基于特征熵值分析的网站分类系统实现

李琛轩

院 （系）：计算机科学与技术　　专　　业：计算机科学与技术

学　　号：1100300803　　指导教师：张宏莉/沈智杰

2014 年 6 月

哈爾濱工業大學

毕业设计（论文）

题　　目　基于特征熵值分析的网站分类系统实现

专　　业　计算机科学与技术

学　　号　1100300803

学　　生　李琛轩

指 导 教 师　张宏莉/沈智杰

答 辩 日 期　2014 年 6 月

哈尔滨工业大学毕业设计（论文）评语

姓名：李琛轩　学号：1100300803　专业：计算机科学与技术

毕业设计（论文）题目：基于特征熵值分析的网站分类系统实现

工作起止日期：2014 年 2 月 25 日起　2014 年 6 月 30 日止

指导教师对毕业设计（论文）进行情况，完成质量及评分意见：

指导教师签字：＿＿＿＿＿＿＿＿　指导教师职称：＿＿＿＿＿＿＿＿

评阅人评阅意见：

评阅教师签字：＿＿＿＿＿＿＿＿　评阅教师职称：＿＿＿＿＿＿＿＿

答辩委员会评语：

根据毕业设计（论文）的材料和学生的答辩情况，答辩委员会作出如下评定：

学生 _____ 毕业设计（论文）答辩成绩评定为：_____

对毕业设计（论文）的特殊评语：

答辩委员会主任（签字）：_____ 职称：_____

答辩委员会副主任（签字）：_____ _____

答辩委员会委员（签字）：_____ _____ _____ _____

_____ _____ _____ _____ _____

_____ _____ _____ _____ _____

_____

年 月 日

哈尔滨工业大学毕业设计（论文）任务书

姓　　名：李琛轩　　院　（系）：计算机科学与技术

专　　业：计算机科学与技术　　班　　号：1003104

任务起至日期：　　2014 年 2 月 25 日 至 2014 年 6 月 30 日

毕业设计（论文）题目：

基于特征熵值分析的网站分类系统实现

立题的目的和意义：

随着互联网技术的飞速发展，Internet 上的 Web 页面呈指数型增长。对于如何自动对这些海量数据有效处理和管理，来取代低效繁琐的人工分类，Web 文本分类技术成了关键技术。目前对于这方面的研究已经有了很大进展，并且产生了一系列分类方法，其中，KNN 算法简单、有效、参数无关，目前的应用非常广泛。但是，传统 KNN 算法有不少的缺陷，最关键的两个缺陷是运行速度太慢和分类精度不高。本设计对 KNN 算法的缺陷产生原因进行详细地分析，并针对缺陷对算法进行了引入属性熵值等一系列的改进，使得改进的 KNN 算法达到高速、高精度的性能，并且基于改进后的新 KNN 算法，搭建一个真正实用性强的网站分类系统。

技术要求与主要内容：

一个高效的基于改进 KNN 算法网站分类系统由网页内容的下载、页面处理、特征提取、特征向量表示、分类器训练、快速运行决策、提高精度的决策等方面组成。

（1）设计一个高效爬虫模块和高效的页面处理算法；

（2）设计一个精准的特征提取算法，以及特征向量的表示方法；

（3）针对 KNN 算法在速度上的不足，分析影响速度的主要因素，并且对这些因素进行改进，使得新算法在速度上达到高速运行；

（4）针对 KNN 算法在分类精度上的不足，分析影响精度的主要因素，并且对这些因素进行改进，使得新算法在精度上得到大幅度提高；

（5）利用设计模式相关思想将各个模块组建成一个实用、操作简易的图形化界面的系统。

进度安排：

2014.2.20-2013.3.1　完成完成项目所需的知识的储备和熟悉

2014.3.1-2014.3.15　完成爬虫种子集群模块和 HTML 内容提取模块

2014.3.16-2014.4.15　完成特征提取模块和基本 KNN 算法分类器模块以供接下来分析改进使用

2014.4.16-2014.4.30　完成对 KNN 算法在运行速度上的改进并进行实验

2014.5.1-2014.5.31　完成对 KNN 算法在分类精度上的改进并进行实验

2014.6.1-2014.6.7　完成系统的组建和图形化界面的设计

2013.4.8-2014.6.19　拟写毕业设计论文，修改并准备答辩

同组设计者及分工：

无

指导教师签字_____

年　　月　　日

教研室主任意见：

教研室主任签字_____

年　　月　　日

摘　要

随着互联网技术的飞速发展，Internet 上的 Web 页面呈指数型增长。对于如何自动对这些海量数据有效处理和管理，来取代低效繁琐的人工管理，Web 文本分类技术成了关键技术。目前对于这方面的研究已经有了很大进展，并且产生了一系列分类方法，比较著名的有支持向量机（VSM）、K 最邻近（KNN）、神经网络和贝叶斯（Bayes）算法等。在这些算法中，KNN 算法由于其简单、有效、参数无关，目前的应用非常广泛。但是，KNN 算法有着不少的缺陷，最关键的两个缺陷是运行速度太慢和分类精度不高。

本设计对 KNN 算法的缺陷产生原因进行了分析，并对其进行了改进：在特征提取上引入了基于改进的 CHI 方法使得特征提取更加合理；在 KNN 分类器运行速度的改进方面引入了 Rocchio 算法的思想和一些其他简单的思路对分类器进行速度的提升，使得新的分类器的分类速度得到大幅度提升；在 KNN 分类器分类精度的改进方面，通过在相似度计算上引入了基于属性熵值的相似度改进和基于 KNN 类别加权的改进，使得改进的 KNN 算法又在分类精度上得到了大幅度提高。在基于上述这些改进后，搭建出了一个真正具备高效、实用的网站分类系统。

本文完成了分类器系统的实现，并且利用个 3578 个真实网站内容作为测试集对系统进行了性能测试。通过对实验结果进行分析，得出本文提出的新的 KNN 分类器在测试集数据的环境下达到了高速分类和分类正确率远高出传统方法的结论。本文提出的新的高效 KNN 算法作为网站分类器比原有的 KNN 分类方法和加权 KNN 方法有更快的速度，同时比两者有更高的分类精度。

关键词：高效网站分类；改进特征提取；快速分类；高精度分类；属性熵值分析

Abstract

With the rapid development of the Internet,web pages on the Internet is growing exponentially. On the issue of how to organize and deal with these massive data effectively, automatically and how to take the place of manual management which is too inefficient and cumbersome, Web text classification has became a key technology. At present,the research in this area has made great progress,and there are a series of classification methods. And there are some well-known methods such as support vector machine (VSM), K-nearest neighbor(KNN), neural networks and Bayes algorithm. KNN method is widely used due to that it is sample,effective and regardless of parameters. However,the traditional KNN method has two critical flaws, one flaw of them is that KNN method is running too slow, the other flaw is that the accuracy of this method is not sufficiently high.

This paper analyzed the causes of detects in KNN method and made major improvements: In the feature extraction module,we introduced an improved method based on CHI method,which makes the feature extraction more reasonable. In the classifier speed improvements, we introduced the idea of Rocchio method and some other simple ideas to improve classification running speed. Thus new classifier will be greatly improved in the speed. In the classifier accuracy improvements, we made the classification accuracy significantly improved through the introduction of an improved similarity calculation based on the entropy of properties and some ideas based on class weighted KNN method.

Based upon these improvements, we built out a definitely efficient and practical web classification system. This paper completed the implementation of the new classification system,and we used 3578 real web content as a test set to test the performance of our system. Through the analysis of experimental results, we drew a conclusion that in our test data set environment, our improved classifier has achieved high-speed sorting and much higher accuracy rate than traditional KNN method. Web classifier based on new efficient improved KNN method proposed in this paper has much faster speed and much higher classification accuracy rate than the original KNN method and weighted KNN method.

Keywords: efficient site classification, improved feature extraction, rapid classification, precision classification, property entropy analysis

目　录

第 1 章 绪　论

1.1 课题的研究背景和意义

1.1.1 目前网站分类的研究情况

随着互联网技术的飞速发展，Internet 上的 Web 页面呈指数型增长。对于如何有效组织和处理这些海量信息，如何更好地搜索、管理这些网络资源，Web 文本分类技术成了关键技术。目前对于这方面的研究已经有了很大进展，并且产生了一系列分类方法，比较著名的算法有：

支持向量机（SVM）：未改进的支持向量机是用来解决二分分类模式识别问题的，基本思路是通过寻找支持向量来确定可以区分类别的决策面，并且使得分类间隔最大。虽说SVM靠着良好的学习能力和灵活的推广能力而得到了广泛的应用，但是该算法也存在着一些问题，主要包括算法复杂而难以实现、学习过程速度慢、对参数敏感等缺点。

神经网络：人工智能领域被广泛利用的一项技术，在网站分类问题中应用这项技术时，每个分类都需要建立一个神经网络，通过学习得到特征词汇到分类的非线性映射。其计算量庞大而且训练时间非常漫长。

贝叶斯（Bayes）算法：机器学习中非常轻量且常见的方法，基本思想是通过单词和类别的关联概率来估计网站所属类别。

K 最邻近（KNN）：知名的模式识别统计方法，是目前最好的文本分类方法之一。KNN 算法的思想非常简答：将测试文档与训练集中所有已标注类别的文档进行对比，得到相似度最高的 K 个邻近，根据这 K 个邻近所标记的类别选出出现最频繁的类别作为测试文档类别。缺点在于计算量非常巨大而且类别判定策略过于简单和武断。

这些算法均立足于机器学习理论。在它们当中，KNN 算法由于其简单、有效、参数无关，目前的应用非常广泛。

1.1.2 现有解决方案的优点与不足

目前对 KNN 算法用于网站分类的研究有很多，而且出现了很多基于改进的 KNN 算法。

针对传统 KNN 算法在分类速度上的不足，目前相关的研究内容有基于训练集裁剪的改进和基于邻近搜索方法的改进等。基于训练集裁剪的改进主要的考虑角

度在于对训练集的裁剪可以直接减小计算开销，还可以适当提高分类精度。而基于邻近搜索方法的改进主要是从KNN算法对每个新样本都要计算它与所有训练样本之间的相似度，而导致算法本身分类效率低下且占用大量内存的角度来入手。但是以上这些方法不仅对算法提速不明显，而且改进本身需要大量的工作，未体现出简单、实用。

而针对传统 KNN 算法在分类精度上的不足，目前比较著名的改进有基于特征加权的改进和基于类别判断的改进。基于特征加权的改进考虑到传统 KNN 算法在进行相似度计算时等同对待各个向量维而影响分类精度，TF*IDF 方法是目前最常用的加权方法，但是该方法没有考虑特征项的类内、类外分布情况。基于类别判断的改进主要在 KNN 算法判断类别的决策中改进，传统 KNN 算法对 K 个邻近等同对待，改进的 KNN 算法会给 K 个邻近中的那些类别不同的权重来综合评价。以上这些方法各自都会适当提高分类器分类精度，但是都会带来新的问题。

纵观 KNN 算法的研究，在 KNN 分类速度的改进上看，目前的研究从应用角度来说改进效果不明显而且改进代价较大，而在 KNN 分类精度的改进上的考虑又稍显片面，往往顾此失彼。此外，目前少有对 KNN 分类器的分类速度和精度两者都进行改进的研究和系统实现。

1.1.3 基于特征熵值分析的网站分类系统的设计目标

设计目标简而言之就是：设计一个高分类速度且高分类精度的网站分类器，分类速度目标分类速度是毫秒级，目标分类精度是大于 80%。

本设计以实用为出发点对KNN算法的缺陷产生原因进行分析并进行对算法的改进：在特征提取上引入了基于改进的 CHI 方法使得特征提取更加合理；在 KNN 分类器运行速度的改进方面引入了 Rocchio 算法的思想和一些其他简单的思路对分类器进行速度的提升，使得新的分类器的分类速度得到大幅度提升；在 KNN 分类器分类精度的改进方面，通过在相似度计算上引入了基于属性熵值的相似度改进和基于 KNN 类别加权的改进，使得改进的 KNN 算法又在分类精度上得到了大幅度提高。

