基于 SVM 的学科试题自动分类研究

刘一然 骆力明
(首都师范大学信息工程学院 北京 100048)

摘 要 针对海量学科试题所引发的试题管理混乱现象,提出一种基于 SVM 的学科试题自动分类法。对学科语料进行分词、去停用词和统一相似词等预处理操作;采用 TextRank 算法对学科语料进行权重赋值,将语料文本向量化;用 SVM 中的 Linear 核函数训练语料得到分类器。用语料库中的 12 类单选题进行测试,整体分类的准确率、召回率、F1 值均达到 97% 以上。结果表明,该分类法可以有效地分类试题文本。

关键词 试题分类 TextRank SVM

中图分类号 TP391 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2019.01.036

AUTOMATIC CLASSIFICATION OF SUBJECT EXAMINATION QUESTIONS BASED ON SVM

Liu Yiran Luo Liming
(*Information Engineering College*, *Capital Normal University*, *Beijing 100048*, *China*)

Abstract Aiming at the confusion of examination questions management caused by massive subject examination questions, this paper proposed an automatic classification of subject questions based on SVM. The subject corpus was preprocessed by word segmentation, stop words removal and similar words unification. TextRank algorithm was used to assign weight to the subject corpus by vectorizing the corpus text. Linear kernel function in SVM was used to train corpus to get classifier. Twelve types of single choice questions in the corpus were tested. The overall classification accuracy, recall rate and F1 value were all above 97 %. The results show that the proposed classification method can effectively classify text of examination questions.

Keywords Question classification TextRank SVM

0 引 言

随着互联网的快速发展,以文本为主的非结构数据急剧增长,在教学领域表现为学科试题的爆炸性增长,这给学科试题管理带来严峻挑战。一方面,数量庞大的学科试题使传统手工试题分类法变得不切实际;另一方面,缺乏管理的学科试题阻碍了用户对其的使用。针对以上现象,本研究借鉴文本分类[1-2]领域中的统计学习方法,采用 SVM[3-7]算法实现了学科试题的自动分类,将试题自动对应到教材各章,达到快速管理学科试题的目的。

文本分类是指在给定分类体系下,根据文本内容自动确定文本类别的过程。目前,针对中文文本的分类研究已取得一些成效。

在中文长文本分类方面,周庆平等[8]提出一种基于聚类改进的 KNN 算法并对 11 类新闻语料进行分类,分类的准确率、召回率及 F1 值均比使用传统的 KNN 分类算法有所提高,而且还在一定程度上解决了 KNN 算法的耗时问题。杨帅华等[9]基于粗糙近似集模型提出了一种 KNN 改进算法,用 λ 近似集和上近似集表示文本类的分布特征,在 λ 取得合适阈值后,可在保持文本分类准确率基本不变的前提下,提高分类效率,并用 10 类作文语料进行实验证明了算法的可行

收稿日期:2018-08-30。国家自然科学基金面上项目(61672361);北京成像技术高精尖创新中心项目(BAICIT-2016004)。
刘一然,硕士生,主研领域:自然语言处理。**骆力明**,教授。

性。赵燕等[10]采用朴素贝叶斯分类法将农业文本分为4类,结果表明该方法可以很好地对农业文本进行分类,分类性能较高。

关于中文短文本的分类,高元等[11]提出一种融合随机森林和贝叶斯多项式的标题分类算法并在22类图书标题语料上进行实验,平均分类的准确率和召回率均达到70%以上,但由于底层使用的分类器数量多所以导致实验效率不高。马丽菲等[12]提出一种基于中文短影评分类的本体和决策树相结合的方法,将1 500篇中文短影评分为8类,结果表明该分类法比传统的分类效果提高3%,查准率达到90.1%。

阅读文献后发现,目前对中文长文本的分类研究已经成果斐然,但对中文短文本的分类研究仍然不足,并且关于学科试题的分类研究更是寥寥无几。因此,本文提出的针对学科试题的自动分类具有研究的价值。

1 系统整体架构

本研究建立在C++面向对象程序设计这门课程的基础上,选择该学科做实验对象有以下两点原因:(1)"C++语言程序设计"既是一门典型的大学课程,又是计算机二级考试中的一项考试科目,故以该学科为实验对象有着广大的应用需求。(2)这门课程的学科语言不仅包含常规的自然语言而且还存在独特的代码语言,故对其的研究成果不但包含了对常规学科的研究而且还具有鲜明的C++学科特征。

