文本分类实验项目日志（最后更新-2019.07.19）

冉亚鑫

目录

[文本分类实验项目日志（最后更新-2019.07.19） 1](#_Toc14448043)

[1 One vs rest & stacking 二分类多分类器文本分类实验 2](#_Toc14448044)

[1.1 描述 2](#_Toc14448045)

[1.2 实验设计 3](#_Toc14448046)

[1.2.1 旧方法 3](#_Toc14448047)

[1.2.2 新方法 5](#_Toc14448048)

[1.3 实验结果和分析 6](#_Toc14448049)

[1.3.1 实验-旧方法 6](#_Toc14448050)

[1.3.2 实验-新方法 9](#_Toc14448051)

[1.3.3 实验-新方法-新数据 10](#_Toc14448052)

[1.4 结论 11](#_Toc14448053)

[1.5 集成学习的资源 11](#_Toc14448054)

# One vs rest & stacking 二分类多分类器文本分类实验

## 描述

根据5-6月份 one vs rest（all）和stacking的融合实验的若干问题，6月中旬进行了改进。

5-6月份最终形成的方案为：融入KB；加入第一层分类器；改变KB返回类目数量，形成多个对比实验。具体方法和结果详见1.3.4

5-6月份实验存在的问题说明如下：

1. 类间的相似性可能是影响最终结果的主要因素；
2. 采取one-hot文本表示出现的维度过大以及OOA问题（如对于5:4:1比例，测试集占1/10，若测试集中某个词未出现在训练二分类器中的词空间里，则不能表示该特征，那么测试时会失去该特征，进而影响测试结果）；
3. 最终准确率难以提升。

6月初提出的改进方案如下：

1. 去掉知识库；
2. 加入第二层层级分类，使用fasttext训练二级分类器。
3. 使用词向量表示文本特征。
   1. 1.3给出了具体的实验设计和结果。

