基于Hadoop MapReduce并行计算框架的邮件自动分类

曹佳涵　白家杨　刘笑今
2019st04小组

摘　要

本实验的目标是通过 MapReduce 编程来实现邮件的自动分类，通过本课程设计的学习，可以体会如何使用 MapReduce 完成一个综合性的数据挖掘任务，包括全流程的数据预处理、样本分类、样本预测等。我们使用Hadoop MapReduce并行计算框架对原始邮件文本进行特征选择、特征向量权重计算、文本分类和样本预测任务，从而搭建起一个完整的文本分类处理流程。我们采用了不同的特征提取方法和分类算法并进行比较和分析。此外，我们使用Spark同样完成了实验。

关键词：邮件分类，并行计算，MapReduce，Spark

1　引入

此模板是基于 LaTeX 的标准文类 article设计，也即意味着你可以把 article 文类的选项传递给本模板，比如 `a4paper`，`10pt` 等等（推荐使用 `11pt`）。本模板支持 PDFLaTeX 和 XeLaTeX[1] 两种编译方式。

数学字体的效果如下：

$$(a+3b)^n = \sum_{k=0}^{n} C_n^k a^{n-k}(3b)^k \tag{1}$$

1.1　全局选项

我在这个模板中定义了一个语言选项 `lang`，可以选择英文模式 `lang=en`（默认）或者中文模式 `lang=cn`。当选择中文模式时，图表的标题引导词以及参考文献，定理引导词等信息会变成中文。你可以通过下面两种方式来选择语言模式：

```
\documentclass[lang=cn]{elegantpaper} % or
\documentclass{cn}{elegantpaper}
```

[1]中文字体均使用 ctex 包设置。

1.2 自定义命令

在此模板中，并没有修改任何默认的命令或者环境，所以，你可以在此模板使用原来的命令和环境。另外，我自定义了 3 个命令：

1. \email：创建邮箱地址的链接；
2. \figref：用法和 \ref 类似，但是会在插图的标题前添加 **<图 n>** ；
3. \tabref：用法和 \ref 类似，但是会在表格的标题前添加 **<表 n>**；
4. \keywords：为摘要环境添加关键词。

1.3 列表环境

你可以使用列表环境（itemize、enumerate、description），示例如下：

```
\begin{itemize}
   \item Routing and resource discovery;
   \item Resilient and scalable networks;
   \item Distributed storage and search.
\end{itemize}
```

- Routing and resource discovery;
- Resilient and scalable networks;
- Distributed storage and search.

1.4 插图

插图的命令和以前一样，也是使用 figure 环境。**图 1** 显示了插图的效果。你可以把你的图放到当前工作目录的如下子目录下 (./image/, ./img/, ./figure/, ./fig/)。

```
\begin{figure}[htbp]
  \centering
  \includegraphics[width=0.6\textwidth]{scatter.pdf}
  \caption{Scatter Plot Example \label{fig:scatter}}
\end{figure}
```

1.5 表格

我强烈建议你使用 booktabs 宏包，这个宏包有三个命令 \toprule、\midrule 和 \bottomrule 能方便你制作三线表。**表 1** 是一个示例：

```
\begin{table}[htbp]
  \small
  \centering
  \caption{Auto MPG and Price \label{tab:reg}}
    \begin{tabular}{lcc}
    \toprule
                    &       (1)          &         (2)       \\
    \midrule
```

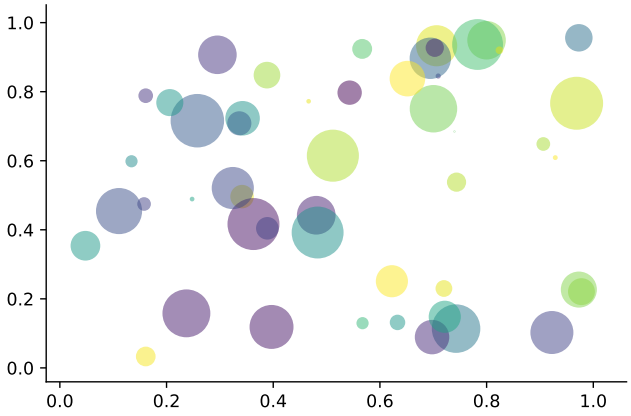

图 1: Scatter Plot Example

```
    mpg             &     -238.90***      &       -49.51      \\
                    &      (53.08)        &       (86.16)     \\
    weight          &                     &       1.75***     \\
                    &                     &       (0.641)     \\
    constant        &      11,253***      &        1,946      \\
                    &      (1,171)        &       (3,597)     \\
    obs             &         74          &          74       \\
    $R^2$           &       0.220         &       0.293       \\
    \bottomrule
    \multicolumn{3}{l}{\scriptsize Standard errors in parentheses} \\
    \multicolumn{3}{l}{\scriptsize *** p<0.01, ** p<0.05, * p<0.1} \\
    \end{tabular}%
\end{table}%
```