采用SVM算法对C++学科试题自动分类的总体实现流程如下:

(1)搜集语料并标注。实验语料包括C++单选题及学科的PPT课件资源。

(2)试题初分类。根据学科试题特点,将其分为三类:代码题、非代码题且题目含解题信息的试题、非代码题且题目不含解题信息的试题。

(3)对PPT语料和试题语料进行预处理操作,具体包括:分词、去停用词、统一相似词。

(4)采用TextRank[13-14]算法对语料进行权重赋值。

(5)用SVM中的Linear核函数训练经上步处理后的语料,得到分类模型。

(6)使用分类模型对测试试题进行分类。

图1给出了基于SVM的C++学科试题分类的系统架构图。

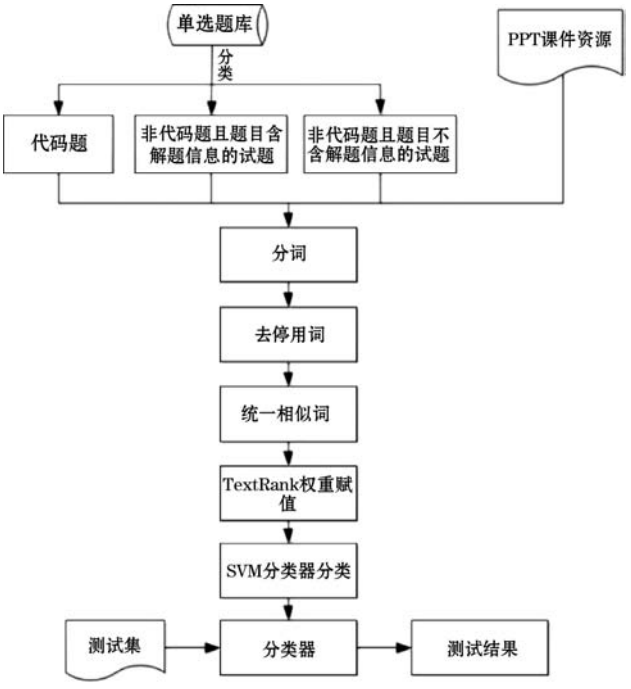

图1 基于SVM的C++学科试题分类的系统架构图

2 数据集介绍

2.1 学科试题

本文从网上各大文库以及考试的纸质试卷中搜集了1 500多道C++单选题并对其进行人工标注分类,分类的依据来源于清华大学出版社出版的《C++语言程序设计》[15],教材共12章,每一章对应一个分类类别。

学科试题通常是由文字组成的,但也不排除由其他非文字(图片等)组成的情况。本研究只对自然语言进行处理,所以如果试题语料中存在非文字试题,则需将其全部找出并改写为文字试题。之后,可以根据文字试题的特征对其进一步分类,比如,根据描述试题的语言类型可将试题分为中文试题和英文试题两类;根据试题题目的特征将试题分为题目含解题信息和题目不含解题信息两类。

分析实验语料库中的单选题,发现部分试题的题目中含有代码,若要确定这类试题的知识点,必须分析代码含义,故需将代码题分离出来,单独处理。分离出代码题后,余留的非代码题中包含有一类试题,在其题目中找不到任何与解题相关的信息,比如:题目"以下说法正确的是"。这类试题四个选项的知识点可能互不相同,故需一一分析。据此,本研究将试题库中的C++单选题分为三类:代码题、非代码题且题目含解题信息的试题、非代码题且题目不含解题信息的试题。

(1)代码题:

此处的代码题是指题目含有代码段或 C ++ 语句的单选题。例如,试题 1 为题目含代码段的单选题,试题 2 为题目含 C ++ 语句的单选题。

试题 1. 下列程序段中具有相同值的是(　)class Base{int b;}; class Base1:virtual public base{int b1;}; class Base2:virtual public base{int b2;}; class derived:public base 1,public base 2{int b3;}; derived d;

A、d. b 与 d. base1::b

B、d. base1::b 与 d. base1::b1

C、d. b 与 d. b3

D、d. base2::b 与 d. base2::b2

A

试题 2. 在"int a[][3] = {{1},{3,2},{4,5,6},{0}};"中,a[2][2]的值是(　)

