文本检索大作业实验报告

2000013150 唐正举

本次文本检索大作业，我完整地完成了文本检索的模型要求和功能要求，在不同的模块中，除了完成基础要求外，还使用了不同的方法并对得到的结果进行了比较。

一、基本要求

1、数据处理

本模块实现在 data_process.ipynb 中，选用英文数据集，使用 nltk 的 stopwords 模块去除停用词、WordNetLemmatizer 模块提取词根、FreqDist 模块过滤低频词、正则表达式去除标点，提取出词典。

部分实验结果如下，body 一栏是处理前的文本，new_body 一栏是处理后的文本。

	title	body	topic	id	new_body
0	Ad sales boost Time Warner profit	Quarterly profits at US media giant TimeWarner...	business	1	quarterly profit medium giant jump three month...
1	Dollar gains on Greenspan speech	The dollar has hit its highest level against t...	business	2	dollar hit high level euro almost three month ...
2	Yukos unit buyer faces loan claim	The owners of embattled Russian oil giant Yuko...	business	3	owner oil giant ask buyer former production un...
3	High fuel prices hit BA's profits	British Airways has blamed high fuel prices fo...	business	4	blame high fuel price drop profit report resul...
4	Pernod takeover talk lifts Domecq	Shares in UK drinks and food firm Allied Domec...	business	5	share drink food firm risen speculation could ...
...	...	...	...	...	...
2220	BT program to beat dialler scams	BT is introducing two initiatives to help beat...	tech	2221	introduce two initiative help beat cost net us...
2221	Spam e-mails tempt net shoppers	Computer users across the world continue to ig...	tech	2222	computer user across world continue ignore sec...
2222	Be careful how you code	A new European directive could put software wr...	tech	2223	new directive could put software writer risk l...
2223	US cyber security chief resigns	The man making sure US computer networks are s...	tech	2224	man make sure computer network safe secure res...
2224	Losing yourself in online gaming	Online role playing games are time-consuming, ...	tech	2225	online role play game time flight reality peop...

