****

**本 科 毕 业 设 计（论文）**

题 目：基于预分类的高效SVM网页分类器

|  |  |
| --- | --- |
| 学生姓名： | 周 俊 辉 |
| 学 号： | 1707020330 |
| 专业班级： | 软件工程17-3班 |
| 指导教师： | 李 村 合 |

摘 要

当今社会，生活的各个方面都离不开互联网的支持。网页分类技术的出现极大地促进了网络的发展，方便了人们的生活。在当今的众多的网页分类算法中，SVM算法由于其分类思想简单和分类效果较好，成为一种比较优秀的网页分类算法。

本次设计使用Java语言来实现网页分类器，根据软件工程开发的方法，对系统依次进行需求分析，概要设计，详细设计，编码，测试，最终开发出一个完备的系统。此系统主要的功能模块包括网络源码的提取，过滤，中文分词，词频统计，特征选择，生成特征向量。事先对语料库的文本的特征向量数据进行训练，得到的SVM模型可持久化的对网页进行分类。通过此模型对测试集进行分类得到了不错的效果。

**关键词**：网页分类；SVM算法；中文分词；特征选择

**Abstract**

In today's society, all aspects of life are inseparable from the support of the Internet. The emergence of web page classification technology greatly promotes the development of the network and facilitates people's life. In today's numerous web page classification algorithms, SVM algorithm has become an excellent web page classification algorithm because of its simple classification idea and good classification effect.

This design uses Java language to implement web page classifier. According to the method of software engineering development, the system needs analysis, outline design, detailed design, coding, testing, and finally develops a complete system. The main function modules of this system include extraction of network source code, filtering, Chinese word segmentation, word frequency statistics, feature selection, generating feature vector. After training the feature vector data of the corpus in advance, the SVM model can classify the web pages persistently. This model is used to classify the test set, and good results are obtained.

**Keywords**: Web page classification, SVM algorithm, Chinese word segmentation,

Feature selection

目 录

[第1章 序言 1](#_Toc72165834)

[1.1 选题目的及研究意义 1](#_Toc72165835)

[1.2 国内外研究现状 1](#_Toc72165836)

[1.3 论文组织工作 2](#_Toc72165837)

[第2章 相关知识介绍 3](#_Toc72165838)

[2.1 网页分类简介 3](#_Toc72165839)

[2.2 Block Level PageRank算法 3](#_Toc72165840)

[2.2.1 Block Level Web Graph 3](#_Toc72165841)

[2.2.2 Page Graph 3](#_Toc72165842)

[2.3 基于视觉的网页分块算法 3](#_Toc72165843)

[第3章 搜索引擎Quark模块 5](#_Toc72165844)

[3.1 模块结构 5](#_Toc72165845)

[3.2 网页分块算法 6](#_Toc72165846)

[3.3 网页主题内容提取 8](#_Toc72165847)

[3.4 算法效果演示 8](#_Toc72165848)

[第4章 SEWM中文Web信息检索评测 10](#_Toc72165849)

[4.1 评测任务介绍 10](#_Toc72165850)

[4.2 评测格式 10](#_Toc72165851)

[4.3 评测结果 10](#_Toc72165852)

[4.3.1 主题型网页发现任务评测结果 10](#_Toc72165853)

[4.4 评测综述 11](#_Toc72165854)

[第5章 网页分块的分布式应用 12](#_Toc72165855)

[第6章 总结与展望 13](#_Toc72165856)

[6.1 总结 13](#_Toc72165857)

[6.2 展望 13](#_Toc72165858)

[致 谢 14](#_Toc72165859)

[参考文献 15](#_Toc72165860)

[附 录 16](#_Toc72165861)

[附录A名词术语及缩略词 16](#_Toc72165862)

# 第1章 序言

## 1.1 选题目的及研究意义

随着当今互联网的迅猛发展，各种网页数据信息呈现出海量的增长趋势。互联网已经越来越成为我们平时生活和发展的不可缺少的一部分。而在当今如此多的网页数据中筛选出我们需要的信息就变得尤为的重要。为了能够处理这些海量的互联网的信息。网页的自动分类的需求就变得尤为的迫切。网页分类技术是一种可以对网页进行快速分类的技术，目前网页分类技术在我们生活中的诸多领域都发挥了重要的作用，比如分类搜索，信息过滤，数字图书馆等。

目前，已经出现了多种基于统计理论和机器学习方法的文本自动分类算法[1-2]，在多种多样的网页分类技术中，由于支持向量机(Support Vector Machine,简称SVM)过于出色的学习能力，成为网页分类技术中一种可以采用的解决方案。SVM是一类可以按照监督学习的方式对数据进行二元分类的广义线性分类器。它的决策边界是对学习样本求解的最大边距超平面[3-4]。虽然SVM的应用领域如此多，但是其也是存在着一些缺陷，SVM对于大规模训练样本难以实施，对于多分类问题的处理也有缺陷，传统的支持向量机只包含二类分类的算法。尽管如此，SVM依然是网页分类领域的一种颇为良好的方法。

## 1.2 国内外研究现状

国外对于文本的自动分类技术研究开始的比较早，早在上个世纪50年代末期，H.P Luhn 就在这一区域进行了开创性的研究，并且提出了词频统计思想用于自动分类[5]。支持向量机是从模式识别中的广义肖像算法发展而来的分类器，最早由前苏联的学者Vladimir N. Vapnik和Alexander Y. Lerner发表在1963年的研究中[6]。

随着各种规划问题求解技术的出现，SVM渐渐被理论化成为统计学习理论的一部分。Bernhard E. Boser、Isabelle M. Guyon和Vapnik又于1992年通过核方法得到了非线性的支持向量机[7]。之后Corinna Cortes和Vapnik于1995年提出了软边距的非线性SVM并且应用于手写字符识别的问题[8]。

国内对于文本的自动分类技术研究起步比较晚。始于上个世纪80年代，所使用的文本分类的方法也相对来说比较的单一，而且其对中文文本的分类从90年代才开始涉及，基本上算是在之前英文文本分类的基础上做一些改进。文本分类最初被运用在文献的分类上，运用了“知识工程”的理论指导分类技术，最具有代表性的是基于专家的文本分类的系统，通过人工定义一些规则，然后根据这些已经定义好的规则，在给定的特殊情况下对文本进行分类。再之后，文本分类逐渐向着基于学习的研究阶段发展，这个时期的代表是基于词典统计的分类系统。目前，我国的文本自动分类技术已经产出了很多成果，并且得到了实际的应用。而这其中基于SVM的文本分类更是取得了不错的成果。

## 1.3 论文组织工作

# 第2章 相关知识介绍

## 2.1 网页分类简介

网页分类技术出现之前，人们已经具有了很多文档自动分类的方法。随着Web的普及和发展，自动分类技术处理的对象就逐渐从文档转向网页。而网页分类技术的多样性也使得分类技术多种多样，在本次研究中选取了SVM来实现网页的分类。

在对文本分类之前需要对语料库文本进行预处理，也就是说要给所分类的文本一个标记，使得在之后的文本分类的过程中加工成的特征更为明确。先对大量的文本进行人工的分类，然后对训练的文本分类构成分类器，以期在后来的文本分类过程中可以利用分类器把这些文本分类到所给定的几个类别之中。从数学上来看，我们进行分类的一个实质也就是要准确地反映文献与属性的关系[9]。分类器系统的结构应分为如图2-1的预处理，特征提取，训练和分类三个大块。

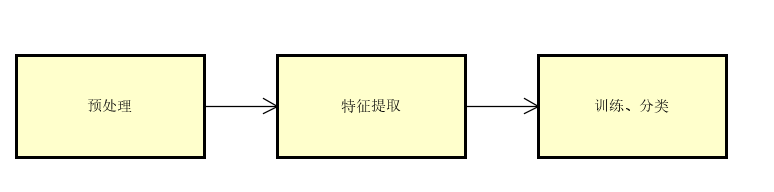


图2-1 分类器系统结构图

在预处理的环节中，主要包括网络源码的提取，过滤，中文分词，词频统计。

其中网络源码的提取采用的HttpClient，根据输入的网址自动解析获得网页的源码。

过滤部分通过对html标签的解析，通过正则表达式过滤掉诸如<script>和<style>等标签，然后根据中文Unicode编码的范围过滤掉无关字符，并且按照一定的格式输出。

中文分词部分将过滤获得的文本内容分割成一个个词语，这里我使用的分词算法是HanLP算法。由于分词中有一些词语不具有代表性，比如“的”，“百度”，“搜狐”，所以在进行中文分词的同时，建立了一个停用词库，将此类无关性的词语进行剔除。同时将保留下来的词语的词频进行统计，一来是查看此网页大概的一个分类情况，二来是为后来计算特征权重提供了方便。

在特征提取环节中，对所提取到的词语进行一个特征权重的计算。这里我才用的是TF-IDF，这是一种用于信息检索与数据挖掘的常用加权技术，其中，TF是词频(Term Frequency)，IDF是逆文本频率指数(Inverse Document Frequency)。下面来介绍一下这种统计方法。

通常来说一个词如果在一个文件集中出现的频率越高，就代表着这个词就越重要，也即一个字词的重要性会随着它在文件中出现的次数成正比增加。但是与此同时如果这个词在所有的文件集中都出现了很多次，说明这个特征词用来表示一个分类的作用就越小，也就是特征词会随着它在语料库中出现的频率成反比下降。

TF-IDF的主要思想是：当某一个词语在一个文件集中出现的频率高，同时其在别的文件集中很少出现，那么就可以认为这个词语具有很好的类别区分能力，适合用来分类。TF的计算公式如公式(2-1)所示：

tfi,j = (2-1)

其中，分子表示这个词在这个文件中出现的次数，分母表示这个文件中文件中所有字词的出现次数之和。IDF的计算公式如公式(2-2)所示：

idfi = lg (2-2)