A、1

B、0

C、6

D、2

C

目前,本研究通过正则表达式可自动识别出代码中的函数、继承、数组、指针、地址和引用。具体内容见表 1。

表 1　通过分析代码可自动识别出的信息汇总表

类别	通过正则表达式可准确识别出的知识
函数	函数、函数重载、递归调用函数、普通的函数调用
继承	私有继承、公有继承、保护继承、虚继承
数组	二维数组、一维数组、对象数组、字符数组、通过地址引用二维数组、通过地址引用一维数组
指针	指针、函数的指针、指向指针的指针、数组指针、指针数组
引用	引用
地址	地址

上述试题 1 中的代码可匹配正则表达式"class[ ]{1}[a - z0 - 9A - Z] + [:]{1}public[ ]{1}[a - z0 - 9A - Z] + "和"class[ ]{1}[a - z0 - 9A - Z] + [:]{1}virtual[ ]{1}public[ ]{1}[a - z0 - 9A - Z] + ",前者表示"公有继承",后者表示"虚继承"。同理,分析试题 2 可自动识别出:二维数组。这些通过分析代码而得到的词语对后续试题的分类发挥着重要作用。

(2) 非代码题且题目不含解题信息的试题:

这类试题的特点是试题题目不含任何与解题相关的学科信息,据此可将这类试题分离出来。单独处理这类试题的原因在于题中四个选项的知识点可能互不相同,所以不能将整道试题当作一个分类语料进行处理。例如:

试题 3. 下列正确的选项是(　)

A、继承是创建一个具有别的类的属性和行为的新类的能力

B、C 语言支持面向对象的程序设计

C、空白符是一种可以被编译的符号

D、标识符不宜过长,一般设定为 16 个字符

A

试题 3 即为一道非代码题且题目不含解题信息的试题,选项 A 的知识点是"继承",可归类到第 7 章(继承与派生)中;选项 B 的知识点是"面向对象",可归类到第 1 章(绪论)中;选项 C 和 D 的知识点均为"词法记号",可归类到第 2 章(C ++ 简单程序设计)中。鉴于这类试题选项知识点的特征,需将每个选项均视作一个分类语料。

(3) 非代码题且题目含解题信息的试题:

分离出上述两类试题后,余下的试题可统一称为非代码题且题目含解题信息的试题,这类试题的分类语料为将正确答案代入到题目后的结果。例如:

试题 4. (　)的功能是对对象进行初始化。

A、析构函数

B、数据成员

C、构造函数

D、静态数据成员

C

试题 4 的分类语料为"构造函数的功能是对对象进行初始化"。

2.2　PPT 课件资源

本研究中,训练语料的来源除 C ++ 学科的单选题外还包括教师授课过程中不可或缺的 PPT 课件资源。因为课件内容涵盖课程的重要知识点,故可将其作为实验部分训练语料。与试题语料一致,PPT 语料也分为 12 类。

3　数据预处理

3.1　分　词

汉语分词是自然语言处理必不可少的一个环节。目前,常用的分词工具有哈工大的语言技术平台(LTP)、中科院的 NLPIR 汉语分词系统、开源自由的 JAVA 工具包(HanLP)以及 Python 中的中文分词组件 jieba 等。

笔者曾尝试使用 LTP、HanLP 以及 jieba 对本研究中的训练语料进行分词操作,结果显示,这三种分词方

法均可较为准确地分割出常用词语,但对学科词语的分词效果却并不理想,这是因为用来训练分词器的语料通常缺乏领域性,故不能有效识别领域词语。鉴于该情况且考虑到本研究是在 Python 环境下进行的,从而选用了 jieba 分词组件。

为提高分词准确率,本研究在分词过程中加入了人机交互的自定义词典,即机器自动分词后,人工判断结果中是否存在错分词,若存在,可向自定义词典中添加正确分词结果,从而满足实际需要。举例说明见图 2。

```
5.在下面有关析构函数的叙述中，正确的是（）
分词结果如下：
5. 在 下面 有关 析 构 函数 的 叙述 中 ， 正确 的 是 （ ）

是否存在错分词（y：存在；n：不存在）：y
请输入新词（输入n时结束）：析构函数
请输入新词（输入n时结束）：n
重新分词结果：
5. 在 下面 有关 析构函数 的 叙述 中 ， 正确 的 是 （ ）
```

