文档归类-开题报告

# 项目背景

随着互联网时代的蓬勃发展，人们积累了大量数字文本文档，如何有效的组织这些数据并从中进行信息挖掘，成为一个课题。最早的文本分类研究可以追溯到20世纪六十年代[1]，当时还以规则处理为主。近些年，伴随着计算机计算能力的大幅提升，以及统计机器学习和深度学习的快速发展成型，文本分类的方法也一直在变化。

文档归类，也称文本分类，指对文本数据集按照一定的分类体系或标准进行分类的标记。文本分类问题与其它分类问题没有本质上的区别，其方法可以归结为根据待分类数据的某些特征来进行匹配，选择最优的匹配结果，从而完成文档所属的一个或多个分类的标记[2]。

本项目将利用机器学习算法，从文档中抽取特征，完成文档的自动归类工作。将采用Usenet的20新闻组分类数据[3]进行模型的训练和验证。

# 问题描述

20新闻组分类问题是一个有监督多分类问题。每篇新闻属于且仅属于一个类别。通过对每篇文档进行特征提取，基于训练集的标注信息，对模型进行训练，并在测试集上验证模型效果，最终得到一个有效的模型可以对输入的文档完成正确分类信息的输出。模型将通过预测的分类结果的准确率来进行评估。

特征提取方面，将尝试TfIDF[4]，词向量[5]以及字符特征等方式来建模。模型方面，将尝试朴素贝叶斯，决策树，SVM以及深度学习等分类模型完成分类工作。

# 输入数据

20新闻组数据可以通过sklearn提供的函数sklearn.datasets.fetch\_20newsgroups直接下载，总共有18846个文档。按照函数提供的训练集、测试集分割方式，即基于某个日期进行数据的划分，训练集有11314个文档，测试集有7532个文档。训练集会进一步按20%的比例分割出验证集。训练集将用于模型的训练。验证集用于模型调参。测试集用于模型最终效果的评估。每个文档独立成为一个文件，类似电子邮件格式，包含头部，正文和尾部三个区块，头部包含发件人、标题等信息，正文是具体内容，尾部为一些署名信息。

在基于词向量的特征提取方案下，考虑到训练集样本规模有限，不足以训练语言模型，将引入text8[6]数据集使用word2vec[7]来训练词向量，该数据集为英文维基百科经过清洗和截断构造而成，每个单词均为小写并空格分隔，一共一行，截断到100MB大小而成。

# 解决方案

## 文本预处理

对于文本的处理主要有几种方式：

大小写转换，统一转换成小写。

去除标点符号。由于是做文档分类，而标点符号主要是些断句或者情感信息，对于分类没有帮助，因此需要去掉。但是在字符特征的建模下，需要保留符号作为分隔符。

每篇文档的头（header），脚注（footer）以及引用（quote）信息，也需要选择性去除。因为，这部分元信息对于文档分类的帮助可能不大，而且容易造成过拟合（比如某些用户只活跃在部分新闻组类别下，导致模型可能会基于用户名而不是文档内容来进行分类），因此需要有选择的去除。但是标题（Subject）需要保留，里面可能有一些用于分类的关键信息。

去除停用词，由于没有语义信息，也需要去除。

低频词，对于出现次数小于一定阈值的词也可以去掉，减少长尾词汇，缩减词表大小。

词干提取（stemming）与词形还原（lemmatization），对于单词的变体进行归一便于统一语义信息来完成分类。

## 特征提取

特征提取方面，会尝试使用TFIDF，词向量，字符等特征来建模。

采用TFIDF来进行特征提取，主要是考虑该方法对于在某篇文章中出现频率高，但是在所有文档中出现次数低的词给予较高的权重，也就是倾向于筛选出文章具有代表性的关键词。

词向量（Word2vec）特征可以压缩语义空间，保留语义信息的同时降低特征维度，可以使模型具备一定的语义理解能力，对于未登录词，使用0初始化特征。

字符特征则进一步简化特征维度，并通过深度模型自动加工提取特征。针对英语字母表，最终选择了69个字符来作为特征表示，包括26个小写字母，10个数字，32个其他字符和一个换行符，采用独热编码[8]进行建模。

## 分类模型

模型方面，由于TFIDF是词袋模型，没有考虑文本顺序信息，因此无法用于CNN等基于位置信息建模的模型，另外，此项目是文章分类，属于长文本，RNN在这种场景下也不适合直接应用。最后，词向量和字符特征把文档表示为2维特征，也无法应用于传统机器学习模型。