在基于上述这些改进后，搭建出了一个真正具备高效、实用的网站分类系统，分类响应速度达到了毫秒级，分类精度在 3578 个真实测试数据中达到了 85.047%。

1.2 论文的研究内容与组织结构

1.2.1 论文的研究内容

本文主要研究内容为：对 KNN 算法用于网站分类的各个步骤进行缺陷分析，

引入属性熵值的概念、Rocchio 算法思想以及一些其他策略，对分类中每个步骤进行全面改进和优化，最终建立一个有实用价值的网站分类系统。

1.2.2 论文的组织结构

本文内容按照下面的结构进行组织安排：

第一章绪论介绍了研究课题的背景，给出了网站分类技术的发展情况与面临的问题，阐述了现有解决方案的优点与不足，明确了研究的内容与目标。

第二章介绍了系统设计的整体架构和技术需求，以及各个模块的详细功能、涉及技术。

第三章详细介绍了系统的前两个模块的实现：爬虫模块和页面处理模块。对于页面处理模块，介绍了一种线性时间复杂度的正文提取方法，并将其应用于本设计的页面处理模块。

第四章详细介绍了特征提取与文本特征表示模块的技术实现，包括对现有特征提取办法的缺陷分析和改进，以及文本特征表示方法的介绍。

第五章详细介绍了 KNN 分类器模块的实现。从传统 KNN 算法的缺陷入手，分别从运行速度的改进上提出 4 种办法和分类精度的改进上提出了 4 种办法，并将这些办法组合成最终的新型改进 KNN 分类器。

第六章介绍了最终的新型改进 KNN 分类器在真实网站数据集上的实验结果，并与传统方法对比，得出本文提出的新型改进 KNN 分类器更加快速和更加准确的结论。

第 2 章 系统模块组成介绍

2.1 系统总体架构

本设计旨在设计一个效果优秀的网站分类系统。

首先确定输入和输出：输入为一个或多个 URL，输出为每个输入 URL 对应网站的类别判断结果。

根据输入输出分析需求：输入为 URL 列表，而网站识别靠的是对网站内容分析，所以首先需要一个爬虫来对 URL 对应的 HTML 文档进行下载；其次，为了提取出 HTML 文档中的重要信息进行进一步的分析，需要一个页面处理模块来提取有价值信息；再次，在有价值信息中，需要提取出对分类有辨识能力的若干特征项，并且要把特征项进行数学抽象以便于作为分类器可读取的输入，所以又需要一个特征提取和文本特征表示模块；接下来，通过把网页文本的数学形式输入到分类器，分类器进行识别后输出分类结果，所以还需要一个分类器模块。

根据上述的分析，可以得出我们所需的模块包括：

（1）一个爬虫模块：输入是 URL，输出是对应 HTML 文档；

（2）一个页面处理模块：输入是 HTML 文档，输出是有价值短文本内容；

（3）一个特征提取和文本特征表示模块：输入是文档的文本内容，输出是数学表示的文本；

（4）一个分类器模块：输入是文本的数学化表示，输出是判定类别结果。

假设我们是系统的用户，我们输入一个 URL，先由爬虫下载页面，之后对页面进行处理，得到网页的短文本，之后将短文本转化为数学表示形式，输入到分类器中，分类器给出分类结果。

再假设我们是分类器的构建者：在机器学习领域，分类器的构建的基本思是先给出类别已人工标注的训练集，并用特征提取技术提取对分类有帮助的特征，再把这些特征数学化表示，而后分类器对数学化表示后的训练集进行学习，自动找出训练数据中的特征与类别的关联规则或者规律，然后给出一个未标注类别的测试输入，分类器根据自己之前的学习，对输入进行判断而后给出分类器认为其属于的类别。而这里提到的分类器模块本身又需要训练集，对于本设计来说即已经人工标记了所属类别的网站。

综上所述，分类器系统的设计可以从两个角度来形象地描述，分别是：使用者（用户）的角度和分类器构建（系统内部）的角度。

用户的角度看：

（1）用户输入某网站 URL，由系统对该网站进行分类，如果该网站已经被分类过，那么直接返回对应的结果；否则爬虫爬取该网站，下载对应的 HTML 文档；

（2）从 HTML 文档中提取出有价值文本，对其进行文本特征的数学化表示；

（3）将网站的数学表示形态输入已构建好的分类器中，分类器对该数学表示形态进行分类，返回分类结果。

系统按照用户角度的架构大致如图 2-1 所示：

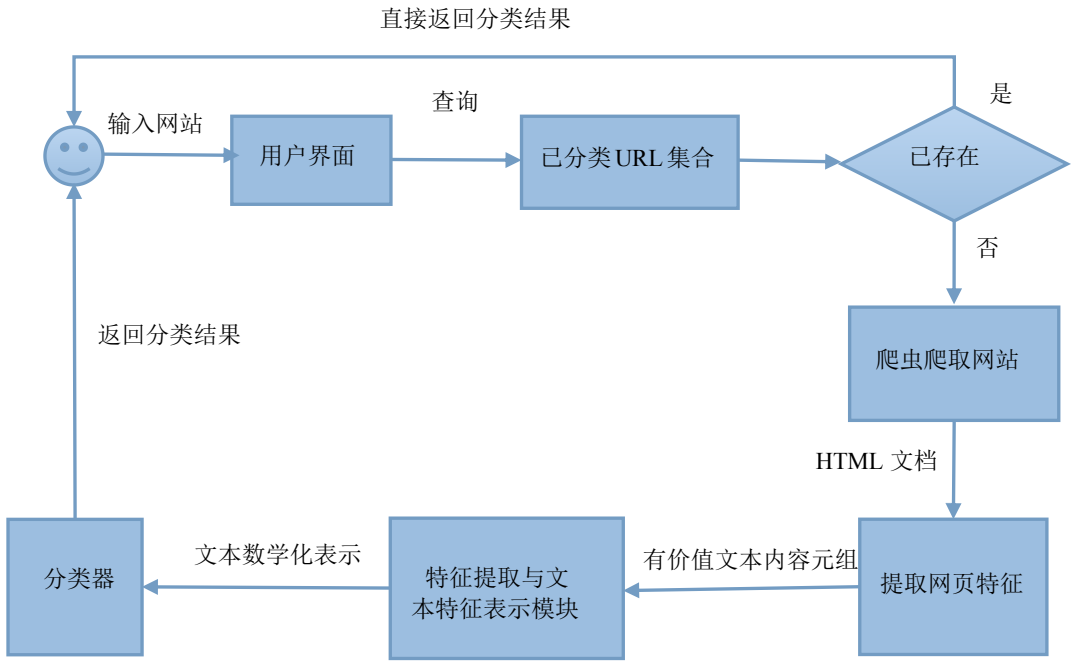

图 2-1 用户角度的系统架构

从分类器构建的角度看：

（1）设计分类规则，找到每个分类下的足够多的 URL 并且标注类别，作为建立训练集的 URL 列表；

（2）用爬虫模块爬取 URL 列表中的所有 URL，下载所有 HTML 文档；

（3）网页内容处理模块，用于对种子站点的内容进行有价值文本的抽取，于是，HTML 文档简化为短文本；

（4）将所有类别下的所有短文本进行特征提取（在本设计中最终研究出一种基于改进的卡方检测方法来特征提取），提取出所有有分类识别能力的特征项，把每个短文本拥有的特征用某种数学模型数学化表示之（在设计中最终确定使用向量空间模型 VSM 和 TF*IDF 方法），于是短文本简化为数学表示形态，到此时，训练集中的每一个网站都是一个数学化（VSM）形态；

（5）用一种机器学习算法（本设计最终研究出一种高效且高精度的新型改进 KNN 算法分类器）对训练集进行学习，建立出分类器模型。

系统按照分类器构建的角度架构大致如图 2-2 所示：

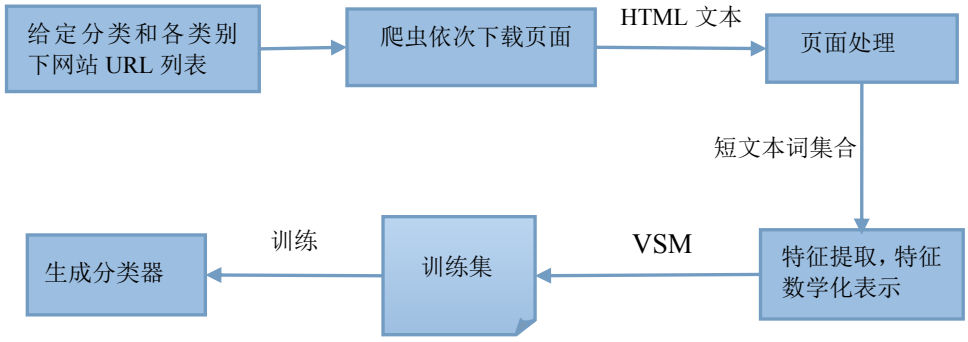

图 2-2 构建分类器角度的系统架构

综上，本设计按照模块划分为：爬虫模块，页面处理模块，特征提取与文本特征表示模块，KNN 分类器模块。下面几小节介绍了各个模块的功能和所需技术。

2.2 爬虫模块功能与技术

爬虫模块的功能：简而言之就是依照给出的 URL 去下载对应的 HTML 文档进而用于下一步分析。从使用者的角度看，用户输入 URL，如果该 URL 尚未被收录，那么下载对应 HTML 文档；从构建分类器的角度看，训练集的建立需要各个类别下大量真实的网站，需要对每个类别下的 URL 进行下载 HTML 文档作为训练集的生成。

涉及的技术实现：这个模块相对简单，需要编写一个高效的爬虫，下载页面内容存入 HTML 文档即可。但是，由于目前中文网站编码良莠不齐，下载页面内容存入 HTML 文档时可能遇到网页编码不统一而导致的下载页面乱码问题，我们需要注意设计的爬虫应该由对各种网页编码的转换处理能力。

2.3 网页处理模块功能与技术

网页处理模块的功能：为了提取出可以反映、暗示网站类别或者对网站所属类别有指导性的一些标签内容，需要对网页进行某些标签内容的提取，比如 HTML 文档的标题提取、META 元标签的提取、正文提取等。提取出这些内容组合成一个短文本并分词后作为网站内容的简写形式，以便特征提取模块进行特征提取。

涉及的技术实现：对于 HTML 文档中的标题 TITLE 标签、元数据 META 标签，使用正则表达式技术可以高效、无误差地提取；而正文提取不是很容易，需要设计或者参考一种正文提取方法，在本设计中，最终参考了一种线性时间复杂度的

基于行块分布函数的正文提取方法。之后，需要一种分词技术来把短文本切分为词的集合。

2.4 特征提取与文本特征表示模块功能与技术

特征提取和模块的功能：对于训练网页集经过页面处理模块生成的短文本词集合形态，本模块需要根据训练集中所有词在每个类别下的分布情况利用一种特征提取方法进行特征的提取，找出最能代表和支持类别的那些词作为特征项，然后用数学的形式将这些训练数据保存，形成最终的训练集用于分类器的训练。而对于测试页面或者用户输入的未标注类别 URL 经页面处理模块生成的短文本词集合形态，本模块也需要将其转化为数学表示形式以作为分类器输入。

涉及的技术实现：我们需要在目前成熟的特征提取技术中选取一个最适合的方法，在目前已有的特征提取方法中，本设计选择了卡方检验（CHI）方法，并且在分析了这个方法的缺陷产生原因后，提出了一种改进的卡方检验方法。文本特征的数学化表示方法中，本设计考虑到向量空间模型（VSM）在目前的文本分类中效果较好，于是采用了 VSM 表示方法。而且为了反映每个特征项的权重，引入了 TF*IDF 方法来计算在每个文本向量中每个维度（即每个特征项）的向量值。

2.5 分类器模块功能与技术

分类器模块功能：顾名思义，分类器就是用来分类的，用户输入的 URL 经过上述几个模块的处理后生成的文本向量 VSM 作为分类器的输入，分类器进行计算后输出自己对输入的类别猜测作为该 URL 的最终判定类别。