图 2 人机交互向自定义词典中添加新词

图 2 表明 jieba 分词器在自动分词时,无法正确识别“析构函数”这个 C++学科词语。为了正确分出该词语,笔者将其加入到自定义词典中。结果表明分词器可依据自定义词典中的词汇准确分割词语。

3.2 去停用词

语料的分词结果中存在一些对后续试题分类无关的词语,如:介词、语气词、连接词等。本研究搜集了哈工大停用词库、四川大学机器学习智能实验室停用词库以及百度停用词表,合并三表且根据实际情况,加入一些在本研究中不需要进行分析的词语。例如,试题 6 去除停用词后的结果见图 3。

试题 6. 关于动态多态性的下列描述中( )是错误的

A、动态联编是以虚函数为基础的

B、动态绑定是在运行时确定所调用的函数代码的

C、动态关联调用函数操作是指向对象的指针或对象引用

D、运行时多态性是编译时确定操作函数的

D

```
【试题6】去除停用词后的结果如下：
动态多态性
动态联编 虚函数 基础
动态绑定 运行时 确定 调用 函数代码
动态关联 调用函数 操作 指向 对象的指针 对象引用
运行时多态性 编译时 确定 操作函数
```

图 3 试题 6 去除停用词后的结果

在去除试题 6 语料中的停用词时,除使用原始的停用词表外,笔者还向其中加入了“描述”和“错误”这两个与后续试题分类无关的词语。

3.3 统一相似词

语料中常常存在一些意思一致的不同词语,出现该现象的原因主要有两点:一是由于汉语同义词较多;二是由于不同出题者对词语的选取有着不同程度的偏好。为减少相似词对后续分类的干扰,在此需将语料中的相似词统一为同一个常用词语。例如,试题 6 统一相似词后的结果见图 4。

```
【试题6】统一相似词后的结果如下：
动态多态性
动态多态性 虚函数 基础
动态多态性 运行时 确定 调用 代码
动态多态性 调用函数 操作 指向 对象指针 对象引用
动态多态性 编译时 确定 操作函数
```

图 4 试题 6 统一相似词后的结果

在试题 6 去除停用词后的语料中,词语“动态多态性”、“动态联编”、“动态绑定”、“动态关联”和“运行时多态性”所表示的意思都是一样的,在此笔者将其统一为“动态多态性”这个词语。

4 TextRank 算法

TextRank 是一种基于图结构的模型,其思想来源于 Google 公司的 PageRank[16] 算法。TextRank 算法的主要构造方法为:首先将文档分词;然后以这些词作为图的顶点,词与词之间的联系作为边,通过这种点与边的连接来构造图结构模型;接着利用投票原理对文本中的重要词语进行排序,顶点所得票数越多,证明它就越重要。

本研究采用 TextRank 的无向图模型,每个节点的初始权值为 1,迭代计算最终权重。计算公式如下:

$$WS(V_i)=(1-d)+d\times\sum_{V_i\in in(V_i)}\frac{w_{ji}}{\sum_{V_k\in out(V_j)}w_{jk}}WS(V_j) \quad (1)$$

式中:$WS(V_i)$ 是节点 V_i 的权重值(称为 PR 值);d 为阻尼系数,表示图中某一节点跳转到其他任意节点的概率,一般取值为 0.85;$in(V_i)$ 是指向点 V_i 的所有节点的集合,$out(V_j)$ 是节点 V_j 所指向的所有节点的集合。式(1)右侧的求和表示与节点 V_i 相邻的每个节点对该节点的贡献程度。实验中,部分语料的权重值如图 5 所示。