综上，最终计划采用如下的方案来实现文档归类：

|  |  |  |
| --- | --- | --- |
| 序号 | 特征提取 | 模型 |
|  | TFIDF | 决策树[9] |
|  | TFIDF | 朴素贝叶斯[10] |
|  | TFIDF | SVM[11] |
|  | 词向量 | CNN (TextCNN[12]) |
|  | 词向量 | CNN+LSTM (C-LSTM [13]) |
|  | 字符 | CNN (Char-CNN [14]) |

其中，决策树，朴素贝叶斯和SVM模型将使用sklearn中的实现，深度模型将使用keras+Tensorflow来完成。

# 基准模型

采用早期的论文[15]中的指标，其中提到的PrTFIDF模型在测试集的准确率为90.3%。论文中使用了标题和正文文本，并随机分割33%的数据作为测试集，其余数据作为训练集进行训练，由于没有引入验证集调参，论文中的模型可能有过拟合的风险。因此最终选择Stanford的分类器[16]的指标，准确率81.7%，该文献也是基于时间来分割测试集，与本文的数据分隔方法一致，可以作为对比。

# 评价指标

通过数据类别分析发现，各个类别的数据量相对比较均衡。因此，模型将基于准确率指标进行效果分析，即

yi 是第i个样本的真实分类标签，是第i个样本的预测分类标签，如果真实分类与预测相同，则I=1，否则I=0

# 项目设计

整体问题解决流程定义如下：

1. 数据获取。将通过sklearn提供的函数来完成数据的下载。
2. 数据分析。查看全部数据在各个分类分布的均匀程度，每篇文档的词数，以及词频统计。如果分类分布不均匀，则需要考虑进行数据增强或者平衡，避免影响模型准确率指标，或者更换为考虑数据分布不均匀情况的指标。
3. 数据预处理。把数据分成训练集、验证集和测试集。并准备大小写转换，停用词去除，标点去除等函数，用于后续模型构建过程按需使用。
4. 数据特征提取。包括词袋模型（BoW）的特征提取，词向量模型构建，字符特征构建。为后续模型训练做准备。
5. 模型训练与调优。在训练集上使用不同的模型进行训练，针对验证集的结果优化参数。根据前文所述，模型将采用决策树，朴素贝叶斯，SVM，CNN和LSTM，与特征抽取进行组合，总计有6种模型结果。在测试集上验证模型效果，绘制学习曲线和混淆矩阵评估模型性能。对于非深度模型，使用GridSearch搜索最优参数，对于深度模型，只能通过调整超参数，引入dropout，来优化模型效果。最终选出最优模型。
6. 结果分析。针对最优模型的表现进行分析，判断该模型是否可有效解决文档分类的问题。

# 参考文献

1. 文本分类概述. <http://blog.csdn.net/chl033/article/details/4733647>
2. 文本分类. 百度百科. <https://baike.baidu.com/item/%E6%96%87%E6%9C%AC%E5%88%86%E7%B1%BB>
3. 20 Newsgroups. <http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html>
4. TF-IDF. <https://baike.baidu.com/item/tf-idf/8816134?fr=aladdin>
5. Word embedding. Wikipedia. <https://en.wikipedia.org/wiki/Word_embedding>
6. Text8. <http://mattmahoney.net/dc/textdata>
7. Word2vec. <https://en.wikipedia.org/wiki/Word2vec>
8. 独热编码. https://en.wikipedia.org/wiki/One-hot
9. DecisionTree. <https://en.wikipedia.org/wiki/Decision_tree>
10. Naïve Bayes Classifier. <https://en.wikipedia.org/wiki/Naive_Bayes_classifier>
11. Cortes C, Vapnik V. Support-Vector Networks[J]. Machine Learning, 1995, 20(3): 273-297.
12. Kim Y. Convolutional Neural Networks for Sentence Classification[J]. empirical methods in natural language processing, 2014: 1746-1751.
13. Zhou C, Sun C, Liu Z, et al. A C-LSTM Neural Network for Text Classification[J]. arXiv: Computation and Language, 2015.
14. Zhang X, Zhao J J, Lecun Y, et al. Character-level convolutional networks for text classification[J]. neural information processing systems, 2015: 649-657.
15. Joachims T. A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization[C]. international conference on machine learning, 1997: 143-151.
16. Stanford Classifier. https://nlp.stanford.edu/wiki/Software/Classifier/20\_Newsgroups