## 实验设计

### 旧方法

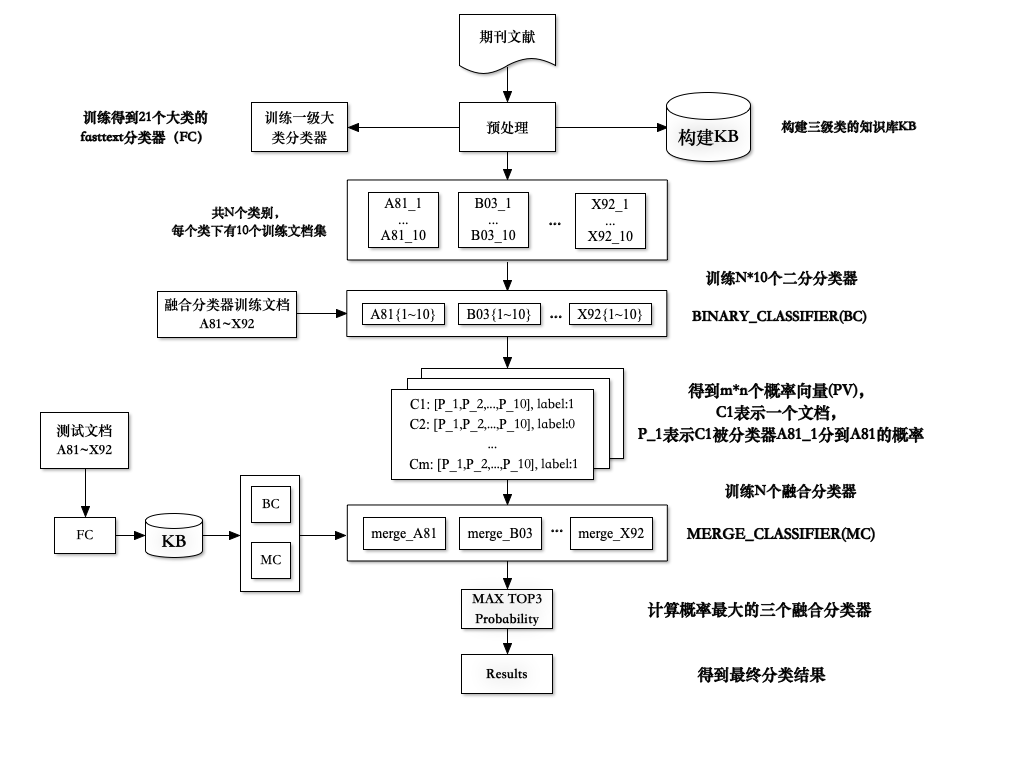
详细的实验方法如图1所示。（旧版本）

图1 文本分类框架

整个实验可分为6个环节：数据预处理，训练一级fasttext分类器，构建三级分类知识库，训练二分类分类器，训练融合分类器和测试分类器。每个环节的详细方法如下：

1. 数据预处理。原始数据集保存在两个txt文件，大小共约680MB，具体数据形状如图2所示。

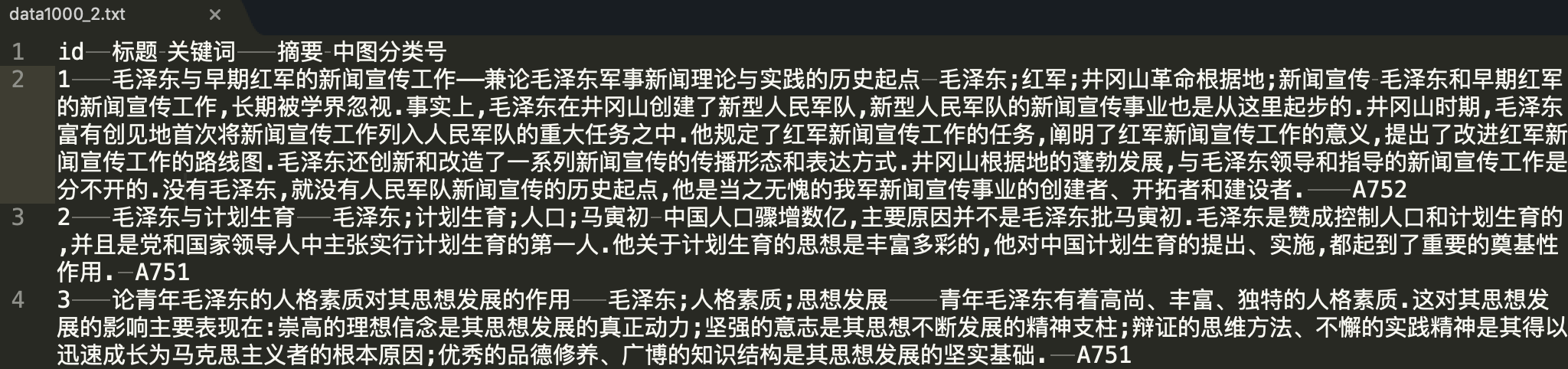


图2 数据集形状

根据实验设计思路，需要抽取并生成新的数据集。表1列出了4次实验的数据集详细信息，其中1-3次是5月份实验用数据集，第4次是6月份数据集，第4次与前3次的**区别**是：（1）类别数量增加一倍左右；（2）每个三级类的文档数量未固定，即类别数据分布不均衡。（3）细化数据清洗环节（如抽取时考虑类号名字特点、去除无效文档、去除重复词语等）。

表1 数据集介绍

|  |  |  |  |
| --- | --- | --- | --- |
| 实验次数 | 大小 | 类别数量 | 文档数量 |
| 1 | 7MB | 10 | 1000\*10=10000 |
| 2 | 332MB | 467 | 1000\*467=467000 |
| 3 | 278MB | 358 | 1000\*358=358000 |
| 4 | 312MB | 773 | 727367 |