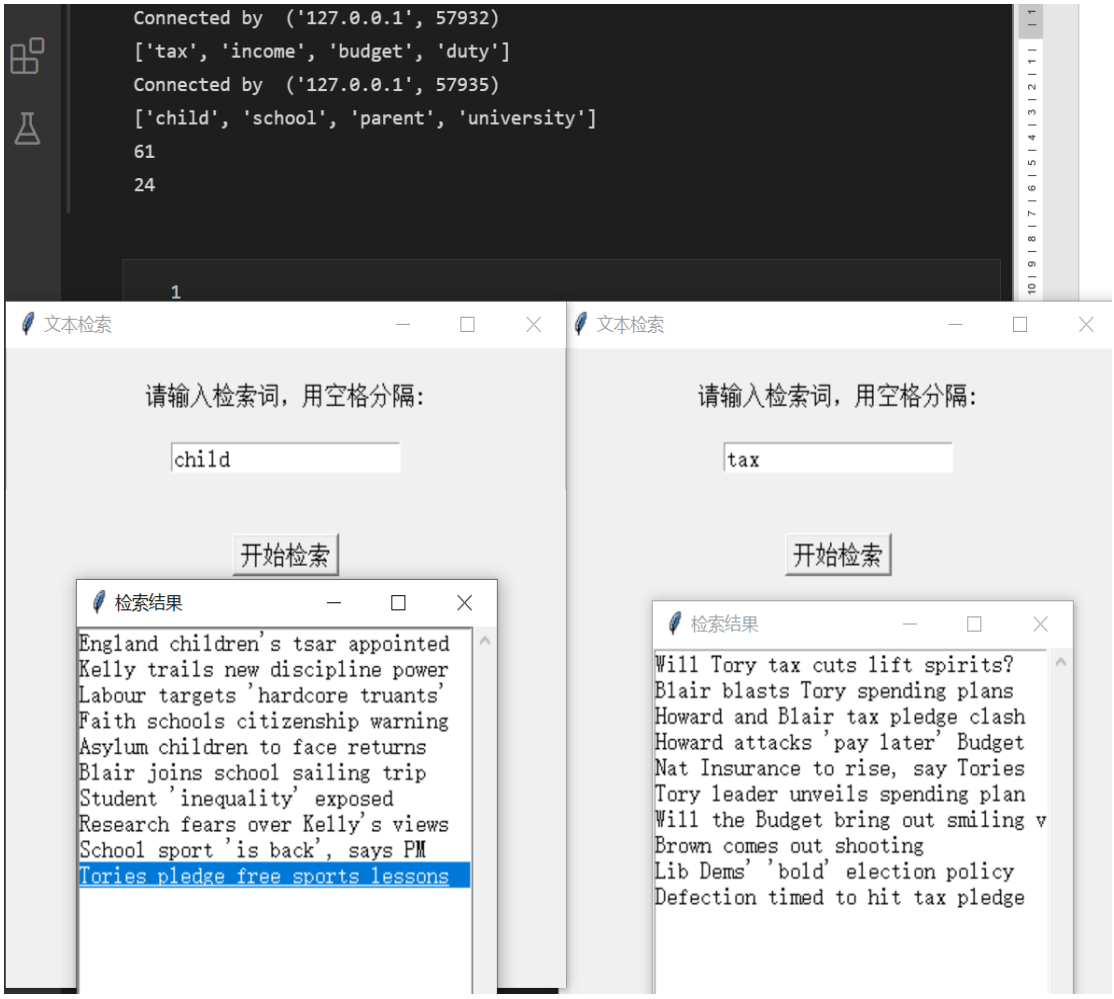
2225 rows × 5 columns

预处理后的数据存放在 data.csv 文件中，词典数据存放在 vocab.txt 中。

2、检索排序

本模块实现在 local_server.ipynb 中，服务器端收到客户端发送的检索词，根据检索词的出现频率对所有文章排序，选出频率最高的前 8 篇文章。此处的“频率”为加权后的频率，第 k 个检索词权重设定为 1/2^(k-1)，更能反映检索者的偏好。根据这 8 篇文章，采用 HITS 算法进一步寻找相似文章并排序，得到与检索词最相关且最权威的 10 篇文章，从服务器端发送回客户端，作为检索结果。测试如下：

关于 C/S 架构，我采用了 threading 模块多线程解决并发问题，经测试有效。下图中，61 对应 tax 的检索，24 对应 child 的检索，说明正在进行并发；检索结果显示正常且确实与检索词高度相关，说明线程操作无误。



3、排序优化

本模块实现在 tfidf.ipynb，pca.ipynb 和 local_server.ipynb 中，其中 tfidf.ipynb 负责计算每篇文章每个词语的 tf*idf 值，并生成文章向量；pca.ipynb 负责提取主成分，并计算文章之间的余弦相似度；local_server.ipynb 负责 HITS 算法的实现。

以下是 tf*idf 的计算结果：

	title	id	word	totcnt	wordcnt	TF	IDF	TFIDF
25654	Ad sales boost Time Warner profit	1	account	183	2	0.010929	2.677074	0.029258
6994	Ad sales boost Time Warner profit	1	advert	183	1	0.005464	4.375308	0.023909
14026	Ad sales boost Time Warner profit	1	advertising	183	2	0.010929	4.375308	0.047818
26985	Ad sales boost Time Warner profit	1	already	183	1	0.005464	1.783256	0.009745
16304	Ad sales boost Time Warner profit	1	also	183	2	0.010929	0.564685	0.006171
...	...	...	...	...	...	...	...	...
42834	Losing yourself in online gaming	2225	worry	1151	3	0.002606	2.989013	0.007791
58736	Losing yourself in online gaming	2225	would	1151	7	0.006082	0.663479	0.004035
185676	Losing yourself in online gaming	2225	write	1151	8	0.006950	2.365178	0.016439
3236	Losing yourself in online gaming	2225	year	1151	9	0.007819	0.425439	0.003327
138412	Losing yourself in online gaming	2225	young	1151	1	0.000869	2.481766	0.002156