其中，分子为语料库中的文件总数，分母表示包含指定词语的文件数目，这个时候如果该词语不在语料库中，就会导致分母为零，所以这个时候我们一般会做特殊处理，让分母自动加一，避免出现分母为零的情况。最后我们计算TF和IDF的乘积。使用TF-IDF有利于过滤掉比较常见的不具有代表性的词语而保留比较能区分出类别重要的词语。

在训练、分类环节中，使用的语料库文本来源于搜狗文本分类语料库，根据划分好的分类进行训练。这里使用开方检验的方法选择出具有标志性的特征，因为不同类别的文件所具有的标志性的词语是不同的，所以初步根据各个类别所具有的的特征词也能够分辨出大致的文本分类情况。经过训练的模型最后通过序列化的方法保存在本地，以后每次判断分类均可避免训练过程，而调用已经持久化保存的模型进行分类的预测。

## 2.2 SVM算法

SVM是一种基于统计学习法的独立二元分类系统。独立二元的意思是对于在我们分出的类别中，分类系统会针对每一个类进行判断，判断待测文档是否属于该类，而结果只有两种：要么属于该类，要么不属于该类。SVM不仅追寻训练的准确性，而且考虑到了学习的空间的复杂性，在准确性和复杂性之中采取了一种折中的方式，对于样本的分类具有良好的效果。

### 2.2.1 SVM算法的技术原理

SVM在几何上的意义是在线性可分的前提下，在将要进行分类的问题中给出输入和目标X = {X1, … ,XN}，y = {y1, … ,yN}，在输入的数据中各个样本都包含经过筛选后的多个特征，并且构成了一个特征空间：Xi = [x1, … ,xn]∈χ，而目标为二元变量y∈{-1,1}表示负类和正类。若输入的数据所在的特征空间存在一个超平面可以将目标按正类和负类分开，并且使任意的样本点到平面距离大于等于1[4]；即满足公式(2-3):

(2-3)

则称这个分类问题线性可分，其中ω为这个超平面的法向量，b为截距。

事实上满足该条件的决策边界是形成了了两个互相平行的超平面作为可能存在的最小间隔边界来对样本的分类进行判断，如公式(2-4)：

(2-4)

我们称在上半部分的样本为正类，同理下半部分的样本我们称之为负类，如图2-2所示。

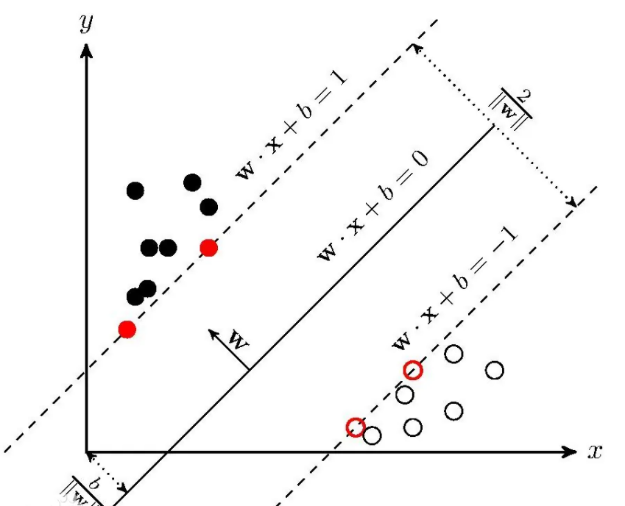


图2-2 线性支持向量机

而在线性不可分的情况下，SVM采用特征映射方法，引入核函数来使得K(*xi*，*xj*) = Φ(*xi*) Φ(*xj*)，实现非线性变化后的线性分类[10]，如图2-3所示。

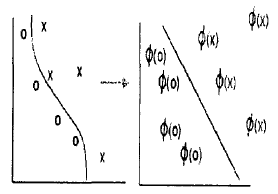


图2-3 非线性支持向量机

### 2.2.2 SVM算法的技术路线

(1) 支持向量机的训练算法

事实上训练一个SVM实际上就相当于求解一个如公式(2-5)的凸二次规划(简称QP)问题[11]。

 (2-5)

在这个公式里，为矩阵，是一个核函数。该问题是一个凸最优化问题，最优点应满足的充要条件是KKT条件[12]，即所有的训练样本都应满足的条件如公式(2-6)。



 (2-6)



其中C为惩罚因子，。

对于小规模的QP问题，通过常用的最优化算法，类似于牛顿法、拟牛顿法等均能够较好地进行求解。但是当训练集很大的时候，尤其是支撑向量个数也比较庞大时，大多数算法需要的内存将会因为与Q矩阵的大小成比例而达到很庞大的程度，这个时候解决QP问题的常用的比较经典的方法就不再可行了[13]。也因为这个原因，研究人员又提出了许多用来解决大规模的训练集的SVM训练算法。

(2) SMO算法

SMO(Sequential Minimal Optimization)即序列最小优化算法是一种将问题分解成为一系列的最小规模的QP 问题的算法，这个算法是由Microsoft Research的John C. Platt在1998年提出来的，并逐渐成为最快的二次规划优化算法，在线性SVM和数据稀疏的情况下性能更优。问题进行求解时，每次迭代只处理包含两个拉格朗日乘子的QP 问题，所以，SMO算法首先就需要找到这两个待优化的Lagrange乘子[14]。具体实际的做法就是：首先找到违反KKT 条件的样本点所对应的拉格朗日乘子，将其作为α1，第二个拉格朗日乘子根据最大化优化步长来定，Platt 以|E1-E2|来近似表示优化步长，即选取满足 max|E1-E2| 的样本点对应的拉格朗日乘子作为α2。

(3) 支持向量机增量学习算法

经典的SVM训练算法是不支持增量学习的，每当有新的样本加入的时候，就要对对样本进行重复性的重新的训练，长此以往下去，浪费了大量的时间。试想一下，倘若可以保存原来训练所得的知识又能够在这其中加入新的样本一起训练，这样的话，就能够保留之前已经学习获得的知识，又可以避免因为新的样本的加入而导致训练时间的激增。并且，我们应当意识到，在实际生活中，我们对于样本的采集不可能一蹴而就，很多情况下都是经过不断的积累才收集到的，所以时常需要加入新的样本。其实SVM的最优分类面仅与支持向量有关。所以，对SVM算法中增量学习的研究具有很重要的理论意义和实用价值。

## 2.3 本章小结

本章主要介绍了本次毕业设计中涉及到的一些知识。例如，网页分类的相关流程：网页源码的获取、过滤、中文分词、词频统计、特征提取、训练和分类，和SVM算法的技术原理和技术路线的一些基本介绍。

# 第3章 需求分析

## 3.1 用户需求

进入21世纪以后，互联网飞速发展，各种各样网页的出现对于网页分类技术提出了不小的要求。面对如此海量的网页数量和如此多的网页种类，精确的对网页进行合理的分类就显得尤为重要。试想如果没有一种合理的网页分类技术，那么用户在众多网页分类中寻找想要的类型就会显得比较困难。

网页分类的基础是文本分类，对于中文文本分类的研究在很早之前就开始出现，本次的毕业设计就是基于SVM算法开发的一个网页分类系统，用户登录分类系统后，只需要输入想要得知分类的网页地址就可以获得该网页的经系统分析的分类结果。同时，为了满足用户多样化的需求，更多的展示分类过程中的细节，提供了包括中文分词，提取词频前N位，特征权重显示，以及经过SVM算法预测分类可能性的归一化数值。

### 3.1.1 用户界面

本次设计的用户界面采用的是网页前端的方式，主要考虑到的是部分用户可能并不熟悉使用Java以及使用打开Java所需要的开发工具。而采用网页的形式，用户只需要输入本次设计的网页地址，就可以进行一系列网页分分类操作。本次设计对网页的样式做了良好的设计，前端主要采用了bootstrap来对组件进行一个优化，按钮布局也符合人们浏览的习惯。左上角的标题可以清晰明了的展示这是一个网页分类系统，色彩和文字属性也采取了一定的设计。

### 3.1.2 用户操作流程

第一步：在请输入网页url右侧输入框输入待分类的网页url。

第二步：点击获取源码按钮，在左侧文本框内会显示出经HttpClient爬取的网页源码。

第三步：点击清洗数据按钮，后台会提取过滤掉如<script>和<style>等标签，并且将中文字符返回到左侧的文本框显示。

第四步：点击中文分词按钮，左侧文本框会显示分词的结果和词频。

第五步：点击词频前N按钮，并且指定N的数值，左侧文本框会显示所指定的前N位词语。

第六步：点击特征权重按钮，左侧文本框会显示经计算后所有特征词的权重值。

第六步：点击分类结果按钮，左侧文本框会显示经SVM模型预测的网页分类结果，并且显示出各个类型的可能性数值。

## 3.2 预期目标

本系统在流程：获取源码→清洗数据→中文分词→词频前N→特征权重→分类结果中均可以显示出经系统处理返回的文本，并且在最后的分类结果步骤，应有较高的分类的准确性。最终通过测试集计算系统的准确率，召回率，F1值应该达到一定的水平。

