

大数据计算技术实践报告



题 目 Novel Authorship Recognition

姓 名 潘朝鑫 侯戚通 陈达

学 号 31601182 31601172 31601112

专业班级 计算1604 1604 1602

所在学院 计算学院

二○ 19 年 4月 22日

1. 课题背景

小说作者识别是给定一个待判定作者的小说内容，比如某一章，甚至是某一段文字，通过作品涉及到的争议作者所著的其他作品推测作者写作风格，并与待判定归属的作品风格进行比较，从而识别出真正的作者。

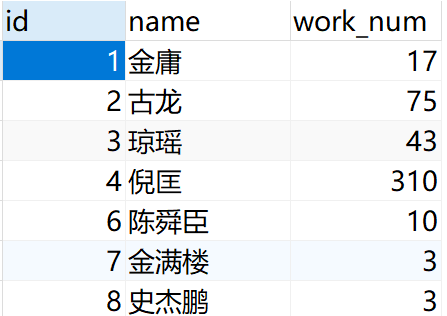
作者的写作风格体现在其文章的语法，词汇，语篇结构，句式以及句法等特点中，是写作过程中所形成的个人语言特征。不同的作者风格不同，基于这一特点可以比较准确推断出某篇文章的作者。

在此背景下，我们提出将写作风格作为特征，结合机器学习分类模型的研究方法，以期获得较好的识别效果。

1. 作者识别概述
   1. 数据采集

数据来源：悠读文学网 https://www.yooread.net/。采集到的数据包括：

作者（author）主要信息

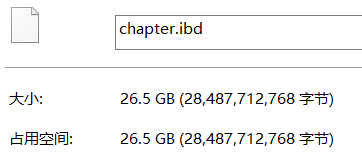


作品（novel）主要信息



章节（chapter）主要信息（文件大小26G左右）





采集策略：

* 作者专区页 https://www.yooread.net/writer/，采集作者信息
* 作者详细页 https://www.yooread.net/writer/387/，采集作品信息
* 作品详细页 https://www.yooread.net/1/1/，采集章节信息
* 章节内容页 https://www.yooread.net/1/1/1.html，采集章节内容
  1. 数据预处理
     1. 目标格式说明

项目的目标是作者识别，所以数据集以作者为单位进行分离，每个作者单独一个文件，该文件包含该作者所有作品的所有章节内容

数据集划分：按7比3的比例将该作者的所有作品分离成训练集和测试集。

* 训练集文件名：作者ID-r-00000\_train\_小说数量
* 测试集文件名：作者ID-r-00000\_test\_小说数量

样本只有1个时，划给训练集。测试集为0，但是还是生成对应的测试文件，读取文件时直接忽略掉

以下是划分样例说明

|  |  |  |
| --- | --- | --- |
| 样本总数 | 训练集样本数 | 测试集样本数 |
| 1 | 1 | 0 |
| 2 | 1 | 1 |
| 3 | 2 | 1 |
| 4 | 3 | 1 |
| 5 | 4 | 1 |
| 6 | 5 | 1 |
| 7 | 5 | 2 |
| 8 | 6 | 2 |
| 9 | 7 | 2 |
| 10 | 7 | 3 |
| 11 | 8 | 3 |
| 12 | 9 | 3 |
| 13 | 10 | 3 |
| 14 | 10 | 4 |
| 15 | 11 | 4 |
| 16 | 12 | 4 |
| 17 | 12 | 5 |
| 18 | 13 | 5 |
| 19 | 14 | 5 |
| 20 | 14 | 6 |
| 21 | 15 | 6 |
| 22 | 16 | 6 |
| 23 | 17 | 6 |
| 24 | 17 | 7 |
| 25 | 18 | 7 |
| 26 | 19 | 7 |
| 27 | 19 | 8 |
| 28 | 20 | 8 |
| 29 | 21 | 8 |
| 30 | 21 | 9 |

每个文件，一行存放一条记录，代表一个章节。格式如下：4个字段，字段之间用\t隔开。其中作者ID、小说ID、章节ID依次升序排列

* 一个文件代表一个作者，所以同一文件中，所有记录的作者ID都相同
* 章节ID升序排序，章节ID小的，代表章节序相对靠前，保证小说内容的连贯性
* 章节内容是以段落为单位的，段落之间以空格作为分隔符

