

基于 SVM 的程序设计相关网页判别系统

摘要

ACM/ICPC 是一个有着重大影响力的国际大学生程序设计竞赛，互联网亦有大量的训练习题与解题报告。有些学校会建设自己的网站用于收集解题报告，供师生参考学习。该系统可服务于这样的网站，用于自动化评判网页。

该判别系统接受一个 url（或本地网页）作为输入 x ，判别后输出结果，一个重要的参数为字符串 y ，用于阐明该网页的类别。记集合 $A = \{\text{网页} \mid \text{它描述的是计算机数据结构编程的相关内容}\}$ ，那么 y 为“yes”或“no”，前者表示 $x \in A$ ；后者表示 $x \notin A$ 。

该系统可分为三个模块。

收集模块用于网页收集与标注，实现网页到原始训练集的处理。

训练模块以上个模块的输出作为输入，先后进行中文分词处理、初始特征空间统计，并通过卡方检验得到抽取后的特征空间，由此可计算每篇文档的特征向量，归一化后送往 SVM 进行训练。这是一个机器学习过程，通过对径向基核函数较关键的参数 γ 与 C 调优，得到训练好的模型。

判别模块通过 web 页面与用户交互，可接受 url 或本地网页文档作为输入，输出结果包括网页的标题、正文、分词结果、特征向量、判别结果、判别结果准确率估计这 6 个参数。除了与人交互外，判别模块同时也提供了 web 服务，可以供其他代码跨平台批量调用。

关键词：程序设计，支持向量机，文本分类，机器学习，统计

A SYSTEM BASED ON SVM TO JUDGE THE WEBPAGE ABOUT PROGRAMMING DESIGN

ABSTRACT

ACM/ICPC is a famous international collegiate programming contest. There are many exercises and problem-solving reports in the Internet. Some schools would build their own websites to collect them for teaching, and this system is helpful to judge the webpages automatically.

This judge system accepts an url (or a local webpage) as the input x , then outputs a result which contains an important parameter y , indicating the type of this webpage. We assign set $A = \{\text{webpage} \mid \text{the content it describes is relative to computer programming or data structure}\}$, so y is "yes" or "no". The former means $x \in A$; and the later means $x \notin A$.

This system is divided into 3 parts. Collecting module collects and labels the webpages. Namely, it transforms webpages to raw training set.

Training module uses the former module's output as input, then complete following tasks: Chinese word segmentation, raw feature space statistics, Chi-square testing, feature vector calculate per document. Next sends the vector set into SVM after being normalized to train. This is a machine learning procedure, gaining a module by adjusting parameters γ and C of the RBF.

Judging module's output contains title, content, word segmentation, feature vector, result and probability estimate. In addition, this judge system provides web service, which makes it convenient that other codes' cross platform and batch calling.

Key words: svm, support vector machine, text classification, machine learning

目 录

摘要.....	I
ABSTRACT.....	II
1 绪论.....	3
1.1 研究背景.....	3
1.2 文本分类研究现状.....	3
1.3 主要研究内容.....	4
1.4 本文章节安排.....	5
2 理论基础与相关工具.....	6
2.1 中文分词.....	6
2.2 卡方检验.....	7
2.3 神经网络.....	10
2.3.1 神经网络简介.....	11
2.3.2 核方法与径向基函数.....	12
2.4 支持向量机.....	12
2.4.1 线性可分模式的最优超平面.....	12
2.4.2 线性不可分模式的最优超平面.....	14
2.5 前端工具.....	15
2.6 后端工具.....	16
3 需求分析.....	19
3.1 功能需求.....	19
3.2 界面需求.....	19
3.3 运行环境需求.....	20
4 系统设计.....	21
5 系统实现.....	24
5.1 数据收集模块.....	24

5.1.1 web 界面实现.....	24
5.1.2 根据 URL 或本地文件解析网页.....	26
5.1.3 数据库设计与实现.....	27
5.1.4 数据库访问对象的单例实现.....	29
5.1.5 爬虫模块实现.....	29
5.1.5 收集数据说明.....	30
5.2 数据处理模块.....	33
5.2.1 中文分词的 IKAnalyzer 实现.....	33
5.2.2 java 并发与同步.....	35
5.2.3 文档结构化存储的数据结构实现.....	37
5.2.4 卡方检验实现.....	39
5.2.5 特征向量计算与归一化.....	41
5.2.6 libsvm 工具箱的 javaAPI 使用.....	43
5.2.7 使用 libsvm 进行训练.....	46
5.2.8 机器学习效果评价.....	49
5.3 判别模块.....	49
5.3.1 用户交互 web 界面实现.....	50
5.3.2 文件上传功能实现.....	53
5.3.3 web service 调用.....	53
6 总结与展望.....	55
6.1 本文工作总结.....	55
6.2 未来研究展望.....	55
参考文献.....	57
致谢.....	58
附录.....	59

1 绪论

1.1 研究背景

ACM/ICPC, 国际大学生程序设计竞赛的重要性与含金量不言而喻, 东华大学一直致力于此竞赛的学生选拔与学生培养, 并搭建了本校自己的 Online Judge 系统 (简称 DHUOJ)。此系统在汇总了一些其他著名 OJ 题库习题的同时, 也在互联网中收集了不少相应的解题报告, 供师生参考学习。

DHUOJ 的解题报告收集模块现在是这样工作的: 以题号为关键字在搜索引擎 (如百度等) 中搜索, 对前面若干个搜索结果进行判别, 判别规则为网站域名对比、关键词匹配等, 较为机械, 准确率也不是很高, 词库需要人工维护且扩展性不强。

此系统利用基于统计的机器学习算法, 期望得到更好的判别效果。

1.2 文本分类研究现状

文本分类是指计算机在接收给定的文本作为输入后将其归属于预先给定的若干个类的过程。计算机并不能够像人一样阅读和理解文章, 所以必须把文本转化成计算机能够理解的形式。向量空间模型^[1] (Vector Space Model) 是目前文本表示的主流模型。它的基本思想是把文档简化为词语的集合, 每个词语作为特征向量的分量表示: (w_1, w_2, \dots, w_n) , 其中 w_i 为第 i 个特征词语的权重^[2]。但是, 文章所携带的信息由多部分组成: 词语、词语间的顺序以及文本的上下文信息。向量空间模型这种文档表示方法, 基本上忽略了词语以外所有的部分, 这使得它能表达的信息量存在上限, 因此该类系统几乎不可能达到人类的分类能力。

当前文本分类的主流实现是基于统计学习的方法。它的实际表现较为优秀,

误判率比较低，主要有以下几种。

第 1 种：k 近邻算法。选择特征向量和新文档特征向量最相近的 k 个文档。它们中的大多数属于哪一类，就倾向于认为这个新文档属于哪一类。这种方法要与每个文档比较，计算量很大^[3]。

第 2 种：决策树方法。决策树可视为一棵树的预测模型，它的结点表示测试特征，边表示特征的每个值，叶结点对应类别。从决策树根结点到不同叶结点的一个个路径就是对待预测对象的预测过程。

第 3 种：反向传播神经网络。它可以实现非线性分类，通常采用两层隐藏网络，输出神经元个数为分类个数。

第 4 种：支持向量机。通过构造合适的核函数，将样本空间非线性投射到高维空间，在高维空间实现线性分类。

第 5 种：朴素贝叶斯。基本思想是在已知先验概率和条件概率的情况下，计算待分类文本属于各个类别的后验概率，然后将待分类文本分到后验概率最大的类别中。

该系统采用支持向量机来解决文本分类问题。

1.3 主要研究内容

该系统要从头到尾实现一个网页分类判别系统。按照信息处理顺序，依次要完成数据收集、数据处理、交付使用三个阶段。

数据收集要完成的内容为网页获取、网页解析、添加标注、数据存入数据库。

数据处理模块又可以分为预处理模块和训练模块。前者完成数据库中读取数据、中文分词、特征空间统计、特征向量计算等所有前期预处理功能；后者通过对训练集的训练得到训练模型，交付给最后的数据判别模块使用。

判别模块提供用户交互与程序交互两种使用模式。它们背后的实现流程是一样的，依次是网页获取、网页解析、中文分词、计算特征向量、送往 libsvm 工具结合已有的训练成果进行预测。

本文对每一部分的设计与实现都进行了详略得当的讨论。

1.4 本文章节安排

本文共分为 6 个章节。

第 1 章绪论介绍了该论文的选题背景、文本分类的主流方法、本文主要研究内容和章节安排。

第 2 章介绍了该系统涉及的理论基础与用到的第三方工具。主要包括了中文分词与卡方检验，它们都是该系统实现过程中的必要处理流程。然后介绍了神经网络与支持向量机，它是该系统用到的 libsvm 机器学习工具的理论基础。

第 3 章为需求分析，主要包括功能需求、界面需求与运行环境需求。

第 4 章给出了该系统的总的逻辑设计与各模块的流程图设计。

第 5 章详细讲述了该系统的实现过程——综合运用多种工具完整实现了从数据收集到机器学习，再到最后的提供接口支持第三方调用。

第 6 章对本文的工作进行了总结，并对未来的进一步工作做出展望。

2 理论基础与相关工具

2.1 中文分词

我们用若干个词组来描述一篇文档的特征，这样做是有道理的。文章的组成单位，从小到大依次是字、词、句、段、篇、章。如果以单个的字为信息描述单位，表述能力太差；如果以句子为单位，包含的信息则过于精确，泛化性能太差。于是词组就成了理想的选择。科技文献的摘要后面一般都附有关键词，既方便于人快速查，也方便于搜索引擎的抓取和分类，就是这个道理。

中文分词，指的是将给定的汉字字符串以词组为单位进行切分。以句子“我今天考了 100 分”为例，那么理想的分词结果就应该是“我/今天/考了/100/分”。

在讲述中文分词原理之前，首先讲一下英文如何分词。在英文当中，单词之间天然有着空格作为分隔，所以分词很容易实现。而中文中，虽然有逗号和句号等作为文章内容的分隔，但词语之间是没有任何分隔的。

对于中文分词，现有的分词算法可分为三大类。它们分别是基于字符串匹配的机械分词方法、基于理解的分词方法和基于统计的分词方法。

(1)机械分词方法

事先准备好一个充分大的词典，在待分词的字符串中对比着词典查词，若找到一个词语，就表示匹配成功。按照扫描方向不同，机械分词方法可以分为正向匹配（从左到右的扫描顺序）和反向匹配（从右到左的扫描顺序）。也可以按照不同长度词语匹配的优先权重不同来分类，分为最大（最长）匹配和最小（最短）匹配。

现代汉语中的词语并非都是两个或两个以上汉字构成的，所以正向最小匹配和逆向最小匹配使用的比较少。实验数据表明，逆向匹配的精度略高于正向匹配，遇到的歧义现象也较少。在分词系统的实际使用中，往往把机械分词作为初步分

词手段，后续还需运用其他手段来进一步提高分词的准确率。

(2)理解分词方法

让计算机来模拟人对句子的理解。在字符串匹配的基础上，通过词法、语法分析来处理歧义现象。歧义现象在中文分词处理中比较常见，它是指同样的一句话，可能有两种或更多种切分方法^[4]。下面是一个交集型歧义：“化妆和服装”可以分成“化妆/和/服装”或者“化妆/和服/装”。计算机遇到此类情况时有时并不能给出令人满意的结果。

(3)基于统计的分词方法

从某种角度上来看，词语是稳定的字与字之间的组合。因此在上下文中，彼此相邻的字同时出现的次数越多，它们就越有可能构成一个词。因此字与字彼此相邻的频率能够在一定程度上反映出成词的可信度。该方法最大的缺点是要有大量预先分好词的语料来作为支撑，且统计过程中程序的时空开销极大。

2.2 卡方检验

对该系统所收集到的所有样本进行词语统计，结果显示共有 1,755,220 个词语，去重之后共有 61,100 种词语。把这么多词语中的每一个词语都作为一个描述样本的特征，显然特征空间是过大的。一个很自然的想法就是，高维的特征向量应该更有利于描述样本特征，提高机器学习的准确度，而实际并非如此^[5]。这么大的维度可能会带来维度灾难，因此就要从大量的特征中选择一些有代表性而又不影响分类效果的特征。特征选择就是从初始特征空间中选择一些代表性的词作为新的特征空间，这个过程也成为降维。

特征提取的常用方法有文档频率，信息增益，互信息，卡方检验等。

一些研究者在对多种特征选择方法和分类方法做了大量的对比试验后，得出结论为卡方统计量特征选择方法的特征选择效果最好，获得的分类精度最高^[6]。故该系统使用卡方检验来降维。

卡方检验的基本思想是通过观察理论值与实际值的偏差来确定理论的正确与否。例如用卡方检验判断两个变量是否独立，具体的做法为先假设两个变量确实是独立的（称为原假设），然后对比实际值（也就是观察值）与理论值的偏差

程度。如果偏差足够小，就接受原假设。如果偏差大到超出阈值，就否定原假设，接受备择假设。

在一个卡方检验中，假设理论值为 E ，多个观测值分别为 x_1, x_2, \dots, x_n ，那么偏差程度为

$$\sum_{i=1}^n \frac{(x_i - E)^2}{E} \quad (2-1)$$

使用 $(x_i - E)^2$ 作为分子是为了避免累加过程中正误差与负误差的抵消，又添加一个 E 作为分母是为了避免均值的大小对差异程度的影响。

计算出来值以后，与事先设定好的阈值进行比较，如果大于阈值，就认为原假设不成立，反之则认为原假设成立。

判断一个网页是否与计算机编程相关，实际上就是一个对网页的分类，而该系统是以网页正文来作为分类依据的，所以它就是一个文本二分类问题。在文本分类问题中，我们主要关心一个词 **token**（一个随机变量，下文简写为 t ）与一个类别 **class**（另一个随机变量，下文简写为 c ）之间是否相互独立。如果独立，就说明词 t 对一篇文章是否属于类别 c 的判断没有价值。但与一般的卡方检验不一样的是此处并不设定阈值，因为很难说明词 t 和类别 c 关联到哪种程度才算是对分类有价值或无价值的。此处的卡方检验的价值在于帮助我们根据卡方值筛选出一些相对来说对分类有价值的词语即可。

该系统在卡方检验阶段使用的原假设为“词 t 与类别 c 不相关”。计算出的卡方值越大，就说明对原假设的偏离越大，就可以越倾向于认为原假设的反面情况是正确的。选择的过程为对每个词计算它与类别 c 的卡方值，然后从大到小排序，取前若干个就可以。

例如，该系统收集了 n 个网页，分类有正样例和负样例，考察特征词“算法”与正样例（即该网页描述内容为计算机编程相关）的相关性。经过统计，得到了四个观察值 a 、 b 、 c 、 d 。

a: 包含“算法”且属于正样例的网页数。

b: 包含“算法”但属于负样例的网页数。

c: 不包含“算法”但属于正样例的网页数。

d: 不包含“算法”且属于负样例的网页数。

现在用表 2-1 来描述四个观察值 a、b、c、d 的关系。

表 2-1 “算法”与正负样例的统计信息

特征选择	正样例	负样例	总 计
包含“算法”	a	b	a+b
不包含“算法”	c	d	c+d
总 计	a+c	b+d	n

如果原假设是成立的，即“算法”和正样例没什么关联性，那么在所有的网页中，“算法”这个词都应该是等概率出现，而不管网页是不是与计算机编程相关的。我们并不能够知道这个概率具体是多少，但它应该体现在观察结果中，因此我们可以说这个概率 P （“算法”出现）接近 $(a+b)/n$ ，因为 $(a+b)$ 是包含“算法”的网页数，除以总网页数 n 就是“算法”出现的概率。那么在正样例网页中，就应该有 $(a+c) \cdot P$ （“算法”出现）个网页包含“算法”，这是我们的理论值，很显然实际值为 a 。于是得到了偏差表达式

$$D_{11} = \frac{(a - E_{11})^2}{E_{11}} \quad (2-2)$$

同理可以计算剩下三种情况的差值 D_{12}, D_{21}, D_{22} 。最后得到“算法”与正样例

$$\text{的卡方值 } \chi^2(\text{“算法”}, \text{正样例}) = D_{11} + D_{12} + D_{21} + D_{22} \quad (2-3)$$