涉及的技术实现：由于 K 邻近算法（KNN）在文本向量模型下是最好的文本分类算法之一，本设计中的分类器基于 KNN 算法。在对传统 KNN 算法进行缺陷研究后，本文罗列出 KNN 算法运行时间慢和分类精度不高的主要原因，在运行时间上结合 Rocchio 算法、建立倒排索引、建立“位置向量”等思路，同时在分类精度上结合了属性熵值分析、类别加权、类别平均相似度、共有特征个数等因素改进分类策略，从而设计出一种新的改进 KNN 算法，这个算法拥有高效、高精度的特性。

2.6 本章小结

本章包括本设计的功能需求、模块划分和各个模块的技术简要介绍。特别地，基础的爬虫搭建和页面处理涉及的分词技术均非设计重点，且稳定性要求较高，所以这两者分别采用了目前相对稳定强大的开源工具 Scrapy 和 Jieba 分词，在这两

个开源工具的基础上，只需根据本设计所需功能需求进行功能定制，而不必去考虑具体底层实现，减少了很多不是很必要的编程工作。考虑到 Python 语言很适合数据分析，而且上面提到的开源项目均是基于 Python 实现的，最终本文采用了强大的编程语言 Python（Python 2.7.5）来对系统进行全方面设计，系统实现的平台是 Unix 操作系统。

第 3 章 爬虫模块和页面处理模块

3.1 爬虫模块详细设计

本设计中爬虫的用途就是把目标 URL 的 HTML 文档保存到对应磁盘路径下。

前面提到，基础爬虫的搭建并非设计重点，而且稳定性要求较高，需要很多额外的知识，所以本文爬虫基本架构采用了开源项目 Scrapy，在 Scrapy 爬虫的基础架构上，我们进行功能代码添加，使得爬虫模块符合需求。

为了添加功能代码，首先需要理解 Scrapy 爬虫的架构，如图 3-1 所示：

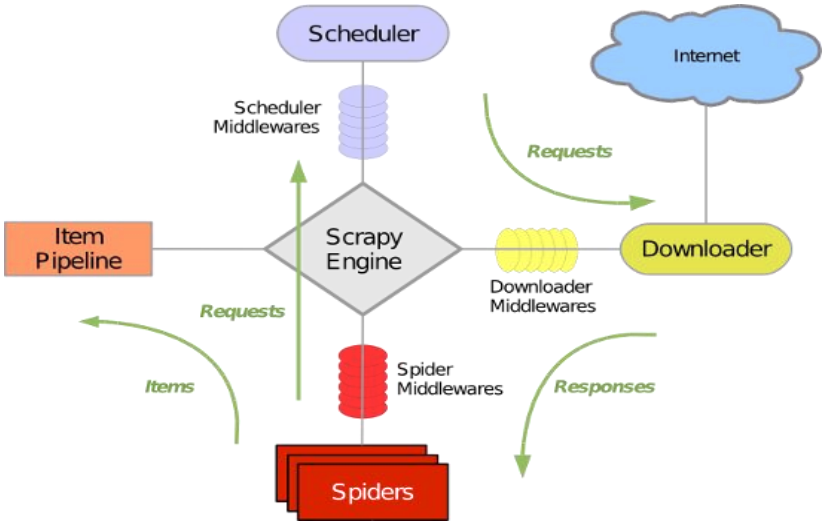

图 3-1 Scrapy 爬虫架构

输入是一个目标 URL，该 URL 被送入 Spider 组件（这一步图上并未体现）。Spider 组件接受这个 URL 作为初始 URL，将其通过 Engine 核心交给 Scheduler 组件进行调度，Scheduler 组件维护一个调度队列，每次 Scheduler 把队头的 URL 作为 Request 送给 Downloader 组件去互联网上下载对应的 HTML 页面。Downloader 下载完成后把页面内容封装入 response 发给 Spider 组件，Spider 组件负责对 response 中的页面内容进一步处理和分析。Spider 组件对页面内容分析的行为由开发者自己定义，比如，我们可以在页面中提取新的超链接，将其再次送入 Scheduler 组件去调度；再比如，我们可以再页面中提取我们感兴趣的数据作为 Item 送到 Item Pipeline 去对这些数据进行进一步操作（保存、计算、分析结构等）。

从 Scrapy 工作流程我们可以看到，当 Scrapy 爬虫爬取得到相应的应答

（Response）后，把处理权交给了 Spider 组件，在这里我们可以自由地去定义我们要做的事情。所以本设计需要在 Spider 组件中定义我们的行为。具体的方案流程图见图 3-2：

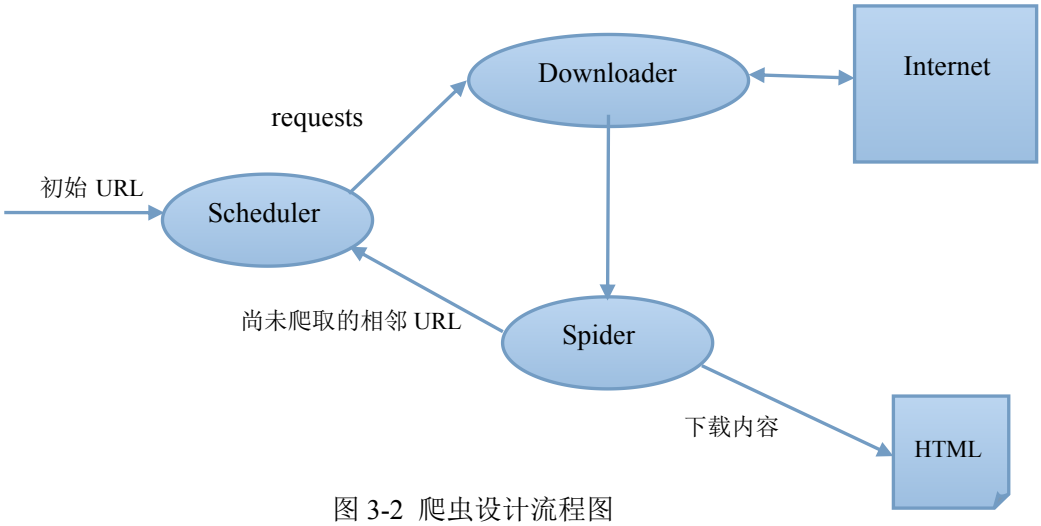

图 3-2 爬虫设计流程图

初始的时候人为输入一个 URL 作为初始请求，当请求返回后回调函数开始执行，在回调函数中解析下载到的页面的 HTML 内容，将 HTML 代码以 HTML 格式存储到与 URL 路径对应的磁盘路径中，同时提取出一切从属本域名下的 URL，在回调函数中执行新的请求，这样爬虫就会继续爬取这些新的 URL。于是，当爬虫工作结束后，形成了种子集群。

由于分类器需要大量真实网页训练集，故从一个人工分类目录 CHINADMOZ 网站提取 18669 个已经归类的网页，用本模块的爬虫来下载这些网页作为训练网页集。

3.2 页面处理模块详细设计

为了提取出可以反映、暗示网站类别或者对网站所属类别有指导性作用的一些标签内容，需要对网页进行对某些标签内容的提取，比如 HTML 文档的标题提取、META 元标签的提取、正文提取等。提取出这些内容组合成一个短文本并分词后作为网站内容的简写形式，以便特征提取模块进行特征提取。

3.2.1 页面内容价值分析

由于网页的内容信息量巨大而且有很多噪音（比如广告代码、注释、脚本代码），而我们只需要对网站类别有价值的那些内容。通过对 HTML 每个标签的意义的学习和对网站 SEO 知识的了解后，本文找到如下几个有价值的文本块或标签：

（1）META 标签：为了推销自己和增大流量，大多数网站都在搜索引擎这一

互联网入口做文章。网站为了让搜索引擎更快更准地找到自己、提高搜索排名，会在 meta 元标签中放置这么两行：

<meta name=“keywords” value=“word1,word2,word3,..”>;

<meta name=“description” value=“content1,conten2,content3,...”>.

从标签的 name 属性容易看出，keywords 对应的 value 里往往存放着一些本网站的关键字，description 对应的 value 里往往存放着本网站的简要介绍。在网站优化排名（百度优化、谷歌优化等）技术中，这两个标签起着不可或缺的作用。显然，这两个文本段对于网站分类非常具有极大的参考价值，对于优化排名做得比较好的网站甚至可以仅凭这两个标签来判断网站所属类别；

（2）TITLE：网站的 title 标签里存放着网站名，而对于静态页面甚至存放着文章正文标题，显而易见，title 对网站分类也有关键参考价值，尤其是那些网站优化排名做得相对差的网站；

（3）网页正文：正文对网站类别也有一定参考价值，不过结合实际情况来说，网页正文对分类的参考价值不是很高，有时甚至可能干扰类别判断。

3.2.2 页面处理方法

根据上一小节的分析，本设计中需要提取的部分包括：

（1）HTML 的<meta[@name=‘keywords’]>标签；

（2）HTML 的<meta[@name=‘description’]>标签；

（3）HTML 的<title>标签；

（4）HTML 的真实正文（如果有有效正文的话）；

由于正则表达式在处理结构化文本的优势和 Python 语言本身对正则表达式的默认支持，META 标签和 TITLE 标签的提取非常简单，只需编写对应的正则表达式即可。然而正文的提取却不是那么容易，考虑到正文对网站内容的参考价值不是很大而且本设计追求高效，本设计不值得在正文提取技术上耗费大量时间，所以，需要一种快速而轻量级的正文提取方法对正文进行准确抽取。

目前的正文提取方法有很多，最常见的就是采用基于 DOM 树的方法，虽然这种方法直观而且有效，但是耗时过长，而且对病态 HTML 文档的处理能力太弱，对本设计来说不是很可取。而对于采用基于机器学习的方法，这种方法需要大量数据集而且同样耗时很长，对于本设计来说显得很小题大做，也是不可取的。

本文参考了一种线性时间复杂度的正文提取算法-基于行块分布函数的正文提取算法，本设计采用了这种算法并对其进行了一些适当的参数修改。下一小节介绍这个算法的详细思想和实现。

3.2.3 一种线性时间的正文提取算法

考虑到本设计要求高效处理 HTML 文档，所以正文提取的设计借鉴了 Google Code 中 CX-extractor 项目的思想：将提取 HTML 文档正文的问题转化为一个在行块分布函数中找波峰区间的问题，完全摆脱了对 HTML 文档结构的考虑。通过扫描一遍 HTML 文档纯文本来建立行块分布函数，该函数的波峰所在区间位置就是正文段所在位置，于是 HTML 正文提取问题可以由求函数波峰区间的方式在线性时间内解决，该方法对 HTML 正文的定位既高效又较准确，适合本设计采用。

由于网页在抛去一切 HTML 标签后也是一行一行的文本，每一行都有自己的长度，文字多的行当然长度很大，文字少的行当然长度很小，而显然正文部分所在的那些行具有很高的行长度，这是一个很好的识别正文的切入点。基于行块的网页正文提取的思路就是来源于这里。

正文提取的第一件事就是把 HTML 文档的所有标签去除掉，剩下的纯文本用 hText 表示。此时先引入几个要用到的定义：

（1）行块：以 hText 中的行号作为轴，取周围 K 行（前 K 行或者后 K 行都可以，具体的 K 值在实验时人工调整），合成一个行块 hBlock，行块 i 是以 hText 行号为 i 合成的行块；