## 3.3开发工具及运行环境

### 3.3.1 开发工具

(1) IntelliJ IDEA

(2) WebStorm

(3) Postman

### 3.3.2 运行环境

(1) JDK1.8

(2) Win10操作系统

(3) Chrome浏览器

## 3.4 本章小结

本章主要介绍了本设计的用户需求、实际过程中的操作流程和预期目标，并且给出了本设计过程中使用到的开发工具和运行环境。

# 第4章 概要设计

## 4.1 整体设计

本次毕设项目是一个基于SVM的网页分类器。对于一个网页分类器大体上应该包括两个方面。分别是训练过程如图4-1和分类过程如图4-2。

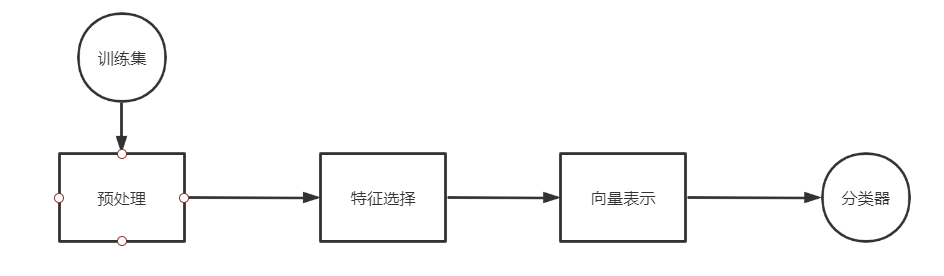


图4-1 训练过程

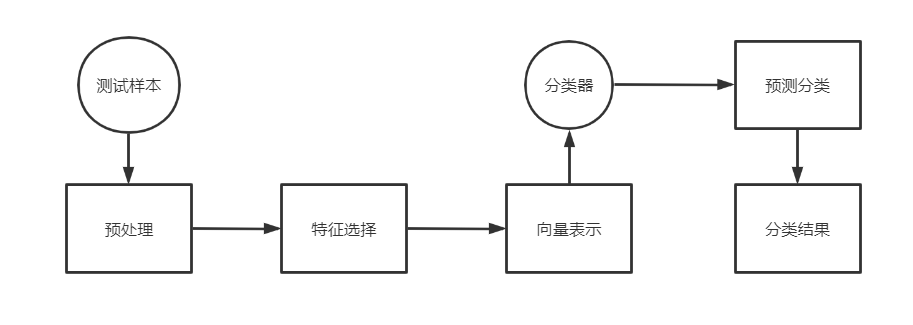


图4-2 分类过程

其中训练集来源于搜狗文本分类语料库，并且已经按照分类将不同的类别的训练集分好类了，训练过程就是根据SVM算法对训练集文本训练最终得到SVM预测模型的一个过程。最终这个得到的SVM模型可以用来对我们所提取的网页进行网页分类的预测。

其中在分类过程中的预处理又可以细分为如图4-3的流程。

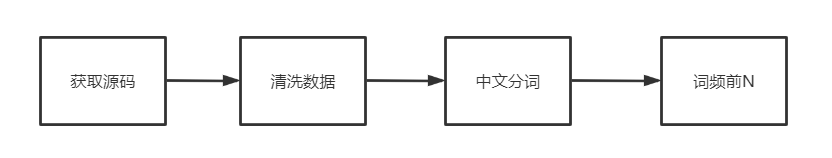


图4-3 预处理

## 4.2 模块设计

### 4.2.1 模块划分

整个网页分类系统主要划分为如下七个模块。

1. 前端界面模块
2. 获取源码模块
3. 清洗数据模块
4. 中文分词模块
5. 词频前N模块
6. 特证权重模块
7. 分类结果模块

### 4.2.2 模块介绍

(1) 前端界面模块

用户前端模块是用来展示分类过程的可视化模块。主要采用的是HTML页面加上BootStrap前端框架来完成的。前后端数据传递通过jQuery来完成。通过用户输入的待分类网页url，返回后续步骤所得的文本内容。

(2) 获取源码模块

当用户输入完网页的url后，jQuery通过获取源码按钮所绑定的后端RequestMapping地址，并且将此url作为参数传递给其中的方法。获取源码这部分是通过HttpClient来进行源码的提取的，当输入的网页url正确时会返回对应的源码到前端，如果输错url，后端会报错，并且返回“返回的状态码不是200”提示。

(3) 清洗数据模块

当输入的待分类的url解析出源码后除了返回到前端进行显示，还保存到了本地文本文件进行保存，第二步的清洗数据模块就是将保存的源码文本先根据正则表达式剔除掉诸如<script>和<style>标签，然后根据中文的Unicode编码范围，按照一定的格式提取出中文字符，然后返回到前端界面展示。

(4) 中文分词模块

在清洗数据模块中得到了网页的中文文本信息，这个时候就需要对这个中文文本进行分词了，方便后续计算特征。网上已有的中文分词的算法非常多，诸如jieba分词，word分词，HanLP分词，ansj分词等，不同分词的区别差异比较小，我这里选择了HanLP中文分词，并且因为在分词过程中诸如“的”，“地”之类的单个字词的特殊性，选择在统计词频的时候将少于两位的字词剔除掉。又因为如“百度”，“新浪”等很多词语不具有任何区别意义，故建立了一个停用词表。在进行中文分词的同时和停用词表中的词语进行比较，如果存在，那么从得到的分词中剔除。

(5) 词频前N模块

因为进行了中文分词处理，所以对于每个词出现的次数已经做了记录，在这一步中只需要按照词出现的次数进行一个排序即可实现该模块的功能。在这里做了一个优化就是提取词频前多少用户可以进行一个自定义的操作。

(6) 特征权重模块

因为后续进行SVM模型的预测分类需要特征向量作为参数，所以这一模块可以说是很重要的一个模块，这里因为SVM分类器的实现基于liblinear软件包，需要的参数是特征id和特征的权重，所以在这一步也是将特征词转化为一个唯一的特征id表示，特征权重的计算采用的是TF-IDF算法。计算完成后将特征词和对应的特征权重返回到前端显示。

(7) 分类结果模块

计算出特征权重之后，将特征节点作为参数传到liblinear软件包的训练和预测方法中进行预测分类的操作，最终会得到一个经过归一化后的double类型数组，用来存储各个分类可能性的数值，最终数值最大的那个分类就是本次SVM分类器预测分类的结果，将分类结果传递到前端界面显示，这样本次分类就结束了。

## 4.3 本章小结

本章主要介绍了该SVM网页分类器整体的体系结构以及体系结构中各个模块所采取的设计方法和所实现的功能。

# 第5章 详细设计

## 5.1 用户界面

本次设计前端页面采用的原始的HTML语言，在样式方面，采用了BootStrap的前端框架。通过jQuery来传递前端的数据到后端。因为使用到了部分图标和字体，故引入了图标库Font Awesome。前端提示输入和提醒目前可分类网页类型代码如代码5-1。

1. **<div** class="span9"**>**
2. **<form** id="Form" method="post" enctype="multipart/form-data" role="form"**>**
3. **<div** class="form-group"**>**
4. **<h3** class="page-title"**>**请输入网页url: **<input** id="url" name="url" type="text" style="width:300px;"**/></h3>**
5. **<div** class="row-fluid"**>**
6. **<div** class="block-heading"**>**暂时可支持分类**<a**
7. href="http://news.sohu.com/" target="\_blank"**>**网页**</a>**类型（体育、健康、军事、教育、旅游、汽车、财经）
8. **</div>**
9. **<textarea** id="show" name="show" rows="26" style="width: 1100px" class="txtshow"**></textarea>**
10. **</div>**
11. **</div>**
12. **</form>**
13. **</div>**

代码5-1 前端提示输入代码

前端各种不同功能按钮实现代码如代码5-2。

1. **<div** class="span3"**>**
2. **<div** class="sidebar-nav"**>**
3. **<button** class="nav-header" style="width: 345px; text-align: left; font-size: medium"
4. name="getSourceCode" formaction="/getSource/getSourceCode"**>**
5. **<i** class="fa fa-chevron-right"**></i>**
6. 获取源码
7. **</button>**
8. **<button** class="nav-header" style="width: 345px; text-align: left; font-size: medium" name="getClearData"
9. formaction="/getSource/getClearData"**>**
10. **<i** class="fa fa-chevron-right"**></i>**
11. 清洗数据
12. **</button>**
13. **<button** class="nav-header" style="width: 345px; text-align: left; font-size: medium" name="getWord"
14. formaction="/getWord/getWordFrequency"**>**
15. **<i** class="fa fa-chevron-right"**></i>**
16. 中文分词
17. **</button>**
18. **<button** class="nav-header" style="width: 345px; text-align: left; font-size: medium" name="getWord"
19. formaction="/getWord/getTopNumberWord"**>**
20. **<i** class="fa fa-chevron-right"**></i>**
21. 词频前
22. **</button>**
23. **<label>**
24. **<select** class="form-control" style="background-color: #eeeeee"**>**
25. **<option** name="number" value="20" selected**>**20**</option>**
26. **<option** name="number" value="30"**>**30**</option>**
27. **<option** name="number" value="40"**>**40**</option>**
28. **<option** name="number" value="50"**>**50**</option>**
29. **</select>**
30. **</label>**
31. **<button** class="nav-header" style="width: 345px; text-align: left; font-size: medium" name="getFeature"
32. formaction="/getFeature/getFeatureWeight"**>**
33. **<i** class="fa fa-chevron-right"**></i>**
34. 特征权重
35. **</button>**
36. **<button** class="nav-header" style="width: 345px; text-align: left; font-size: medium" name="getLinearSVM"
37. formaction="/getLinearSVM/getLinearSVMClassifier"**>**
38. **<i** class="fa fa-chevron-right"**></i>**
39. 分类结果
40. **</button>**
41. **</div>**
42. **</div>**

代码5-2 前端功能按钮显示代码

这其中可以注意到代码5-1和代码5-2的<div>对应的class有一个是span9另一个是span3，这是BootStrap的栅格系统将网页的一行分成12个栅格，其中显示模块占9格，按钮模块占3格。

前端传递到后端对应的RequestMapping使用jQuery来进行标签中内容的传递，主要的js代码如代码5-3所示。

1. $(**function** () {
2. **function** upload(data, url, number) {
3. $.ajax({
4. async: **false**,
5. method: 'post',
6. url: url,//传给后端@RequestMapping
7. data: {"url": data, "number": number},
8. datatype: 'json',//返回格式
9. success: **function** (data) {
10. $("#show").val(data);
11. }
12. })
13. }
14. $("button").bind('click', **function** () {
15. let url = $(**this**).attr('formaction');
16. **const** data = $("input").val();
17. **const** number = $("select").val();
18. upload(data, url, number)
19. })
20. })

代码5-3 前端参数传递代码

其中“data”是输入框中用户输入的请求分类的网址，“url”是用来处理不同的按钮对应的后端代码的路径，“number”是用户输入的提取词频的数目。“show”表示的是用来显示返回数据的textarea区域。