把 $D_{11}, D_{12}, D_{21}, D_{22}$ 的值分别代入并化简，可以得到

$$\chi^2(\text{“算法”}, \text{正样例}) = \frac{n(ad - bc)^2}{(a+b)(a+c)(b+d)(c+d)} \quad (2-4)$$

词 t 与类别 c 的平方值更一般的形式可以写成

$$\chi^2(t, c) = \frac{n(ad - bc)^2}{(a+b)(a+c)(b+d)(c+d)} \quad (2-5)$$

在计算过程中，对于每一个词语 t ，公式中的 $(a+c)$ 、 $(b+d)$ 与 n 都是固定的。因此为了简化计算，将公式改为

$$\chi^2(t, c) = \frac{(ad - bc)^2}{(a+b)(c+d)} \quad (2-6)$$

并不影响词语之间卡方值大小的比较。

卡方检验也有它的缺点。它只统计文档是否出现某个词，而却不管它出现了几次。这样就会导致对低频词作用的夸大，甚至会出现某个词在正样例的每篇网页中都只出现了一次，其开方值却大过了在正样例 99% 的网页中出现了 10 次的词。而要知道，其实后面的词才是更具代表性的，而它们却因分数不高在卡方检验中被淘汰出去。这就是卡方检验著名的低频词缺陷。

2.3 神经网络

该系统使用支持向量机作为机器学习模型，它又是神经网络的一种实现形式。此章节对神经网络与支持向量机作出简要讲解。

人脑计算与传统的数字计算机相比有着完全不同的工作方式。人脑是一个非线性的和并行的计算机器，它依靠神经元完成复杂的诸如感知、模式识别等的计算。最普通的人工神经网络（下文简称神经网络），就是模仿人脑，以人工神经元为基础进行建模的机器^[7]。

本章中有部分数学定理阐述，用到了一些符号与约定，见表 2-2。

表 2-2 符号与缩写约定

缩写或符号	英文	中文
\mathbf{a}^T	transpose of vector \mathbf{a}	向量 \mathbf{a} 的转置
$\mathbf{a}^T \mathbf{b}$	inner product of vectors \mathbf{a} and \mathbf{b}	向量 \mathbf{a} 和 \mathbf{b} 的内积
$\mathbf{a} \mathbf{b}^T$	outer product of vectors \mathbf{a} and \mathbf{b}	向量 \mathbf{a} 和 \mathbf{b} 的外积
$\ \mathbf{x}\ $	Euclidean norm of vector \mathbf{x}	向量 \mathbf{x} 的欧几里得范数

2.3.1 神经网络简介

人工神经元的模型见图 2-1。

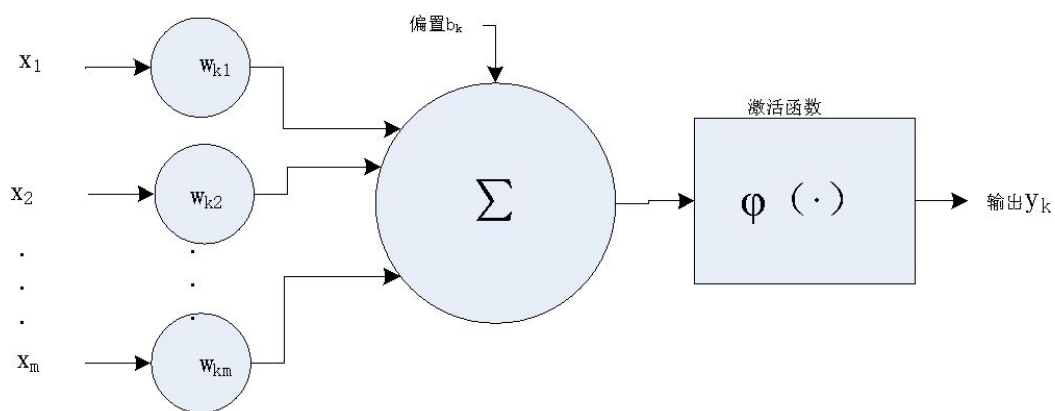


图 2-1 人工神经元模型示意

其中 x_1, x_2, \dots, x_m 是输入信号， $w_{k1}, w_{k2}, \dots, w_{km}$ 是神经元 k 的突触权值， b_k 为偏置。求和节点的输出为

$$v_k = \sum_{i=0}^m w_{ki} x_i \quad (2-7)$$

最终的输出

$$y_k = \varphi(v_k) \quad (2-8)$$

该系统所使用的支持向量机，以径向基函数作为核函数，所以下面简要介绍核方法与径向基函数。

2.3.2 核方法与径向基函数

核方法是用来解决非线性模式分析问题的一个手段，它将原始数据变换到高维特征空间，然后在输出层进行分类^[8]。Cover 定理说明了这样做的潜在合理性，该定理可以定性地表述如下。假设空间不是稠密分布的，将复杂的模式分类问题非线性地投射到高维空间将比投射到低维空间更可能是线性可分的。

径向基函数是这样一类函数，

$$\varphi(\mathbf{x}, \mathbf{c}) = \varphi(\|\mathbf{x} - \mathbf{c}\|) \quad (2-9)$$

即函数值仅依赖于变量间的距离（通常为欧式距离）。欧式距离的计算方法为，以一个 m 维的向量 $\mathbf{a} = (x_1, x_2, \dots, x_m)$ 为例，则该向量的欧几里得范式

$$\|\mathbf{a}\| = \left(\sum_{i=1}^m x_i^2 \right)^{0.5} \quad (2-10)$$

2.4 支持向量机

支持向量机（SVM，全称 Support Vector Machine）的主要思想是建立一个超平面作为决策曲面，使得正例和反例之间的隔离边缘被最大化。支持向量机的理论基础是统计学习理论，更精确的说，支持向量机是结构风险最小化的近似实现。Joachims 最早将 SVM 用于文本分类^[9]。

2.4.1 章节简要介绍了 svm 在线性可分模式下的工作原理，2.4.2 章节作为扩展，简述在处理更加复杂的线性不可分的模式时的工作原理。

2.4.1 线性可分模式的最优超平面

考虑训练样本 $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$ ，其中 \mathbf{x}_i 是输入模式的第 i 个样例， d_i 是对应的期

望响应。假设 d_i 的取值只有 +1 和 -1，这两种模式是线性可分的，则用于分离的超平面形式的决策曲线方程是：

$$\mathbf{w}^T \mathbf{x} + b = 0 \quad (2-11)$$

其中 \mathbf{x} 是输入向量， \mathbf{w} 是可调的权值向量， b 是偏置。因此可以写成：

$$\begin{aligned} \mathbf{w}^T \mathbf{x}_i + b &\geq 0 \quad \text{当 } d_i = +1 \\ \mathbf{w}^T \mathbf{x}_i + b &< 0 \quad \text{当 } d_i = -1 \end{aligned} \quad (2-12)$$

对于一个给定的权值向量 \mathbf{w} 和偏置 b ，由式 (2.4.1-1) 定义的超平面和最近的数据点之间的间隔被称为分离边缘。支持向量机的目标就是要找到一个，使分离边缘最大的超平面。

图 2-2 描述二维输入空间中最优超平面的几何示意图。

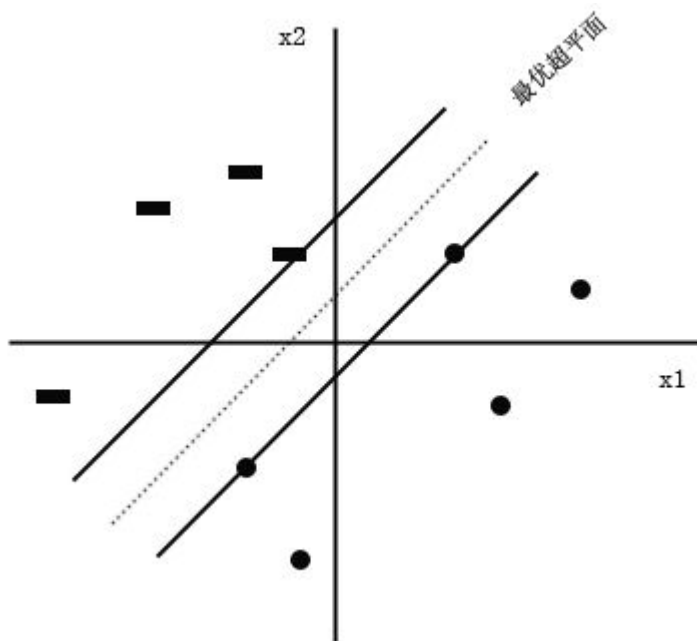


图 2-2 线性可分模式最优超平面的思想示意图

图 2-2 中黑色方块表示标签为 +1 的数据点，黑色圆点表示标签为 -1 的数据点。对于找到的一对最优超平面的参数 \mathbf{w}_0 和 b_0 ，它们一定满足

$$\begin{aligned} \mathbf{w}_0^T \mathbf{x}_i + b_0 &\geq 1, \text{ 当 } d_i = +1 \\ \mathbf{w}_0^T \mathbf{x}_i + b_0 &\leq -1, \text{ 当 } d_i = -1 \end{aligned} \quad (2-13)$$

落在最优超平面两侧边缘上的特殊数据点 (\mathbf{x}_i, d_i) (也即满足式 2.4.1-3 中等

号条件下的点) 称为支持向量, “支持向量机” 因此得名。

2.4.2 线性不可分模式的最优超平面

章节 2.4.1 中给定的训练样本是模式可分的, 然而现在给定这样一组训练数据, 使其肯定不能建立一个不具有分类误差的分离超平面。此时, 支持向量机的目标就变成了找到一个最优超平面, 使其对整个训练样本的分类误差的概率达到最小。

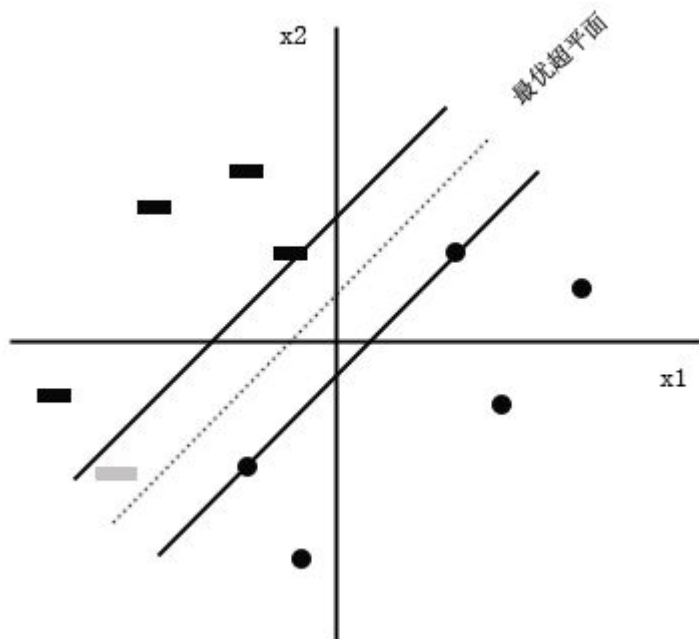


图 2-3 线性不可分模式的最优超平面

可以把式 (2.4.1-3) 中的两行合并为一行

$$d_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, i=1, 2, \dots, N \quad (2-14)$$

如果某个数据点不满足式 (8.2-1), 我们就称它为不可分离数据点, 对应着图 2-3 中的灰色方块。为了处理不可分离数据点, 现在引入一组新的非负标量变量 $\{\varepsilon_i\}_{i=1}^N$ 到分离超平面的定义中, 表示为

$$d_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \varepsilon_i, i=1, 2, \dots, N \quad (2-15)$$

这里 ε_i 称为松弛变量, 他们度量一个数据点对模式可分的理想条件的偏离程度。

我们可以通过最小化关于权值向量 \mathbf{w} 的泛函达到使平均错误分类的误差最小的

目的

$$\Phi(\varepsilon) = \sum_{i=1}^N I(\varepsilon_i - 1) \quad (2-16)$$

泛函满足式 (2.4.2-2) 的约束条件。函数 $I(\varepsilon)$ 是一个指标函数，定义为

$$I(\varepsilon) = \begin{cases} 0, \varepsilon \leq 0 \\ 1, \varepsilon > 0 \end{cases} \quad (2-17)$$

不幸的是， $\Phi(\varepsilon)$ 对 \mathbf{w} 的最小化是非凸的最优化问题，它是 NP 完全的。

为了使最优化问题数学上易解，为了逼近泛函 $\Phi(\varepsilon)$ 重写函数：

$$\Phi(\varepsilon) = \sum_{i=1}^N \varepsilon_i \quad (2-18)$$

而且，通过形成泛函对权值向量 \mathbf{w} 的最小化公式简化计算，得出

$$\Phi(\mathbf{w}, \varepsilon) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^N \varepsilon_i \quad (2-19)$$

$\sum \varepsilon_i$ 是测试错误数目的上边界。参数 C 控制着机器的复杂性和不可分离点数之间的平衡。当它的值选得比较大的时候，暗示着支持向量机的设计对训练样本的质量具有高度的信心。相反当参数 C 选择比较小的时候，认为训练样本中存在噪声，因此将对其不太强调。

2.5 前端工具

(1) bootstrap (css 前端框架)

bootstrap 是一个 css (Cascading Style Sheets) 前端框架，它简洁、直观、漂亮，特点之一是能够根据不同尺寸的设备（如手机、平板、笔记本与台式机的显示器尺寸不一）来响应式地布局，背后原理是 css 的媒体查询功能。有了响应式布局，就能够灵活地面对不同分辨率的设备，做到一次开发普遍使用。另一个特点是栅格化，把浏览器一行的宽度均分成十二等份，以方便对表单、按钮、表格等元素进行精致的网页排版。

(2) angularjs (MVC 前端框架)

angularjs 是一个前端 MVC 框架。MVC 是一种流行的开发理念——M, Model, 数据模型; V, View, 视图, 负责向用户展示的部分; C, Controller, 业务逻辑和控制逻辑。它们的模块化有利于维护与复用。本系统中用的最多的是数据双向绑定功能和依赖注入。以数据收集模块的网页解析内容为例, 当爬虫系统将网页解析结果以 json 格式返回给浏览器时, 不需要调用 `$('#id').val=jsonObj.theValue` 此类代码, angular 框架会根据此前指定的绑定关系自动刷新。