作者ID 小说ID 章节ID 章节内容

特别说明：数据包含中国小说和外国小说，外国小说大部分都是译文，由于译者和原作者的身份不同，所以数据分析阶段不包括外国小说。

* + 1. 处理策略

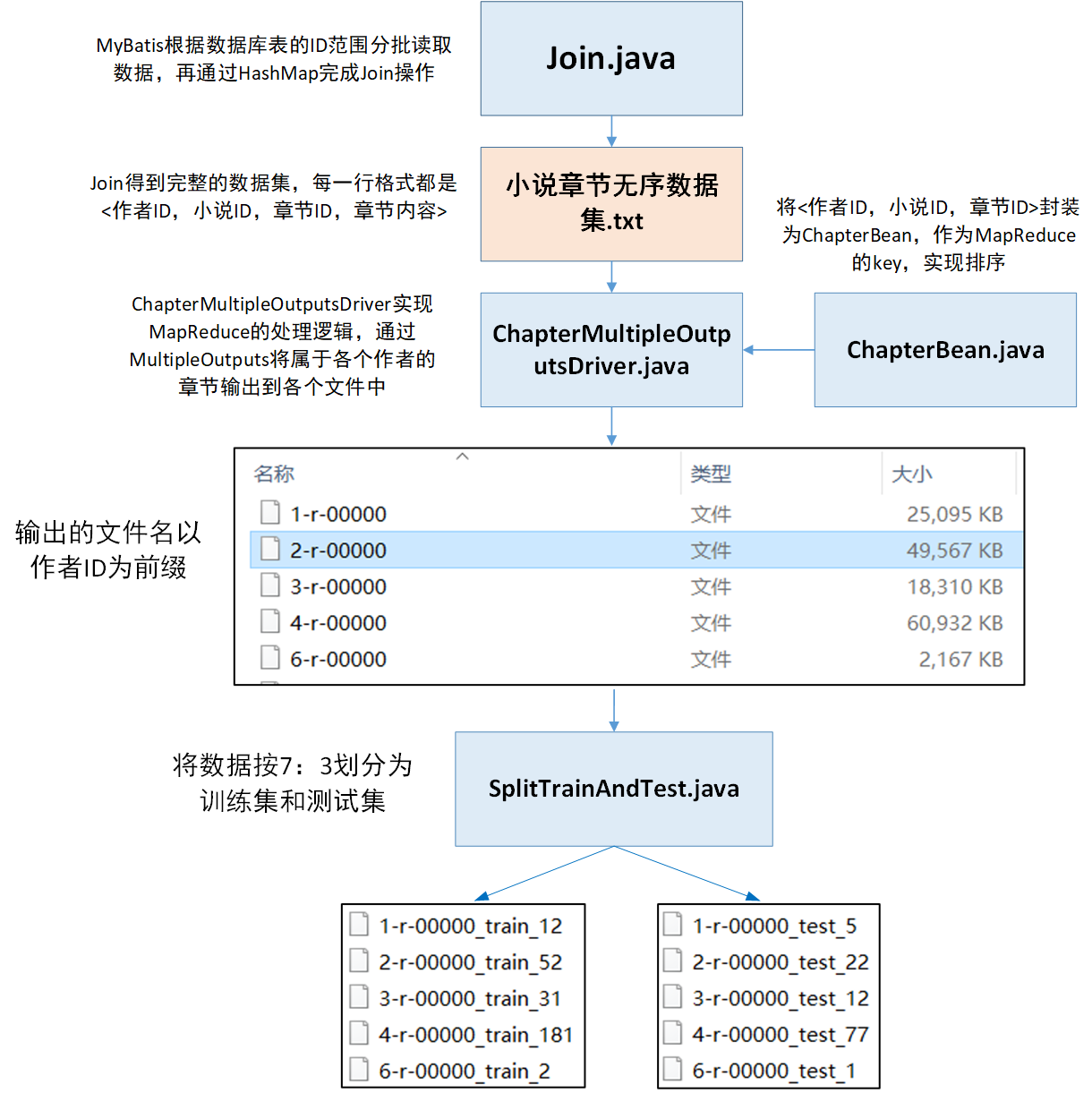
在采集数据阶段，文章内容是一段一段采集的，此时已经过滤掉无意义的空段，这样就不存在缺失值了。为了便于观察，以空格作为段落分隔符，保存到数据库中。因为文章原内容一般不包含空格，即使有，也不影响我们后续的分词分析处理，所以没有关系。