数据集划分：每个类下，训练二分类数据集：训练融合分类数据集：最终测试集 = 5:4:1。

下面的流程主要以实验4的数据集为例。

1. 训练一级分类器。将所有数据集按照每行数据为‘\_\_label\_\_A,word1\tword2’的格式存储。A代表一级分类号，word1代表文档分词后的第一个词。然后使用fasttext模型训练分类器，最后抽取总数据集的20%作为一级分类器测试。
2. 训练二分分类器。对于每个三级类，长度为le3\_length，那么该类的二分训练集长度为le3\_binary\_length = le3\_length\*0.5，从这些数据中随机抽取10个子集，然后加入其它三级类的若干数据（长度大致与属于本类数据相同，记为sub\_n），最终的目标是使得每个子集包含sub\_n个属于该类的文档，sub\_n个不属于该类的文档。这2\*sub\_n个文档使用机器学习分类算法训练一个二分类器。共得到773\*10=7730个分类器。
3. 训练融合分类器。将1中训练的每个类的10个分类器融合训练为一个分类器。方法是：对于某个类（如A81类），长度为le3\_length，那么该类的融合训练集长度为le3\_merge\_length = le3\_length\*0.4，抽取le3\_merge\_length条该类文档和不属于该类的le3\_merge\_length条文档组合成2\*le3\_merge\_length条文档，然后用10个该类的二分分类器进行分类，结果以概率形式给出（分到类A81的概率），并保存为长度为10的向量，标签为1或0（如“A81”类数据，则标签记为1，非“A81”类记0）。然后将这些向量进行训练，得出融合的分类器。这样共得到773个融合分类器。
4. 测试分类器。测试过程包括5个方面：测试一级分类器；知识库（KB）分类；测试二分分类器；测试融合分类器；测试系统分类器。其中，测试系统分类器有多种方案，将在1.3.4节的结果和分析中提到。

### 新方法

新方法与旧方法的区别如下：

1. 去掉KB。原因是经过实验发现，加入KB会过滤掉正确的类号；去掉KB的副作用是增加测试的时间开销。
2. 加入二级分类。在一级分类的基础上加入二级分类，原因是为了降低测试的时间开销。
3. 文本表示阶段采用词向量方法。原因是原来方法（tf-idf）会生成词汇映射表文件（将近900MB），并且会造成维度过大和OOA问题；使用词向量可以避免前两个问题，OOA问题与词向量模型以及数据集有关。
4. 使用词向量中的词作为自定义词典**分词**。这样做的好处是在文本表示阶段尽可能使每一个特征（词语）都能够用词向量表示。（目前还在实验阶段）

新方法的具体步骤如下：

1. 数据预处理。同1.2.1的1，仅使用第4次抽取的数据集。
2. 训练一级分类器。同1.2.1的2。
3. 训练二级分类器。将所有数据集按照每行数据为‘\_\_label\_\_A8,word1\tword2’的格式存储。A8代表二级分类号，word1代表文档分词后的第一个词。然后使用fasttext模型训练分类器，最后抽取总数据集的20%作为测试。
4. 训练二分分类器和融合分类器。步骤与1.2.1的3和4大致同，仅文本表示阶段不同。
5. 测试分类器。测试过程包括5个方面：测试一级分类器；测试二级分类器；测试二分分类器；测试融合分类器；测试系统分类器。其中，测试系统分类器较复杂，将在1.3中提到。

## 实验结果和分析

### 实验-旧方法

本次实验加入了层级分类（FC）和知识库（KB），第一层分类器使用fasttext算法，第二层分类器使用svm算法。其中，fasttext的测试结果为0.95.