2.6 后端工具

(1) jersey (RESTful Web Service 框架)

jersey 是一个 RESTful Web Service 框架。web service 即远程函数调用, 通过该特性, 在互联网中跨机器调用其他服务器上的函数, 就像调用本地代码一样简单又方便。原理是框架把请求对象序列化为 json 形式的字符串, 发送 http 请求到指定的服务器上, 服务器端把 json 字符串再反序列化为对象, 找到函数代码入口后开始执行, 得到返回对象后序列化为 json 字符串, 作为客户端 http 请求的回应返回, 客户端再将结果反序列化为对象。至此, 完成一次远程调用。虽然细节上比较麻烦, 但框架已经为我们封装好了这些细节。还有一个好处就是——把服务分布在不同的机器上有利于负载均衡, 该请求还是无状态的, 对于此次请求的处理不依赖于上一次的状态。

(2) jackson (实现 json 与 java 对象的转化)

jackson 用于实现 json 与 java 对象的序列与反序列化。web service 要实现跨机器传送对象那么就需要有一种技术能把对象转换为特定格式的信息。为了实现跨平台 (比如把 java 对象转化为 c#或 c++对象), 又考虑到方便人们阅读, 所以基于字符串规则的转换是最理想的。json 就是这样一个轻量级的数据交换格式, 而 jackson 是一个实现该功能的很好用的框架。

(3) tomcat-jdbc-pool (数据库连接池)

连接池的基本思想是在程序第一次启动的时候, 将数据库连接 (通常为 `java.sql.connection`) 作为对象存储在内存中。当某段代码需要访问数据库时, 不

再建立一个新的连接,而是从预先准备好的连接池中取出一个已建立的空闲连接对象。使用完毕后,用户也不必主动地将连接关闭,而是将连接放回连接池中,以供下一个请求访问使用。整个过程中,数据库访问者与数据库服务端之间连接的建立、断开都由连接池自身来管理。同时,数据库连接池中的初始连接数、连接的上下限数以及每个连接的最大使用次数、最大空闲时间等等都是可以通过文本文件来配置的,非常灵活。使用数据库连接池可以将负责与数据库连接的模块独立出来,方便维护。以此为中转,避免了对数据库服务端频繁的连接与断开操作,减少资源占用,获得更快的响应时间。

(4) spring (依赖注入)

spring 的核心是依赖注入。本质就是通过配置 xml 文件或注解来自动生成你需要的对象,放在 spring 的大容器中。spring 是一个独立的框架,既可以用在一般的 java 本地代码中,也可以用在 Web 网站模块中。程序先创建 bean 容器,再调用 bean 容器的 `getBean()` 方法来获取 Spring 容器中的 bean。上面讲到该系统中需要使用数据库连接池,为了灵活地配置,我们通常把与数据库连接的参数写在在文本文件中而不是硬编码在代码里。该系统 web 模块的入口是 web 容器 tomcat,所以在 tomcat 中配置 spring,让 spring 读取数据库配置文件帮我们生成需要的 jdbc datasource 对象。

(5) maven (项目依赖管理工具)

它是一个项目构建管理工具,用于告诉编译器项目中各文件之间的依赖关系等。pom.xml 文件是 maven 项目的入口,这个 xml 文件指定了要构建该项目的几乎所有配置信息。maven 的仓库分为本地仓库和远程仓库。本地仓库是 maven 在我们本机设置的仓库目录,默认目录为当前用户目录 `/.m2/repository/`。远程仓库即默认的远程中央仓库,为互联网中的某台专用服务器。maven 的工作原理为读取我们在 pom.xml 中指定的依赖项目坐标(包含开发商,项目名称,版本号等),maven 从远程中央仓库下载得到我们需要的文件并存放到本地仓库中,供本地的若干个项目共享,实现了一份依赖随意共用的灵活性。当多个项目具有共同的多个依赖时,可以配置一个依赖关系在父文件中,通过继承实现复用,该系统就用到了这一特性。

(6) commons-fileupload (文件上传的后台处理框架)

它用于文件上传的后台处理。文件上传涉及到大量的二进制流数据处理，使用该工具可显著减少代码量。

(7) jsoup (网页解析工具)

它是一个网页解析工具，将在论文 5.1.2 章节进行介绍。

(8) ikalyzer (中文分词工具)

它是一个中文分词工具，将在论文 5.2.1 章节进行介绍。

(9) 还有一些其他常用的 tomcat 网站容器、jsp/servlet 技术、Junit 单元测试、log4j 日志记录等工具等不再介绍。

3 需求分析

3.1 功能需求

该系统是为了弥补人工判别网页的繁琐低效与机械评判网页的不够灵活而开发的，所以要保障判别模块的判别效果。为了实现这一目的，数据的收集加工都要认真对待，每个阶段的需求见下。

数据收集任务量不小，靠人工去一个个地保存网页，比较麻烦且效率很低，所以考虑实现一个爬虫模块来辅助收集。该系统的机器学习是一个有监督的学习过程，所以收集到的网页要想用于训练还需添加标注，该系统也要实现方便加标注的需求。

机器学习接受的输入为多个带标签的向量，所以该系统要实现网页解析、中文分词、计算特征向量这些需求。为了得到更有效的特征向量，还需要降维与向量归一化。

判别模块接受网页文件、url 两种用户提交。判别结果要详实完备，输出结果应包括网页的标题、正文、分词结果、特征向量、判别结果。为了给用户一个参考依据，还要附带判别结果准确率估计这一参数。此外，为了让该系统用途更广，除了与人交互外，判别模块同时也提供了 web 服务，可以供其他代码跨平台批量调用。

3.2 界面需求

该系统有些模块有用户交互的过程，选用 web 页面来实现效果很好，因为它可以跨设备、跨操作系统来展现。

收集模块包含爬虫模块。设定完初始 url 之后，它就应该展现 Jsoup 解析后的标题、正文内容，然后等待用户判别。如果它是正样例网页，就可以点击“是”

按钮，存入到数据库存放正样例的表中。如果它不是正样例网页，就可以点击“否”按钮，存入到数据库存放反样例的表中。如果该网页内容偏少，无法确定，就可以点击“不清楚，跳过”按钮，忽略对该网页的处理。当写入数据库成功时，也需要有对话框进行反馈。

机器学习是一次性的过程，代码直接实现直接又简单，这部分不需要界面。

判别模块接受网页文件、url 两种用户提交，所以需要给出两个提交框。考虑到页面整洁且这两个提交框是二选一的情况，提交框并不直接显示，而是由用户选择哪一种输入后触发相应的模态框来交互。上文提到的 6 个输出参数都要醒目地以 h2 标题形式展现。

3.3 运行环境需求

需求 1: java 运行环境。这是显然的，要求版本为 jre 1.7 及以上。

需求 2: 电脑联网。该系统需要从互联网中抓取数据，所以必须联网。

需求 3: web 容器。该系统有 web 网页模块，所以需要部署在 web 容器中，推荐 tomcat 工具。

需求 4: 数据库。该系统一些信息需要持久化保存，所以只靠内存是不够的，推荐 mysql 数据库。

4 系统设计

该系统按照流程进行划分，分为数据收集模块、数据处理模块和最后的数据判别模块。该系统的总架构图见图 4-1。

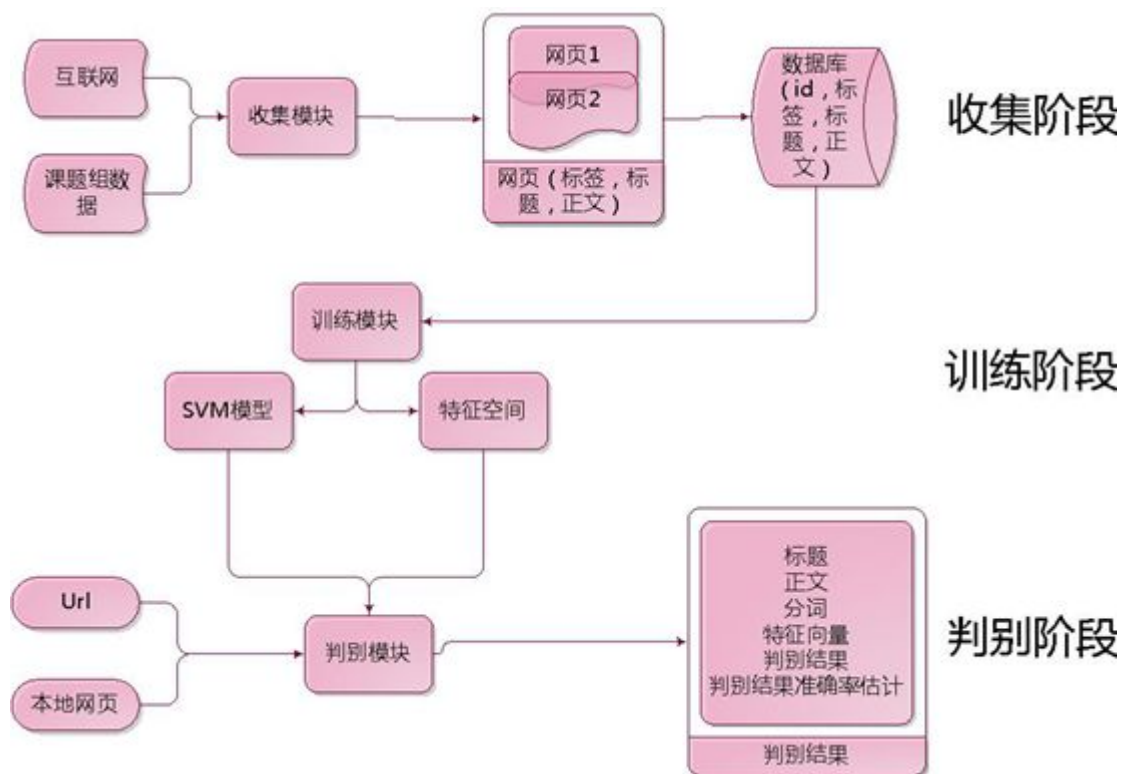


图 4-1 系统总架构图

数据收集模块用来收集网页，我们给这些网页贴上正标签或负标签，把它们分成两类，分别表示与程序设计相关和不相关的网页，作为最初的数据放到数据库保存起来。以后的数据处理都在这里开始进行，可以不再访问互联网。数据收集任务量不小，靠人工去一个个地保存网页，比较麻烦且效率很低，所以考虑实现一个爬虫模块来辅助收集。爬虫工作之前需要设定初始的爬取地址，且爬取到的结果需要我们再给它贴上标签。考虑到这些交互，数据收集模块可以通过 web 模块来实现。它的流程图见图 4-2。

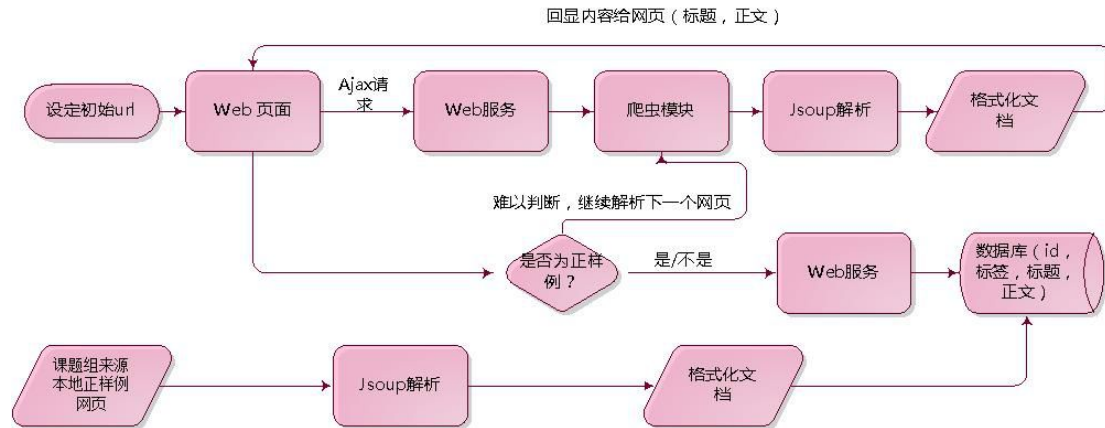


图 4-2 收集模块流程图

数据处理模块又可以分为预处理模块和训练模块。前者完成分词、统计、特征向量计算等所有前期预处理功能，后者通过对训练集的训练得到训练模型，交付给最后的数据判别模块使用。它的流程图见图 4-3。

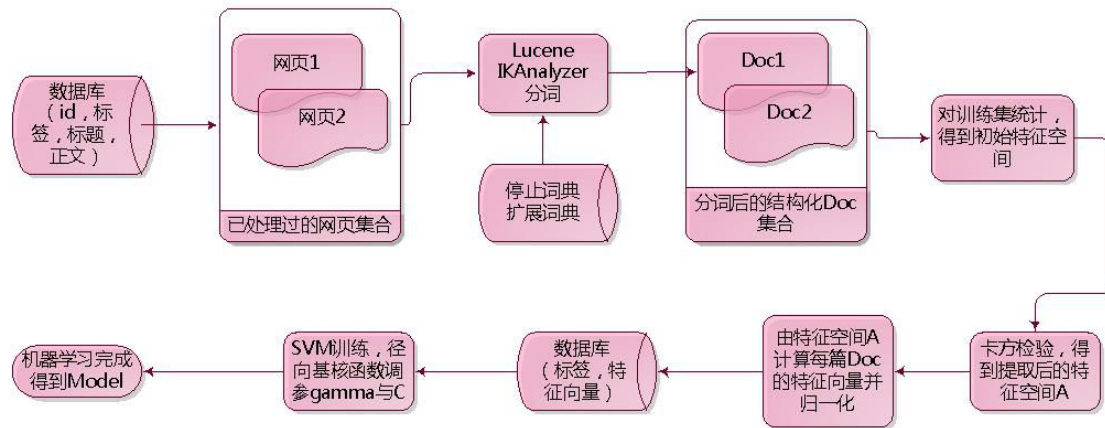


图 4-3 训练模块实现流程图

数据判别模块可接受本地网页文件和网页 url 两种输入，也可提供 api（应用程序接口）供其他程序调用，满足用户多样的需求。输出的判断结果有以下内容——解析出来的网页标题，网页正文，正文分词之后的词语列表，计算出来的特征向量，SVM 训练模型给出的判别结果以及该判别结果的估算准确率。该模块也由 web 实现。

5 系统实现

5.1 数据收集模块

5.1.1 web 界面实现

说明页面，由 bootstrap 导航条和三个子页面组成。见图 5-1。



图 5-1 收集模块 web 首页

“收集统计”子页面用来读取数据库，点击“读取收集系统数据库的统计信息”，统计出总共样本的数目，还有更为详尽的细粒度的统计信息。见图 5-2。



图 5-2 收集模块统计页面

“收集添加”子页面用来添加新的网页，它通过 web 服务与爬虫模块相连。设定完初始 url 之后，下面就会展现 Jsoup 解析后的标题、正文内容。见图 5-3。

[illegible]

图 5-3 收集模块添加页面

如果它是正样例网页，就可以点击“是”按钮，存入到数据库存放正样例的表中。

如果它不是正样例网页，就可以点击“否”按钮，存入到数据库存放反样例的表中。

如果该网页内容偏少，无法确定，就可以点击“不清楚，跳过”按钮，忽略对该网页的处理。

当写入数据库成功时，会有对话框进行反馈，见图 5-4。



图 5-4 收集模块添加网页成功提示

可以注意到它有一个 docId 编号，它用于唯一标记一篇文档，在数据库不同处理阶段的多个表中，docId 一样的文档就是同一篇文档。

由于以下三种原因造成的失败，该系统也会给出提示：

原因 1：网络通讯失败；

原因 2：该网站采取反爬虫策略，不能够爬取到有效的信息；

原因 3：数据库中已经有同样的网址记录，因为重复所以拒绝添加。

5.1.2 根据 URL 或本地文件解析网页

(1) Jsoup 简介

jsoup 是一个解析网页的框架，支持解析 url 和本地网页文件。

要解析 url 必须发送 http 请求，所以 jsoup 内部封装好了 http 客户端的功能。通过网络通讯能够拿到最原始的资源文件，此时里面有大量的 html 标签。