基本数据类型@0.14552223965 运算符重载@0.14552223965 运算对象@0.177238
运算对象@0.133008327207 调用@0.156727250664 运算符重载@0.178930368451
编译@0.233463005532 机制@0.233463005532 运算符重载@0.299610983405 函
重载@0.25 已有的运算符@0.25 C++语言@0.25 运算符重载@0.25
运算符@0.333333333333 运算符重载@0.333333333333 结合性@0.333333333333
个数@0.333333333333 操作数@0.333333333333 运算符重载@0.333333333333
重载@0.25 运算符@0.25 操作数@0.25 运算符重载@0.25
重载@0.291970755552 运算符重载@0.416058488896 友元函数@0.291970755552
运算符重载@0.14552223965 函数@0.177238880175 形参@0.14552223965 运算
操作数@0.177238880175 运算符重载@0.177238880175 后置@0.14552223965
操作数@0.177238880175 运算符重载@0.14552223965 友元函数@0.177238880175
运算符重载@0.162074557254 友元函数@0.162074557254 对象@0.162074557254
运算符@0.2 操作数@0.2 友元函数@0.2 形参@0.2 函数@0.2
友元函数@0.171818917628 形参@0.170887650298 后置@0.171818917628 运算
动态多态性@0.5 编译时多态性@0.5

图 5　部分语料的权重值

5　SVM 分类算法

SVM 是一种基于统计学习理论的有监督机器学习算法。该算法常用的核函数有 4 种,分别为:线性核函数、多项式核函数、高斯(RBF)核函数以及 sigmoid 核函数。

利用 SVM 训练语料得到分类模型的步骤如下:

(1) 将训练语料转化为 SVM 算法可以处理的形式。

(2) 选择合适的核函数。由于线性核函数参数少,速度快,对线性可分数据,其分类效果理想,故一般情况下首先尝试用线性核函数来做分类。

(3) 改变实验参数值,寻找最优参数。

(4) 用最佳参数训练语料得到分类模型。

图 6 显示了当惩罚因子 $C=1$(默认)时,用十折交叉法验证线性核函数的分类准确率实验结果。

```
D:\python27\python.exe "D:\软件\PyCharm Community Edition 2017.1.3\helper
Testing started at 9:48 ...
Launching unittests with arguments python -m unittest discover -s D:/pyth
分类准确率:
[ 0.78091873  0.81205674  0.87096774  0.88530466  0.8057554   0.88129496
  0.91666667  0.92028986  0.91272727  0.86131387]
平均分类准确率:
0.864729588973
```

图 6　十折交叉法验证线性核函数的分类准确率

从图 6 结果可看出,选用线性核函数后的平均分类精度达到 86.47%,说明语料是线性可分的,适合用线性核函数训练得到分类模型。

图 7 显示了不同惩罚因子下,十折交叉法的平均分类精度(横坐标为惩罚因子数,纵坐标为其对应的分类精度)。

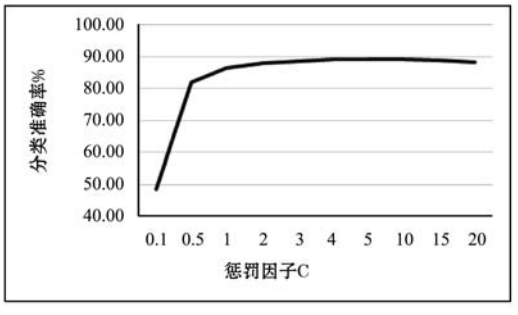

图 7　不同惩罚因子下的分类精度

图 7 表明,当惩罚因子为 0.1、0.5、1、2、3、4、5、10、15、20 这 10 个不同数值时,平均分类精度先增大后趋近相同。在设定惩罚因子时需注意:该数值设置太大容易导致过拟合。故本实验中,惩罚因子数值为 3。

6　实验结果

6.1　实验数据

实验数据来源于语料库中的 1 500 多道 C ++ 单选题。试题在各章节的数量分布如表 2 所示。

表 2　试题分布情况

章节	数量	章节	数量
绪论	74	C ++ 简单程序设计	295
函数	182	类与对象	239
数据的共享与保护	105	数组、指针与字符串	190
继承与派生	109	多态性	135
群体类和群体数据的组织	94	泛型程序设计与 C ++ 标准模板库	14
流类库与输入输出	78	异常处理	14

6.2　TextRank 窗口长度值

实验时,TextRank 算法中窗口长度 W 的数值会对结果产生影响。窗口长度用来确定词与词之间是否存在联系,即节点与节点能否连接形成一条边。设置好 W 后,从一个词连接到下一个词的距离必须限定在 W 之内。图 8 显示设置不同 W 后,十折交叉法验证下的平均分类准确率。