## 5.2 核心模块

本次设计的核心模块主要分为获取源码、清洗数据、中文分词、词频前N、特征权重、分类

### 5.2.1 获取源码

(1) 流程图

构建网页分类器的第一步自然是获取网页的源码，系统获取源码的流程如图5-1所示。

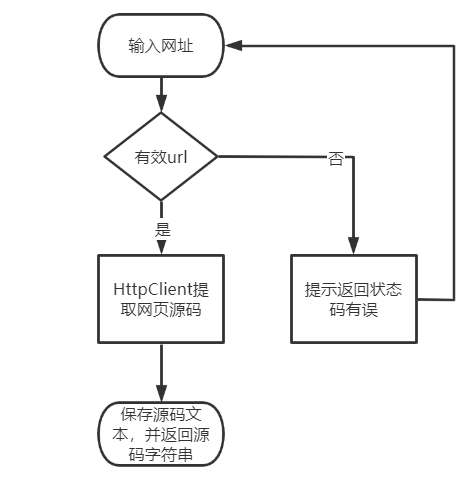


图5-1 获取源码流程图

(2) 核心代码展示

提取网页源码的主要代码如代码5-4所示。

1. //生成httpclient，相当于该打开一个浏览器
2. CloseableHttpClient httpClient = HttpClients.createDefault();
3. CloseableHttpResponse response = **null**;
4. //创建get请求，相当于在浏览器地址栏输入网址
5. HttpGet request = **new** HttpGet(url);
6. request.setHeader("User-Agent", "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36");
7. **try** {
8. //执行get请求，相当于在输入地址栏后敲回车键
9. response = httpClient.execute(request);
10. //判断响应状态为200，进行处理
11. **if** (response.getStatusLine().getStatusCode() == HttpStatus.SC\_OK) {
12. //5.获取响应内容
13. HttpEntity httpEntity = response.getEntity();
14. **return** EntityUtils.toString(httpEntity, "utf-8");
15. } **else** {
16. //如果返回状态不是200，比如404（页面不存在）等，返回打印信息
17. **return** "返回的状态码错误,请输入正确的网页url！";
18. }
19. } **catch** (IOException e) {
20. e.printStackTrace();
21. } **finally** {
22. //关闭
23. HttpClientUtils.closeQuietly(response);
24. HttpClientUtils.closeQuietly(httpClient);
25. }

代码5-4 获取源码核心代码

HttpClient通过模仿人类行为，携带一个请求头打开浏览器，并且抓取目标网页的源码包转化成字符串进行保存。在获取源码之后除了返回到前端异步显示外，也保存在本地文本文档中进行存储。

### 5.2.2 清洗数据

(1) 流程图

当获取了网页源码之后，将保存在本地的源码文本读出，进行去除无用标签和无用字母标点的操作，并将最后的中文文本按照一定格式进行保存和返回到前端，清洗数据的流程图如图5-2所示。

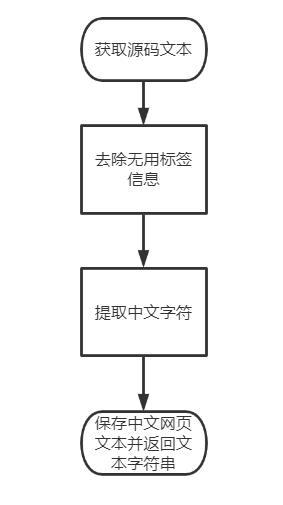


图5-2 清洗数据流程图

(2) 核心代码展示

清洗数据的主要代码如代码5-5所示。

1. HttpClearHtmlUtil hch = **new** HttpClearHtmlUtil();
2. String sourceCode = FileOperationUtil.readFile(SOURCE\_CODE\_PATH);
3. String html = hch.replaceHtml(sourceCode);
4. System.out.println(html);
5. **char**[] c = html.toCharArray();
6. **int** len = c.length;
7. **char**[] tmp = **new** **char**[len];
8. **int** k = 0;
9. //根据换行符，汉字，符号分行表示的文本解析之后的字符串
10. **for** (**int** i = 0; i < len - 1; i++) {
11. **if** (((c[i] >= 0x4E00 && c[i] <= 0x9FA5)) && ((c[i + 1] >= 0x4E00 && c[i + 1] <= 0x9FA5))) {
12. tmp[k++] = c[i];
13. } **else** **if** ((c[i] >= 0x4E00 && c[i] <= 0x9FA5)) {
14. tmp[k++] = c[i];
15. tmp[k++] = 0x000A;
16. }
17. }
18. **if** (c[len - 1] >= 0x4E00 && c[len - 1] <= 0x9FA5) {
19. tmp[k] = c[len - 1];
20. }
21. //网页正文无中文字符时返回
22. **if** (k == 0) {
23. **return** "当前解析网页正文无中文字符";
24. }
25. String clearData = String.valueOf(tmp).substring(0, k - 1);
26. FileOperationUtil.writeFile(CLEAR\_DATA\_PATH, clearData);
27. **return** clearData;

代码5-5 中文字符提取

将获取源码步骤中的源码字符串作为参数传递给replaceHtml方法，replaceHtml方法中是用正则表达式去除诸如<script>和<style>等无用标签的方法。去除无用标签后，使用中文Unicode字符的范围介于0x4E00和0x9FA5之间的方法过滤掉其他字符，仅保留中文字符，并按照在同一个标签里的中文字符自成一行的方式进行字符串的存储。存后的字符串按照相同的方式写到本地的文本中进行保存。

### 5.2.3 中文分词

(1) 流程图

提取到待分类网页的中文字符后下一个步骤就是对中文字符进行一个分词的操作了，因为我们的分词的结果将作为一个个的特征来供分类器进行分析，所以中文分词是的一个环节。中文分词的方法很多，本次设计使用的是HanLP分词进行的处理。中文分词的流程如图5-3所示。

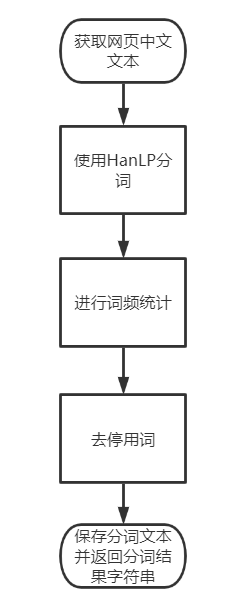


图5-3 中文分词流程图

(2) 核心代码展示

中文分词过程的第一步使用HanLP进行分词操作，直接调用提供的分词方法即可，词频的统计就是对出现词语的个数进行计数，但是分词过程中会产生一些无用的词语，对于词语长度小于1的直接剔除，对部分不具有辨识度的单词，进行一个停用词的处理。停用词表从网上收集。由于停用词的个数过多，时常可能需要更新，故将停用词使用文本进行保存，方便随时更新，去除停用词的核心代码如代码5-6所示。

1. **public** **void** removeStopWords(Map<String, Integer> map, Set<String> set) {
2. Set<String> keySet = map.keySet();
3. **for** (String term : keySet) {
4. **if** (set.contains(term)) {
5. map.remove(term);
6. }
7. }
8. }

代码5-6 去停用词

其中map存储的是词频，即词语和词语出现的次数，set集合是停用词，将map中的词语和set集合中的词语进行对比，如果set集合里包含，那么就从词频map集合中去除这一项。

### 5.2.4 词频前N

(1) 流程图

词频统计完成之后紧接着就是进行对词频进行一个排序，提取我们想要的前N位词频。词频前N模块的流程图如图5-4所示。

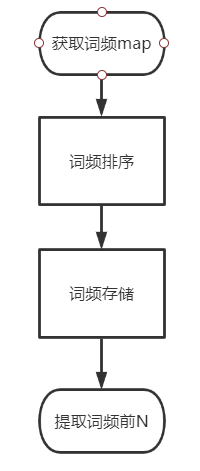


图5-4 词频前N流程图

(2) 核心代码展示

在中文分词模块中，已经统计好了词频，这一步也就是对词频进行一个排序的操作，为了避免之后用户改变需要提取词频的个数所造成的重复性计算，这里我采取了将所有词频一次性进行排序，并且按照从高到低一行一行进行保存，这样，当用户重复性修改需要的提取词频的个数时，只需要读取相应的行数即可。核心代码如代码5-7所示。

1. **public** **void** getTopNumberWord() {
2. Map.Entry<String, Integer> entry;
3. **int** size = wq\_map.size();
4. **for** (**int** i = 0; i < size; i++) {
5. entry = getMaxEntry(wq\_map);
6. list.add(entry);
7. }
8. **for** (**int** i = 0; i < size - 1; i++) {
9. top\_word.append(list.get(i)).append("\n");
10. }
11. top\_word.append(list.get(size - 1));
12. FileOperationUtil.writeFile(TOP\_WORD\_PATH, top\_word.toString());
13. }

代码5-7 提取词频前N

其中getMaxEntry方法是按照wq\_map集合中的value值进行一个排序提取的方法，排序后的词频保存在TOP\_WORD\_PATH指定的路径中。当用户指定需要提取词频前N的N值后，会返回前N行数据，代码如代码5-8所示。

1. **public** **static** **void** readTopNumberWord(String path, **int** number) **throws** IOException {
2. FileReader fr = **new** FileReader(path);
3. BufferedReader br = **new** BufferedReader(fr);
4. **int** i = 0;
5. String line;
6. **while** ((line = br.readLine()) != **null** && i < number) {
7. i++;
8. top\_word.append(line).append("\n");
9. }
10. br.close();
11. fr.close();
12. }