为了查找我们感兴趣的特定内容，就需要使用 DOM 或 CSS 选择器来查找、取出数据。jsoup 提供了一系列函数，可以像使用 jquery 一样方便地根据标签名或标签属性来抓取数据。

(2) Jsoup 使用

需要注意的是，我们平时上网用的浏览器，除了根据用户输入的 url 完成通信，还要解析执行 javascript 代码。

浏览器在发送 http 请求时，会在头部里面加入很多信息，比如 Cookie、userAgent 等。所以网站可以根据这些信息来确定这个请求是正常的用户请求还是爬虫机器请求，对于后者，为了减轻网站压力服务器通常不予回应，所以该系统在使用 jsoup 工具时会附上猎豹浏览器的真实 userAgent，降低失败率。

浏览器在拿到网页以后，会执行 java script 代码，有一些代码会再次让浏览器发送请求，拿到一些内容来展现在网页上。因为 jsoup 不能执行 java script 代码，所以会出现它拿到的内容少于真实内容的情形。

中文网页的流行编码集有“utf-8”、“gb2312”“gbk”等，网页的编码声明一般放在<meta charset=“xxx”>标签中，所以 jsoup 会首先扫描编码信息，再进行内容解析。

5.1.3 数据库设计与实现

数据收集模块对应两张表，`collect_positive_table` 与 `collect_negative_table`，分别存放正样例网页和负样例网页。它们的设计都是一样的，见图 5-5。

名	类型
no	int
id	int
title	varchar
isURL	bit
URL	varchar
isLocal	bit
path	varchar
content	text
time	datetime

图 5-5 `collect_positive_table` 与 `collect_negative_table` 表字段设计

`collect_positive_table` 表中各列的含义依次是，本表中的文档编号，文档 id，文档的标题，该文档是否根据 url 获得，url 的值，该文档是否从本地获得，本地路径的值，文章的内容，数据库中这条记录的修改时间。

数据预处理模块会关注这个表中的 id，title 与 content 这三列内容。

数据预处理模块对应两张表，`token_positive_table` 与 `token_negative_table`，它们分别对应 `collect_positive_table` 与 `collect_negative_table` 表中处理过后的内容。它们的设计见图 5-6。

名	类型
no	int
id	int
title	varchar
tokenList	text
time	datetime

图 5-6 `token_positive_table` 与 `token_negative_table` 表字段设计

`token_positive_table` 表中各列的含义依次是，本表中的文档编号，文档 id，文档的标题，文章正文分之后的分词列表，数据库中这条记录的修改时间。

卡方检验模块会批量读取这两张表的内容，然后得到抽取后的特征空间，存

入 non_relational_table 这张表中。non_relational_table 这张表就是为了持久化存储抽取后的特征空间，也就是一个 List 的字符串形式。因为数据只有一条，所以 non_relational_table 这张表模仿非关系数据库，设计见图 5-7。

名	类型
► theKey	varchar
theValue	text
time	datetime

图 5-7 non_relational_table 表字段设计

非关系型数据库存储的主要内容就是 key-value 这样的键值对形式，所以该系统尝试将两列的名字定为“key”与“value”。可能是 mySql 数据库的关键字原因，插入 sql 语句会报错，所以换成了“theKey”与“theValue”这两个名字。

在得到了抽取后的特征向量空间之后，就可以计算先前收集的正样例网页与反样例网页的特征向量了，并存入到 vector_positive_table 与 vector_negative_table 这两张表中。表的设计见图 5-8。

名	类型
► no	int
id	int
title	varchar
vector	text
time	datetime

图 5-8 vector_positive_table 与 vector_negative_table 表字段设计

vector_positive_table 表中各列的含义依次是，本表中的文档编号，文档 id，文档的标题，文档的特征向量，数据库中这条记录的修改时间。

需要注意的是，id 用于唯一标识一篇网页，所以多张表中同样的 id 对应的是同一个网页。这样做的好处在于开发过程中方便地调试与排错。

该系统的每一张表中都有一列的属性为 text，是 mySql 数据库用来存放长文本的专用格式，它支持的最大内容长度为 $2^{16}=65536$ 个字节。对于爬虫爬取到的网页的正文长度是未知的，有可能出现上万字。所以用 text 类型来保存网页内容的 content 字段。

5.1.4 数据库访问对象的单例实现

为了降低资源开销，提升程序响应性能，该系统使用了数据库连接池。该系统多个类对数据库都有读写行为，所以又在 `com.likeyichu.dal` 包下编写了 `Dao` 类，所有对数据库的访问都由此类完成。`dal` 的全写是 `data access layer`，即数据访问层。`dao` 的全写是 `data access object`，即数据库访问对象。`Dao` 类有一个代表着数据库连接池的 `dataSource` 成员，因为在程序的整个生命周期中，此成员只被初始化了一次且一直有效，所以可以把 `Dao` 类设计为单例模式，避免了 `new` 对象带来的开销。

单例模式是一种很有用的程序设计思想，经常用在日志记录的类 `Logger` 上。`Logger` 类需要将程序运行期间的异常信息或其他的重要信息记录在同一个文件中，所以必须避免多段代码对同一文件的同时写入。试想不采用单例模式，`Logger` 类的第一个对象 `obj1` 正在写入文件，而程序的其他地方又 `new` 了一个 `Logger` 类的对象 `obj2`，由于文件写入是互斥的，这时 `obj2` 对文件的任何内容写入都将是失败的。

同样地，想要避免在 `Dao` 类的外部调用该类的构造函数，就需要将构造函数的可见性设置为 `private`，`Dao` 对象的生成交给该类的具有 `public` 权限的方法来完成。该方法可以在类外被任意次调用，所以需要保证每次返回的 `dao` 对象都是同一个，也即返回的多个引用都指向内存中的同一片区域。考虑到此，将 `dao` 对象前面添加 `static` 关键字即可。

5.1.5 爬虫模块实现

爬虫模块接受数据收集模块 `web` 页面用户的输入作为初始 `url`，然后执行爬虫任务，工作步骤见下。

第 1 步：设置初始 `url`，放入爬取队列，设置两个读游标——用户读游标和爬虫读游标。用户读游标用来确定用户已经处理过的网页对应 `url` 位置，用户每读一个 `url` 该游标后移一个元素。

第 2 步：在爬取队列中取出一个 `url`，交给 `jsoup` 工具，解析出该页面的所有 `url`，筛选后放入爬取队列。

第 3 步：爬虫读游标后移一个元素。

第 4 步：若用户未读 url 个数大于 500，暂停工作；否则转入步骤 2 继续工作。

注意，这里的爬取队列并没有采用 queue 的数据结构。queue 是一个先进先出的数据结构，插入元素时会自动在尾部追加，读取元素时会自动从头部拿出。如果采用 queue 存放爬取的 url 数据，不需要再去维护用户读游标。但为什么不用 queue 而用 list 呢？这样做是有一个判重的考虑。例如，urlA 对应的页面中有指向 urlB 的超链接，而 urlB 对应的网页中又有指向 urlA 的超链接，这种情形并不少见。所以使用 list.contains() 方法来对待加入爬取队列的 url 集合做一个筛选，防止同一页面被多次爬取。

jsoup 解析网页后，回显给用户，让用户去给该网页添加标签，人机交互的速度是远远慢于爬虫爬取速度的，所以该系统设定了一个阈值，当爬取队列中用户未处理的 url 个数大于 500 时，爬虫模块暂停工作。若在某一次用户读取后检查到用户未处理的 url 个数不足 500，爬虫继续工作。

步骤 2 中提到的筛选操作，除了 url 判重筛选外，还有无效 url 筛选。有些 `` 标签中的超链接地址为相对于该网页在服务器中位置的相对位置，即不含有“http”前缀，jsoup 工具在处理这样的请求时会抛出资源找不到的异常。所以该系统需要事先剔除掉这些无效的链接。

一些网站会做反爬虫处理，虽然该系统已经添加了真实的 userAgent 这一 http 请求头部参数，但有一些反爬取是基于 cookie 或 session 机制的，此时爬虫模块就不能爬取到网页内容，所以这里还要有相应的容错处理。当这里发生异常时，把异常捕获掉，把爬虫读游标后移一个元素，继续工作。人们有时会有这样一个疑问，百度、搜狗等搜索引擎是怎么做到对这些网站内容的抓取呢？因为这些网站欢迎这些搜索引擎的抓取，根据 ip 地址反查域名可以确定爬虫请求是否来自于这些网站。

5.1.5 收集数据说明

(1) 正样例收集

正样例为与计算机编程相关的网页，此处举个例子明确一下。比如有这么一个句子：“一个图的连通分量是该图的极大连通子图；而一个连通图的生成树是极小连通子图。最小生成树算法有 Prime 与 Kruskal。原图为 $G=(V,E)$ ；生成树 $G_1=(V_1, E_1)$ ”，它讲的就是计算机编程中的算法。

url 收集部分以 csdn 博客、博客园、itEye 等专业 it 网站的相应页面依次作为爬虫初始 url 进行爬取，针对性强。正样例的另一部分来源是课题组提供的本地网页文件。这些网页是事先人工分类好的，所以与 url 部分的收集有差异。本地网页由系统指定文件位置，不需要经过爬虫模块，且由 jsoup 解析好后自动贴上正样例标签，省略了人工标注的交互。

(2) 负样例收集

负样例为与计算机编程无关的网页，比如“做自己喜欢的事，和自己喜欢的一切在一起”这句话就显然可以作为负样例收集。

负样例收集全部为 url 方式收集，由于收集量大，还邀请了其他同学帮忙收集。一些爬虫初始 url 分布见表 5-1。

表 5-1 负样例收集简表（部分）

爬虫初始页面	内容分类
东华大学首页	教育
新华网首页	新闻综合
搜狐体育	体育
淘宝网首页	零售
携程网首页	旅游
百度贴吧首页	话题讨论
起点中文网首页	网络文学
...	...

内容分类一列表示以该页面作为初始爬取页面后所能抓取到内容的主要类别。因为在域名 domain.com 网站中，各页面的绝大多数链接指向的网页还是属于该网站的。

从表 5-1 中可以看到，内容分类多样化，收集到的数据不会过于片面而影响

svm 的训练效果。

还需要指出的一点是，初始 url 设置为某个网站的页面，爬取到的内容并不全是该网站的内容。以起点中文网首页设为初始 url 为例，在爬取过程中就抓取到了大量的站外 url 链接，有和讯读书、4399 小游戏、驴妈妈旅游网、威锋网等。

另外，同一网站的不同二级域名，展现的内容也会很不一样。以淘宝网首页为例，它会含有淘宝女人 (<http://nvren.taobao.com>)、淘宝搭配 (<http://dapei.taobao.com>) 二级域名的超链接，这两个二级域名分别用来展现美妆技巧、搭配风格等内容。这一特性再次扩展了负样例收集的丰富性与多样性。

收集模块完成的样例收集统计见表 5-2。

表 5-2 样例收集统计表

正样例收集	url 来源个数	100
	本地网页个数	1600
	小计	1700
负样例收集	url 来源个数	1700
	本地网页个数	0
	小计	1700
总样例收集	总计	3400

收集模块插入到数据表中的内容格式见图 5-9。

no	id	title	isUR	URL	isLoc	path	content
1	1	动态规划优化之使用Map记忆化搜索 - 我是传奇	1	http://b	0	(Null)	动态规划优化之使用Map记忆化搜索 - 我是传奇
2	2	数据结构/基础算法 - 我是传奇	1	http://b	0	(Null)	数据结构/基础算法 - 我是传奇
3	16	zoi1383-----Binary Numbers	0	(Null)	1	D:\东华的	zoi1383-----Binary Numbers
4	17	poj4045树形dp_递归算法_新浪博客	0	(Null)	1	D:\东华的	poj4045树形dp_递归算法_新浪博客
5	18	zoi 1383 Presentation Error - 已回答 - 搜搜	0	(Null)	1	D:\东华的	zoi 1383 Presentation Error - 已回答 - 搜搜
6	19	zoi 3600 - 默、 - 博客频道 - CSDN.NET	0	(Null)	1	D:\东华的	zoi 3600 - 默、 - 博客频道 - CSDN.NET
7	20	[ZOJ3309]Search New Posts - lovekid的日记	0	(Null)	1	D:\东华的	[ZOJ3309]Search New Posts - lovekid的日记
8	21	简单题 HDU 1390 Binary Numbers_~小丫头	0	(Null)	1	D:\东华的	简单题 HDU 1390 Binary Numbers_~小丫头
9	22	ZOI 3305 拆点 最大流_JosiahChiu_新浪博客	0	(Null)	1	D:\东华的	ZOI 3305 拆点 最大流_JosiahChiu_新浪博客
10	23	Zoi 3600 Alice's present_A Crazy_新浪博客	0	(Null)	1	D:\东华的	Zoi 3600 Alice's present_A Crazy_新浪博客
11	24	ZOI3305 - 终究我要华丽逆袭 - 博客频道 - CSDN	0	(Null)	1	D:\东华的	ZOI3305 - 终究我要华丽逆袭 - 博客频道 - CSDN
12	25	Binary Numbers (向量, 迭代器) zoi 1383	0	(Null)	1	D:\东华的	Binary Numbers (向量, 迭代器) zoi 1383
13	26	zoi 3600 Taxi Fare - yzl_rex - 博客频道 - CSDN	0	(Null)	1	D:\东华的	zoi 3600 Taxi Fare - yzl_rex - 博客频道 - CSDN
14	27	zoi 3305 - I am coming ! - 博客频道 - CSDN	0	(Null)	1	D:\东华的	zoi 3305 - I am coming ! - 博客频道 - CSDN
15	28	hdu 1390 - ysc504的专栏 - 博客频道 - CSDN	0	(Null)	1	D:\东华的	hdu 1390 - ysc504的专栏 - 博客频道 - CSDN

图 5-9 收集模块插入到数据表中的内容格式

从 no 这一列可以很直观的看出本表中网页的编号，id 这一列的取值规则为先查询 collect_positive_table 表中 id 的最大值，记为 idMax1，再查询 collect_negative_table 表中 id 的最大值，记为 idMax2，则本条插入数据的 id 值为 $\max(\text{idMax1}, \text{idMax2}) + 1$ 。

5.2 数据处理模块

5.2.1 中文分词的 IKAnalyzer 实现

IK Analyzer 是国内的一个有名的开源分词器，它基于 java 实现，以另一个国际开源项目 Lucene（基于倒排实现的全文搜索引擎）为应用主体，结合了词典分词和文法分析算法的中文分词组件。它的官方使用文档中并没有说明该工具的分词准确率。

IK Analyzer 支持细粒度和智能分词两种切分模式。现在对这两种切分模式做一个对比。实验文本原文为“张三说的确实在理”。

智能分词结果：张三 | 说的 | 确实 | 在理

最细粒度分词结果:张三 | 三 | 说的 | 的确 | 的 | 确实 | 实在 | 在理
智能分词效果较好, 该系统使用它的智能分词模式。

IK Analyzer 2012 的架构图见图 5-10.