每一条记录有 作者ID、小说ID、章节ID、章节内容 这四个字段，这需要将novel表和chapter表连接。因为数据集太大，无法通过内存一次性完成处理，所以选择通过Java分批完成Join操作（代码见Join.java）。

Join完成之后，得到小说章节无序数据集。此时通过MapReduce，对数据集进行排序（作者ID、小说ID、章节ID依次升序排列），并以作者为单位划分为各个文件（代码见ChapterMultipleOutputsDriver）。

最后通过SplitTrainAndTest.java，将数据集划分为训练集和测试集。

整个过程的流程图如下：



* 1. 特征提取

作者的写作风格体现在其文章的语法，词汇，语篇结构，句式以及句法等特点中，我们将这些特点分别用不同特征进行表示。其中包括标点符号，词频，句长，词性等特征，共计29维。

汉语中的词分为实词和虚词，其中虚词包括副词，介词，连词，助词，叹词，拟声词。虚词通常不代表任何具体的含义，只代表一定的语法含义。大量的研究表面，虚词比实词更有规律，更能体现作者的写作风格，故我们在结巴词库中找到词性为以上6种的词，然后按词频进行排序，选择前300个词作为虚词特征，共计300维。

故，共计提取329维特征，以特征的方式来数字化的表示作者的写作风格。

以下为特征说明。（特征以一章节为单位）

段落与句长特征（3）

|  |  |
| --- | --- |
| 特征名 | 特征说明 |
| 段落数 | 段落数 |
| 段落平均长度 | 平均每段落字长 |
| 句子平均长度 | 平均句子长度 |

标点符号特征（9）

|  |  |
| --- | --- |
| 特征名 | 特征说明 |
| 问号占比 | 问号数/标点符号总数 |
| 句号占比 | 句号数/标点符号总数 |
| 逗号占比 | 逗号数/标点符号总数 |
| 感叹号占比 | 感叹号数/标点符号总数 |
| 引号占比 | 引号数/标点符号总数 |
| 冒号占比 | 冒号数/标点符号总数 |
| 省略号占比 | 省略号数/标点符号总数 |
| 分号占比 | 分号数/标点符号总数 |
| 单括号占比 | 单括号数/标点符号总数 |

词长特征（8）

|  |  |
| --- | --- |
| 特征名 | 特征说明 |
| 1字词占比 | 词长为一的词数/总词数 |
| 2字词占比 | 词长为二的词数/总词数 |
| 3字词占比 | 词长为三的词数/总词数 |
| … |  |
| 8字词占比 | 词长为八的词数/总词数 |

词性特征（9）

|  |  |
| --- | --- |
| 特征名 | 特征说明 |
| 名词占比 | 词性为名词的词数/总词数 |
| 副词占比 | 词性为副词的词数/总词数 |
| 形容词占比 | 词性为形容词的词数/总词数 |
| 介词占比 | 词性为介词的词数/总词数 |
| 连词占比 | 词性为连词的词数/总词数 |
| 助词占比 | 词性为助词的词数/总词数 |
| 叹词占比 | 词性为叹词的词数/总词数 |
| 动词占比 | 词性为动词的词数/总词数 |
| 拟声词占比 | 词性为拟声词的词数/总词数 |

虚词特征（300）

|  |  |
| --- | --- |
| 特征名 | 特征说明 |
| ‘在’（介词） | ‘在’的词数/词性为介词的总词数 |
| ‘和’（连词） | ‘和’的词数/词性为连词的总词数 |
| ‘也’（副词） | ‘也’的词数/词性为副词的总词数 |
| ‘等’（助词） | ‘等’的词数/词性为助词的总词数 |
| ‘呀’（叹词） | ‘呀’的词数/词性为叹词的总词数 |
| ‘叮’（拟声词） | ‘叮’的词数/词性为拟声词的总词数 |
| … |  |

1. 实验结果与模型评价

由于小说作者识别是给定一个待判定作者的小说内容，通过作品涉及到的争议作者所著的其他作品推测作者写作风格，从而识别出真正的作者。故是以作者风格为特征的分类问题，即将小说内容进行分类，标签即为作者身份。