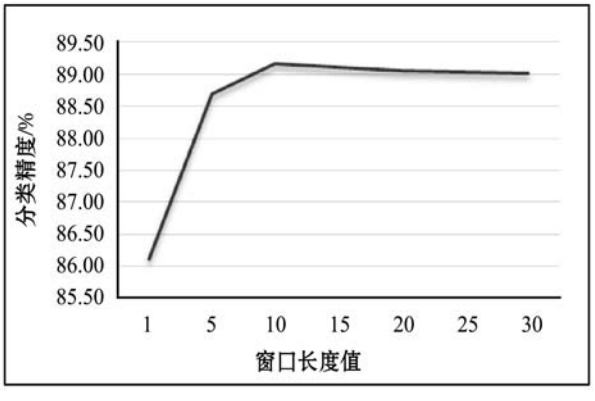

图 8　不同窗口长度下的平均分类准确率

从图 8 可看出,当窗口长度在 5 以内时,分类准确率会随窗口长度的增大而较大幅度升高;在 5 至 10 以内时,分类准确率小幅度上升;当窗口长度继续增大时,分类准确率会有所下降,这是因为若窗口长度太

大,语料的图连接会将语料中大部分词语互相联系起来,使相互的权重投票结果太平均,从而不能明确表示出关键词的权重值。本实验中,窗口长度取值为5。

6.3 分类器性能评价

本研究使用准确率、召回率和 F1 值对基于 SVM 的学科试题分类模型进行质量评价。其中,准确率是指一个文档被分类器分类到某一类别而且这个分类是正确的概率。召回率是指一个文档应该属于某一类别而分类器也确实将其分到该类别的概率。准确率和召回率反映了分类质量的两个不同方面,F1 值是对分类器性能的综合评价指标,计算公式如下:

$$准确率 = \frac{a}{a+b} \tag{2}$$

$$召回率 = \frac{a}{a+c} \tag{3}$$

$$F1 = \frac{准确率 \times 召回率 \times 2}{准确率 + 召回率} \tag{4}$$

式中:a 表示人工分类中属于该类的文档被分类器分到该类中的文档数目。b 表示人工分类中不属于该类的文档却被分类器分到该类中的文档数目。c 表示原本属于该类却被分到了其他类中的文档数。

本研究对比了 TF-IDF 和 TextRank 这两种不同的权重赋值算法下的试题分类评价结果,分析结果得出使用 TextRank 算法对语料赋权值后的分类效果更佳,具体分析如下。

图 9 对比了 TextRank 和 TF-IDF 这两种不同的权重赋值算法下的分类准确率。

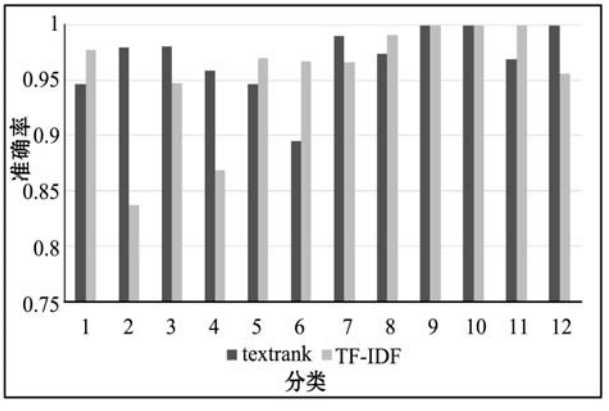

图 9 分类准确率对比图

从图 9 可看出,用 TextRank 算法对语料赋权值后的分类准确率在 2、3、4、7、12 这 5 类上的结果优于 TF-IDF;对于类别 9 和 10,使用两种算法对语料赋权值后的分类准确率相同;而对于 1、5、6、8、11 这 5 类,采用 TF-IDF 算法对语料赋权值后的分类准确率高于 TextRank 算法。通过计算可得,采用 TextRank 算法赋权值后的平均分类准确率为 97.02%,采用 TF-IDF 算法赋权值后的平均分类准确率为 95.70%。从而得出:在准确率性能上,用 TextRank 算法赋权值后的分类结果优于 TF-IDF。