图 5-10 IK Analyzer 2012 架构图

由图 5-10 可以看到, IK Analyzer 有一个扩展配置管理单元, 在这里可以配置我们自己的停止词典与扩展词典。出现在停止词典中的词语将不会出现在分词的结果中, 这里主要放置出现频率很高但对反映内容特征帮助不大的词语, 如“的”、“了”、“那么”等部分语气词、连词和副词。该系统配置了一个停用词典 stopword.dic, 从网上搜集了一些常见的停用词放在了里面。

IK Analyzer 系统内部有一个默认词典, 作为中文分词的主要依据, 用户也可以配置一个扩展词典, 作为对默认字典的补充。该判别系统没有配置扩展词典。

IK Analyzer 对一个字符串进行分词的步骤如下。

第 1 步：创建 IKAnalyzer 对象 analyzer，对构造函数传参 true 表示使用智能分词模式。

第 2 步：将待分词字符串作为 StringReader 构造函数的参数得到 StringReader 对象 stringReader。

第 3 步：执行语句 `TokenStream tokenStream= analyzer.tokenStream("", reader)` 得到 tokenStream，需要注意该类来自 IK Analyzer 以外的 Lucene 框架。

第 4 步：执行语句 `tokenStream.getAttribute(CharTermAttribute.class)` 得到 CharTermAttribute 对象，它代表了分之后的一个词语^[10]。

第 5 步：`while (ts.incrementToken())` 循环用来遍历分词后的词语序列，就像 java 和 c++ 中 `iterator()` 遍历的思想一样。至此得到结果。

分词后，有些词语可能前面或后面还有空格，所以在放入数据库之前调用 `String.trim()` 函数来去掉空格。

分词后插入数据库中的效果见下图 5-11。

no	id	title	tokenList
1	1	动态规划优化之使用Map记忆化搜索	[动态, 规划, 优化, 使用, 记忆, 化, 搜索, 传奇, 博
2	2	数据结构/基础算法 - 我是传奇 - 博客	[数据结构, 基础, 算法, 传奇, 博客, 频道, 传奇, I
3	16	zsj1383-----Binary N	[博客, 频道, 热热, 热热, 热热, 目录, 视图, 摘要,
4	17	poj4045树形dp_盾算法_新浪博客	[树形, 盾, 算法, 新浪, 博客, 发, 博, 文, 博, 文, 盾
5	18	zsj 1383 Presentation Error - 已回	[回答, 搜, 搜, 问问, 网页, 图片, 视频, 音乐, 问问
6	19	zsj 3600 - 默、 - 博客频道 - CSDN.I	[默, 博客, 频道, 默, 默, 乖, 孩子, 追随, 大神, 脚
7	20	[ZOJ3309]Search New Posts - love	[日志, 网易, 博客, 网易, 新闻, 微, 博, 邮箱, 相册
8	21	简单题 HDU 1390 Binary Numbers.	[简单, 题, 小丫, 私房, 菜, 百度, 空间, 相册, 广
9	22	ZOJ 3305 拆点 最大流_JosiahChiu_亲	[拆, 大流, 新浪, 博客, 发, 博, 文, 博, 文, 订阅, 手
10	23	Zoj 3600 Alice's present_ACrazy_新	[新浪, 博客, 发, 博, 文, 博, 文, 订阅, 手机, 订阅
11	24	ZOJ3305 - 终究我要华丽逆袭 - 博客	[终究, 华丽, 逆, 袭, 博客, 频道, 终究, 华丽, 逆,

图 5-11 分词后插入数据库中的效果图

5.2.2 java 并发与同步

论文前面部分提到过，正样本中有 1600 个本地网页，它们是课题组其他同学手工判别的劳动结果。在数据处理模块需要将这些网页也放入到

collect_positive_table 表中。虽然本地 I/O（磁盘读取）速度并不会太慢，但与在内存中直接运行编译好的 java 代码相比，前者的速度还是显得太慢。为了提高收集效率，该系统采用三个线程并发处理。

c++中，想要执行多线程，只需要把想要并行的函数在定义时定义好形参列表即可，然后让系统函数进行回调。

而 java 中没有 c++中所谓的函数指针概念，并发编程时的代码量比 C++会多一点。想要并行运行的代码段必须放置在某一个类中，记为 A，则类 A 需要实现 Runnable 或 Callable 接口，这两个接口都有 run()函数，这里就是多线程执行的入口。执行下行代码即可在 jvm 中开辟一个新线程开始执行。

```
Future<?> java.util.concurrent.ExecutorService.submit(A task)
```

该系统中并发收集本地网页的类为 ConcurrentCollect，它有 List<File> fileList; 与 Queue<LocalPage> LocalPageQueue;两个属性。fileList 指明了它要收集的文件列表，参数由构造函数传入。LocalPageQueue 是一个队列，由 ConcurrentCollect 对象放入解析好的网页，由 DAL（数据库访问层）取出数据执行数据库插入操作。很显然，这里是一个生产者消费者模型的实现。

java.util.concurrent.ConcurrentLinkedQueue 类是 Queue 接口的一个线程安全实现，该系统使用它作为 ConcurrentCollect 类 LocalPageQueue 属性的实现。

在本地网页解析完之后，需要通知 DAL 来消费数据。多线程中规定不同代码执行顺序的手段叫同步。该系统采用信号量实现。

java.util.concurrent.Semaphore 类是 java 中的信号量，构造函数中可以传入一个整数，它代表着系统中可用资源的数量^[1]。Semaphore.acquire()函数是关键，它令该信号量代表的可用资源数量减一，当值小于或等于 0 不够减时，线程阻塞，直到可用资源数量大于 1 时该线程重新被操作系统恢复。与此相反，Semaphore.release()函数将 Semaphore 的值加 1。因此，该系统将 semaphore 初始值设为 0，DAL 对象首先执行 Semaphore.acquire()方法，三个 ConcurrentCollect 对象在并发完成任务后执行 Semaphore.release()方法。

系统运行过程中，可以在控制台窗口清晰地看到线程池 pool-2 中的三个线程在并发执行，见下图 5-12。

```
[pool-2-thread-3] INFO com.likeyichu.jsoup>AboutJsoup - 得到一篇本地文档, 标题: pku 2594 Treasure E
[pool-2-thread-2] INFO com.likeyichu.jsoup>AboutJsoup - 得到一篇本地文档, 标题: poj 1556_ccsu_zqx_
[pool-2-thread-1] INFO com.likeyichu.jsoup>AboutJsoup - 得到一篇本地文档, 标题: P0J3411-Paid Roads
[pool-2-thread-3] INFO com.likeyichu.jsoup>AboutJsoup - 得到一篇本地文档, 标题: poj2594_刀仔_百度空间
[pool-2-thread-2] INFO com.likeyichu.jsoup>AboutJsoup - 得到一篇本地文档, 标题: poj 1556 The Doors
[pool-2-thread-2] INFO com.likeyichu.jsoup>AboutJsoup - 得到一篇本地文档, 标题: pku 1556 The Doors_
[pool-2-thread-3] INFO com.likeyichu.jsoup>AboutJsoup - 得到一篇本地文档, 标题: poj 2594 Treasure E
```

图 5-12 线程池 pool-2 中的三个线程并发执行

5.2.3 文档结构化存储的数据结构实现

网页是该系统处理的重点，所以与网页有关的类有 WebPage 与 Doc。前者只有标题、正文、标签三个属性，用来保存较为原始的数据。后者由 WebPage 对象的数据加工而成，它的 UML（统一建模语言）图见图 5-13。该图由 starUml 软件通过解析已经编写好的 java 代码逆向自动生成。

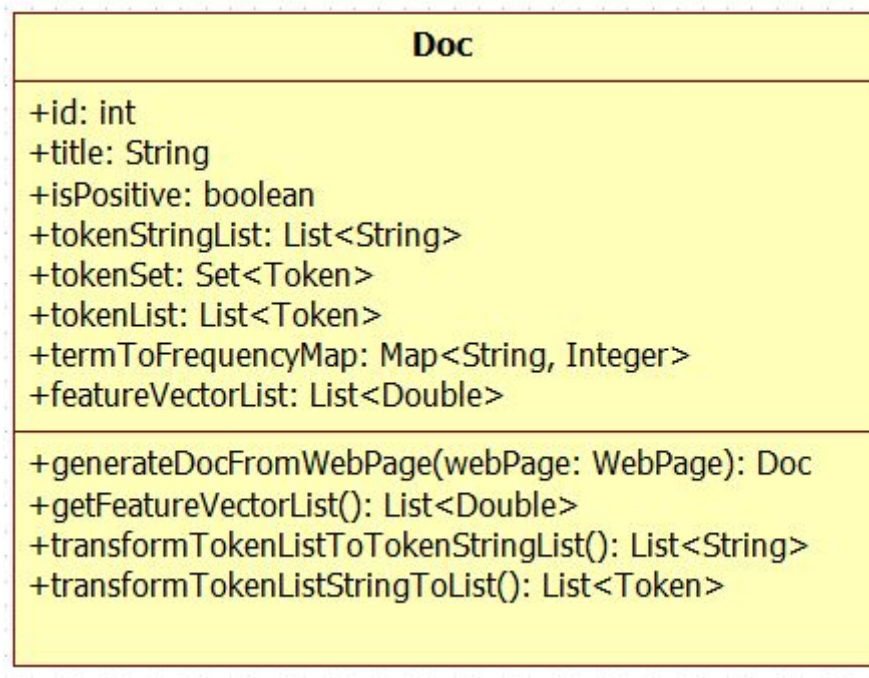


图 5-13 Doc 类 UML 图

现在对该类属性进行说明。

id: 该文档的 id，同一个网页在信息处理的各个阶段，id 始终不变。

title: 该文档的标题。

isPositive: 用于标记该文档是正例还是反例。

tokenStringList: 按顺序存放文档正文的分词序列，为了在卡方检验阶段得到词语出现的频度，所以这里不会去掉重复的词语，也即某个词语在文章中出现了两次，那么 List 中该词语也会出现两次。

tokenSet: 存放 token 的 Set。有了 tokenList，为何要再设置一个 tokenSet 呢？在卡方检验阶段，要判断某个词是否在一个网页中出现。而 set 接口的一个数据结构实现 HashSet 的优势就是能够根据哈希散列实现快速查找。在 list 结构中的查找是顺序查找，平均查找长度为 list 大小的一半。假设一个网页有 2000 个词语，那么使用了 set 后这里的查找速度就可以提升 1000 倍！

tokenList: 存放 token 的 List。token 为 Token 类的对象，在卡方检验章节给出描述。

termToFrequencyMap: 这是一个用来存放键值对的集合，键为词语，值为该词语在本文的出现频度。用来实现特征向量的归一化。

featureVectorList: 用来存放计算后的特征向量，向量的维数在卡方检验阶段给出，每一维的数据均为 Double 类型。

下面对该类的方法进行说明。

generateDocFromWebPage(): 从 collect_positive_table 与 collect_negative_table 表中读取信息赋值给对象 WebPage，再将 WebPage 对象的标题、正文、标签三个属性相应地赋值给 Doc。有了 WebPage 作为中转，逻辑上更为清楚。

getFeatureVectorList(): 用来计算特征向量。

transformTokenListToTokenStringList(): 此方法在判别模块中被调用。判别模块从数据库中读出特征空间，是 String 形式，此方法将其转化为 List<Sting> 形式，也即对 “[示例 1, 示例 2]” 字符串去掉首尾的中括号后按照逗号进行切割，得到一个个词语后插入 List<Sting>。

transferTokenListStringToList(): 与 transformTokenListToTokenStringList() 方法类似，也是对字符串进行切分。不同之处在于它将切割出来的词语传递给 Token 类的构造函数，最后得到的是 List<Token> 而不是 List<Sting>。

5.2.4 卡方检验实现

定义一个类 `Token` 用于描述一个词语的内容和 `a,b,c,d` 四个属性，这四个属性在卡方检验中的作用已经在上个章节中给出。它的 UML 图见图 5-14。

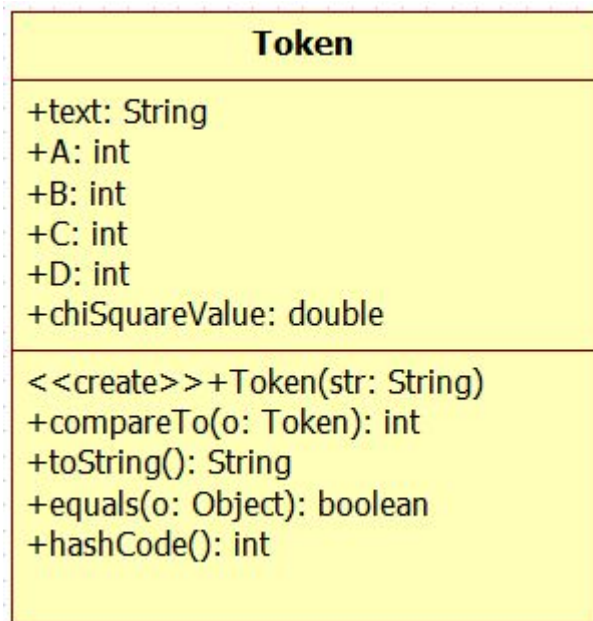


图 5-14 `Token` 类 UML 图

它的 6 个属性的作用见名知意，不再详述。下面讲一下它的 5 个方法。

`Token()`: 这是带 `String` 类型参数的构造函数，用来把参数值赋值给 `text`，表示该词语的内容。

`compareTo()`: 此方法重写 `Object` 类的方法，用来定义 `Token` 对象之间的大小比较规则。java 中有 `<Token> void java.util.Collections.sort(List<Token> list, Comparator<? super Token> c)` 现成的泛型库函数来支持 `List` 中自定义对象之间的排序，我们只要重写 `compareTo()` 方法就可以了。在 `sort()` 函数的第二个形参中传入 `Collections.reverseOrder()`，就可以实现倒序排序。

`toString()`: 此方法重写 `Object` 类的方法，用来定义把对象转换为 `String` 字符串的规则。因为字符串能被人直接看懂，所以这里输出它的各个属性方便调试。

`equals()`: 此方法重写 `Object` 类的方法，用于实现自定义类的对象之间是否相等的判断。此处设定相等的规则为若两个 `token` 对象的 `text` 属性相等，则它们相等。

hashCode(): 此方法重写 Object 类的方法。此处定义为 text.hashCode(), java 默认实现了 String 类型的 hashCode()方法。

必须同时重写 equals()与 hashCode()方法,才能正确使用泛型集合 Set<Token> 的自动去重功能。

TokenStatistics 类用来存储所有网页中 token 对象的统计信息,为卡方检验提供数据支持。它的 UML 图见图 5-15。

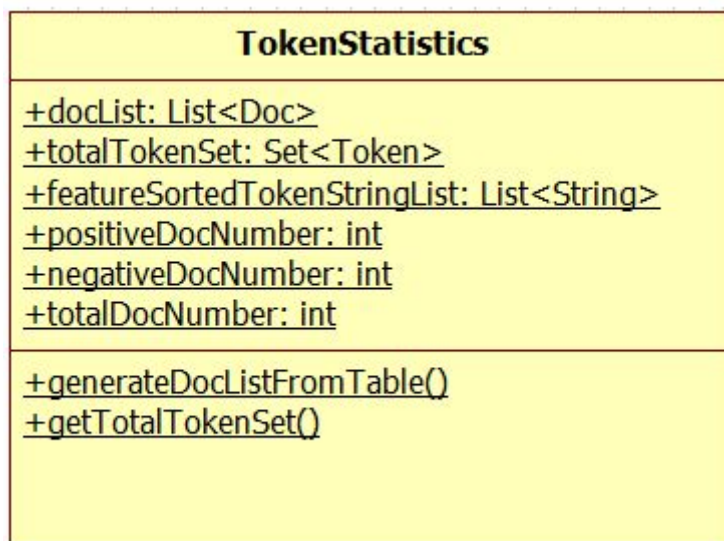


图 5-15 TokenStatistics 类 UML 图

它的 6 个属性依次说明如下。

docList: 存放从 token 表中拿到的所有 Doc。

totalTokenSet: 所有样本中的 token 对象的集合。

featureSortedTokenStringList: 选取后的特征词语。

positiveDocNumber、negativeDocNumber、totalDocNumber 分别表示正样本数、负样本数与总样本数,依次对应着卡方检验中的(a+c)、(b+d)与 n。