228701 rows × 8 columns

tf_idf 数据保存在 tfidf.csv 文件，文章向量数据保存在 newsvec.csv 文件，余弦相似度数据保存在 cos_sim.csv 文件。

4、文章聚类与评价

本模块实现在 cluster.ipynb 中，采用 sklearn 的 KMeans 包和 normalized_mutual_info_score 模块，对文章向量进行聚类和互信息计算，分析聚类效果。

聚类纯度在 0.5-0.7 之间，互信息在 0.45-0.65 之间。

5、相似词

本模块实现在 similar_word.ipynb 中，利用 tf-idf 所得矩阵的转置得到词向量，进行降维和相似度计算，为每个单词保留与之相似度最高的 3 个单词。

以下是部分相似词结果，每个单词的下方存放该单词的 3 个相似词。

```
···   Output exceeds the size limit. Open the fu
     ['claim',
      'accuse,deny,tell',
      'story',
      'direct,cast,classic',
      'relay',
      'medal,medallist,gold',
      'attention',
      'indoor,medallist,pole',
      'tough',
      'lose,good,defeat',
```

相似词数据存放在 synonym.txt 文件中。

6、模糊匹配

本模块实现在 local_server.ipynb 中，在检索中将相似词顺序加入检索词列表中，赋予较低权重参与频率计算，再进行排序。

二、拓展要求

1、排序优化

关于 HITS 算法，我进行了效率上的优化。由于文章数量较多，迭代每个文章的 authority 和 hub 向量的计算时间较长，不满足搜索速度要求。于是，我在提取出待选文章集合（包括经频率排序的 8 篇文章和经这 8 篇找到的相似文章）后，对于他们 authority 和 hub 向量的计算只限于待选文章集合，这样将 2000 余维矩阵降至数十维，大大提高运算效率。这样做的想法是，一个文章的 authority 和 hub 向量很大程度上取决于它与类似文章的关系，而与不那么类似的文章关系不大，所以只选取与它类似的文章进行计算在很大程度上是保真的。最终的检索时间在 3-5s 之间。

2、文章聚类与评价

我在这一模块中测试了多种文章向量的纯度 purity 和互信息 NMI，这些文章向量由原文章向量经不同参数的 PCA 得到，包括原向量、降至 50 维的向量和降至 10 维的向量。由于 KMeans 算法每次得到的结果之间有差距，故我对于每种向量都取了 5 次实验的均值，发现维数越低，聚类效果越好。

```python
# original 原始数据
table = pd.read_csv("newsvec.csv")
table = table.drop(columns=['Unnamed: 0'])
p1, n1 = 0, 0
for i in range(5):
    purity, Fv, Nmi = clustering(table)
    p1 += purity
    n1 += Nmi
print("Purity: ", p1/5, "NMI: ", n1/5)
```

[159]

```
Purity:  0.5753707865168 54 NMI:  0.4907050722497015
```

```python
# pca 降至50维
table = pd.read_csv("vecpca2.csv")
table = table.drop(columns=['Unnamed: 0'])
p2, n2 = 0, 0
for i in range(5):
    purity, Fv, Nmi = clustering(table)
    p2 += purity
    n2 += Nmi
print("Purity: ", p2/5, "NMI: ", n2/5)
```

[158]

```
Purity:  0.624629213483146 NMI:  0.5715147520673607
```

```python
# pca 降至10维
table = pd.read_csv("vecpca.csv")
table = table.drop(columns=['Unnamed: 0'])
p3, n3 = 0, 0
for i in range(5):
    purity, Fv, Nmi = clustering(table)
    p3 += purity
    n3 += Nmi
print("Purity: ", p3/5, "NMI: ", n3/5)
```

[157]

```
Purity:  0.6893483146067416 NMI:  0.6352787216021816
```