最终测试时，经过和老师及学长讨论形成了两种方案，后来经过实验，又总结了几种优化策略。下面是这几种方案的介绍。

1. 文档首先经过KB，得到20个三级类号，然后仅在这20个类的20\*10=200个二分类器和20个融合分类器中进行分类测试。记为way\_1。
2. 文档首先经过7730个二分类器和773个融合分类器，然后将所有分类结果按概率从大到小排序，与KB的结果取交集。记为way\_2。
3. 加入一级分类。文档首先经过一级分类器，得到一级分类号（如A），并经过KB，得到20个三级分类号（如A81,A84,B02,…,C81），然后仅在一级分类号下并且存在于20个三级分类号的分类器中进行分类测试（如在A81,A84中测试）。记为way\_3。

以上三种方案，把最后得到排名前三的三级类号作为最终的分类结果。

此外，加入了baseline，即不加入KB和层级分类的结果，如下表所示。

表2 实验4结果

|  |  |  |
| --- | --- | --- |
| **实验代号** | **准确率** | **运行时间/s** |
| Baseline | 0.184 | 58113(约16h) |
| Way\_1 | 0.215 | 14886(约4.1h) |
| Way\_2 | 0.216 | 28106(约7.8h) |
| Way\_3 | 0.231 | 7035(约2h) |

从表2可以看出，增加KB，准确率提高了近0.03，并且时间性能大大改善。way\_1和way\_2主要差别在于时间，way\_1运行时间更短，这与方案（1）和方案（2）的步骤有关。（1）和（2）是计算所有三级类的分类结果并排序，而（1）仅需要计算由知识库得到的三级类下的分类器并排序，因此（1）的时间性能更好。

实验Way\_3验证了加入第一层分类器是有效的，时间性能提高了一倍，准确率提高了0.015。因此下一步将考虑以way\_3为基础进行优化。

实验发现，对于方案（3），增加从知识库中返回的结果数目，如经过KB得到30个结果，最终的准确率得到提高。因此，进行了几组优化实验，如表3所示。

表3 way\_3的优化实验结果

|  |  |  |  |
| --- | --- | --- | --- |
| **实验代号** | **KB类号数** | **准确率** | **运行时间/s** |
| Way\_3 | 20 | 0.231 | 7035(约2h) |
| Way\_4 | 30 | 0.269 | 10951(约3h) |
| Way\_5 | 50 | 0.305 | 20411(约5.6h) |
| Way\_6 | 60 | 0.312 | 20150(约5.6h) |

由表3，增加KB的类号后，准确率显著提高，但相应的时间开销也增加。

为了分析具体分类时每个类的效果，实验统计了每个文档在分类时的第一层分类结果、KB分类结果、最终分类结果列表（top10）。以way\_6为例，选取三个有代表性的类别，其中其整体的准确率如表4所示，选取其中若干个文档具体分析结果如表5所示。

表4 三个类的准确率

|  |  |  |
| --- | --- | --- |
| **三级类号** | **文档数** | **准确率** |
| A849 | 1526 | 0.84 |
| D08 | 820 | 0.475 |
| Q593 | 732 | 0.000 |

表5 三个文档的具体分类结果

|  |  |  |  |
| --- | --- | --- | --- |
| **文档** | **KB结果** | **Le1结果** | **最终结果（取top10，不足10个全部选择）** |
| A849\_C1 | A:12;B:1;C:1;D:20… | A | ['A849', 'A81'] |
| D08\_C1 | B:1;C:1;D:5;Q:41… | D | ['D52','D43','D90'] |
| Q593\_C1 | S:7;Q:40… | Q | ['Q942','Q945','Q95-3','Q25', 'Q53','Q935','Q954','Q936', 'Q948', 'Q24'] |

由表5知，在D08类中一个文档的KB结果中出现了较多的Q类，并且最终分类结果中未命中D08；在Q593类中一个文档中虽然大部分都是Q类，但最终结果（top3）并没有命中。

表4和表5只是列出了个别文档的分类情况，这三个类全部分类情况见附件1.

此外，为了分析其它类的分类效果，统计了那些效果较差的类别，如图5所示。其中，横坐标表示准确率，纵坐标表示低于准确率的三级类类别数量。类别总数为773.

图5 三级类准确率分布情况

从图5可以看出，大部分三级类的准确率低于0.5，有将近一半低于0.3，有20个三级类准确率为0.0.