代码5-8 返回词频文本前N行

前端用户传入的number作为readTopNumberWord的一个参数，直接将排序好的前N行数据传递到前端进行输出。

### 5.2.5 特征权重

(1) 流程图

这一步是进行SVM分类预测最关键的一步，分类器无法直接识别文本，必须要对文本进行特征向量的表示。提取特征权重的流程图如图5-5所示。

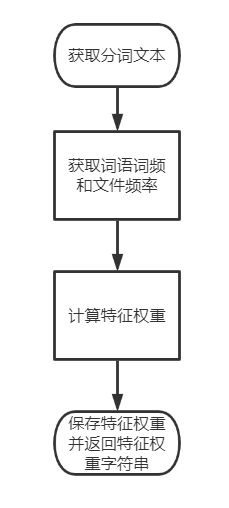


图5-5 提取特征权重流程图

(2) 核心代码展示

计算特征权重的的方法有很多种，这里采用的是TF-IDF方法，需要提取特征词的词频(term frequency)和文本频率(document frequency)。词频是之前步骤中已经得到的，文本频率是在训练集中包含该词语的文件的个数，通常可以认为如果包含一个词的文件个数越少，这个词就越有代表性。其中词频可以通过中文分词后获得，而文件频率则是通过对训练集文本采用卡方检验的方法进行特征的统计和计算。即提取特征的同时，建立数组记录该特征所存在的文件的个数。计算特征权重的核心代码如代码5-9所示。

1. **public** **double** weight(**int** feature, **int** frequency) {
2. **return** Math.log10(frequency + 1) \* Math.log10((**double**) **this**.numDocs / (**double**) **this**.df[feature] + 1.0D);
3. }

代码5-9 计算特征权重

其中为了避免文件频率为0导致分母为0无法计算，给分母自动做了加1，使用对数计算可以使运算的结果保持在一个均匀的范围内。

对训练集进行卡方检验统计特征和文件频率。卡方检验实际上就是就是统计实际情况下的测量值和理论情况下的推断值之间偏离的程度。其中，如果卡方值越大，表示两者之间差距越大，卡方值越小，差距也就越小。而当二者完全相等时，卡方值为0，表示实际结果与理论情况下完全相同。因为采用的语料库事先就已经经过分类，当我们使用卡方检验来统计特征也比较方便。使用卡方检验的具体方法如下。

先提出一个假设：假设词语“考研”与“教育”类不相关。统计出包含“考研”且属于“教育”类的文档数为A，包含“考研”但不属于“教育”的文档数为B，不包含“考研”属于“教育”类的文档数为C，不包含“考研”且不属于“教育”类的文档数为D。做出四格表如表格5-1所示。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 属于“教育” | 不属于“教育” | 总计 |
| 包含“考研” | A | B | A+B |
| 不包含“考研” | C | D | C+D |
| 总计 | A+C | B+D | N |

表格5-1 卡方检验四格表

计算公式如公式5-1所示。

χ2(考研，教育) = (5-1)

计算出卡方值后就可以按照卡方值排序选择特征了。

### 5.2.6 分类结果

(1) 流程图

根据之前步骤得到的特征节点和训练阶段得到的SVM模型，将其作为参数，调用liblinear软件包中的predictProbability方法进行分类结果的预测，此模块的的流程图如图5-6所示。

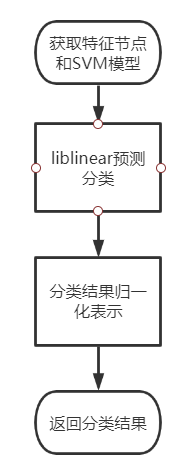


图5-6 获取分类流程图

(2) 核心代码展示

获取分类过程的核心代码如代码5-10和5-11所示。

1. **public** **double**[] categorize(Document document) {
2. FeatureNode[] x = buildFeatureVector(document, model.featureWeighter);
3. probs = **new** **double**[7];//保存7个分类试验结果
4. //进行SVM预测，得到向量归一化后的数组,传入的参数为svm的model,特征feature,和用来保存七个分类得分的数组
5. Linear.predictProbability(model.svmModel, x, probs);
6. //probs为归一化后的数组
7. **return** probs;
8. }

代码5-10 liblinear分类

1. // 对词向量进行归一化(L2标准化).得到分词后每个词节点的权重值
2. **double** normalization = 0;
3. **for** (**int** i = 0; i < termFrequency; i++) {
4. **double** weight = x[i].getValue();
5. normalization += weight \* weight;
6. }
7. normalization = Math.sqrt(normalization);
8. **for** (**int** i = 0; i < termFrequency; i++) {
9. **double** weight = x[i].getValue();
10. x[i].setValue(weight / normalization);
11. }

代码5-11 特征向量归一化

方法categorize是调用liblinear软件包进行分类的方法。根据输入的文档先构造文档向量，在之前计算特征权重过程中就已经获得了经过TF-IDF计算后的特征权重，需要对计算出的特征权重按照代码5-11进行归一化(L2标准化)处理，处理之后所得的特征权重作为predictProbability方法的一个参数进行分类预测。其中L2标准化即把之前计算出的特征向量x(x1,x2,…,xn)的L2范数定义为公式(5-2)。

normalization(x) = (5-2)

要让x归一化到L2的范数，需要建立一个映射满足映射结果x’的L2范数为1，推导公式如公式(5-3)。

normalization(x’)=== = (5-3)

即：

通过L2标准化后可将特征权重全部归一化处理，方便后续liblinear的分类。

## 5.3 本章小结

本章主要介绍了整个毕设过程中核心模块的流程图和核心代码。详细介绍了分类系统各个模块的设计方法。

# 第6章 系统测试与运行结果

## 6.1 运行环境

(1) JDK1.8

(2) Win10操作系统

(3) Chrome浏览器

## 6.2 运行和测试结果

### 6.2.1 运行流程

启动IntelliJ IDEA，运行SVMClassifierTest，开始对语料库进行训练，训练出的SVM模型序列化写入本地，供持久化的使用。训练的文本类型包括体育、健康、军事、教育、旅游、汽车、财经七个类别。训练的过程如图6-1所示。

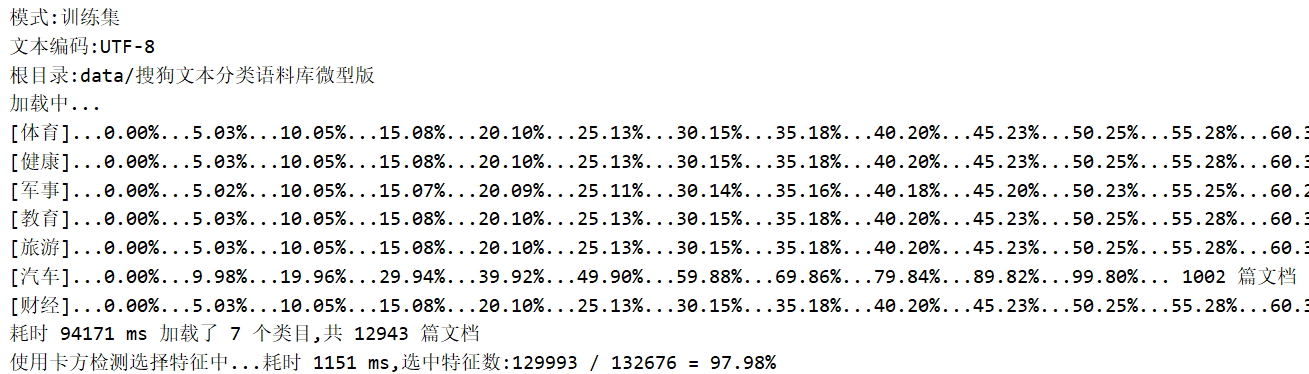


图6-1 语料库训练

训练得到分类模型后，启动SVMWebApplication，开始对网页进行分类操作。前端网页端口为设置为8080，在Chrome浏览器输入: <http://localhost:8080/>，显示用户界面如图6-2所示。

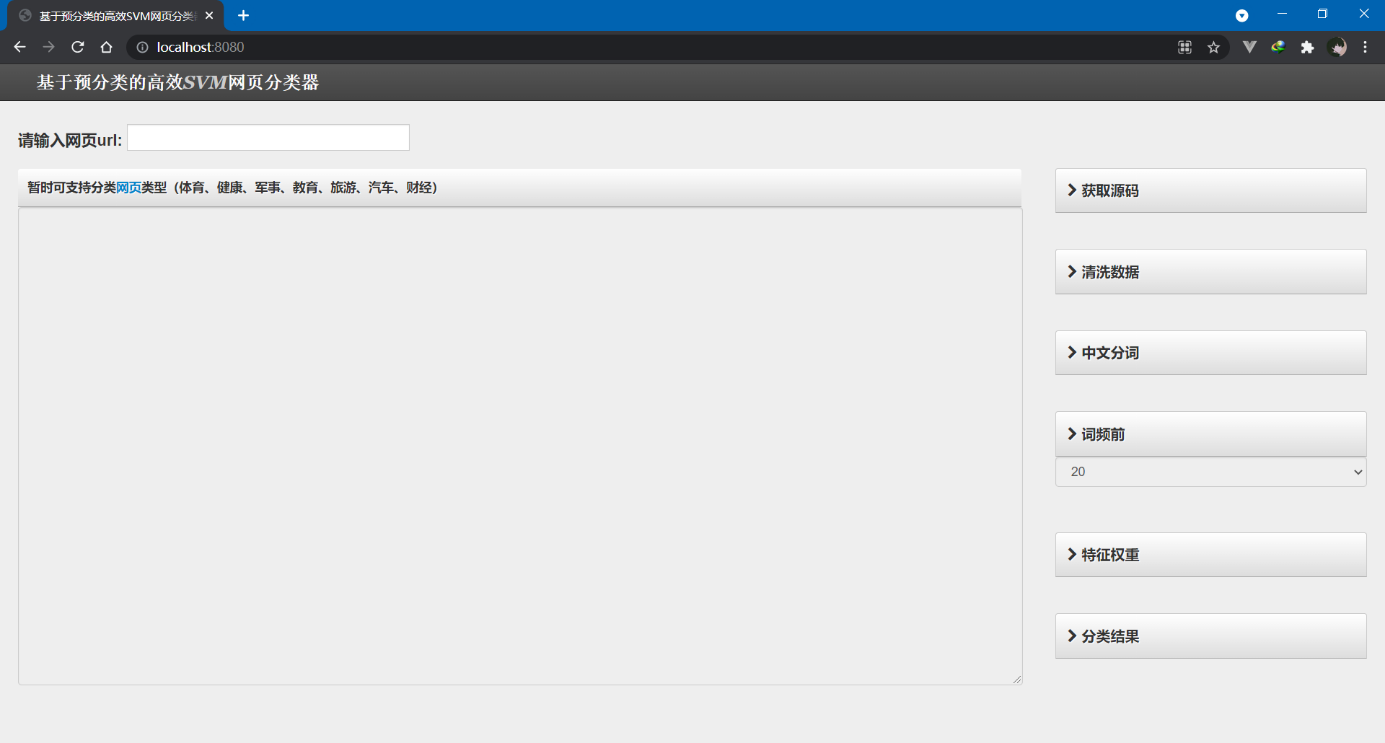


图6-2 用户界面

用户界面主要有八个部分组成，分别是输入网页的输入框、获取源码按钮、清洗数据按钮、中文分词按钮、词频前N按钮、特征权重按钮、分类结果按钮和左侧显示文本框。接下来通过一个具体的案例展示分类过程。