（2）行块长度：一个行块在抛去一切空格、回车符等后的字符总数称为该行块的行块长度，即 K 行非空字符总长度和；

（3）行块分布函数：以行块 hBlock 的行块号为 X 轴，以对应的行块长度为 Y 轴，建立行块分布函数。

算法实际做的事情就是在行块分布函数中找到最大值所在的一段连续区间[a,b]，将行块号为 a 和行块号为 b 分别对应的行区间取出来视为正文。

算法具体的步骤如下：

（1）首先，把 HTML 文档的一切标签、注释全部删掉，生成了一个包含很多行的纯文本；

（2）对于每一行统计这一行的长度；

（3）从第一行开始，每相邻的 K 行我们定义为一个“行块”，而”行块“长度为这 K 行的总和；

（4）以行块为 X 轴，对应的行块长度为 Y 轴，建立行块分布函数；

（5）扫描函数，将具有最值且连续的一块区域取出视为目标正文；

我们用数学语言来描述“寻找最值且连续的一块区域”这一问题：假设 X 为行号，Y(X)为行号 X 对应行的长度，要求正文区域所在的起始行块号 X_{start} 和终止

行块号 X_{end}，需要满足以下四个条件（其中：X_t 是第一个行块长度急剧上升的点，K 是行块长度）：

（1）Y(X_{start})>Y(X_t)；

（2）Y(X_n)≠0（n∈[start+1,start+K]，急升点之后 K 个点的行块长度不能为 0，用于防止噪声）；

（3）Y(X_m)= 0（m∈[end,end +1]，急降点行块长度为 0，正文结束）；

（4）Y(X)最大的行块对应的行块号 X∈[X_{start}，X_{end}]（保证此区域是波峰所在区域，即是取到行块最大值的区域）。

分析算法效率：由于算法仅需要扫描一遍 HTML 文档的纯文本，假设纯文本有 n 行，那么算法时间复杂度为 O(n)，达到了线性时间复杂度下的正文提取。

3.2.4 页面处理关键流程图

模块设计流程图如图 3-3：

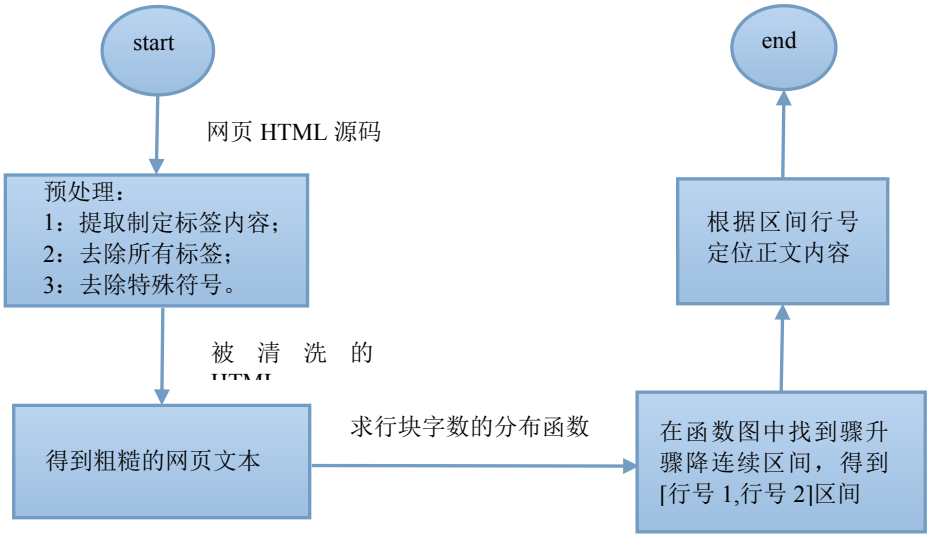

图 3-3 正文提取流程图

对于一个网页，有用标签被提取后，我们就得到了这个网页的关键文本。对每一个网页的关键文本利用开源工具 Jieba 分词工具进行分词后，每个网页生成一组词的集合。值得说明的是，训练网页经过在这一步后，每个类别下的每个网址都从对应的 HTML 文档里生成了各自的关键词集合，这些词集合就是网页内容的简单表示，接下来需要做的事情就是在每个类别下的关键词集合中找出可以区分和代表类别的特征词，即特征提取。

3.3 本章小结

本章介绍的是分类器设计实现的两个前期模块：爬虫模块和页面处理模块。爬虫模块用来下载每个 URL 对应的 HTML 文档，至此 URL 用对应的 HTML 文档来代表；页面处理模块用来对 HTML 文档进行关键文本提取并分词，至此 HTML 文档用关键词来表示。

这两个模块相结合，初始的网站 URL 的表示形式转化为一组关键词集合。对于训练集 URL，接下来需要在关键词集合上提取出各类别的特征词；而对于测试 URL 或者用户输入 URL，接下来需要结合系统提取的特征词对测试 URL 对应的关键词集合进行数学化表示。具体的工作将在下一章详细介绍。

第 4 章 特征提取与文本特征表示模块

4.1 特征提取技术介绍

对于训练网页集经过页面处理模块生成的短文本词集合形态，本模块需要根据训练集中所有词在每个类别下的分布情况利用一种特征提取方法进行特征的提取，把最能代表类别和区分类别的那些词全部找出来作为特征项，然后根据最终的特征项集合把每个 HTML 短文本转化为数学的形式保存下来，形成最终的训练集用于分类器的训练。而对于测试页面或者用户输入的未标注类别页面经页面处理模块生成的短文本词集合形态，本模块也需要将其转化为数学表示形式以作为分类器的输入。

本模块在特征提取技术上以在特征提取上表现较为出色的卡方检验（CHI）统计方法为基础，进行了该算法的缺陷分析和改进，产生了一种新的改进型 CHI 特征提取方法。

而在文本特征表示技术上，本文采用了向量空间模型的数学表示方法，并且用 TF*IDF 为向量的每个维度（即特征词项）加权，将最初的 URL 经爬虫和页面处理模块产生的关键词集合形式最终转化成了空间向量形式，成为了分类器可接受的数学化输入形式。

4.1.1 传统的卡方检验方法（CHI）

在数理统计领域，卡方检验方法是一种很常见的用于检验两个变量独立性的方法，其核心思路就是通过比较实际值和理论值两者之间的差异来判断假设是否成立，先假设两个变量互相独立，然后计算实际值和理论值的实际偏差程度来判定是否可以接受假设成立。实际偏差我们称为卡方值，如果卡方值较大，则不接受假设，如果卡方值较小，则认为假设成立。

应用到特征提取问题上，就是假设某个词 t 和特征 c 不相关，然后计算 t 和 c 的卡方值，卡方值越大则说明 t 和 c 越相关，t 对于 c 类就越有参考价值，t 就越应该被当做特征。

原理基本说明白了，现在来看一个简单的例子：

假设此时我们有 N 篇训练文档，其中有 M 篇是属于类别“教育”类的，现在要来研究一个词“高考”和类别“教育”的相关程度。虽然人脑一下就可以看出两者非常相关，但是电脑不同于人脑，只能老老实实地计算两者的相关性。

我们需要在训练集里统计四个值：

（1）包含关键字“高考”而且属于“教育”类的文档数，设为 A；

（2）包含关键字“高考”但不属于“教育”类的文档数，设为 B；

（3）不包含关键字“高考”而且属于“教育”类的文档数，设为 C；

（4）不包含关键字“高考”而且不属于“教育”类的文档数，设为 D；

明显，A+C=M，B+D=N-M，A+B+C+D=N。

接下来按照卡方检验思路建立公式（以包含关键字“高考”而且属于“教育”类为例）：

首先，需要一个假设，我们假设“高考”和“教育”没有关联；

其次，思考理论值，对于所有的类别，“高考”一词出现的文档数量应该是相等的，而无视文档是否是“教育”类，即每个类别下有“高考”出现的文档概率应该是相等的，这个概率由式（4-1）给出：

$$p=\frac{A+B}{N} \tag{4-1}$$

那么，理论上“教育”类包含“高考”的文档数 X 即为教育类文档数和概率的乘积，如式（4-2）：

$$E=\frac{(A+C)\times(A+B)}{N} \tag{4-2}$$

再次，思考实际值，其实实际值显而易见，就是 A；

最后，计算偏差值d_{11}，见式（4-3）：

$$d_{11}=A-E \tag{4-3}$$

而包含关键字“高考”但不属于“教育”类、不包含关键字“高考”而且属于“教育”类、不包含关键字“高考”而且不属于“教育”类这三种情况的差值计算也类似。在这四种情况的实际偏差值$d_{11},d_{12},d_{21},d_{22}$都计算出来后求和，得到卡方检测公式见式（4-4）：

$$\chi^2(\text{教育},\text{高考})=\frac{N(AD-BC)^2}{(A+C)(B+D)(A+B)(C+D)} \tag{4-4}$$

由于 A+C=M，B+D=N-M，A+B+C+D=N。上式的所有常量都可以略去，如此一来，最终的词 t 和类别 c 的卡方检验公式如式（4-5）：

$$\chi^2(t,c)=\frac{(AD-BC)^2}{(A+B)(C+D)} \tag{4-5}$$

其中：A 表示 c 类文档中包含词 t 的文档的数量，B 表示非 c 类文档中包含词 t 的文档的数量，C 表示 c 类文档中不包含词 t 的文档的数量，D 表示非 c 类文档中不包含词 t 的文档的数量。

CHI(t,c) 的值越大，特征 t 与类别 c 越相关；CHI(t,c) 的值越小，特征 t 与类别 c 越不相关。特别地，当 CHI(t,c)=0 时，表示特征 t 与类别 c 互相没有关联。

设类别 c 中所有文档共有 n 个词，每个词表示为$t_i(1<=i<=k)$，则类别下特征提取问题就是选择出该类别下$\chi^2(t_i,c)$最大的那 k 个词，至于 k 值的选择由实验效果不断进行人工调整。

4.1.2 传统的卡方检验方法的缺陷分析

分析传统卡方检验公式，我们可以发现以下几点缺陷：

（1）由于 A+C=M，B+D=N-M，A+B+C+D=N，可以得出 A 值越大，则 C 值越小，从而$\frac{A}{C}$越大；对应的是，B 值越小，D 值越大，从而$\frac{B}{D}$值越小。所以，卡方检验公式的核心就是寻找$\frac{A}{C}>\frac{B}{D}$的特征项，亦即 AD-BC 越大的值越应该被选中。但是从式（5）的分母$(AD-BC)^2$可看出，虽然 AD-BC 越大，卡方值也越大，但是这里只要$|AD-BC|$越大卡方值就越大，也就是说，BC-AD 越大（即 AD-BC<0 且越小），卡方值也越大。此时 B 和 C 都比较大，而 A 和 D 都比较小，对应的情况就是某词在特定类别下出现次数比较少，而在其他类别下出现次数比较多，此时按公式计算出的卡方检验值也比较高，所以在特征提取时这个词也有可能被选出来作为特征词。这说明传统卡方检验公式有时会产生负相关现象：某个在指定类别下出现次数很少而在其他类别下出现很多的词可能会被选中作为特征词；

（2）CHI 统计只统计了特征出现在哪些文档，而没有考虑特征在每个文档中出现的次数。也就是说，它仅考虑了特征词出现的文档数量，而没有考虑这个特征词在每个文档中的词频，所以使得低频词和高频词在特征提取时被一视同仁，但是实际上高频词和低频词相比更应该作为特征项。我们考虑这么一种情况：某词在特定类别下仅在少量篇文档中出现，但是在这些文档中该词的词频都比较高，由于传统卡方检验对词频的无视，计算出的 CHI 值可能会很小，但是这个词由于词频较高，可能对分类指导作用很大；

（3）考虑这样一种情形，某个特征项在 10 个类别中均有出现，而另一个特征仅在 2 个类别中出现，那么明显后者应该比前者更具类别区分能力，但是在传统卡方检验方法中却没有考虑一个词出现在的类的数量，如果一个词出现在的类别比较多，权重应该低一些；

（4）由于我们的训练数据比较大，直接造成的影响就是使得 D 值远远大于 A、B、C 值。再看式（5），由于 D 值远大于 A、B、C，使得当 A 很小而 B 很大时，

