信息检索作业2
基于lnc-ltn和向量空间的信息检索程序

计算机科学与技术
1511186
梁宸
qq:446822609

February 25, 2020

1 程序简介

本程序模型与作业要求一致, 权重模型为tf for docs, idf for queries, 相似度评分模型为向量空间, 文档和查询都以词干的形式处理. 语言为Python. 使用莎士比亚全集作为样例文档集. 本程序为控制台程序.

2 运行环境

本程序理论上运行在Python 3.6环境中, **应该**用到了两个外部包: nltk用于分词和提取词干, numpy用于矩阵运算.

3 运行

在包依赖满足的情况下, 运行launch.py. 程序启动时检查是否建立了词典和权重等文件, 如果没有就检索文档以构建. 之后程序接受3个命令: update用于重新构建词典和权重矩阵, 当文档集有更新时; query用于开始一个查询; close用于结束程序. 开始一个查询后, 需要输入一些查询单词.

4 实现简介

4.1 处理文档

先对每个文档分词、提取词干, 这里使用了nltk包中的Porter Stemmer提取算法, 之后只需进行一次扫描即可建立词典、tf和df矩阵. 这要求先遇到的单词出现在词典的前面, 这样对文档中的每一个词干, 先检查是否在词典中, 如不在则添加到词典的尾部, 之后, 如果该词干在当前文档的tf为0, 则该词干的df增加1. 同时更新tf.

扫描一遍文档集后, 获得了tf和df矩阵, 以及词典. 之后利用numpy包可以简便地得到lnc pattern的权重矩阵.

4.2 查询

查询的处理与上述文档的处理相似, 不同的是使用ltn机制. 值得注意的是ltn计算方式中零向量的产生: 当查询的所有单词都在所有的文档中出现时, idf因子为0, 出现零向量间断点, 本程序这里使用类似Laplace Smoothing的方式对idf因子进行了处理, 即对数分子项加1, 这样此情况下idf不会等于0而是趋于0; 当查询的所有单词都不在任何一个文档中出现时, tf因子为0, 出现零向量, 本程序的处理方式是警告用户, 并按自然顺序给出文档列表作为结果. 事实上tf也可以使用很多方式来进行smoothing, 但是作业好像要求不使用.

我个人认为零向量的出现, 归根结底, 应该是tf-idf模型的不足, 查询的tf为0的话, 与所有文档无关, 这也尚可解释; 但idf因子为0时, 该模型仅根据"所有doc都有这个term所以这个term就没有意义", 就在该维度给出了0的weight, 这显然是不能令人信服的, 之前发现tf-idf的定义与信息熵有很多相似之处, 以此问题为导火索, 查了一下得知, 它的定义其实是交叉熵(cross entropy), 简单解释(脑补)就是, 交叉熵可以理解为用分布p去拟合分布q时需要的信息量(编码长度), 在ML领域经常被用作损失函数. 对于tf-idf模型, p被解释为tf, 一个文档中出现某词项的个数, q被解释为df/N, 随机挑选一个文档, 出现该词项的概率. 那关于零向量的解释性, 应该问题就出现在这里了, 似乎cross entropy基本都是和smoothing一起使用的, 就是因为这个0.