故采用十种机器学习常用分类算法，将写作风格作为特征，作者身份为标签，进行分类，并统计准确率。

以下为十种分类算法及其准确率，包括决策树（Decision Tree）、K近邻算法（K-Nearest Neighbor）、Logistic回归、随机森林(Random Tress)、梯度提升树(GBDT)、多层感知机（MLP）、支持向量机（SVM）、朴素贝叶斯（Naïve Bayes）、xgboost、Rocchio算法。

|  |  |  |
| --- | --- | --- |
| 作者数 | 分类算法 | 准确率 |
| 5 | K-Nearest Neighbor | 0.94 |
| Logistic | 0.9860288534548216 |
| Random Forest | 0.9148063781321184 |
| GBDT | 0.9564160971905846 |
| xgboost | 0.9778283978739559 |
| MLP | 0.988914198936978 |
| Rocchio | 0.9599088838268792 |
| SVM | 0.979195140470767 rbf  0.3233105542900532 poly |
| Naive Bayes(GaussianNB) | 0.7902809415337889 |
| Naive Bayes(BernoulliNB) | 0.9249810174639331 |
| Naive Bayes(MultinomialNB) | 0.9804100227790433 |
| Decision Tree | 0.6813971146545178 |

|  |  |  |
| --- | --- | --- |
| 作者数 | 分类算法 | 准确率 |
| 10 | K-Nearest Neighbor | 0.9384057971014492 |
| Logistic | 0.9841897233201581 |
| Random Forest | 0.9181488801054019 |
| GBDT | 0.9288537549407114 |
| xgboost | 0.9713438735177866 |
| MLP | 0.9871541501976284 |
| Rocchio | 0.9562747035573123 |
| SVM | 0.9681324110671937 |
| Naive Bayes(GaussianNB) | 0.8111824769433466 |
| Naive Bayes(BernoulliNB) | 0.944828722002635 |
| Naive Bayes(MultinomialNB) | 0.9826251646903821 |
| Decision Tree | 0.6195652173913043 |

|  |  |  |
| --- | --- | --- |
| 作者数 | 分类算法 | 准确率 |
| 20 | K-Nearest Neighbor | 0.9203102961918195 |
| Logistic | 0.9651237338120272 |
| Random Forest | 0.8398512629824336 |
| GBDT | 0.8495319912809335 |
| xgboost | 0.92184895499423 |
| MLP | 0.9742915758430568 |
| Rocchio | 0.9177458648544685 |
| SVM | 0.8692781125785357 |
| Naive Bayes(GaussianNB) | 0.7892678548531863 |
| Naive Bayes(BernoulliNB) | 0.9159507629183229 |
| Naive Bayes(MultinomialNB) | 0.9509552506731632 |
| Decision Tree | 0.4733299140915502 |

|  |  |  |
| --- | --- | --- |
| 作者数 | 分类算法 | 准确率 |
| 50 | K-Nearest Neighbor | 0.8454347935930473 |
| Logistic | 0.8892931130311758 |
| Random Forest | 0.6888989944924461 |
| GBDT | 0.7651963013490981 |
| xgboost | 0.8633217118892426 |
| MLP | 0.9385579303723914 |
| Rocchio | 0.8497802031226315 |
| SVM | 0.681396650277811 |
| Naive Bayes(GaussianNB) | 0.5907230559345157 |
| Naive Bayes(BernoulliNB) | 0.8787327573139305 |
| Naive Bayes(MultinomialNB) | 0.8705977464504068 |
| Decision Tree | 0.43014501541104544 |

|  |  |  |
| --- | --- | --- |
| 作者数 | 分类算法 | 准确率 |
| 100 | K-Nearest Neighbor |  |
| Logistic | 0.780983627649651 |
| Random Forest | 0.5849014120750105 |
| GBDT |  |
| xgboost |  |
| MLP | 0.8110585737720737 |
| Rocchio | 0.7629386599761974 |
| SVM |  |
| Naive Bayes(GaussianNB) | 0.47055228537424815 |
| Naive Bayes(BernoulliNB) | 0.8826272958281064 |
| Naive Bayes(MultinomialNB) | 0.7565376821383769 |
| Decision Tree | 0.31876226317990286 |

优化：

Logistic L2比L1好

SVM rbf远好于poly

测试网址：http://nahook.gitee.io/author1