由于 D 值的影响，AD-BC 的值仍然很大。但是显而易见的是，A 值很小而 B 值很大的词项不太适合被当做一个特征项。

4.1.3 一种改进的卡方检验方法

对上一小节分析得出的四个缺陷，我们需要逐个改进：

（1）在计算 CHI 前先判断 BC>AD 是否成立，如果成立，不再计算，直接 CHI=0，否则继续计算，如此一来，直接避免了负相关问题；

（2）引入词频因素，每个特征的平均词频设为 TF，则引入的词频因素α=TF；

（3）引入特征词出现的类的数量因素，假设特征词 t 在训练集中的 x 个类别中都有出现，y 为训练集总类别数，则引入的类数量因素$\beta=\log\frac{y}{x}$，从这个因素公式看到，如果特征词出现在很多数类中，β值较小；而特征词出现在很少数类中，β值较大；

（4）为了适当改进 D 太大时 A 少 B 多仍然 CHI 很大的问题，引入一个比重$\gamma=(\frac{A}{B})^2$，即比较特征词在本类中的数量和不在本类中的数量比，引入这个因子，可以适当提高“A 大 B 小”情况的卡方值，且适当减小“B 大 A 小”情况的卡方值；

综上，新的卡方统计公式见式（4-6）：

$$\chi^2(t,c)=\begin{cases}\dfrac{(AD-BC)^2}{(A+B)(C+D)}\times\alpha\times\beta\times\gamma & \text{AD>BC}\\ 0 & \text{AD<BC}\end{cases} \qquad (4\text{-}6)$$

这样，基于改进的新型卡方检验方法基本成型。在这个新的改进公式中，消除了传统 CHI 方法的负相关问题，弥补了传统 CHI 方法对词频考虑的欠缺，弥补了传统 CHI 方法对词在类别间分布考虑的欠缺，缓解了训练数据太大造成的 A、B 对比度不明显的问题，特征提取的效果可以得到一定提高，剔除了大量对于分类作用比较小的特征和错误特征。

4.2 文本特征表示介绍

计算机不是人脑，它不会轻易地“读懂”词，无法理解词的意思，所以必须将词转化为计算机可以理解的形式，即数学化形式。在信息处理领域，为了方便文本的分析和处理，向量空间模型（VSM）的文本表示方式被广泛采用。向量空间模型的主要思想在于：用一个高维向量$W=(w_1,w_2,...)$来表示文本，其中，向量的每个维度代表每一个特征项（特征提取后的特征），w_i表示特征的权重。一个

HTML 文档经页面处理过后的短文本词集合设为 set，则该文档转换为向量空间表示的步骤为：

（1）先建立一个向量 V=(0,0,0,...)，维度为特征的个数；

（2）对比特征集合和 set，对 set 拥有的每个特征 i 对应的维度下计算权重 w_i；

如此，HTML 短文本词集合便转化为数值化的向量。那么权重应该如何计算？在最初的向量表示是以 0-1 形式表示的，即如果特征存在于文档，则对应的文档向量的那一维的权重为 1，否则为 0。但是这种方法过于简单，完全忽略了词在文档中的作用程度。所以我们需要分析词在文档中的重要性由哪些因素来决定。

4.2.1 体现词在文档中权重的关键因素分析

对于一篇文档而言，文档中出现次数较多的词是否应该在文档中的权重较大？凭直觉考虑的话，答案是肯定的；

对于多篇文档而言，如果一个词 t_1 在每一篇中都有出现，而另一个词 t_2 仅在一篇文档（设为 D）中出现，那么对于 D 文档而言，t_2 的权重和 t_1 的权重哪个更大一些？

更复杂的情况，如果一个词 t_1 在每一篇中都有出现，而且在一篇文档 D 中出现的次数较多；而另一个词 t_2 仅在文档 D 中出现，但是出现次数较少，那么此时对于 D 文档而言，t_2 的权重和 t_1 的权重哪个又更大一些？

这三种情况仅仅是一些最一般的情况，真实的情况要复杂得多。可见我们不能单单考虑词的出现次数，或者单单考虑词在各个文档中分布情况，我们需要一种方法可以相对合理地描述词的权重。下一小节介绍一种在信息检索领域上很常见的特征权重计算方法：TF*IDF 方法。

4.2.2 TF*IDF 方法

TF*IDF 方法是信息检索常用的词汇权重度量标准。对于某篇文档中的某个词，它用这个词在文档中的词频和这个词本身的预测主题的能力来度量词的权重。这也是 TF*IDF 方法的核心思想。

从 TF*IDF 字面来看，方法实际上就是词频 TF 和逆文档频率 IDF 的乘积，也就是对两者的综合考量。其中，TF 就是词在某文档中出现的频率，而 IDF 则可以反映一个词的主题预测能力。

TF 是词在某特定文档的频率，反映了词在文档中的出现频繁度。之所以用频率而不是直接使用词的计数值，主要是考虑到文档各自长短不一，在长文档中出现次数多的词未必就比短文档中出现次数少的词更重要。某文档 D_j 中词 t_i 的词频

$tf_{i,j}$，用式（4-7）来计算：

$$tf_{i,j}=\frac{n_{i,j}}{\sum_k n_{k,j}} \tag{4-7}$$

其中，$n_{i,j}$表示词t_i在文档D_j中的出现次数，而$\sum_k n_{k,j}$表示文档D_j中所有词的总计数和。

IDF 指逆向文档频率，是对一个词的主题预测能力或者重要性的一种数学化的度量。先从一个简单的例子入手：1000 篇文档中，有 10 篇文档包含“云计算”这个词，有 600 篇文档包含“技术”这个词，更有 1000 篇包含“我们”这个词，这里面哪个词最有主题预测能力？哪个词最没有这种能力？答案分别是“云计算”和“我们”这两个词。因为前者仅出现于 10 篇文档，说明该词很罕见，而后者出现在所有文档中，说明该词很常见。IDF 就是基于总文件数与包含给定词的文档数相除的方式（当然，是除法的改进）来反映词汇的主题预测能力或普遍重要性的。IDF 计算方法见式（4-8）：

$$idf_i=\log\frac{|D|}{|\{j:t_i\in d_j\}|} \tag{4-8}$$

其中，D 表示所有文档，$|D|$表示总文档数，$|\{j:t_i\in d_j\}|$表示包含词汇t_i的文档数。假如不在任何文档中，$|\{j:t_i\in d_j\}|$值为 0，所以一般以$|\{j:t_i\in d_j\}|$+1 做分母来保证除法的有效性。

TF*IDF 的最终公式见式（4-9）：

$$tfidf_{i,j}=tf_{i,j}\times idf_i \tag{4-9}$$

应用到本设计中就是：对于每个特征项t_i，计算特征在给定文档D_j短文本词集合中对应的 TFIDF，从而来表示这个文档的文档向量。

至此，从最初的 URL 到最终的向量空间模型表示过程已经完成。经过这一步，以向量形式表示的训练集已经完全建立，用户输入的 URL 到向量的转化流程也完全建立。

4.3 本章小结

本章介绍了特征提取技术和文本特征表示的技术。

特征提取技术的章节中重点引入了卡方检验（CHI）方法，以及对 CHI 方法本身缺陷的分析，在对缺陷的改进和修补后，产生了一种基于改进的新型 CHI 特征提取方法并且应用到本设计中。

文本特征表示技术主要介绍了向量空间模型（VSM）以及特征词权重的计算

的 TF*IDF 方法，这一步是为了将前一章生成的 HTML 短文本词集合最终转化为可供分类器接受的数学化形式。

在特征提取技术和文本特征表示模块完成后，分类器模块可以正式工作了：一方面训练集在经过 VSM 表示后终于可以供分类器去学习了，另一方面 URL 分类器相当于终于可以接受用户输入了。下一章将详细介绍 KNN 分类器模块，这一模块更是本文的重中之重。

第 5 章 KNN 分类器模块

5.1 传统 KNN 算法介绍

K-邻近（KNN）算法是一种基于实例的学习方法，代表 K 个最邻近分类法，通过分析与输入样例最相似的 K 个历史记录（训练数据）来为输入样例分类。KNN 算法被认为是 VSM 下最好的分类方法之一，方法简单、易于实现、实用性强，但是传统的 KNN 算法的缺陷也非常明显，比如饱受诟病的分类速度太慢和分类精度不高等缺陷。

传统的 KNN 算法应用于文本分类的基本思路是：在给定新文本后，考虑在训练文本中与新文本最相思的 K 篇文本，根据这 K 篇文本所属的类别，统计类别出现次数，出现次数最多的那个类别被认为是新文本的类别。

具体的步骤如下：

（1）输入待分类的文本的 VSM 表示形式；

（2）在训练集合中找出与新文本向量最为相似的 K 个文本，其中两个文本向量的相似度计算比较常用的是余弦值法，计算公式如式（5-1）：

$$Sim(d_i,d_j)=\frac{\sum_{k=1}^{M}W_{ik}\times W_{jk}}{\sqrt{(\sum_{k=1}^{M}W_{ik}{}^2)(\sum_{k=1}^{M}W_{jk}{}^2)}} \qquad (5\text{-}1)$$

其中，K 值的确定一般由实验结果来进行人为调整；

（3）在输入文本的 K 个最邻近邻居中依次计算每类的权重，如式（5-2）：

$$p(\vec{x},C_j)=\sum_{\vec{d}_i\in KNN}Sim(\vec{x},\vec{d}_i)y(\vec{d}_i,C_j) \qquad (5\text{-}2)$$

其中，$\vec{x}$ 为新文本的特征向量，$Sim(\vec{x},\vec{d}_i)$ 为相似度计算公式，而 $y(\vec{d}_i,C_j)$ 为类别属性函数，即如果 $\vec{d}_i$ 属于类 C_j，那么函数值为 1，否则为 0；

（4）比较类的权重，将权重最大的那个类别赋给文本的类别 c_{out}，见式（5-3）：

$$c_{out}=\arg\max_{c_j\in C}p(\vec{x},C_j) \qquad (5\text{-}3)$$

5.2 传统 KNN 算法的缺陷

KNN 算法作为一种惰性学习方法，训练集对于 KNN 算法而言平时只是一般地保存起来，直到输入未标注样本时才真正建立分类器，用训练集中每个训练样本

和未标注样本进行余弦值相似度计算，而后选取相似度最高的 K 个训练样本，看哪个类别出现最多，就把输入样本标注为哪个类别。

这里可以看出三个明显的缺陷：

（1）用训练集中每个训练样本和未标注样本进行相似度计算，带来了巨大的计算开销；

（2）在相似度计算上，仅仅用特征向量进行简单余弦计算，没有考虑特征项对类别的指导作用，影响分类精度；

（3）在 K 个最相似样本中，仅仅简单地统计每个类别出现的次数，选取出现最多的那个类作为分类依据，这样的决策过于简单、武断，严重影响分类精度。

以上是比较明显的几个问题，更多影响分类速度和分类精度的因素在之后的章节中进行详细分析。

得知传统 KNN 算法的缺陷是分类速度和分类精度后，我们接下来从速度和精度这两个角度分析各自缺陷的产生原因并逐步改进。

5.3 在运行速度上改进 KNN 算法

本节对传统 KNN 算法在分类速度低下的缺陷产生原因上进行了详细分析，并对每个产生原因给出了对应的解决方案，通过在分类速度角度对传统 KNN 算法的改进，分类速度得到了大幅度改进，达到了高效分类的效果。

5.3.1 传统 KNN 算法运行速度低下的原因分析

前面提到，用训练集中每个训练样本和未标注样本进行相似度计算，为传统 KNN 算法带来了巨大的计算开销。这是算法速度慢的主要原因之一。

我们再来看相似度计算公式，是利用了向量余弦值计算的方式。为了看清楚相似度计算的过程先举一个简单的例子：

向量 $v_1 =< 1,2,3 >$ 和向量 $v_2 =< 4,5,6 >$ 计算相似度，需要在 v_1 和 v_2 一一对应的每个维度上做乘法，之后把所有乘积做加法作为分子；接下来需要分别求 v_1 的长度和 v_2 的长度，也需要遍历每一维才能算出结果。

虽然这个计算过程非常简单而且在计算机上计算非常快，但是我们把实例拓展一下，我们在特征提取和文本特征表示模块后生成的 VSM 形式的训练集中每一个向量都有近 35000 维，我们就假设 v_1 和 v_2 的维度都是 30000，相似度计算由于需要遍历每一个维度而变得耗时，虽然此时的运算过程在计算机上仍是 1 秒以内完成，但是 KNN 算法在做相似度计算上需要计算多少相似度？大量的相似度计算和每个相似度计算的耗时进行加成后，会很明显地体现出相似度计算耗时的影响。