(1) 输入网页

第一步先输入待分类的网页的url，这里测试网页使用腾讯体育的官网为例，在输入框输入腾讯体育官网地址：<https://sports.qq.com/>。

(2) 获取源码

点击获取源码按钮，显示如图6-3。

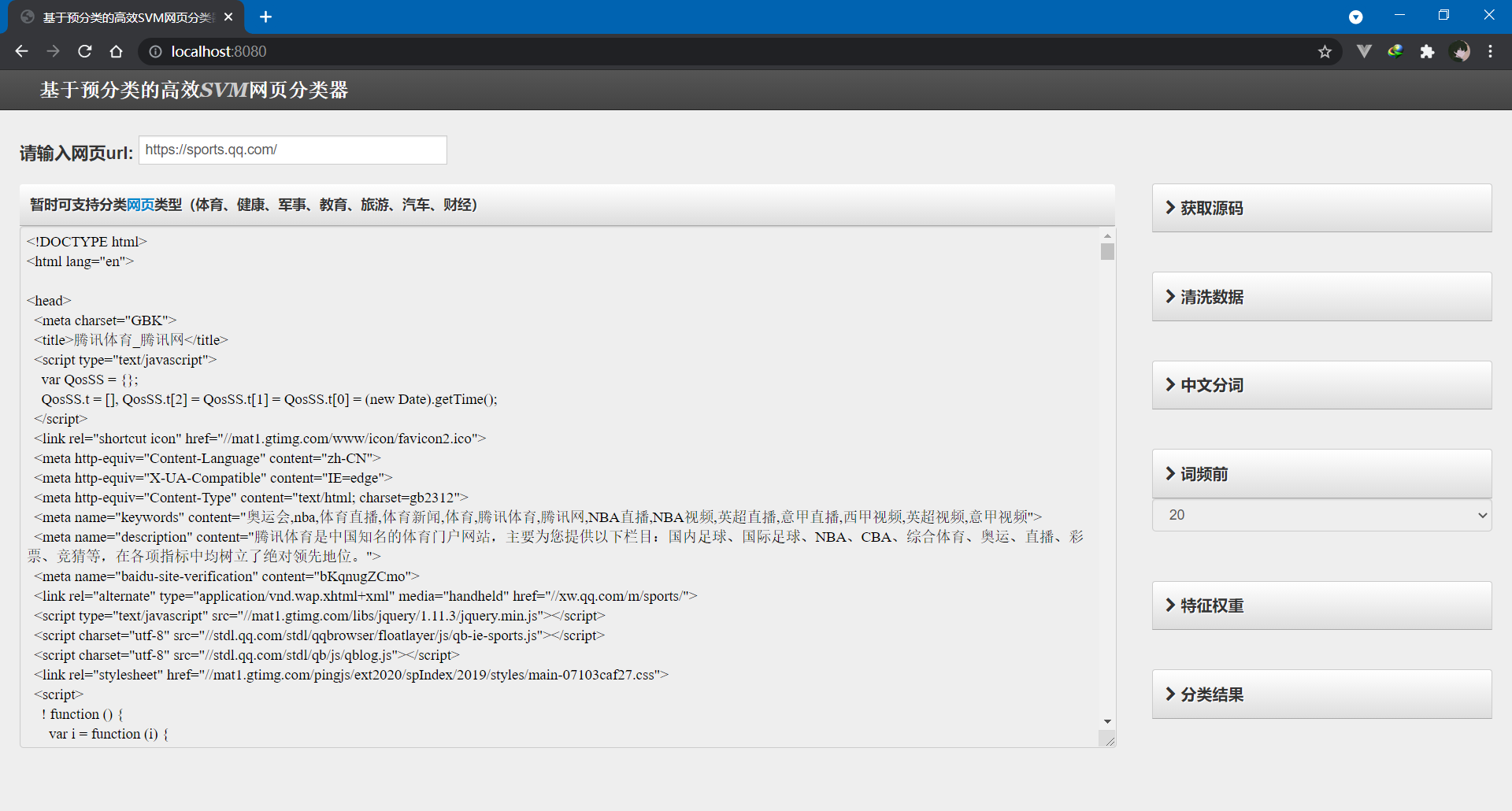


图6-3 获取源码

(3) 清洗数据

完成获取源码操作后，对保存在本地的源码文件进行处理，提取出有用的中文文本数据，按照一定格式显示在文本框。显示如图6-4。

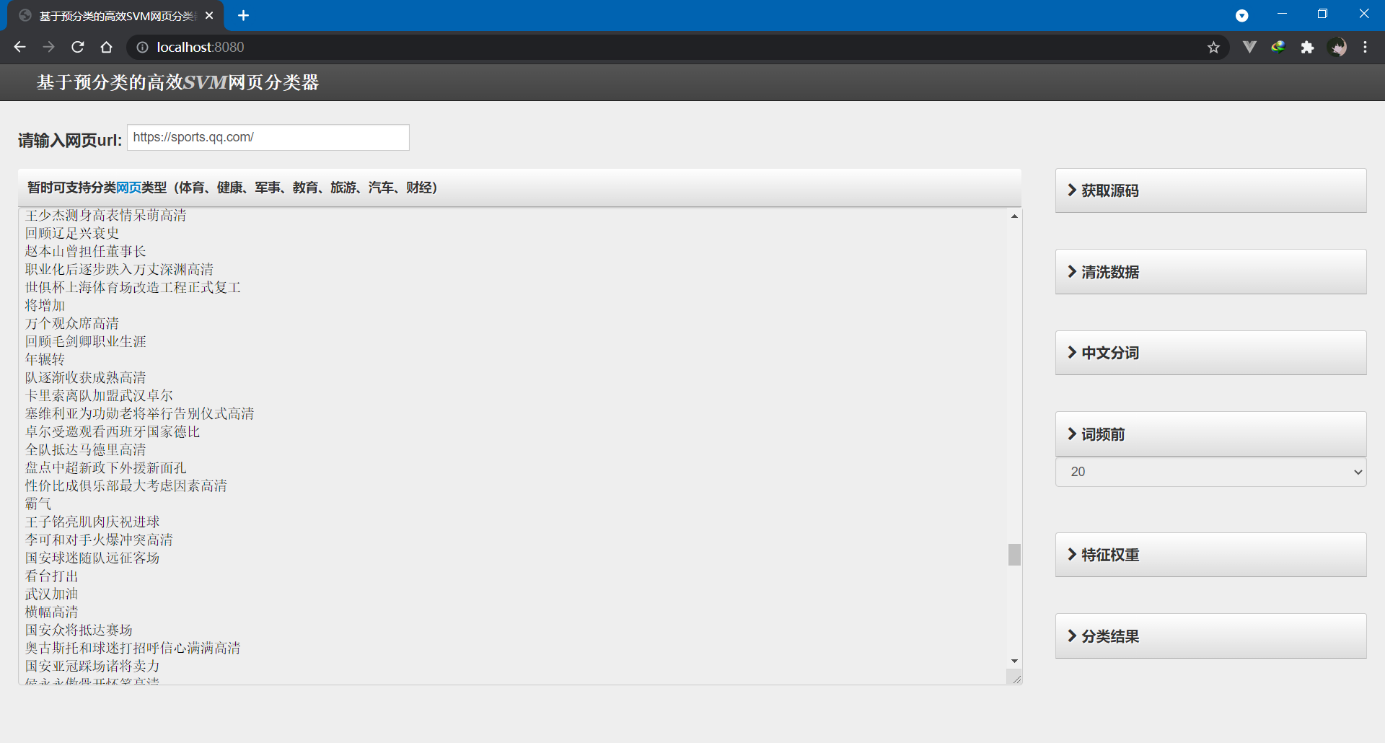


图6-4 清洗数据

(4) 中文分词

对清洗过的中文数据进行分词操作，分词结果统计词频并且去除停用词后按照一定格式显示到文本框。显示如图6-5。

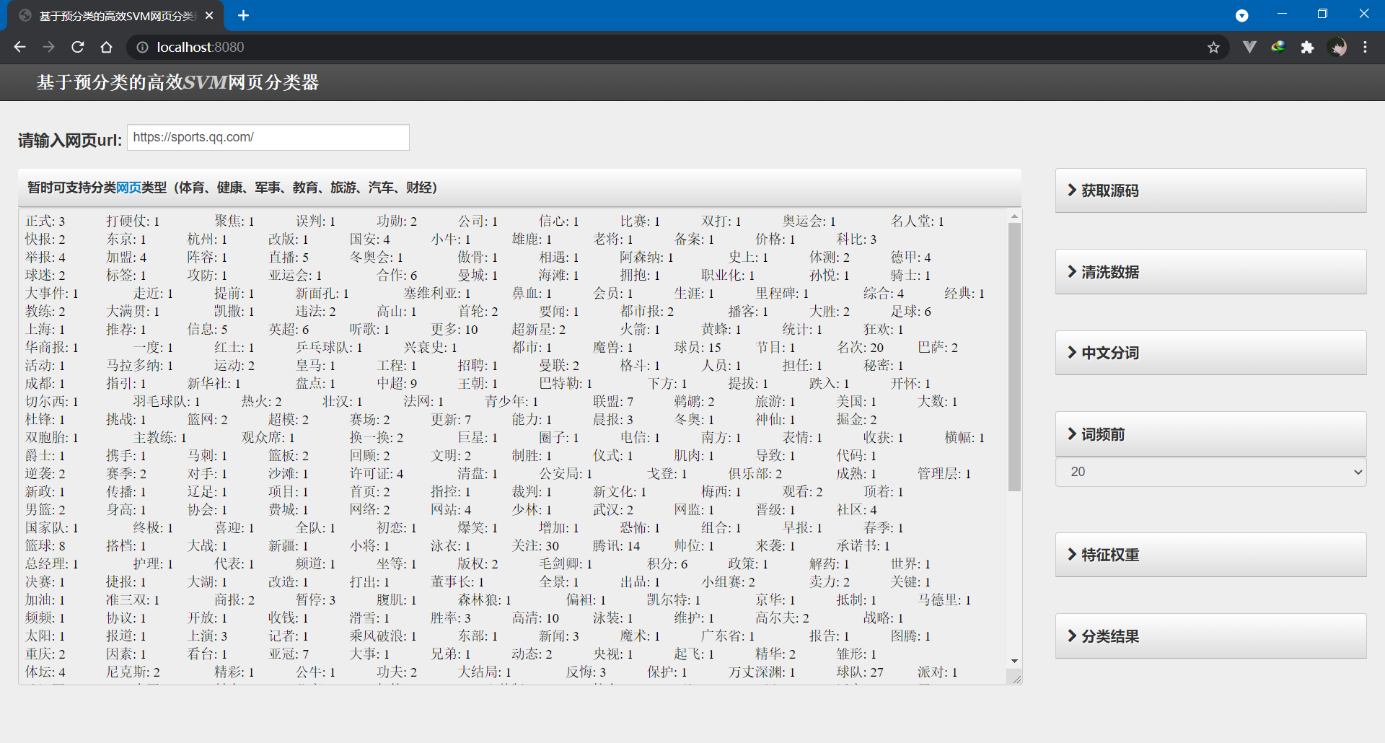


