\2020-fall\NLP\homework2\evaluate.py"
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\ASUS\AppData\Local\Temp\jieba.cache
Loading model cost 0.699 seconds.
Prefix dict has been built successfully.
C11-Space's correct rate is 87.69%.consuming time：505.33
C19-Computer's correct rate is 94.86%.consuming time：1013.66
C3-Art's correct rate is 92.57%.consuming time：575.72
C31-Enviornment's correct rate is 91.23%.consuming time：973.94
C32-Agriculture's correct rate is 92.19%.consuming time：800.55
C34-Economy's correct rate is 93.49%.consuming time：1193.75
C38-Politics's correct rate is 95.25%.consuming time：726.80
C39-Sports's correct rate is 94.83%.consuming time：962.30
C7-History's correct rate is 67.05%.consuming time：338.72
total correct rate is 91.86%
```

对其进行5折交叉验证后，模型平均分类正确率在91.86%，理论上达到了朴素贝叶斯模型的分类精度上限，如果要有更高的分类精度应该选择更复杂的模型。具体来看，大部分的类分类精度达到了90%，但是History类的分类正确率只有67.05%，出现这个结果主要原因是，History类的数据和Art类的数据太接近了，导致模型偏向把原本属于History类的数据预测为Art类，我们从词典的关键词分布就可以看出来。这时候只能用更复杂的模型才能把这两个类分开，基于词频的朴素贝叶斯模型在解决这个问题上效果不佳。

C3-Art	tfidf	C7-History	tfidf
艺术	0.008376916039905391	历史	0.0040721636836853195
文艺	0.004269891432107159	小说	0.0020045000862919126
文学	0.003298573751566718	史学	0.001987276093238599
作品	0.0021140182348950536	文学	0.0019489176842015195
文化	0.0021124714056314367	文化	0.001549971653626454
文艺学	0.001976658311269081	中国	0.0014084774812034887
审美	0.0019705204510220113	民族	0.0010313226573472243
小说	0.0019547940244792443	政治	0.0010090338343435314
创作	0.0018808189203929806	革命	0.0009621935016723979
电影	0.0017315640973008112	创作	0.0009591525722454604
美学	0.0016575845128088368	作品	0.0009480241501369966
文论	0.001544445185822235	艺术	0.0009349117266910528
作家	0.0013024273684837909	社会	0.0009172099857895499
诗	0.001270502145150608	人物	0.0008756698468915009
精神	0.001268664593724987	作家	0.0007922293159569333
艺术家	0.0012343865559303455	史	0.0007913560783447926
批评	0.001229552631571279	鲁迅	0.0007906953350687796
人物	0.0011854856251811061	人	0.0007902965213626904
历史	0.001180915266152882	说	0.0007289438496171922
叙事	0.0011173461061188177	西方	0.0007281826756708023

C3-Art	tfidf	C7-History	tfidf
文艺理论	0.0010951692421403445	历史学	0.00072453607315172 33