综上分析，影响分类速度的因素主要有两个：

（1）测试样本需要和所有训练样本比对；

（2）测试样本和训练样本都是高维度向量，相似度计算很耗时。

一方面，我们需要引入一些决策来减少与测试样本与的训练样本的个数；另一方面，我们需要思考从哪里入手改进相似度计算的时间复杂度。

5.3.2 用 Rocchio 算法进行预选候选类

当分类器收到一个输入样本，我们可以先为这个样本找出它可能属于的类别，即为这个样本找出一个候选类别集合，仅计算属于候选类的训练样本和输入样本的相似度。这样一来，省去了输入样本和不太可能是其分类下的训练集的相似度计算，大大减少了要做相似度计算的样本数量。

那么，如何为输入样本找出其可能属于的类别来组成一个候选类集合？

这里引入 Rocchio 算法的思想：为训练集的每个类别构建一个“原型向量”来代表这个类别，输入样本仅与原型向量进行相似度计算，找出与输入样本相似度最高的那个原型向量，其所属类别判断为输入样本的类别。

虽然 Rocchio 算法看起来非常武断，但是我们可以借鉴其思想作为输入样本的候选类的选取方式：为训练集的每个类别构建一个“原型向量”来代表这个类别，输入样本先与这些原型向量进行相似度计算，找出与输入样本相似度相对比较高的那些原型向量，它们所属的类别作为候选类别集合，之后，KNN 分类器仅需要和所有属于候选类别的训练样本进行相似度计算，这样一来，计算量的开销得到大幅度减小。

这里又有一个新的问题，原型向量怎么建立？我们采用了在信息检索中的最右代表向量来做原型向量。每个类的最优向量计算公式如式（5-4）：

$$\vec{q}_{opt} = \alpha \times \frac{1}{|C_r|} \sum_{\vec{d}_j \in C_r} \vec{d}_j - \beta \times \frac{1}{|C_{nr}|} \sum_{\vec{d}_j \in C_{nr}} \vec{d}_j \qquad (5\text{-}4)$$

其中，C_r 表示本类的文档集合，C_{nr} 表示非本类的文档集合，$\vec{d}_j$ 表示训练样本向量，α 和 β 是参数，由人为给定，经过试验认为 α=16，β=4 比较合适。

从上面这个公式看出，每个类别的最优代表向量就是本类向量集合的质心向量与非本类向量集合的质心向量差的改进，α 和 β 分别用来扩大本类质心向量权重和减小非本类质心向量的权重。

在我们为训练集中每个类别建立好最优代表向量之后，当新文本被输入时，先让新文本的特征向量与这些质心向量做相似度对比，找出相似度大于平均值的那些质心向量，把这些质心向量所代表的类作为 KNN 的候选类，只计算这些类下的

训练文本，极大减少了计算量。

通俗一点说就是我们利用了 Rocchio 算法思想对输入文本所属类别进行了一次预先分类，这一决策可以比喻为选秀节目的海选：参选人是每个类别，海选规则是根据和输入文本的相似度得分来排名，海选结果是得分高于平均以上的那些类别，之后再让从海选脱颖而出的类别去参加下一轮“比赛”。

5.3.3 根据文本的特征集与每类特征交集再次筛选候选类

如果输入样本所拥有的特征项（即向量维度不为 0）集合如果和某个类别的特征集合没有任何交集，那么输入样本就不会属于这个类别，也没有必要和这个类别下的那些训练样本作比较。

这个思路很简单，实现也很简单：我们承接从 Rocchio 预分类得到的候选类，在候选类中对每一个类统计这个类的特征集与新文本拥有的特征的交集，如果交集为空，那么这个类被抛去，最后形成再次被筛选的新候选类集。

5.3.4 建立倒排索引

前面两小节尝试通过减少需要计算相似度的训练样本所属类别的方式来削减计算量，下面从削减类内训练样本的角度来继续减少需要与输入样本进行相似度计算的训练样本个数。

试想一个场景：当输入样本在某特定类下对所有从属该类的训练样本做相似度计算时，如果输入样本和某个训练样本恰好在各自拥有的特征项上没有交集（即同样的维度上不会同是非 0 值），那么按照相似度计算公式，这两个样本向量的相似度必然为 0。如此一来，输入样本与这个训练样本做了一次无用的相似度计算，本文的思路就是事先从类别下的训练样本集中排除那些相似度必然为 0 的样本，然后再和剩余训练样本计算，通过排除相似度必然为 0 的训练样本，相似度计算量也会得到一定减少。

怎么确定和输入样本相似度必然为 0 的训练样本？很简单，只要这个向量与输入样本向量没有公共的非 0 维度，即两个向量包含的特征项是没有交集的。

所以，对于输入样本中的每个特征项，我们仅需要找出所有包含这个特征项的训练样本，在每个输入样本的特征项都找到对应的训练样本后，未被找出来的训练样本就是和输入样本包含的特征集没有交集的样本，即相似度必然为 0 的样本。

于是，我们需要建立一个用于查询某特征项与包含这个特征项的训练样本的映射关系的数据结构，具体实现如下：对于一个类别，以每一个特征项作为键 key，以所有包含这个特征的训练文本 ID 集合作为值 value，建立倒排索引，即建立一

个链式哈希表。

在为每个类别建立了倒排索引之后，输入样本在和某类的训练样本计算相似度之前，先在该类的倒排索引上查询哪些训练样本需要和输入样本进行计算，之后再和查询到的训练样本做相似度计算。

综上所述，本方法对特定类内参与相似度计算的训练样本集进行裁剪，也一定程度减少了需要相似度计算的训练集数量，进而提高分类速度。

5.3.5 引入位置向量表示法来降低高维向量计算量

前面几小节尝试从减少相似度计算次数的角度来不断裁剪需要参与运算的训练样本个数，进而提高分类速度，接下来我们从提高相似度计算公式效率的角度来提高分类速度。

我们在前面提到，由于向量的维度非常大，而在相似度计算中需要遍历每个维，使得相似度计算上很耗时。

从相似度计算公式上我们很容易看到，对于两个向量，相似度计算公式的分子其实就是计算公共的维度上值的乘积和。假设相似度计算时正在遍历某一维，一个向量上对应的维的值为 0，而另一个向量该维不为 0，乘积仍然是 0，使得遍历这一维不会对相似度有任何影响；而一个向量上对应的维的值不为 0，另一个向量该维也不为 0，乘积则非 0，相似度的值会变化（增加）。

显然，真正对相似度计算有影响的仅仅是两个向量上对应值非 0 的公共维度的乘积和。如果两个向量同一维度上有 0 值，那么这一维度完全可以跳过不计算。而训练集的每个训练数据 VSM 向量都是由各自对应的 HTML 短文本词集合变换而来，在词到 VSM 的变换中，我们提到，如果在短文本中没有某个特征，那么其 VSM 上对应的特征维下的值是 0。那么可想而知，由于短文本中本身也没有太多的词，其 VSM 上真正非 0 的维度也不会有太多。

目前我们得出的分析是，向量虽然维度高，但是非 0 维少，恰好相似度只受两个向量的公共非 0 维的影响。那么，对于两个向量，由于各自非 0 维很少，两者的公共非 0 维普遍会更少（至多和非 0 维较少的那个向量的非 0 维大小一样），我们仅对两个向量的公共非 0 维进行遍历，可以大大减少相似度计算中因对每个维度都遍历造成的耗时问题。

本文决定用一种叫做“位置向量”的方式来表示向量，“位置向量”定义如下：

对于一个向量 $W=(w_1,w_2,...)$ 的每一位，如果这一位上值非 0，则记录位置和对应值，否则略过，那么向量就可以表示为公式（5-5）：

$$W=<(position_1,value_1),(position_2,value_2),...> \qquad (5\text{-}5)$$

其中 position 为位置，value 为该位置的对应值。

从上面的公式可以看出，“位置向量”其实就是仅保存非 0 维的位置和对应值的表示形式。

现在我们用一个例子来看这种新的向量表示方法的有效性：

设位置向量 $pv_1 =< (1,3),(100,4),(3000,2),(10000,7),(25000,6),(30000,8) >$，向量 $pv_2 =< (1,7),(8,9),(100,4),(3020,2),(6700,7),(10000,9),(25000,1),(30000,3) >$，两者的真实维度都是 35000；

从位置向量看出，pv1 在 1,100,3000,10000,25000,30000 维上有非 0 值，pv2 在 1,8,100,3020,6700,10000,25000,30000 维上有非 0 值；

那么两者的相似度计算过程如下：

先计算 pv_1 长度：

$$| pv_1 |= \sqrt{3^2 + 4^2 + 2^2 + 7^2 + 6^2 + 8^2} \approx 13.342$$

再计算 pv_2 长度：

$$| pv_2 |= \sqrt{7^2 + 9^2 + 4^2 + 2^2 + 7^2 + 9^2 + 1^2 + 3^2} \approx 17.029$$

计算向量乘积，选取两个向量的公共维，即 1,100,10000,25000,30000：

$$multi = 3 \times 7 + 4 \times 4 + 7 \times 9 + 6 \times 1 + 8 \times 3 = 130$$

带入相似度公式求相似度：

$$sim = \frac{multi}{| pv_1 | \times | pv_2 |} \approx 0.572$$

从仅用两个位置向量的公共位来计算相似度时，在 pv_1 长度计算中遍历了 6 维，在 pv_2 长度计算中遍历了 8 维，在向量乘积中遍历了 5 维，而传统的向量在同样的三个步骤则分别需要遍历 35000、35000、35000 维。

相比来看，“位置向量”大大降低相似度计算时的维度遍历，抛弃了在大量维度上无用的计算（计算结果为 0），使得相似度计算的效率得到大幅度提升。

在验证了“位置向量”的可行后，我们从此将不再使用传统的向量，我们需要修改之前的特征提取和文本特征表示模块，把其中定义的“短文本词生成 VSM 传统向量”的函数改为生成 VSM 位置向量的新函数。

5.3.6 快速 KNN 算法的系统流程

在经过了上面几个小节的策略设计，KNN 算法在运行速度上的问题已经得到了比较完善的解决，在实现环境下，传统 KNN 算法平均分析一个输入样本需要近 1 分钟之久，而改进后的快速 KNN 算法则平均仅需 880ms。