图 10 对比了 TextRank 和 TF-IDF 这两种不同的权重赋值算法下的分类召回率。

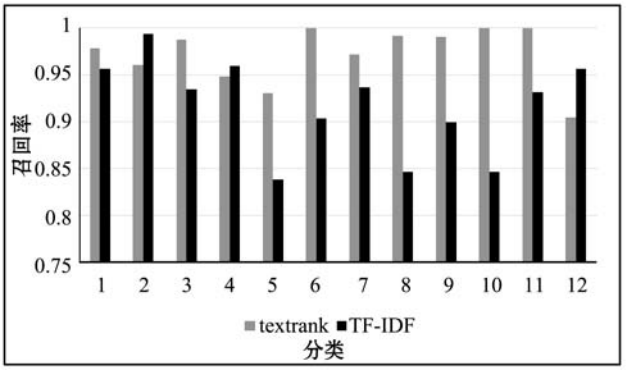

图 10 分类召回率对比图

从图 10 可看出,用 TextRank 算法对语料赋权值后的分类召回率在 1、3、5、6、7、8、9、10、11 这 9 类上的结果优于 TF-IDF;而对于 2、4、12 这 3 类,采用 TF-IDF 算法对语料赋权值后的分类召回率高于 TextRank 算法。通过计算可得,采用 TextRank 算法赋权值后的平均分类召回率为 97.18%,采用 TF-IDF 算法赋权值后的平均分类召回率为 91.67%。由此可得:在召回率性能上,用 TextRank 算法赋权值后的分类结果优于 TF-IDF 算法。

图 11 对比了 TextRank 和 TF-IDF 这两种不同的权重赋值算法下的 F1 值。

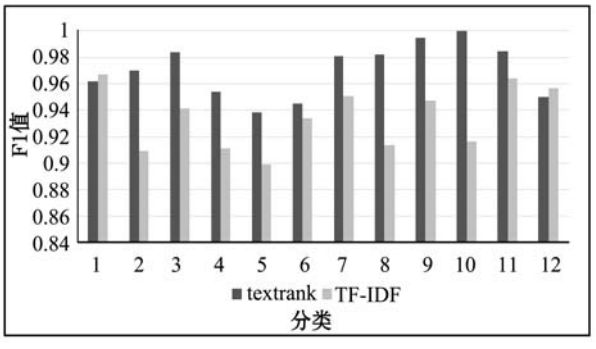

图 11 F1 值对比图

从图 11 可看出,用 TextRank 算法对语料赋权值后的分类 F1 值在 2、3、4、5、6、7、8、9、10、11 这 10 类上的结果优于 TF-IDF;而对于 1 和 12 这两类,采用 TF-IDF 算法对语料赋权值后的 F1 值高于 TextRank 算法。通过计算可得,采用 TextRank 算法赋权值后的平均分类 F1 值为97.05%,采用 TF-IDF 算法赋权值后的平均分类 F1 值为93.42%。由此可得:在 F1 值性能上,用

TextRank 算法赋权值后的分类 F1 值高于 TF-IDF 算法。

实验结果表明,用 TextRank 算法对语料赋权值后的分类结果优于用 TF-IDF 算法对语料赋权值后的实验结果。并且由于该方法下的分类准确率、召回率、F1 值均达到97% 以上,说明本研究提出的基于 SVM 的学科试题分类方法是可行的。

7　结　语

研究表明,基于 SVM 的学科试题分类法是有效的,该方法可对学科试题进行较为准确地分类。在本研究中,将学科试题按照课本章节分为 12 类,因为类别是按照"章"划分的,故目前根据分类结果只能确定试题知识点的大概范围。下一步将尝试对试题进一步分类,得到更加具体的知识点。

参　考　文　献

[1] 古丽娜孜·艾力木江,乎西旦·居马洪,孙铁利,等. 基于支持向量的最近邻文本分类方法[J]. 智能系统学报,2018, 13(5):799-807.

[2] 贺鸣,孙建军,成颖. 基于朴素贝叶斯的文本分类研究综述[J]. 情报科学,2016,34(7):147-154.

[3] 黄正伟,唐芳艳. 基于 SVM 分类模型的垃圾文本识别研究[J]. 数学的实践与认识,2016,46(7):144-153.

[4] 崔建明,刘建明,廖周宇. 基于 SVM 算法的文本分类技术研究[J]. 计算机仿真,2013,30(2):299-302,368.