表 1: Auto MPG and Price

	(1)	(2)
mpg	-238.90***	-49.51
	(53.08)	(86.16)
weight		1.75***
		(0.641)
constant	11,253***	1,946
	(1,171)	(3,597)
obs	74	74
R^2	0.220	0.293

Standard errors in parentheses

*** p<0.01, ** p<0.05, * p<0.1

1.6 参考文献

此模板使用了 BibTEX 来生成参考文献，默认使用的文献样式（bib style）是 GB/T 7714-2015[2]。参考文献示例：？使用了中国一个大型的 P2P 平台（人人贷）的数据来检验男性投资者和女性投资者在投资表现上是否有显著差异。

你可以在谷歌学术，Mendeley，Endnote 中获得文献条目（bib item），然后把它们添加到 wpref.bib 中。在文中引用的时候，引用它们的键值（bib key）即可。注意需要在编译的过程中添加 BibTEX 编译。如果你想在参考文献中添加未引用的文献（部分或者全部），可以使用

```
\nocite{EINAV2010, Havrylchyk2018} % add the two reference.
\nocite{*}   % add all the reference in the bib file.
```

如果你想修改参考文献的样式（比如改为 aer），你可以在导言区将下面代码注释掉。

```
\usepackage[authoryear]{gbt7714}
```

并且文档末尾添加

```
\bibliographystyle{aer}
```

2 数据分析

共有20个类别，20000个文件.....

3 MapReduce处理流程

3.1 特征选择

本任务的主要工作是对原始的邮件文本中进行特征选择，选择出能够表征邮件主题的特征词，为后续的文本分类做准备。

对于输入的未分词的邮件训练样本全集和停词表，我们需要输出全局邮件文本特征，并对它们进行相应的编号，此外，对于训练数据集的目录，将目录名（即文本类别）转换为相应的类别序号。

对于该步骤，我们采用了两种计算方法，分别为非并行化和并行化计算方法。

3.1.1 非并行化计算方法

非并行化计算方法主要思路是顺序执行，首先读取20_newsgroup文件夹下的子文件夹，将子文件夹名（即类别名）转化为类编号并进行存储。将 Lucene 的 Standard Analyzer 分词器作为基类，输入停词表，自定义自己的带有停词表功能的停词器，然后顺序依次读取每个子文件夹下的所有

[2]通过调用 gbt7714 宏包

文件，使用自定义的停词器对文本进行分词，并将分词结果储存在目标文件中。为了后续处理的方便性，我们修改输出文件的格式为：filename-classNum，filename为原文件的文件名，classNum为它所属的类别的编号，两者用分隔符 '-' 隔开。

伪代码如下：

Algorithm 1 特征选择非并行化算法

Input: 邮件训练样本全集 U，停词表 S

Output: 用停词表分割后的干净文本，全局邮件文本特征集

初始化类别名-类别编号对应表 classMap

初始化停词器 stopAnalyzer=StopAnalyzer(S)

for each 文本 u in U **do**

新文本 $u\prime$ = stopAnalyzer(u)

通过类别名-编号对应表查找文本u的文件名filename对应的类别编号num=classMap(u)

在目标目录下创建filename-num文件，并将新文本$u\prime$的内容写入该文件。

end for

3.1.2 并行化计算方法

非并行化顺序执行，缺点也显而易见：计算速度慢，效率低，因此我们设计了并行化的计算方法，利用 MapReduce 计算框架，很好地并行处理大量的训练文本，极大地加快了处理速度。

在Map阶段读取原文本，利用停词器将其分词后输出到目标目录下，同时对于每一个单词，发射(word,filename-classNum)键值对。值得注意的是，因为停词表是所有Mapper都需要各自初始化的，因此将其初始化放在Map阶段的setup过程中，首先将停词表文件路径存在DistributedCache中，然后在setup过程中读取该路径并利用其初始化停词器，在map过程中使用该停词器。