图6-5 中文分词

(5) 词频前N

根据已经得到的中文分词选择想要提取的词频前N位词语。这里选择N为40，提取词频前40，并将结果显示到文本框。显示如图6-6。

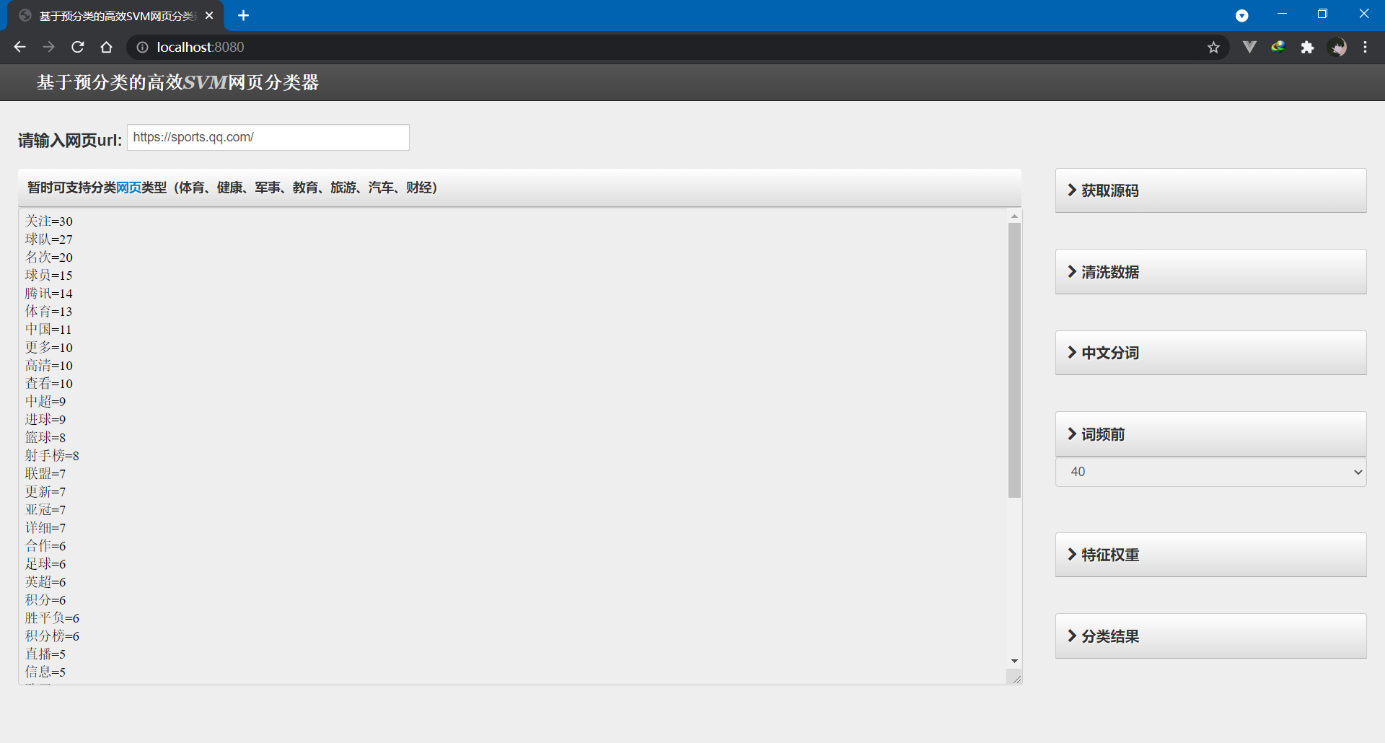


图6-6 词频前N

(6) 特征权重

在得到网页的特征词和词频后，结合训练过程中获得的对应特征词的文件频率，计算特征词的TF-IDF值，将所有特征词的权重经归一化处理后按照一定格式显示到前端文本框。显示如图6-7。

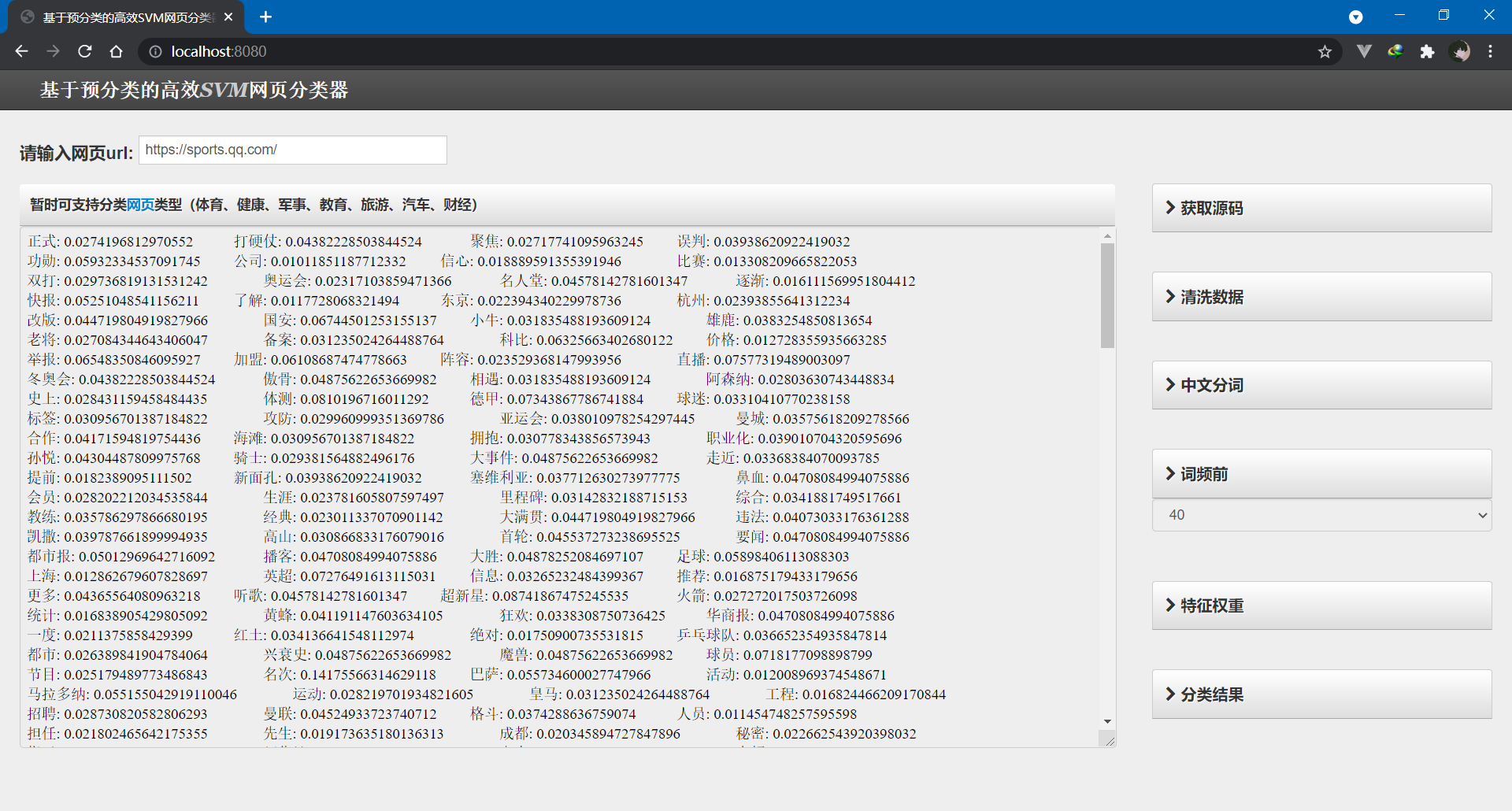


图6-7 特征权重

(7) 分类结果

将得到的特征节点和训练所得SVM模型作为liblinear预测分类方法的参数，进行分类结果的预测。预测结果数值归一化后显示各个分类可能性，并显示到文本框。显示如图6-8。