它的三个方法说明如下。

generateDocListFromTable(): 从 token 表中拿到所有的 doc,此时该系统只关注 doc 的标签、id、分词序列三个属性,不再关注标题。

getTotalTokenSet(): 遍历拿到的 doc 对象的 token,将它们放入此 totalTokenSet 集合。此方法中会给出所有样例的词语总个数,得到的 set 的大小为所有样例的词语总种类数,前者要远大于后者。

FeatureSelect 类从 TokenStatistics 对象中拿到信息，进行计算，它的 UML 图见图 5-16。

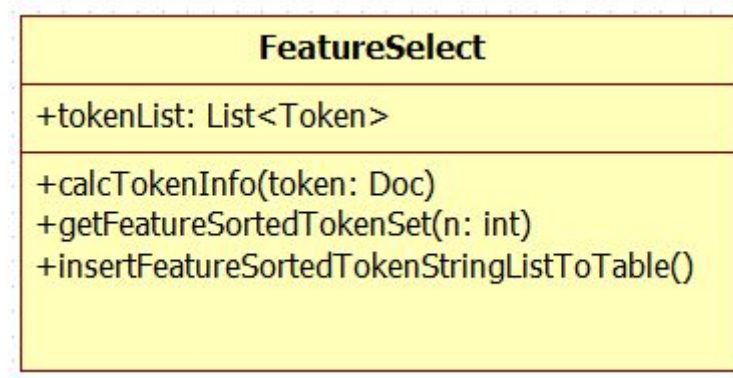


图 5-16 FeatureSelect 类 UML 图

它的属性只有一个，用来接收 TokenStatistics 对象产生的 tokenList。

calcTokenInfo() 方法是重点。开始拿到的 token 只有 text 属性，此方法将遍历所有的样本，为每一个 token 对象的 a、b、c、d 属性进行赋值。

getFeatureSortedTokenSet() 方法用来产生降维后的特征空间，传入参数 n 表示取卡方值最大的前 n 个词语作为新的特征空间。

insertFeatureSortedTokenStringListToTable(): 将提取后的特征空间插入数据库，它将调用 Dao 对象的一个方法，由数据访问层完成具体实现。

该系统选取排序后的前一千个词语作为特征空间。选取出来排名前十的词语是：[偶数，不放心，连通分支，数学题，整除，操守，辛子，升序，菜价，英杰]。

以词语“偶数”为例进行观察，它所对应的 token 对象的卡方检验信息为：text=偶数，A=27，B=0，C=1673，D=1700，chiSquarevalue:23840.0。

这个结果与我们的预期较为符合。

5.2.5 特征向量计算与归一化

在卡方检验阶段提取出来 1000 个得分最高的词语之后，我们就得到了特征空间。

有了提取后的特征空间，就可以批量的计算网页的特征向量。为了方便说明特征向量计算过程，假设该系统的特征空间为[你好，算法，我的，大学，小明]。有以下 A、B 两个句子。A 为“你好，我的名字是小明，小明这个名字很常见”，

表 5-3 句子的特征向量计算示意

特征空间	你好	算法	我的	大学	小明
A 句特征向量	1	0	1	0	2
B 句特征向量	1	1	0	1	0

该系统将数据归一化到[0,1]区间中。归一化也成为标准化处理^[12], 该系统采用向量求模标准化。以向量 $\mathbf{a} = (1, 2, 3)$ 为例, 计算得模为

$$|\mathbf{a}| = \frac{1^2 + 2^2 + 3^2}{\sqrt{1^2 + 2^2 + 3^2}} = 3.742 \quad (5-1)$$

则

$$\mathbf{a}_{new} = (\frac{1}{3.742}, \frac{2}{3.742}, \frac{3}{3.742}) \quad (5-2)$$

最终将数据保存在数据库当中，保存格式见下图 5-17。

[illegible]

图 5-17 向量归一化后存入数据库示意

5.2.6 libsvm 工具箱的 javaAPI 使用

libsvm 是台湾大学林智仁教授等开发的一个支持向量机软件包，除了可以在 matlab 中使用以外，还提供了 java 程序接口^[13]。下面讲一下 java 接口的使用。

(1) 工具框架

通过 maven 工具得到 libsvm-3.17.jar，它的结构见图 5-18。

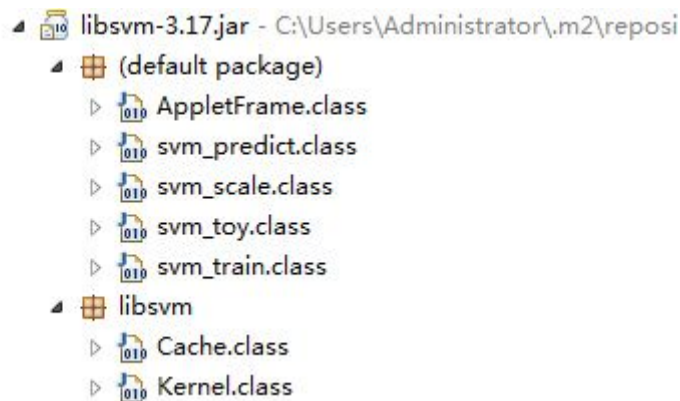


图 5-18 libsvm-3.17.jar 结构示意图

default package: 这里是官方封装出来的一些的类，它们都有 `main()` 函数，可直接作为小工具使用。但不方便的是这些类不在任何 `package` 中，所以该系统的工程文件无法引用，解决办法是在源码中强行加入 `package` 声明语句。

libsvm package: 核心文件。

`svm_train` 的输入为训练集，输出为得到的训练模型。

`svm_predict` 的输入为待预测数据，输出为得到的预测结果。

它们的输入输出都是文件。

(2) 提供训练集

libsvm 官方提供了一些训练集：

<http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/>

其中有一个 `breast-cancer` 训练集，见图 5-19。



图 5-19 libsvm 官网提供的 breast-cancer 训练集

从图中可以看到，它格式为：

label featurIndex1:value1 featurIndex2:value2 ...

在一些情况下，我们会对训练数据进行缩放，缩放的目的在于：

目的 1：避免一些特征值范围过大而另一些特征值范围过小；

目的 2：避免在训练时为了计算核函数而计算内积的时候引起数值计算的困难。

因此，通常将数据缩放到 $[-1,1]$ 或者是 $[0,1]$ 之间。

libsvm 提供了 `svm_scale` 类来进行缩放。

缩放参数有 `-l lower : x scaling lower limit (default -1)` 与 `-u upper : x scaling upper limit (default +1)`。

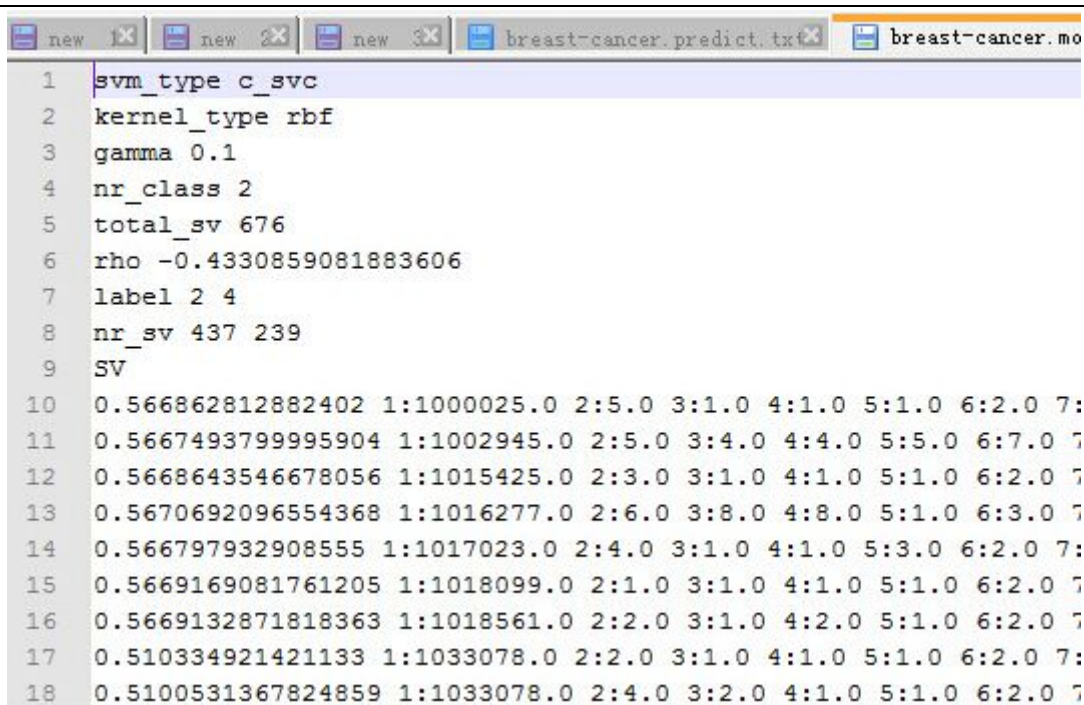
将 breast-cancer 训练集缩放后的部分结果见图 5-20。

```
4.0 1:-0.8318433374950909 2:0.3333333333333326 3:-0.7777777777777778 4:-0.
4.0 1:-0.8318403504090852 2:1.0 3:1.0 4:0.5555555555555556 5:0.111111111111
2.0 1:-0.8316748658443667 2:-0.3333333333333337 3:-1.0 4:-1.0 5:-1.0 6:-0.
2.0 1:-0.8316612746030405 2:-1.0 3:-1.0 4:-1.0 5:-1.0 6:-0.7777777777777778
4.0 1:-0.8313864626905116 2:-0.1111111111111116 3:-0.1111111111111116 4:-
2.0 1:-0.8312720572964916 2:-1.0 3:-0.7777777777777778 4:-0.77777777777777
2.0 1:-0.8312542841347573 2:-0.7777777777777778 3:-1.0 4:-1.0 5:-1.0 6:-0.7
4.0 1:-0.8307935261183705 2:0.7777777777777777 3:0.7777777777777777 4:1.0 5
```

图 5-20 将 breast-cancer 训练集缩放后的部分结果

(3)得到训练模型 model

训练模型 model 被 libsvm 转存为文本文件，内容见图 5-21。



```

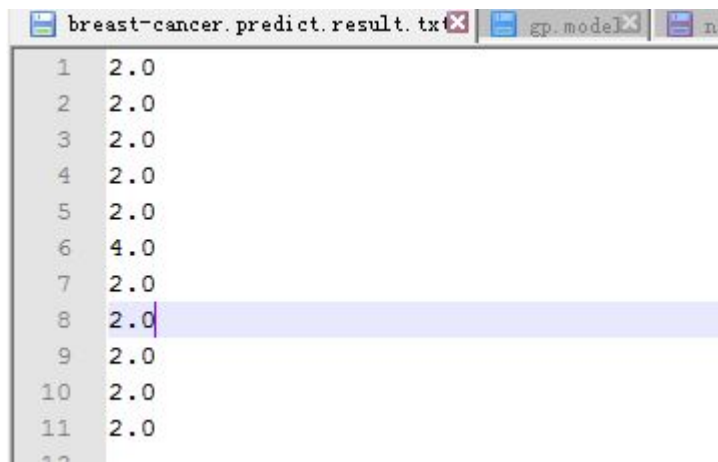
1 svm_type c_svc
2 kernel_type rbf
3 gamma 0.1
4 nr_class 2
5 total_sv 676
6 rho -0.4330859081883606
7 label 2 4
8 nr_sv 437 239
9 SV
10 0.566862812882402 1:1000025.0 2:5.0 3:1.0 4:1.0 5:1.0 6:2.0 7:
11 0.5667493799995904 1:1002945.0 2:5.0 3:4.0 4:4.0 5:5.0 6:7.0 7:
12 0.5668643546678056 1:1015425.0 2:3.0 3:1.0 4:1.0 5:1.0 6:2.0 7:
13 0.5670692096554368 1:1016277.0 2:6.0 3:8.0 4:8.0 5:1.0 6:3.0 7:
14 0.566797932908555 1:1017023.0 2:4.0 3:1.0 4:1.0 5:3.0 6:2.0 7:
15 0.5669169081761205 1:1018099.0 2:1.0 3:1.0 4:1.0 5:1.0 6:2.0 7:
16 0.5669132871818363 1:1018561.0 2:2.0 3:1.0 4:2.0 5:1.0 6:2.0 7:
17 0.510334921421133 1:1033078.0 2:2.0 3:1.0 4:1.0 5:1.0 6:2.0 7:
18 0.5100531367824859 1:1033078.0 2:4.0 3:2.0 4:1.0 5:1.0 6:2.0 7:

```

图 5-21 model 文件内容示意

(4)进行预测

这里只是为了说明用法，简单起见我们直接把训练集作为待预测集，结果见图 5-22。



```

1 2.0
2 2.0
3 2.0
4 2.0
5 2.0
6 4.0
7 2.0
8 2.0
9 2.0
10 2.0
11 2.0
12

```

图 5-22 libsvm 的预测结果

预测集与训练集一样，每一行都要有 label 标签。当我们用已知的结果来检验预测的准确性时，那么下行输出的准确性统计就是真实的：

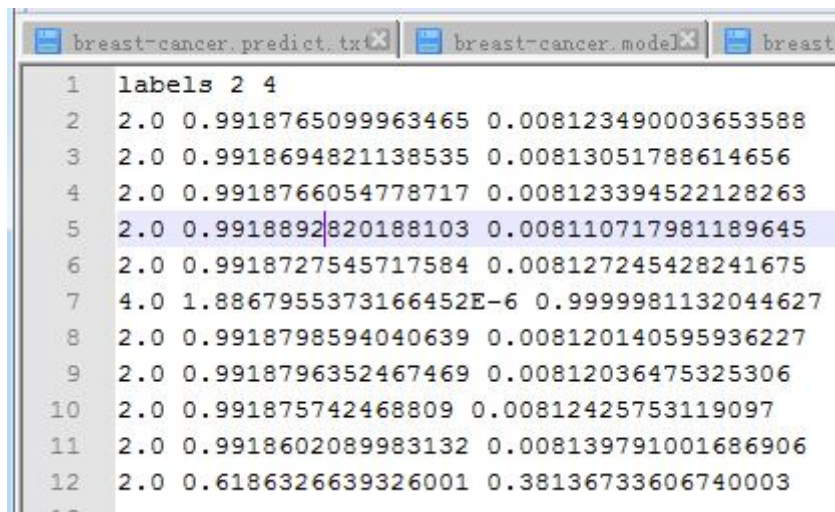
Accuracy = 90.9090909090909% (10/11) (classification)

