**论文分析**

一、理论部分

1、概述

本文主要介绍了传统的用于文本分类的方法和作者新提出的两种方法，在文本分类任务的性能上面的比较，传统分类任务常用的方法有tf，tf\_idf，tf\_chi，tf\_ig，tf\_eccd，tf\_rf，iqf\_qf\_icf。而本文作者提出了两种新的基于熵的词的权重计算方案，即dc和bdc，然后本文在两个数据集reuters和snippets上面分别用两个分类器SVM和KNN去完成对数据集的分类任务，得到不同方案下不同分类器的分类效果，分类效果用MicroF1和MacroF1去衡量，结果中作者提出的基于熵的dc和bdc以及tf\_dc和tf\_bdc均实现了比较好的分类效果，验证了作者的理论。

2、词权重算法

* tf\_idf:

公式：

tf(t，d) = 某个词t在某个文档d中的个数/文档d中的总词数

idf（t）= (文档总数/含有词t的文档数)

tf\_idf(t) = tf(t，d)\*idf（t）

参数说明：

tf:词频，词t在某个文档中的频率，即是是相同词因为可能在不同的文档中存在，所以tf的值也是不同的，所以不能加入词库作为词的初始权重存在

idf：逆向文档频率，相同词的idf都相同所以在实验中加入了词库，作为词的初始权重

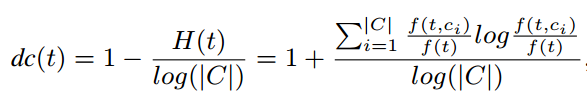
解释：

是经典的词的权重计算算法。词在某个文档中出现次数越多，对这个文档的分类作用就越大，如‘排球’在某个文档中广泛存在，则这个文档的类别很有可能是体育。词越分布在不同的文档中，对文档分类的作用越不明显，比如很多停用词‘的’，‘啊’广泛的分布在不同类别的文档中，对文档分类作用很低。

* tf\_dc:

公式：

tf(t，d) = 某个词t在某个文档d中的个数/文档d中的总词数



tf\_dc(t) = tf(t, d)\*dc(t)

参数说明：

f(t,ci)：含有词t且属于ci类别的文档数

f(t)：含有词t 的所有的文档数。

解释：

熵的定义：-

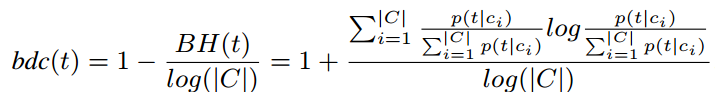
因为熵代表了混乱程度，熵越大则代表越混乱，即一个词分布在很多个类别的文档中所以这个词对于区别文档类别的作用就越小，所以dc的值就越小。而引入log|C|是为了使得dc的值取值在（0,1）。

* tf\_bdc:

公式：

tf(t，d) = 某个词t在某个文档d中的个数/文档d中的总词数

p(t|ci)=f(t,ci)/f(ci)



tf\_bdc(t) = tf(t, d)\*bdc(t)

参数说明：

f(t,ci)：含有词t且属于ci类别的文档数

f(ci)：代表类别为ci的文档数

解释：

p(t|ci)=f(t,ci)/f(ci)，这样就避免了因为某一类中文档数较多所带来的误差。

|  |  |  |  |
| --- | --- | --- | --- |
| 类别 文档数 | f(t,Ci) | f(Ci) | P(t|Ci) |
| C1 | 200 | 800 | 0.25 |
| C2 | 100 | 200 | 0.5 |

如果用dc则词t的f(t, C1)的值更大表明词t更加倾向于存在类别C1中，但是由于C1中含有的文档数远多于C2的文档数所以由p(t|Ci)的大小可知t更加倾向于存在类别C2中。所以引入了bdc以平衡因类别的大小带来的误差。

1. 实验部分
2. 数据、算法、模型说明

* 数据集：

只用了reuters，训练集和测试集分别为data文件夹下的Reuters\_train.txt，Reuters\_test.txt。其中data下的Reuters\_train01.txt用于写程序算法时调试和测试程序的正确性。

* 算法（词权重算法）：

只实现了传统的tf\_idf以及作者提出的tf-dc和tf\_bdc

* 模型（分类器）：

使用了KNN和SVM，并且没有进行调参

2、运行说明

* 先运行data\_preprocess构建词库，并将词库写入到data文件夹下的word\_lib.csv中
* 再运行experiment进行实验

3、框架说明

* data\_preprocess：

用于构建词库并且将词库写入文件word\_lib.csv中

* data\_util:

用于读取文件中数据

* algorithm:

定义了三种词权重算法从而构建每个doc的VSM

* classifier：

定义了两种分类器SVM和KNN，得到测试集样本的的预测标签

* experiment：

用于跑通整个文本分类的实验

1. 复现说明

* 数据预处理：

在复现过程中，是根据\t来切分不同文档的，所以要确保最后一行数据最后边没有进行换行，在用空格切分一个文档中的词要注意用strip()去掉每一个文档的末尾处可能含有的空白符。

* 构建词库：

词库中建立了三个列分别为word、idf、index，其中索引（主键）要设置成word，其中idf作为词的初始权重。因为相同的词均有相同的idf，不会因为所在文档的不同而不同。词库包含了训练集中所有出现过的词有19000多个。

* 构建词的权重算法：

这部分是复现的核心，注意要引入异常机制，因为词库是根据训练集得到的，有可能测试集中有些词是不存在。词的权重算法主要是将各个文档的原始数据转换成一个向量的表示形式，根据不同算法得到某文档中各个词的权重，然后根据词库将这些词对应的权重映射到19000多维上就得到了一个文档的向量表示。而19000多维超出内存，所以要引入稀疏矩阵来存储，行数就是文档列表的长度，列数就是词库的长度即19000多，存储的值就是不同算法得到的权重。在复现算法dc和bdc时读取了词库为df后要加入若干列，这些列的列名是不同的标签名，这列用于记录不同词在不同类中所在的文档数，以便算法的实现。

1. 复现收获与缺陷

* 收获

熟悉了稀疏矩阵以及df操作

强化了代码能力，强化了独立思考能力，因为复现dc和bdc的算法时是自己想出来要在df加上不同标签列，这是实现算法的核心创意。

提升了阅读文献能力。

熟悉了调分类器的操作。

知道了评价标准Micro\_F1和Macro\_F1的含义。

* 缺陷

没有进行对模型的调参。

没有学可视化数据的操作。

没有去阅读文章的引文。

没有熟悉tf\_idf的包的调包过程。

1. 参考资料

1、理论说明及代码实现

* [理论说明](http://zedom1.top/2018/06/01/VSM/)
* [代码实现](https://github.com/zedom1/nlp)