在Reduce阶段，设置一个全局变量n，用来表示每一个单词的唯一标号。输入为(word, value)。由于在Reduce阶段，已经将相同key的键值对都整合到了一起，因此读取的word值是唯一不重复的，只需要利用全局变量n为每一个单词分配相应的标号，并将单词和编号输出到目标文件中即可。

伪代码如下：

3.2 朴素贝叶斯

3.2.1 原理推导

朴素贝叶斯算法是生成模型中分类算法之一。所有朴素贝叶斯分类器都假设特定特征的值独立于给定类变量的任何其他特征的值。例如，如果水果是红色的，圆形的，直径约10厘米，则可以认为它是苹果。朴素贝叶斯分类器认为这些特征中的每一个都独立地贡献于该果实是苹果的概率，而不管任何可能的颜色，圆度和直径特征之间的相关性。

Algorithm 2 特征选择并行化算法

Input: 邮件训练样本全集 U，停词表 S

Output: 用停词表分割后的干净文本，全局邮件文本特征集

初始化类别名-类别编号对应表 classMap

初始化停词器 stopAnalyzer=StopAnalyzer(S)

for each 文本 u in U **do**

新文本 $u\prime$ = stopAnalyzer(u)

通过类别名-编号对应表查找文本u对应的类别编号n=classMap(u)

在目标目录下创建文件filename-num文件，并将新文本$u\prime$的内容写入该文件。

end for

抽象来看，朴素贝叶斯是一个条件概率模型：给定一个要分类的问题实例，用向量表示 $\mathbf{x} = (x_1, \ldots, x_n)$代表一些n个特征（自变量），它为这个实例分配概率$p(C_k \mid x_1, \ldots, x_n)$其中$C_k$表示$K$个可能的类别中的一个。

使用贝叶斯定理，条件概率可以分解为

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

“朴素”的条件独立假设发挥作用：假设所有特征都在$\mathbf{x}$是相互独立的，对类的条件C_k。在这个假设下，

$$p(x_i \mid x_{i+1}, \ldots, x_n, C_k) = p(x_i \mid C_k)$$

因此，可以表示为

$$p(C_k \mid x_1, \ldots, x_n) = \frac{p(C_k)}{p(\mathbf{x})} \prod_{i=1}^{n} p(x_i \mid C_k)$$

因此只要比较$p(C_k)\prod_{i=1}^{n} p(x_i \mid C_k)$的值就可以判断类别。即为求下式

$$\hat{y} = \underset{k \in 1,\ldots,K}{\operatorname{argmax}}\ p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$

常用的有高斯朴素贝叶斯，伯努利朴素贝叶斯Bernoulli和多项式朴素贝叶斯Multinomial三种。

高斯朴素贝叶斯当处理连续数据时，典型的假设是与每个类相关联的连续值根据正态（或高斯）分布分布。

多项式朴素贝叶斯使用多项事件模型，样本（特征向量）表示多项式生成某些事件的频率$(p_1, \ldots, p_n)$其中 p_i是事件i发生的概率（或多类情况下的K个多项式）。在词袋模型中事件表示单个文档中单词的出现。

伯努利朴素贝叶斯：在多变量伯努利事件模型中，特征是描述输入的独立布尔值（二元变量）。

尽管朴素贝叶斯分类器具有朴素设计和明显过于简单的假设，但它在许多复杂的现实世界中仍然运行良好。综合比较表明，贝叶斯分类的表现优于其他方法，特别是在文本处理方面。

3.2.2 hadoop实现

词袋模型就是表示文本特征的一种方式。给定一篇文档，它会有很多特征，比如文档中每个单词出现的次数、某些单词出现的位置、单词的长度、单词出现的频率……而词袋模型只考虑一篇文档中单词出现的频率(次数)，用每个单词出现的频率作为文档的特征（或者说用单词出现的频率来代表该文档）。利用词袋模型表示文本特征来进行朴素贝叶斯分类器的实现。

实际上就是求

$$\hat{y} = \underset{k \in 1,\ldots,K}{\operatorname{argmax}}\ p(C_k) \prod_{i=1}^{n} p(w_i \mid C_k)$$