我们再通过流程图来总结一下本文采用的 KNN 提速的 4 个改进策略，以及这些策略的互相结合。改进后的快速 KNN 算法流程图如图 5-1：

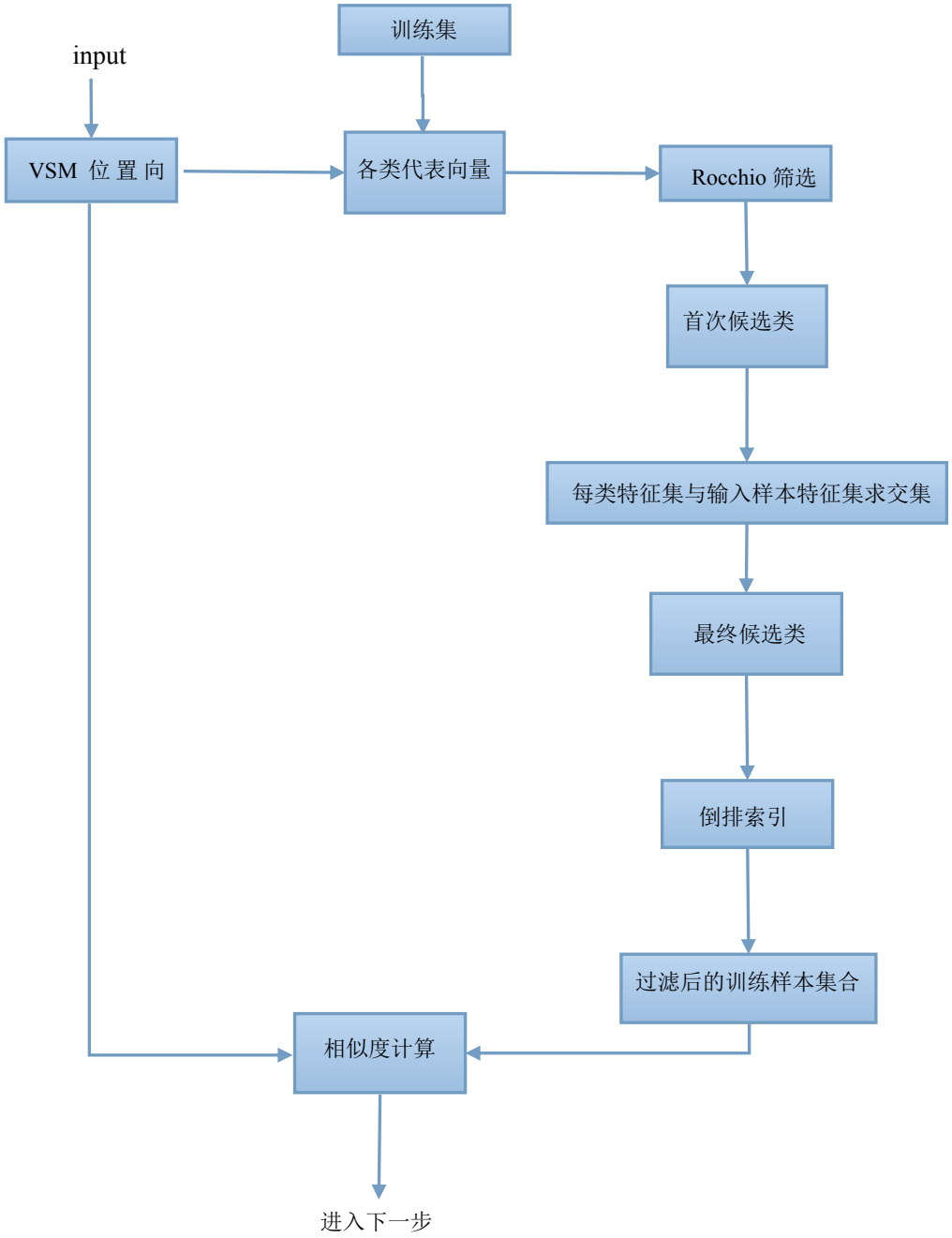

图 5-1 快速 KNN 算法流程图

由于对 KNN 算法分类精度的改进将在后面介绍，于是本流程图中将具体的分类算法先暂时略去。

5.4 属性熵介绍

在正式开始分析传统 KNN 算法在分类精度上的缺陷之前，本文先引入对熵的介绍。通过对熵的定义和熵的意义的理解，才能合理地改进 KNN 算法在分类精度上的不足。

5.4.1 熵的定义

现在，引入本次设计的一大核心-属性熵，设数据集为 S，系统中共有 $C_1, C_2, ...C_n$ 共 n 类。对于数据集中的每一个特征词 i（即属性），其属性熵值 $Entroy_i$ 反映了特征词在每个类别中的分布情况，计算公式如式（5-6）：

$$Entroy_i = -\sum_{j=1}^{n} p_{ij} \ln p_{ij} \qquad (5\text{-}6)$$

其中，$p_{ij} = \frac{|v_{ij}|}{|v_i|}$，$|v_{ij}|$表示在 j 类中属性 i 出现的文档数，$|v_i|$表示在数据集中一切包含属性 i 的文档数。

根据上式可知，如果一个特征词出现在的类别数较少，那么相应的熵就比较小，而如果特征词出现在很多类，那么其熵值很大，特别的，如果一个特征词只属于一类，那么可以看出其熵值为 0。

5.4.2 属性熵值的意义

由于从式（1）可以得出属性信息熵的意义：

如果一个特征词的属性熵值很小，则说明这个特征分布比较集中于较少的类别之中，而如果一个特征词的属性熵值很大，则说明这个特征分布相对分散，在较多的类别中有分布。属性信息熵描述了特征项的分类指向不确定性，一个特征项的分布越是混乱，属性熵值就越高，对分类的指向性就越不确定；一个特征的分布越是有序，属性熵值就越低，对分类的指向性就越明确。

5.5 在分类精度上改进 KNN 算法

本节对传统 KNN 算法在分类精度上的缺陷产生原因上进行了详细分析，并对每个产生原因给出了对应的解决方案，通过在分类精度角度在快速 KNN 分类器基础上的再次改进，分类精度得到了大幅度改进，达到了高精度分类的效果。

5.5.1 传统 KNN 算法分类精度低的原因分析

首先，在相似度计算上，传统 KNN 算法仅仅用特征向量进行简单余弦计算，

没有考虑特征项对类别的指导作用，影响分类精度；

其次，在 K 个最相似样本中，仅仅简单地统计每个类别出现的次数，选取出现最多的那个类作为分类依据，这样的决策过于简单、武断，严重影响分类精度；

再次，在传统 KNN 算法中，每个特征项和每个邻近样本被等同对待，也在影响分类的精度；

最后，还有一些别的因素会对算法造成影响，比如 K 值的选取不当，训练集数据分布不均匀等。

接下来的章节针对这些影响分类精度的因素进行算法改进。

5.5.2 引入共有特征个数改进相似度计算公式

首先，相似度计算上面，用余弦值作为计算标准，会带来向量计算本身的问题，我们以一个例子来说明这种问题：

假如目前有向量 $v_1 =<10,1,1,1,1,1,0,1,0,0>$ 和向量 $v_2 =<10,0,0,0,0,0,0,0,1,1>$ 进行相似度计算。我们先观察到这两个向量只有一个公共维，即这两个向量的共有特征只有一个，相似度不应该很高。但是用余弦值计算得出两者相似度值为 0.96，这种不合理情况应该被稍加改进。

两个向量的共有特征数量在一定程度上反映了相似度的高低，而目前本文使用的相似度计算却没有考虑这一因素，于是我们引入新的相似度计算公式（5-7）：

$$Sim(d_i,d_j)=\frac{\sum_{k=1}^{M}W_{ik}\times W_{jk}}{\sqrt{(\sum_{k=1}^{M}W_{ik}{}^2)(\sum_{k=1}^{M}W_{jk}{}^2)}}\times\frac{n}{\min(|d_i,d_j|)} \qquad (5\text{-}7)$$

其中，新的分母表示两个向量中较小的维度，n 表示两个向量共同拥有的向量维的个数。这样一来，新的因子必然是一个介于 0,1 的数，可以对上述可能出现的相似度太大进行一定的降低，而且对共同维度较多的向量来说，这个因子相对比较大，可以适当提高相似度。

5.5.3 引入属性熵值再次改进相似度计算公式

再考虑相似度计算公式，公式使用的是 VSM 向量来进行相似度计算的，而 VSM 向量每一维的权重是 TF*IDF，也就是说，这里特征词的权重仅仅是特征在单一文本中权重。此时的相似度计算中，文本中更关键的词对相似度的改变有比较大的影响，而不是很关键的词对相似度的改进影响不大。但是，在文本中更关键的特征词却不一定比别的特征词更具分类的指导意义，有可能一个特征词在文本中很重要，但是对分类的指导能力太小，而另一个特征词在文本中虽然不重要，

但是对分类的指导能力很大。

从上面看到，目前的相似度计算上没有给对分类指导能力较大的特征词更高的权重，即没有体现每个特征项对分类的指导作用的大小。在相似度计算上，对分类指导能力强的特征项应该对相似度计算有更重要的影响，而对分类指导能力不强的特征项应该对相似度计算的影响比较弱。对于目前的相似度计算来说，这是一个很大的缺陷。

那么这里有两个问题：

（1）如何体现每个特征词的分类指导能力；

（2）如何把每个特征词的分类指导能力引入相似度计算公式，使得对分类指导能力强的特征项应该对相似度计算有更重要的影响，而对分类指导能力不强的特征项应该对相似度计算的影响比较弱。

对于第一个问题，之前我们引入了属性熵的概念：属性信息熵描述了特征项的分类指向不确定性，一个特征项的分布越是混乱，属性熵值就越高，对分类的指向性就越不确定；一个特征的分布越是有序，属性熵值就越低，对分类的指向性就越明确。也就是说，特征词的分类指导能力可以由其属性熵来表示。第一个问题引刃而解。

而对于第二个问题，属性熵值越低，属性对分类指导能力越强，而相似度计算要体现出指导能力强的特征词（属性）的更大的权重。也就是说，属性熵值越低的特征词，权重应该越大。这里我们在公式（5-7）的基础上再一步改进了新的相似度计算公式，见式（5-8）：

$$Sim(d_i, d_j) = \frac{\sum_{k=1}^{M} \frac{W_{ik} \times W_{jk}}{Entropy_k}}{\sqrt{(\sum_{k=1}^{M} W_{ik}^2)(\sum_{k=1}^{M} W_{jk}^2)}} \times \frac{n}{\min(|d_i, d_j|)} \qquad (5\text{-}8)$$

其中，$Entropy_k$ 代表两个向量中第 k 个公共特征词的熵值大小。

分析这个公式，由于对分类贡献度比较大的特征词 k 的熵值比较小，所以上式可以增大在相似度计算时词汇 k 所占的权重；反之，对于对分类贡献度比较小的词汇 k，计算相似度时会缩减词汇 k 所占的权重。

对于 $Entroy_i$=0 的属性 i，由于无法直接当分母，需要人工将这些属性的熵值设置为一个不大的非 0 值，根据分类效果不断调节，在本设计的数据集上，最终选择了 0.5。注意，这个值不宜选得太小。

如此一来，新的相似度计算公式（5-8）使得对分类指导能力强的特征项应该对相似度计算有更重要的影响，而对分类指导能力不强的特征项应该对相似度计算的影响比较弱。这个全新的相似度计算公式是特征词重要性权重和特征词指导

能力权重的结合考虑，改正了初期相似度计算公式对特征词分类指导能力的无视。

5.5.4 引入类别平均相似度改进在 K 邻居中各类权重公式

前面提到，传统 KNN 方法在计算出新文本的 K 个邻居后，仅对它们进行简单地统计类别，选出类别出现最多的那一类作为输出类别。这种决策方式太武断。

考虑如下情景：对于某个输入网站，设置 K=30，进行了如上的改进 KNN 后，得到相似度最高的 30 个训练样本，其中：

有 8 个属于 A 类，相似度依次是：0.9,0.85,0.8,0.75,0.7,0.65,0.6,0.55；

有 22 个属于 B 类，相似度依次是：0.6,0.58,0.5,0.45,0.43,0.4，0.38,0.36,0.33,...；

很明显，对于输入网站来说，被决策为 A 类更加合理，但是按照传统 KNN 的决策方式，22>8，于是把输入网站错误地归结为 B 类。

这个简单的例子充分体现了传统 KNN 决策的武断。传统办法仅仅考虑了个数，而没有考虑每个类别对于输入文本的相似度，所以本文首先引入了类别平均相似度进而提出了一种结合平均相似度和类别出现个数的新决策方式，为每个类别 K 邻近中出现的类别建立一个权重函数，函数公式如式（5-9）：

$$Weight_{c_i} = \alpha \times avg(sim_{c_{ij},input}) + \beta \times N_{c_i} \qquad (5\text{-}9)$$

其中，α，β 为权重因子，$avg(sim_{c_{ij},input})$ 是属于 K 邻近中属于 C_i 类的训练样本与新样本的相似度的算术平均值，N_{c_i} 是 K 邻近中 C_i 出现的个数。根据实验效果，α，β 取值为 0.8,0.2 比较合适。

5.5.5 引入类别贡献度再次改进在 K 邻居中各类权重公式

在上述改进后，仍然还有改进空间，在目前的改进中，没有体现出每个类别对输入样本的支持程度。传统 KNN 在决策之前，将各个类别等同对待，而更恰当的方式是引入一个类别间影响因子，用以体现一个本文对某个类别的隶属程度。回忆上面在改进 KNN 速度的时候引入的 Rocchio 预分类方法，在预分类时新文本与每个类的最优查询向量进行相似度计算 $sim(C_i, j)$，恰好可以表示输入样本对类别的隶属程度。