[5] 阿力木江·艾沙,吐尔根·依布拉音,库尔班·吾布力,等. 基于 SVM 的维吾尔文文本分类研究[J]. 计算机工程与科学,2012,34(12):150-154.

[6] 朱刘影,杨思春. 基于 SVM 的地理试题自动分类[J/OL]. 计算机应用研究,2018,35(9). [2017-08-28]. http://www.arocmag.com/article/02-2018-09-025.html.

[7] 夏晔,钱松荣. SVM 算法在网站分类中的应用研究[J]. 计算机应用与软件,2012,29(11):222-224.

[8] 周庆平,谭长庚,王宏君,等. 基于聚类改进的 KNN 文本分类算法[J]. 计算机应用研究,2016,33(11):3374-3377,3382.

[9] 杨帅华,张清华. 基粗糙集近似集的 KNN 文本分类算法研究[J]. 小型微型计算机系统,2017,38(10):2192-2196.

[10] 赵燕,李晓辉,周云成,等. 基于朴素贝叶斯的农业文本分类方法研究[J]. 节水灌溉,2018(2):98-102.

[11] 高元,刘柏嵩. 基于集成学习的标题分类算法研究[J]. 计算机应用研究,2017,34(4):1004-1007.

[12] 马丽菲,莫倩,杜辉. 面向中文短影评的分类技术研究[J]. 山东大学学报(理学版),2016,51(1):52-57.

[13] 李航,唐超兰,杨贤,等. 融合多特征的 TextRank 关键词抽取方法[J]. 情报杂志,2017,36(8):183-187.

[14] 杨林青,余瀚,费宁,等. 一种基于 TextRank 的单文本关键词提取算法[J]. 计算机应用研究,2018,35(3):705-710.

[15] 郑莉,董渊,何江舟. C++语言程序设计[M]. 4 版. 北京:清华大学出版社,2010.

[16] 李稚楹,杨武,谢治军. PageRank 算法研究综述[J]. 计算机科学,2011,38(10):185-188.

(上接第 196 页)

参　考　文　献

[1] 亓瑞环. 浅析汽车控制的研究现状与展望[J]. 时代汽车,2017(8):40-41.

[2] 张情. 汽车电子前沿技术及未来发展趋势[J]. 时代汽车,2017(12):46-47.

[3] 闫利利. 汽车电控单元刷新的标准化研究[C]//第十二届中国标准化论坛文集,2015:1720-1724.

[4] 汪庆武,郇钲,黎泽清,等. 基于 SPI 接口 DSP 程序引导加载方法实际与实现[J]. 仪表技术,2016(5):6-8.

[5] 刘坤,韩朝智. 浅析基于 ARM 嵌入式开发的 BootLoader 设计及其实现[J]. 电子技术与软件工程,2016(2):203-204.

[6] 陈姿霖,宋磊锋,张龙岗,等. 基于 UDS 的整车诊断系统设计方法[J]. 汽车电器,2017(4):14-17.

[7] 罗峰,郁静华. 汽车统一诊断服务诊断协议网络层测试方法[J]. 同济大学学报(自然科学版),2016,44(4):632-636.

[8] 游长能. 基于 LabVIEW 的 CAN 总线 UDS 诊断工具开发[J]. 电子测试,2016(19):59-60.

[9] 汪春华,白稳峰,刘胤博,等. 基于 CAN 总线 UDS 服务 BootLoader 应用开发[J]. 电子测量技术,2017,40(2):166-170.

[10] ISO 14229:2013, Road vehicles-Unified diagnostic services (UDS)-Specification and requirements[S].

[11] 王琦,黄悦鹏,邢正阳,等. 基于 CAN 总线的 Bootloader 设计与实现[J]. 微型机与应用,2015,34(18):14-16.

[12] 刘一平. 基于 HIS 协议的车载 Bootloader 的研究与实现[D]. 成都:电子科技大学,2015.

[13] 曾其林,肖大伟,王志民,等. 基于 CAN Bootloader 的整车控制器程序更新系统设计[J]. 东方电气评论,2016,30(4):20-23.

[14] 陈彤,黄立梅. 一种用于汽车电控单元 CAN Bootloader 的设计与实现[J]. 汽车实用技术,2016(9):156-160.