其中w_i表示文档中的第i个单词。

由于每个概率值很小（比如0.0001）若干个很小的概率值直接相乘，得到的结果会越来越小。为了避免计算过程出现下溢，最终无法比较，引入对数函数Log，在 Log 空间中进行计算。然后得到如下公式

$$\hat{y} = \underset{k \in 1,\ldots,K}{\operatorname{argmax}}\ log\ p(C_k) + \sum_{i=1}^{n} log\ p(w_i \mid C_k)$$

训练朴素贝叶斯的过程其实就是计算先验概率$p(C_k)$ff和后半部分似然函数的过程。

3.2.3 先验概率p(c)的计算

$P(C_k)$的意思是：在所有的文档中，类别为C_k的文档出现的概率有多大。假设训练数据中一共有N_{doc}篇文档，只要数一下类别C_k的文档有多少个就能计算$P(C_k)$了，类别C_k的文档共有N_k篇，先验概率的计算公式如下：

$$p(C_k) = \frac{N_K}{N_{doc}}$$

3.2.4 似然函数$p(w_i|C_k)$的计算

由于是用词袋模型表示一篇文档d，对于文档d中的每个单词w_i，找到训练数据集中所有类别为C_k的文档，数一数单词w_i在这些文档（类别为C_k）中出现的次数：$count(w_i, C_k)$。

然后，再数一数训练数据集中类别为$C_k$ff的文档一共有多少个单词

$$\sum_{w \in v} count(w, C_k)$$

其中V是词库，即所有在文本中出现过的单词。（有些单词在词库中，但是不属于类别C_k，那么$count(w, C_k) = 0$ff

计算二者之间的比值，就是似然函数的值。似然函数计算公式如下：

$$p(w_i \mid C_k) = \frac{count(w_i, C_k)}{\sum_{w \in v} count(w, C_k)}$$

需要考虑特殊情况某一类的文档不包含某个词时，考虑add-one-smoothing，类似于拉普拉斯修正，其实就是将“出现次数加一”。此时似然函数变为：

$$p(w_i \mid C_k) = \frac{count(w_i, C_k) + 1}{\sum_{w \in v}(count(w, C_k) + 1)} = \frac{count(w_i, C_k) + 1}{\sum_{w \in v} count(w, C_k) + |V|}$$

其中$|V|$表示词库的单词数。

在hadoop的实现中分成了两个Job来完成，分别是训练阶段的Job和预测阶段的Job

Algorithm 3 朴素贝叶斯训练阶段第一个Job

Input: 邮件训练样本全集 U

Output: 输出训练结果文件，记录键值对<类名#单词，数量>

Map阶段：分割每个单词并根据文件名获得类名编号，输出<类名#单词，数量>

for each 单词 w in U **do**

类名编号$classnum = GetClassNum(GetFileName(w))$

Emit($classnum$ + ”#” + w, 1)

end for

Combine阶段：合并<类名#单词，num1>、<类名#单词，num2>为<类名#单词，num1+num2>

Reduce阶段：合并并输出<类名#单词，数量>到输出文件中

sum = 0

for all <类名#单词，数量num> with same key **do**

类名编号$sum+ = num$

end for

Emit(类名#单词, sum)

Algorithm 4 朴素贝叶斯预测阶段第二个Job

Input: 第一个Job训练结果键值对集合<>训练邮件预测样本全集 U，类编号和类文件数对应表B

Output: 输出训练结果文件，记录键值对<类名#单词，数量>

Map阶段：分割每个单词并根据文件名获得类名编号，输出<类名#单词，数量>

Map阶段setup：将训练结果A键值对保存到Map<String, int>结构的TrainSet中，并提取键值对<类名#单词，数量>中的单词存入词库WordSum中

Map阶段setup：将对应关系B保存到Map<String, int>结构的ClassSet中

Map阶段map：

for each 单词 w in U **do**

类名编号$classnum = GetClassNum(GetFileName(w))$

for each 类别 c in ClassSet.keys **do**

类名编号$classnum = GetClassNum(GetFileName(w))$

Emit($classnum$ + ”#” + w, 1)

end for

Emit($classnum$ + ”#” + w, 1)

end for

Combine阶段：合并<类名#单词，num1>、<类名#单词，num2>为<类名#单词，num1+num2>

Reduce阶段：合并并输出<类名#单词，数量>到输出文件中

sum = 0

for all <类名#单词，数量num> with same key **do**

类名编号$sum+ = num$

end for

Emit(类名#单词, sum)