当我们预测未知数据时（大多数情况都属于这一种），那么这行信息忽略就可以了。

(5)-b 参数

此参数表示是否携带准确性估计信息。若要使用，训练阶段与预测阶段都需要携带-b 参数。

此时，预测结果的输出格式见图 5-23。



1	labels 2 4		
2	2.0	0.9918765099963465	0.008123490003653588
3	2.0	0.9918694821138535	0.00813051788614656
4	2.0	0.9918766054778717	0.008123394522128263
5	2.0	0.9918892820188103	0.008110717981189645
6	2.0	0.9918727545717584	0.008127245428241675
7	4.0	1.8867955373166452E-6	0.9999981132044627
8	2.0	0.9918798594040639	0.008120140595936227
9	2.0	0.9918796352467469	0.00812036475325306
10	2.0	0.991875742468809	0.00812425753119097
11	2.0	0.9918602089983132	0.008139791001686906
12	2.0	0.6186326639326001	0.38136733606740003

图 5-23 携带“-b”参数的 libsvm 预测结果

第一行是标签的集合。

从第二行开始，格式为： 预测结果 该结果是正样本的概率 该结果是负样本的概率。

也就是说，当预测结果为正样本时，第二列永远都大于等于 0.5。

5.2.7 使用 libsvm 进行训练

该系统得到的训练集 train.txt 文件大小为 29,891KB。其中的数据格式见图 5-24。

```

516 1 1:0.0 2: 0.0 3: 0.0 4: 0.0 5: 0.0 6: 0.0 7: 0.0 8: 0.0 9: 0.0 10: 0.0 11: 0.0 12: 0.0 13: 0.0
517 1 1:0.0 2: 0.0 3: 0.0 4: 0.0 5: 0.0 6: 0.0 7: 0.0 8: 0.0 9: 0.0 10: 0.0 11: 0.0 12: 0.0 13: 0.0
518 1 1:0.0 2: 0.2773500981126145 3: 0.0 4: 0.0 5: 0.0 6: 0.2773500981126145 7: 0.2773500981126145 8
519 1 1:0.0 2: 0.0 3: 0.0 4: 0.0 5: 0.0 6: 0.0 7: 0.0 8: 0.0 9: 0.0 10: 0.0 11: 0.0 12: 0.0 13: 0.0
520 1 1:0.0 2: 0.0 3: 0.0 4: 0.0 5: 0.0 6: 0.0 7: 0.0 8: 0.0 9: 0.0 10: 0.0 11: 0.0 12: 0.0 13: 0.0
521 1 1:0.0 2: 0.0 3: 0.0 4: 0.0 5: 0.0 6: 0.0 7: 0.0 8: 0.0 9: 0.0 10: 0.0 11: 0.0 12: 0.0 13: 0.0
522 1 1:0.0 2: 0.0 3: 0.0 4: 0.0 5: 0.5 6: 0.0 7: 0.0 8: 0.0 9: 0.0 10: 0.0 11: 0.0 12: 0.0 13: 0.0
523 1 1:0.0 2: 0.0 3: 0.0 4: 0.0 5: 0.98058067569092 6: 0.0 7: 0.0 8: 0.0 9: 0.0 10: 0.0 11: 0.0 12:
524 1 1:0.0 2: 0.0 3: 0.0 4: 0.0 5: 0.4714045207910316 6: 0.0 7: 0.0 8: 0.0 9: 0.0 10: 0.0 11: 0.0 1
525 1 1:0.0 2: 0.0 3: 0.0 4: 0.0 5: 0.9428090415820632 6: 0.0 7: 0.0 8: 0.0 9: 0.0 10: 0.0 11: 0.0 1
526 1 1:0.0 2: 0.0 3: 0.0 4: 0.0 5: 0.0 6: 0.0 7: 0.0 8: 0.0 9: 0.0 10: 0.0 11: 0.0 12: 0.0 13: 0.0
527 1 1:0.3333333333333333 2: 0.0 3: 0.0 4: 0.0 5: 0.0 6: 0.0 7: 0.0 8: 0.0 9: 0.0 10: 0.0 11: 0.0 1
528 1 1:0.0 2: 0.0 3: 0.0 4: 0.0 5: 0.0 6: 0.0 7: 0.0 8: 0.0 9: 0.0 10: 0.0 11: 0.0 12: 0.0 13: 0.0
529 1 1:0.0 2: 0.0 3: 0.0 4: 0.0 5: 0.0 6: 0.0 7: 0.0 8: 0.0 9: 0.0 10: 0.0 11: 0.0 12: 0.0 13: 0.0
530 1 1:0.0 2: 0.0 3: 0.0 4: 0.0 5: 0.0 6: 0.0 7: 0.0 8: 0.0 9: 0.0 10: 0.0 11: 0.0 12: 0.0 13: 0.0
531 1 1:0.0 2: 0.0 3: 0.0 4: 0.0 5: 0.0 6: 0.0 7: 0.0 8: 0.0 9: 0.0 10: 0.0 11: 0.0 12: 0.0 13: 0.0
532 1 1:0.0 2: 0.0 3: 0.0 4: 0.0 5: 0.0 6: 0.0 7: 0.0 8: 0.0 9: 0.0 10: 0.0 11: 0.0 12: 0.0 13: 0.0

```

图 5-24 训练集的部分数据，说明格式用

程序控制台中可以看到 libsvm 在训练过程中自己动态参数调优的过程。第一次优化后的参数为（保持控制台原样输出格式）：

optimization finished, #iter = 1356

nu = 0.9882352941176469

obj = -2666.344483635098, rho = 1.2899248898029327

nSV = 2689, nBSV = 2687

Total nSV = 2689

表 5-4 给出了这些参数的说明。

表 5-4 控制台输出参数说明

参数	说明
#iter	迭代次数
nu	SVC 参数
obj	二次规划的最小值
rho	决策函数常数项
nSV	支持向量数
nBSV	边界上支持向量数
Total nSV	支持向量总数

训练完成后得到 model 文件，大小为 26,577KB。该文件也是文本文件，它的一些内容见图 5-25。



```
breast-cancer.predict.result.txt gp.model new
1 svm_type c_svc
2 kernel_type rbf
3 gamma 0.001
4 nr_class 2
5 total_sv 3210
6 rho 1.2984605699098033
7 label 1 -1
8 probA -229.61897766629957
9 probB -223.3068450333695
10 nr_sv 1600 1610
11 SV
12 1.0 1:0.0 2:0.0 3:0.0 4:0.0 5:0.0 6:0.0 7:0.0 8:0.0
13 1.0 1:0.0 2:0.0 3:0.0 4:0.0 5:0.0 6:0.0 7:0.0 8:0.0
```

图 5-25 训练得到的 model 文件，部分数据用来展示格式

该 model 文件的一些参数说明见表 5-5。

表 5-5 model 文件参数说明

参数名	参数值	说明
svm_type	c_svc	支持向量机类型
kernel_type	rbf	核类型为径向基函数
gamma	0.001	惩罚系数
nr_class	2	分类数目
total_sv	3310	所有的支持向量数目
rho	1.2984605699 098033	决策函数中的常数项的相反数（-b）
label	1 -1	标签种类
probA	-229.61897766 629957	此参数用于表示预测结果的准确概率
probB	-223.30684503 33695	此参数也用于表示预测结果的准确概率
nr_sv	1700 1610	正反样例的支持向量的数目

5.2.8 机器学习效果评价

该系统采用 K 折交叉验证 ($k=10$) 来验证效果。该系统的收集样例为正样例 1700 个, 负样例 1700 个。该系统将这些样例按顺序分成 10 组, 依次标记为 A,B,C,D,E,F,G,H,I,J。也就是说每组样例都有 170 个正样本与 170 个负样本。

第一次以 B,C,D,E,F,G,H,I,J 的样本作为训练集, A 的样本作为测试集; 第二次以 A,C,D,E,F,G,H,I,J 的样本作为训练集, B 的样本作为测试集。以此类推, 总共进行 10 次。实验结果见表 5-6。

表 5-6 10 折交叉统计结果

训练集	测试集	准确率
B, C, D, E, F, G, H, I, J	A	84%
A, C, D, E, F, G, H, I, J	B	88%
A, B, D, E, F, G, H, I, J	C	81%
A, B, C, E, F, G, H, I, J	D	83%
A, B, C, D, F, G, H, I, J	E	73%
A, B, C, D, E, G, H, I, J	F	76%
A, B, C, D, E, F, H, I, J	G	82%
A, B, C, D, E, F, G, I, J	H	78%
A, B, C, D, E, F, G, H, J	I	72%
A, B, C, D, E, F, G, H, I	J	77%

平均这 10 次的实验结果, 得到平均分类准确率为 79.4%。

5.3 判别模块

判别系统提供用户交互与程序交互两种使用模式。该判别系统已发布至互联网, 地址为 <http://me.likeyichu.com/GraduationProjectUse/views/index.html>。欢迎

体验。

5.3.1 用户交互 web 界面实现

用户通过网页与判别系统交互。用户界面见图 5-26。



图 5-26 使用模块人机交互界面

用户可以提交本地网页文件，也可以提交一个 url。点击任何一个按钮，都会弹出一个模态对话框。以点击“判断本地网页文件”为例，效果见下图 5-27。



图 5-27 模态对话框-提交 Url

以点击“判断本地网页文件”为例，效果见下图 5-28。



图 5-28 模态对话框-提交文件

点击“选择文件”按钮，可以选择文件。以上传一个名为“图论总述”的技术博客网页为例，上传成功后点击“查看结果”按钮，显示效果见图 5-29。

标题

图论总述 - chuchus - 博客频道 - CSDN.NET

正文内容

图论总述 - chuchus - 博客频道 - CSDN.NET chuchus 目录视图 摘要视图 订阅 Markdown博 Spark+Scala, 这样学才够值! 图论总述 分类: 图论 2014-04-08 20:31 452人阅读 评论 (Edge)的集合。图的物理存储,有两种方法。1.邻接矩阵,就是二维数组,较直观,但不能答:至少要有 $(n-1)$ 条边。对于简单图而言至多有 $n*(n-1)/2$ 条边,此时即是完全图。二部图个邻接顶点分属不同的顶点集。匹配:二部图中没有公共顶点的边集。边数最多的匹配叫最大点。增广路:从未盖点出发,经非匹配边、匹配边、非匹配边……交替进行,最终以未盖点多一条边。匈牙利算法:不断寻找增广路,最终得到最大匹配。二部图最小覆盖:选择尽量部图的独立集:该集合中任意两点不相邻接。独立集中顶点个数=总顶点数-最大匹配中边的数。两点都邻接,但同一集合内的任意两点都不相邻,则称G为完全三部图。网络流-概念 容量C (记为 V_s),以及另一个顶点,称为汇点(记为 V_t);对于每一条弧 $\langle u, v \rangle \in A$,对应有一个

分词结果

[图论, 述, 博客, 频道, 目录, 视图, 摘要, 视图, 订阅, 博文, 大赛, 清新, 开启, 天天, 爱, 答题, 论, 收藏, 举报, 图论, 述, 图, 存储, 图, 通常, 表示, 顶点, 集合, 集合, 图, 物理, 存储, 两种, 储, 顶点, 连通, 图, 至少有, 多少, 答, 至少, 要有, 对于, 简单, 图, 而言, 至多, 此时, 即是, 完全, 意, 一条, 两个, 邻接, 顶点, 分属, 不同, 顶点, 集, 匹配, 二部, 图中, 公共, 顶点, 集, 数最多, 匹配, 顶点, 增广, 路, 从未, 盖, 出发, 经, 非, 匹配, 匹配, 非, 匹配, 交替, 进行, 最终, 盖, 结尾, 路, 法, 不断, 寻找, 增广, 路, 最终, 得到, 最大, 匹配, 二部, 图, 最小, 覆盖, 选择, 尽量, 少, 使得, 任意, 两点, 不相, 邻接, 独立, 集中, 顶点, 个数, 顶, 点数, 最大, 匹配, 数目, 图, 顶点, 集, 划, 分]

计算得到特征向量

[0.09549105211188455, 0.0, 0.12732140281584606, 0.031830350703961514, 0.0318303501338577435767, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.06366070140792303, 0.0, 0.06366070140792303, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.031830350703961514, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

svm模型判断结果

yes

判断结果的准确率

0.8527222772187009

图 5-29 点击“查看结果”按钮, 效果示意

5.3.2 文件上传功能实现

MVC 设计思想要求视图与数据分离，提高系统各模块之间的独立性，带来的好处是可维护性与可重用性。所以用户看到的网页显示效果是由多次 http 请求得到的。第一次向服务器发送请求得到 html 页面，此时并没有动态数据。浏览器得到网页后执行 javascript 代码，在用户感知不到的情况下发送 ajax 请求到服务器，得到数据（通常是 xml 或 json 格式）后操纵页面的 DOM 元素将结果回显给用户。

目前主流的上传文件操作并没有采用 ajax 请求，而是使用 form 表单来提交。一个范例见下。

```
<form action="webService/predictFileService" method="post"
    enctype="multipart/form-data">
    上传文件:
    <input type="file" name='fileName'/>
    <input type="submit" value="提交网页文件" />
</form>
```

<form>标签的 action 属性指定了该请求发往何处。它并不像 ajax 那样在发送请求的时候当前页面静止不动，而是连同整个页面一块跳转到 action 指定地址。为了实现页面不变的效果，可以在前端采用<iframe>标签嵌套网页或后端实现页面重定向再次返回提交前的页面。

该系统采用第二种方法。jersey 框架并不能直接响应 http 请求，它处理后的请求要交给 tomcat 等 web 容器，然后再响应给浏览器。于是上传文件的后台只能用 servlet 实现。

5.3.3 web service 调用

该判别系统的设计初衷就是为了实现自动化判断，所以它提供了方便易用的 API（程序调用接口）。为了实现跨操作系统、跨编程语言调用，该 API 以 web service 方式提供。由于一篇网页可能有上万字，特征向量也要有 1000 个 double 数据，为了返回内容的整洁，这里只返回最重要的三个参数。该 web service 调

用契约见表 5-7。

表 5-7 web service 调用契约

请求地址	<u>http://ip:port/predictService/predictUrlList</u>
请求方法	post
头部信息	Content-Type:application/json
报文体信息	{[url1,url2,...,urln]}
返回内容	{[{ "title": "标题示例", "result": "yes 或 no", "degree": "判断准确度估计", }]}

6 总结与展望

6.1 本文工作总结

本文综合运用多种工具与理论,从头到尾实现了一个基于机器学习的网页判别系统。该系统从数据收集到机器学习再到最后的提供接口支持第三方调用,本人做了大量的编码工作,整个系统成功实现。在此过程中做了以下工作。

(1)比较多种主流的文本分类方法,决定采用实际结果较好的支持向量机来作为机器学习的模型。

(2)梳理整个系统的实现逻辑,将系统划分为数据收集模块、数据处理模块与使用模块,并制定了每个模块的流程图,确定了各个环节要用到的工具。

(3)数据收集的过程需要人工标注,考虑到对用户友好,舍弃命令行交互模式,转而采用 web 页面。该收集模块背后需要有爬虫系统来支撑,为此使用 Jsoup 工具予以实现。

(4)完成对初始训练集的分词处理与特征选择。该部分阐述了中文分词的原理,选用 ikAnalyzer 工具予以实现。为了提升分词效果,在 stopwords.dic 停用词词典中又扩充了一些停用词。考虑到卡方检验在特征提取中较为常用,该系统也选用卡方检验进行降维。

