目录

1、运行说明

1.1 项目文件（包含外部数据）

./util.py 项目的基础操作方法

./train_model.py 项目的模型训练文件，包括候选词的选择，候选词的特征选取、关于候选词的分类模型的训练

./generate_submit.py 对测试文件进行关键词抽取的主文件

./data/stopword.txt 停用词

./data/mingxing.dict
./data/zyi.dict
./data/yaoping.dict
./data/yinshi.dcit
搜狗词库字典

./data/train_1000.csv 处理好的初始 1000 条训练文本集(id, title, doc, key_words)

./data/test_107295.csv 需要抽取的 107295 条文本集合((id, title, doc)

./data/all_docs.txt 初始的 108295 条文本数据

./data/train_1000_candidate_add_title.pickle: 1000 条训练文本中抽取出的候选词及其对应的特征。

./data/test_107295_candidate_add_title.pickle 107295 条测试文本中抽取出的候选词及其对应的特征

1.2 项目运行

step 1: python train_model.py

主要步骤：

a.训练文本 topic 相关的模型包括 LSI 和 LDA

b.抽取候选词,并获取各个候选词对应的特征

step 2: python generate_submit.py 主要步骤：

a.将候选词及其对应的特征整理成分类样本
b.训练决策树二分类模型
c..对测试集文本中的候选词进行预测(获取为 key word)的概率
d.对每个文本进行关键词抽取

（因为 step1 比较耗时，所以这里保存了 step1 的结果，这里只需要运行 step2 即可）

2、解题思路方案说明

主要解题思路是把文本关键词抽取看成一个二类分类问题，将候选词分成关键词和非关键词两类，问题的解决就是从抽取出的候选词中找出两个 Score 更高的两个关键词。

2.1 候选词的选取

a.候选词的词性限定[‘n’,’nr’,’nz’,’ns’,’eng’,’nt’,’j’]

b.候选词的长度大于 1

c.候选词不在停用词表里

2.2 关键词的特征选择

a.是否在标题中(0,1)
b.候选词的词性[‘n’,’nr’,’nz’,’ns’,’eng’,’nt’,’j’]
c.候选词首次出现的位置

$$\text{feature(c)} = first\_index/doc\_len$$

d.候选词的长度
e.候选词的组成字符是否都是数字或者字母
f.候选词的 tf-idf 的值
g.第一句中候选词出现的次数（取前 30 个词为第一句）
h.最后一句中候选词出现的次数（取后 20 个词为最后一句）
i.LDA 模型中的候选词的主题分布与文档的主题分布的余弦相似度
j. LSI 模型中的候选词的主题分布与文档的主题分布的余弦相似度
k.词跨度长度((第一次该词出现的位置和该词最后一次出现的位置差)/文档的长度)。

$$\text{feature(k)} = last\_index - first\_index/doc\_len$$

2.3 候选词分类模型训练及标题关键词的偏重

a.构建分类模型训练样本。候选词出现在 train_1000.csv 样本中的 key_words 时记为 1.不出现则为 0。

b.训练二分类模型。主要需要解决分类不平衡的问题，这里通过设置 class_weight 来缓解分类不平衡问题。

c.从预测为关键词的候选词中选取出前两个。根据从 train_1000.csv 数据中发现关键词大部分来源于标题，制定了一些规则。

1）Score 计算方式：分类模型的输出预测为关键词的概率
2）标题中书名规则：从标题中查找书名，有则加入为关键词
3）标题中人名规则：从标题中查找词性为 nr 的候选词，加入为关键词
4）Score 排名规则：先从 Title 的候选词 Score 进行排名，选取关键词，如没有达到两个，再将 Doc 中的候选词 Score 进行排名，选取关键词

代码及数据下载地址:
https://pan.baidu.com/s/1-z9TBqKa8i5vUvESL48jQQ