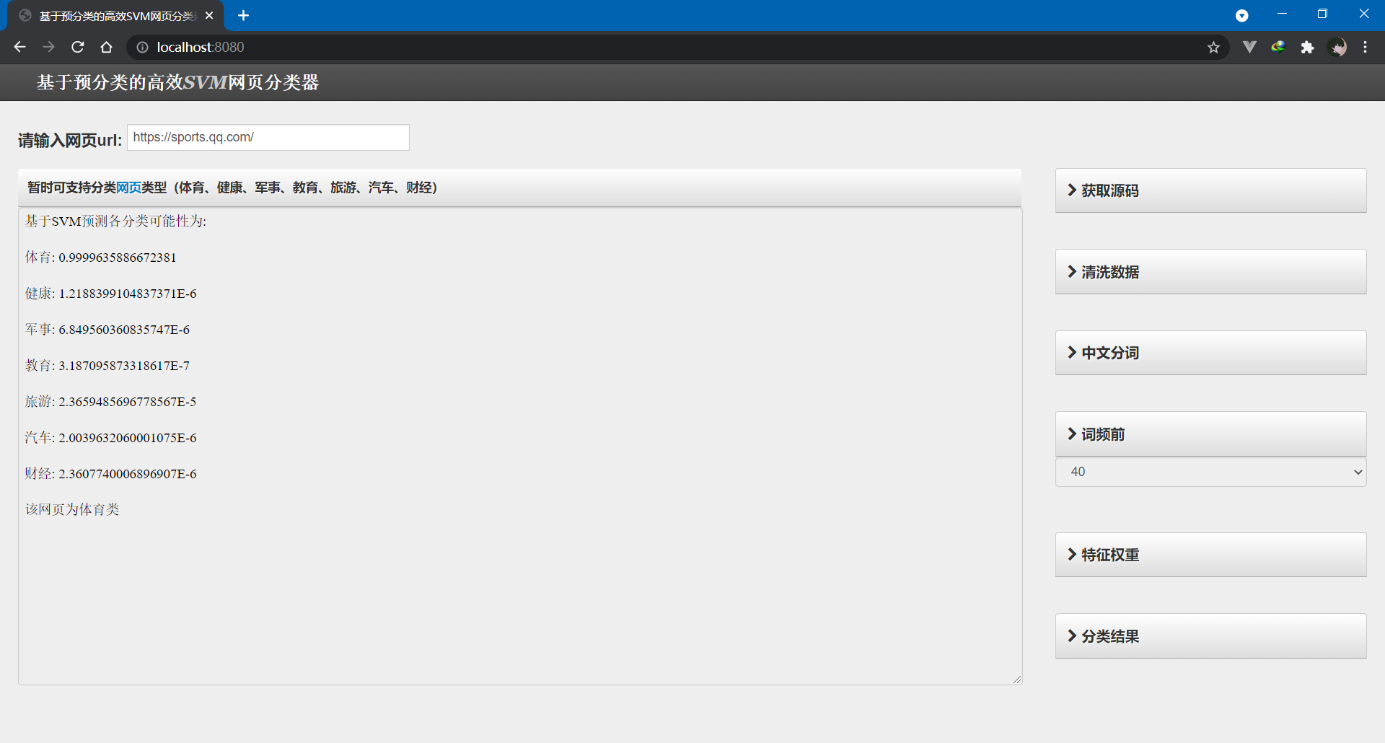


图6-8 分类结果

### 6.2.2 系统测试

为了更好的测试该分类系统的能力，对语料库中的文本进行划分，从七个分类中各挑选出200篇文档，共1400篇文档进行系统性的验证操作，最终测试结果如图6-9所示。

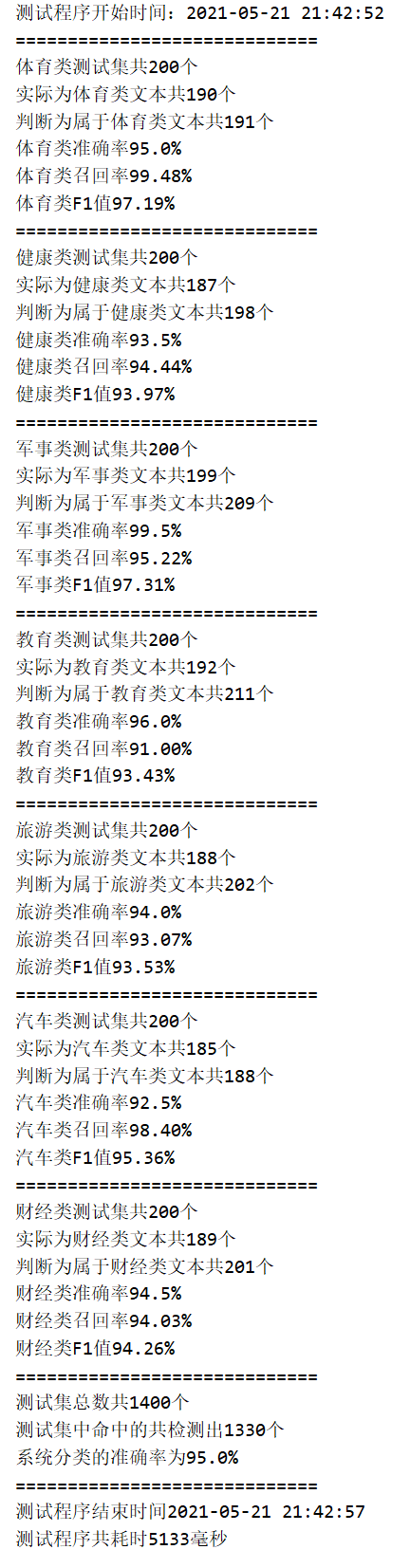


图6-9 测试结果

系统评价的指标有很多种，常用的有准确率、召回率以及F1值。接下来对本系统采用这三种评价方法进行评测。

(1) 准确率和召回率

准确率和召回率的计算公式如公式6-1和公式6-2所示。

准确率： P = (6-1)

召回率： R = (6-2)

其中上述公式中各字母解释如表格6-1所示。

|  |  |  |
| --- | --- | --- |
|  | 实际属于该类文档数 | 实际不属于该类文档数 |
| 判断属于该类文档数 | TP | FN |
| 判断不属于该类文档数 | FP | TN |

表格6-1 参数表

(2) F1值

准确率和召回率分别反映了分类不同方面的的质量，为了能够综合两者，后续提出了使用F1值来评估分类的质量。F1值的计算公式如公式6-3所示。

F1 = (6-3)

根据测试程序计算结果，得到各个分类准确率，召回率，F1值的情况如表格6-2所示。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 类别 | 体 | 健 | 军 | 教 | 旅 | 汽 | 财 |
| 指标 | 育 | 康 | 事 | 育 | 游 | 车 | 经 |
| 准确率(%) | 95.00 | 93.50 | 99.50 | 96.00 | 94.00 | 92.50 | 94.50 |
| 召回率(%) | 99.48 | 94.44 | 95.22 | 91.00 | 93.07 | 98.40 | 94.03 |
| F1值(%) | 97.19 | 93.97 | 97.31 | 93.43 | 93.53 | 95.36 | 94.26 |

表格6-2 系统性能评估

## 6.3 本章小结

本章主要对系统进行了运行和性能测试。测试结果显示该分类系统的准确率达到了95%，各系统评价指标也显示出该系统性能良好。

# 第7章 总结

## 7.1 主要工作

## 7.2 特色

## 7.3 不足与展望

## 7.4 收获

# 致谢

# 参考文献

[1] Yang Yi-ming.An evaluation of statistical approaches to text categorization[J].Information Retrieval，1999，1（1）：76-88.

[2] Yang Yi -ming，Slattery S，Ghani R.A study of approaches to hy－pertext categorization [J].J Intelligent Information System ，2002，18（2/3）：219-241.

[3] Vapnik, V.．Statistical learning theory. 1998 (Vol. 3). ．New York, NY：Wiley，1998：Chapter 10-11, pp.401-492

[4] 周志华．机器学习．北京：清华大学出版社，2016：pp.121-139, 298-300

[5] 吴艳玲. 基于SVM 的网页分类器的研究

[6] Vapnik, V.N. and Lerner, A.Y., 1963. Recognition of patterns with help of generalized portraits. Avtomat. i Telemekh, 24(6), pp.774-780.

[7] Boser, B.E., Guyon, I.M. and Vapnik, V.N., 1992, July. A training algorithm for optimal margin classifiers. In Proceedings of the fifth annual workshop on Computational learning theory (pp. 144-152). ACM.

[8] Cortes, C. and Vapnik, V., 1995. Support-vector networks. Machine learning, 20(3), pp.273-297.

[9] 申红, 吕宝粮, 内山将夫, 等. 文本分类的特征提取方法比较与改进.计算机仿真，2006，23(23):222-224

[10] Yang Y, Liu X. A re-examination of text categorization methods. Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval.

[11] Burges C. A tutorial on support vector machines for pattern recognition [ J ] . Data Mining and Knowledge Discovery ,1998 ,2 (2) :121 – 167

[12] 李亮，刘万春，徐泉清，等.一种基于支持向量机的专业中文网页分类器.计算机应用，2004，24(4)：58-61

[13] 孙剑, 郑南宁, 张志华. 一种训练支撑向量机的改进贯序最小优化算法. 软件学报.2002,13(10):2007-2013

[14] Platt,J. Sequential Minimal Optimization-A Fast Algorithm for Training Support Vector Machines.1999.