(5)简述了神经网络的工作原理,讲解了支持向量机的核思想与最优超平面的构造思想,并选用 libsvm 工具予以实现。

(6)使用 10 折交叉检验对学习效果进行评价,然后实现了人机交互的 web 使用模块与代码调用的 web service 模块。

6.2 未来研究展望

互联网的迅猛发展使得人们在生活、商业、教学等各个方面都越来越离不开

它。让机器去加工处理互联网中的海量数据是一个美好的愿景，也是人们持续关注、研究的重点。该系统服务于教学，80%的准确率这一判别效果已有一定的使用价值。

本科毕业设计时间有限，虽然理解了支持向量机的工作原理，但在 libsvm 具体工具使用上还是不够有经验。此外，该系统在以下几个方面仍有改进空间。

该系统面向的是计算机编程相关网页，互联网上并没有现成的数据集可以用，全靠课题组提供。若有更多更典型的原始网页作为训练数据，相信系统的实现效果会更好。

分词模块采用 IK Analyzer 工具，分词效果较好，但更好的分词效果肯定是有益的。

训练阶段对初始特征空间进行降维操作，选取了前 1000 个卡方值最大的词语。除了卡方检验有着夸大低频词作用的负面效果以外，特征空间的维数控制在哪个范围内会更好呢？我们希望特征向量能够较好地体现文本特征，除了按照词频来选择，是不是还有其他更优的选择方法呢？以上这些问题都需要进一步考虑。

参考文献

- [1]RicardoBaeza-Yates, Berthier Riberio-Neto. Modern Information Retrieval. Chinese Machine Press, 2003:143-145
- [2] 马玉春, 宋瀚涛. Web 中文文本分词技术研究. 计算机应用, 2004, 24(4):134-135
- [3]王煜. 基于决策树和 K 最邻近算法的文本分类研究. 天津大学, 博士论文, 2006
- [4]张毅波. 中文结构化信息检索系统的研究与实现. 中国科学院软件研究所, 博士论文, 2001
- [5]王小青, 中文文本分类特征选择方法研究, 西南大学, 硕士论文, 2010
- [6]荣光, 中文文本分类方法与研究. 山东: 山东师范大学. 2009
- [7] (土耳其) Ethem Alpaydin 著, 范明, 咎红英, 牛长勇等译. 机器学习导论. 北京: 机械工业出版社, 2014
- [8] (加) Simon Haykin 著, 申富饶等译. 神经网络与机器学习. 北京: 机械工业出版社, 2014:168-169
- [9]T. Joachims. Text categorization with support vector machines learning with many relevant feature. Proceedings of 10th European Conference on Machine Learning, 1998:137-142
- [10]Michael McCandless, Erik Hatcher, Otis Gospodnet 著, 牛长流, 肖宇译. Lucene 实战(第 2 版). 北京: 人民邮电出版社, 2012:80-85
- [11]埃史尔著, 陈昊鹏译. Java 编程思想(第 4 版). 北京: 机械工业出版社, 2007:733-735
- [12]同济大学数学系. 工程数学线性代数. 北京: 高等教育出版社, 2012
- [13]陈明 等著. MATLAB 神经网络原理与实例精解. 北京: 清华大学出版社, 2014

致谢

机器学习是目前表现活跃且有一定深度的研究与应用领域，很早以前就有一个想法，要趁着毕业设计的机会来了解与应用一下。作为一名本科生，在有限的时间内除了要编码，还要学习机器学习这一课堂中从未接触过的知识，感到很充实。感谢朱国进老师在选题上对我的建议与帮助，让我能做到自己喜欢的事情，也感谢在系统开发过程中他对我的指导与关怀，让我获益匪浅。

该系统中用到的第三方工具不下十种，不少是与 JEE 相关，在配置与使用过程中难免会有偏差，与周围同学的交流也让大家得到了共同进步，向可爱的同学们致谢。

计算机学院的老师与领导重视课堂教学，在课外创新实践上也积极地为同学们提供机会与指导，这些无形中的积累都是我在计算机领域继续前行的财富。本科四年结束了，在毕业离校之际，借此机会对培养我的东华大学说声感谢与不舍！

附录

卡方检验提取出来的权重最高的前 1000 个词语。

[偶数, 不放心, 连通分支, 数学题, 整除, 操守, 辛子, 升序, 菜价, 英杰, 序数, 奇数, 蛇年, 哈希, 模拟题, 知友, 表达式, 计算出, 源点, 匈牙利, 邻接矩阵, 显得, 白酒, 补贴, 产业链, 垄断, 赞同, 找对象, 债, 授, 见面, 摘编, 早在, 男团, 智能家居, 耍, 骑, 鲜, 赤道, 企鹅, 华尔街, 古代, 总监, 郑爽, 恩爱, 恭喜, 澳门, 力度, 利好, 意欲何为, 宝马, 开发商, 美金, 企业家, 王健林, 涨停, 一幕, 京报, 温柔, 中国经济, 声称, 博物馆, 问道, 表态, 一卡通, 全局变量, 博弈论, 人间仙境, 读入, 操作符, 雪松, 善意, 解禁, 严, 吕, 农民工, 进化, 草, 出于, 先后, 亲自, 影音, 凌晨, 而来, 安全性, 维持, 异类, 仅供, 五年, 会对, 不得不, 被告, 白领, 风采, 发货, 飞行, 最全, 西甲, 真爱, 各方, 卫生, 厮杀, 盛大, 新作, 甜美, 妙, 抛, 掀, 智取, 谁也, 罚, 秘, 贸, 传记, 战役, 霍, 自拍照, 战车, 战记, 同事, 违规, 企业邮箱, 我妈, 助力, 打伤, 成交, 出手, 宣言, 美艳, 滴滴, 导购, 居家, 重试, 重视, 前三名, 影子, 上周, 三款, 两岁, 幅度, 买断, 年初, 主管, 阅兵, 奢侈, 大巴, 多款, 参考消息, 签署, 高科技, 境内, 预售, 女郎, 集结, 公职人员, 神经, 骆, 闭环, 多边形, 膜拜, 告知, 火影忍者, 农, 依靠, 去向, 初恋, 过时, 愿意, 金山, 二是, 上班, 亏损, 陆续, 空姐, 楼市, 变革, 唱歌, 飞速, 明朝, 名校, 停不下来, 价格战, 画质, 侨, 诉讼, 跌倒, 兽, 撸, 时报, 新歌, 毅, 硕, 运作, 账, 谭, 黑帮, 威虎山, 恶魔, 车内, 鹤, 铭, 韦, 王国, 通讯员, 下跌, 两岸关系, 惊天, 辛酸, 牛市, 选自, 辞职, 适应, 信息网络, 净利润, 潜力, 中关村, 甄子丹, 二次元, 家用, 导游, 舆情, 两国, 歌舞, 海上, 索赔, 介入, 两家, 新西兰, 伸冤, 演艺圈, 获悉, 精英, 大妈, 奖品, 信用卡, 天体, 夜间, 有钱, 李冰冰, 领海, 大街, 丙, 请教, 曾在, 供, 偷, 无数, 王子, 不靠, 爆炸, 出色, 勇士, 一代, 之最, 卧槽, 您来, 甜蜜, 潘, 赖, 枪击案, 靓, 豪门, 乔布斯, 另一方面, 千元, 自行, 落后, 钻石, 一致, 奖励, 大涨, 张国荣, 被

查, 传感器, 晚报, 直升机, 改装, 盗版, 渐入佳境, 卖肉, 升至, 贪官, 操控, 光明网, 用车, 共产党, 擎天, 摔, 取得了, 洁, 撤资, 大逆转, 视野, 搜身, 警, 赔, 基础设施, 牢记, 传言, 专车, 超市, 重回, 爱心, 发型, 邻居, 自卫权, 充满, 决战, 过度, 注重, 帝国, 节操, 被称为, 翻倍, 三家, 经销商, 污染, 主任, 开房, 绝世, 并购, 压力大, 治理, 两岸, 精彩内容, 常态, 万象, 影业, 便利, 亡灵, 伤害, 她们, 随手, 降至, 欧盟, 圆通, 奇闻, 定价, 福特, 家具, 对付, 电影票, 风口, 报告书, 上海滩, 首富, 数表, 回文, 图形学, 高斯, 减去, 新人, 凌, 招, 说法, 气, 鼠, 引领, 强烈, 庞大, 铁血, 太极, 覆辙, 约瑟夫, 破坏, 发射, 士, 支撑, 穷, 蓉, 惊险, 开车, 一方面, 抓住, 续航, 脱衣, 一台, 当场, 宝马, 作出, 多数, 频繁, 合数, 畅通, 大致, 试题, 平均寿命, 新片, 领袖, 触摸, 事业单位, 另有, 人民日报社, 饮食, 勒索, 同名, 合同, 无锡, 疾病, 吉星高照, 俗, 冥, 土地, 调研, 忠, 领导干部, 喜爱, 盼, 炉, 绑, 跟贴, 肌, 购车, 煎饼, 国际足球, 这部, 情报, 慈善, 当红, 台港澳, 车祸, 意甲, 假货, 热度, 征途, 给谁, 中新社, 末日浩劫, 破坏者, 炒作, 配音, 澳洲, 中国女足, 营救, 一夜, 万名, 七一, 但也, 派出所, 头套, 大招, 神魔, 出货量, 衣柜, 隧道, 裸体, 桂林, 陈冠希, 面膜, 被判, 外逃, 奇趣, 骗局, 更有, 私, 碎, 延续, 质数, 无过错, 社会风气, 绿领巾, 色香味, 小点, 乘积, 知耻, 争端, 吃法, 四月, 便捷, 透视, 爱之, 几年, 潮流, 居民, 两名, 事后, 示例, 发贴, 反对, 珠, 制造商, 球员, 特定, 台北, 南方, 本土, 议员, 幸存者, 监测, 名牌, 后果, 旋风, 西方, 播报, 仇, 厂, 堵, 壁, 在京, 抵, 技, 整治, 犯, 熙, 中国高校, 手环, 臭, 耀, 净化, 重新认识, 泥石流, 龚, 全力, 像是, 空谷幽兰, 戏曲, 抗议, 人造, 云集, 不贵, 关门, 卸妆, 十亿, 内衣, 快照, 违者, 克里, 爱你, 前妻, 制成, 炮弹, 小额, 常务会议, 采购, 网球, 无人驾驶, 市场份额, 之争, 歧视, 三只, 法庭, 亚太, 念念, 正面, 细数, 交往, 港台, 之时, 年在, 会为, 布兰妮, 阻止, 奇才, 签订, 近年来, 汪峰, 阴谋, 间谍, 市场上, 青睐, 集资, 伴君如伴虎, 小区, 礼包, 硬汉, 口罩, 看天, 豪华, 胖, 这项, 一把, 治疗, 竞技, 面前, 正是, 梦幻, 头文件, 一棵树, 局部变量, 余数, 加法, 每行, 下界, 随缘, 外接圆, 读书笔记, 更为, 冬, 朝, 理念, 野, 必看, 维修, 歌曲, 线程, 求得, 哪家, 摇滚, 嘴, 推理, 具, 玛, 药, 蒋, 低调, 物语, 庆祝, 之中, 不愿, 海底, 复古, 小型, 处长, 口袋, 预警, 和平, 司, 惨, 摆脱, 炒, 放榜,

移动用户, 磊, 艳, 充分, 越狱, 片中, 开枪, 世上, 钱包, 沈阳, 等方面, 食堂, 正整数, 预防, 吉林, 叫板, 经济发展, 名家, 江苏省, 卫浴, 杀死, 首都, 相亲, 皇家, 亏, 贴心, 券, 役, 披, 抡, 抖, 掌, 撑起, 敦煌, 棺, 布隆迪, 赌, 刑侦, 有关部门, 驴, 新闻发布会, 假日, 跑步, 勇气, 又名, 冰箱, 惊呆, 午休, 协助, 世界第一, 军情, 手机号, 关爱, 演过, 游泳, 约谈, 努比亚, 美的, 色情, 脆弱, 绑架, 缺席, 部队, 商业模式, 火了, 粉丝团, 之作, 年纪, 富士康, 为官, 引爆, 仪式, 库存, 公安机关, 尝鲜, 人流, 大作战, 大山, 女尸, 天王, 死于, 限流, 斗地主, 期货, 行政, 成功率, 穿帮, 大白, 处罚, 飞碟, 失足, 相貌, 坏了, 床, 嫣, 意, 莎, 热情, 遇冷, 演技, 重蹈, 天堂, 每条, 表白, 后用, 用法, 求和, 奈, 戳, 明年, 教师, 退, 透明, 行数, 十进制, 聘礼, 对齐, 式子, 瓶颈, 净, 型号, 妍, 涵, 证据, 须, 制定, 酒香, 自带, 艺人, 已是, 淘汰, 队长, 限量, 这组, 局部, 瓦, 切换, 尼玛, 引擎, 食谱, 白宫, 扣, 探险, 全家, 王者, 战神, 腰, 已经成为, 摘下, 变了, 战场, 不少人, 中介, 丢了, 怎么看, 五月, 大地, 石油, 变迁, 故宫博物院, 来华, 高官, 放缓, 口碑, 李敏, 新锐, 时隔, 政变, 掀起, 坎, 承销, 投资人, 除此之外, 匠, 募, 在即, 奴, 百亿, 调戏, 戈, 欣, 演唱会, 总经理, 救援, 磁, 诀, 入围, 儿时, 凭借, 催生, 伤者, 事项, 民主党派, 航空公司, 超模, 东风, 华文, 感人, 剧组, 配套, 军舰, 酷刑, 都将, 违章, 郑州, 炒股, 不见了, 养老, 兼职, 卢森堡, 悄然, 邮编, 实行, 清白, 年中, 巴士, 部长, 群星, 航天, 老鹰, 张掖, 红人, 比特, 万一, 人均, 美食家, 弓箭, 底线, 网事, 之父, 舰艇, 舰船, 仙境, 银幕, 买房, 伙伴, 今夏, 亲生, 开奖, 药方, 此事, 粗口, 大批, 侵华日军, 此后, 门槛, 在国外, 殴打, 面世, 副局长, 大面积, 增幅, 震荡, 霍金, 雷锋, 实录, 装机, 姐妹花, 霸道, 女童, 还让, 咖, 船, 损失, 增加了, 激活, 演讲, 下滑, 流程, 陌生, 极速, 双飞, 对话框, 字节, 更换, 开通, 演示, 而这, 奇特, 首相, 患, 效应, 瑟, 穗, 诚, 胃, 兼容, 公关, 美腿, 实惠, 闹洞房, 二分法, 晕了, 献辞, 拍案, 毛新宇, 二叉, 数字图像, 挂, 童, 不大, 之下, 重叠, 花瓶, 帕, 逐步, 中学, 小学生, 一部分, 昆明, 周日, 咱们, 果在, 超人, 岁, 堂, 热销, 手机用户, 适配, 为主, 带领, 中外, 不让, 等着, 替身, 将军, 取保候审, 议会, 驾到, 万里行, 商场, 第一期, 月内, 上班族, 助阵, 贵金属, 馆里, 丨, 雁门关, 挑衅, 刑, 尤, 财务, 回国, 捞, 中国概念股, 六大, 先导, 超高, 感受到, 退休, 手术, 逐,

玄机, 批评, 社科院, 抗联, 技艺, 上身, 加班, 凡本网, 炮轰, 利率, 木乃伊, 特价, 半价, 分站, 剥夺, 力克, 军用, 初期, 出场, 或以, 惨烈, 这对, 农民, 岛礁, 清爽, 海警]