基于目前的实验，可考虑的优化方法包括这几个方面：改善KB；改善第一层分类器性能；剔除效果非常差的类别。

### 实验-新方法

本次实验加入了第二层级分类（FC），并且文本表示使用词向量方法（百科训练的词向量模型，大小1.7GB），第二层分类器使用fasttext算法，第三层分类器使用svm算法。其中，第一层分类器的测试结果为0.95，第二层分类器的测试结果为0.93.

新方法的几次过程结果如表6所示。

表6 新方法实验过程结果

|  |  |  |  |
| --- | --- | --- | --- |
| **实验代号** | **描述** | **准确率** | **运行时间/s** |
| Baseline(Way\_6) | 旧方法最好结果 | 0.312 | 20150(约5.6h) |
| New\_way\_1 | 加入二级分类器 | 0.334 | 14169(约3.9h) |
| New\_way\_2 | 在new\_way\_1基础上去掉KB | 0.484 | 64741(约18h) |
| New\_way\_3 | 在new\_way\_2基础上使用词向量 | 0.455 | 33425(约9.3h) |

由表6，New\_way\_1加入二级分类器的效果提升不大，但是时间开销降低。New\_way\_2去掉KB后准确率大幅提升，达到0.484，但是时间开销增大3倍多。New\_way\_3使用词向量，准确率降低了0.03，但时间开销降低了一倍。

由此分析，新方法是有效的。

对于新方法New\_way\_3，使用词向量仍会出现OOA问题，故使用词向量中的词作为自定义词典分词。这样做的好处是在文本表示阶段尽可能使每一个特征（词语）都能够用词向量表示。这个方法目前在实验阶段。

### 实验-新方法-新数据

本次数据选取的是规模为7.32GB数据，经过预处理后，大小为5.16GB。预处理包括分词、去除无效数据等。分词使用百度词向量生成的词典进行分词。5.16GB数据的情况如表7所示。

表7 新数据集介绍

|  |  |  |  |
| --- | --- | --- | --- |
| **实验次数** | **大小** | **类别数量** | **文档数量** |
| 5 | 5.16GB | 1679 | ？ |

由于类别数据少会影响分类性能，故去掉数据小于500条的三级类别，共得到851个三级类。

根据这851个类进行实验，方法与1.3.2相同。

使用fasttext训练一级分类器的准确率为0.92，训练二级分类器的准确率为0.77。

表8给出了实验的结果。

表8 新方法-新数据的结果

|  |  |  |  |
| --- | --- | --- | --- |
| **实验代号** | **描述** | **准确率** | **运行时间/s** |
| New\_way\_data\_1 | 和New\_way\_3不同的是未加入2级分类器 | 0.20 | 147581(约41h) |
| New\_way\_data\_2 | 方法和new\_way\_3同 | 0.41 | 35556(约9.8h) |

## 结论

根据新-旧实验和若干次讨论，得到结论如下：

1. 加入第二层分类器是有效的。对比表6的几个方案结果，发现加入第二层分类对分类效果和时间性能有一定提升。
2. 去掉知识库是有效的。根据表6的结果，去掉KB能大幅提升准确率。
3. 使用词向量可以缩短训练和测试时间。根据表6结果，词向量在准确率方面和TF-IDF方法相差不大，但时间开销降低一半，此外，使用词向量不需要生成额外的映射文件。

根据以上结论，新方法是目前最好的文本分类方案，将应用到实际的标引项目中。

## 集成学习的资源

集成学习-1：<https://blog.csdn.net/qq_20386411/article/details/82985219>

集成学习-2: <https://blog.csdn.net/qq_32690999/article/details/78759463>

Stacking原理：<https://blog.csdn.net/wstcjf/article/details/77989963>

Stacking代码实现：<https://www.jianshu.com/p/5905f19c4df6>

模型融合(stacking&blending) ：<https://blog.csdn.net/u014356002/article/details/54376138>