我们再次用到了 Rocchio 算法的思路，幸运的是，在本文改进 KNN 分类速度时利用 Rocchio 算法恰好计算了隶属程度。

我们具体这样来使用它：在 Rocchio 预分类得到的候选类别中，把输入样本对每个类别的隶属程度进行归一化，这样就得到了新文本在候选类别中分别隶属于每个类的最终隶属度 $\delta_{C_i,j}$。把隶属度引入上一小节建立的类别权重公式中，对类别

权重公式再次改进，现在修正权重计算公式见如式（5-10）：

$$Weight_{c_i} = (\alpha \times avg(sim_{c_{ij},input}) + \beta \times N_{c_i}) \times \delta_{C_i,input} \quad (5\text{-}10)$$

如此一来，新的类别权重计算公式又结合了新文本对每个类别的隶属度，类别的加权同时使得新的 KNN 算法有更好的鲁棒性，降低了 K 值选取对 KNN 算法分类的敏感度。

5.6 本章小结

本章重点在传统的 KNN 方法上进行缺陷分析，对 KNN 算法的运行速度问题和分类精度问题上做出改进。

在运行速度角度，本文的改进基于两个出发点：（1）减少需要与输入样本进行相似度计算的训练样本大小；（2）减少相似度计算的时间消耗。用 Rocchio 算法进行预选候选类，再根据文本的特征集与每类特征交集再次筛选候选类，先筛选出可能被判别的类别，再对每个类别建立倒排索引，逐步筛选出值得参与相似度计算的训练样本，从而需要与输入样本进行相似度计算的训练样本得到了大幅度降低。本文同时引入了“位置向量”的向量表示方法，通过这种方法避免相似度计算上遍历无用的维度，极大减少了相似度计算的时间损耗。

在分类精度角度，本文的改进重点在于在相似度计算上引入属性熵值来提高对分类指导能力较强的特征的作用，在 KNN 在 K 个邻居的决策时不仅考虑类别出现次数，更考虑了类别的平均相似度，输入样本对各类别的隶属程度，使得新的 KNN 方法在 K 个邻居的决策中不再片面、武断，而综合考虑了各方面因素。

高效、高精度的分类器通过改进 KNN 算法建立完成，接下来就需要对分类器的性能进行考验。下一章对分类器的实验环境和实验结果进行介绍。

第 6 章 实验测试与评价

6.1 分类标准和训练数据

本系统将目前网页分为 33 大类：

餐饮烟酒副食、电邮、房产家居、交友、体育、政法军事、宗教、时尚美容、影视、百货购物、文化、工业行业、新闻、游戏动漫、旅行、阅读、社科、音乐、交通物流、广告营销、休闲娱乐、农林牧渔、金融保险、卫生健康、人力招聘、咨询服务、教育培训、计算机类、家政礼仪、网络资源下载、门户搜索、博彩、色情。

训练集在数据库中的存储结构见图 6-1：

```
+-------------+------------------+------+-----+---------+----------------+
| Field       | Type             | Null | Key | Default | Extra          |
+-------------+------------------+------+-----+---------+----------------+
| id          | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| url         | varchar(100)     | NO   |     | NULL    |                |
| title       | varchar(100)     | YES  |     | NULL    |                |
| keyword     | varchar(200)     | YES  |     | NULL    |                |
| description | varchar(355)     | YES  |     | NULL    |                |
| content     | varchar(355)     | YES  |     | NULL    |                |
| icp         | varchar(20)      | YES  |     | NULL    |                |
| accesscode  | int(11)          | YES  |     | NULL    |                |
+-------------+------------------+------+-----+---------+----------------+
8 rows in set (0.00 sec)
```

图 6-1 训练集存储结构

部分训练集内容见图 6-2：

```
| 117 | www.yanpk.com | 烟网+香烟+零售+零售商+卷烟+香烟+网站 | 烟草+烟草+公司+卷烟+供货+假烟+名单+黑名单+卷烟+香烟+价格+烟草+公司+烟店+假烟+常识+烟草+公司+招标+烟草 | 烟网+香烟+网站+卷烟+商店+烟民+卷烟+品牌+评价+卷烟+零售+零售店+主页+烟草+草行+行业+烟草行业+招标+卷烟+零售+客户+卷烟+品牌+资料+烟民+交流+网站+烟民+交流+社区+香烟+价格+香烟+图片+权威+卷烟+品牌+资料+资料库 | uk | 13038831 | 0 |
| 118 | www.yanyue.cn | uk | 香烟+烟民+香烟+价格+真假+烟草 | 烟民+交流+网站+烟民+交流+社区+香烟+价格+香烟+图片+香烟+权威+香烟+资料+资料库+本站+出售+香烟 | uk | 090037 | 0 |
```

图 6-2 部分训练集内容

正如图中所示，存储结构包含了 URL、以及网页的 TITLE 标签、META 标签、正文段文本分词后的词集。

6.2 测试结果

本系统采用了 3578 个真实网页作为测试集。

我们将传统 KNN 算法、加权 KNN 算法以及本设计的基于属性熵值分析的快速 KNN 方法进行了分类时间和分类精度的对比。

为了方便，本设计的基于属性熵值分析的快速 KNN 方法在接下来的内容中我们为其命名为 LCX-KNN 方法。

首先，在运行时间测试上我们只用 100 个测试数据（因为传统 KNN 方法在 3578 个测试数据上运行时间相当漫长，不方便测试），测试效果见表 6-1：

表 6-1 分类时间测试统计

分类方法	平均分类时间(s)
传统 KNN	58.54s
加权 KNN	58.43s
LCX-KNN	0.88s

其次，在分类精度上的测试我们完全使用 3578 个测试数据对三种方法进行测试，测试效果见图 6-3：

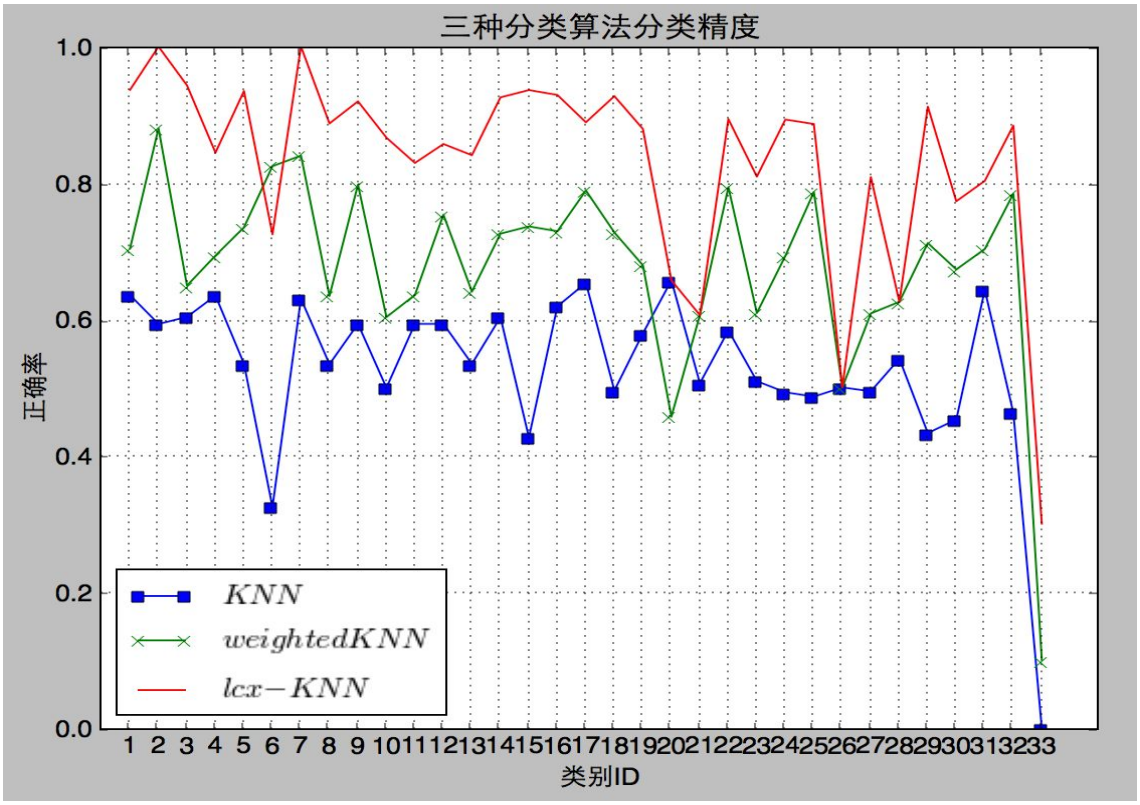

图 6-3 分类速度测试统计

总分类精度测试结果统计见表 6-2：

表 6-2 总分类精度测试统计

分类方法	分类精度(%)
传统 KNN	51.40%
加权 KNN	69.98%
LCX-KNN	85.05%

从表 6-1 可知，由于传统 KNN 方法和加权 KNN 方法由于在运行时间上没有任何改进，在 100 个测试数据上耗时严重，平均分类每个数据要用将近 1 分钟之久；而 LCX-KNN 方法运行时间达到了毫秒级，1 秒钟都不到。

从图 6-3 和表 6-2 可知，由于传统 KNN 方法由于没有任何改进，分类效果最差；加权 KNN 方法分类效果达到了 69.98%，比传统 KNN 方法有所提高；而 LCX-KNN 方法分类效果最佳，而且远高于传统 KNN 方法和加权 KNN 方法。

6.3 本章小结

本章一方面介绍了本系统的详细分类内容，训练数据的内容，另一方面，对传统 KNN 方法、加权 KNN 方法以及本设计的基于属性熵值分析的快速 KNN 方法进行了测试对比。

在 100 个测试数据上的运行时间测试中，传统 KNN 算法和加权 KNN 算法都非常耗时，平均每个测试数据耗时将近 1 分钟，而 LCX-KNN 分类器实现了毫秒级运行，平均每个测试数据耗时在 1 秒以内。从运行时间测试上看，本设计的 LCX-KNN 分类器达到了高效的目的。

在 3578 个测试数据上的分类精度测试中，传统 KNN 算法的分类精度最差，总分类正确率仅达到 51.40%，加权 KNN 算法比传统 KNN 方法分类精度高出 18.58%，分类精度有一定提高，而本文提出的 LCX-KNN 分类器分类精度高达 85.05%，远远高于前两者。从分类精度上看，LCX-KNN 分类器达到了高精度分类的要求。

从本章的实验中我们可以得出，本文提出的 LCX-KNN 分类器是一个高效、高精度的网站分类器。

结　论

KNN 算法由于其简单、有效、参数无关，使得其在 Web 文本分类技术中被广泛应用。但是，KNN 算法有着不少的缺陷，最关键的两个缺陷是运行速度太慢和分类精度不高。

本文的目的是搭建一个真正高效、精准的基于改进KNN方法的网站分类系统。

本文首先阐述了网站分类系统的系统架构，以及所需要的每个模块对应的功能需求和输入输出。

其次，本文对每个模块的实现进行了详细的分析和阐述，本系统设计的模块包括爬虫模块，页面处理模块，特征提取与文本特征表示模块，以及 KNN 分类器模块。

最后，本文对 KNN 算法的缺陷产生原因进行分析并进行对算法的改进：在运行速度角度，本文的改进基于两个出发点：（1）减少需要与输入样本进行相似度计算的训练样本大小；（2）减少相似度计算的时间消耗。用 Rocchio 算法进行预选候选类，再根据文本的特征集与每类特征交集再次筛选候选类，先筛选出可能被判别的类别，再对每个类别建立倒排索引，筛选出值得参与相似度计算的训练样本，从而使得需要与输入样本进行相似度计算的训练样本得到了大幅度降低。本文同时引入了“位置向量”的向量表示方法，通过这种方法避免相似度计算上遍历无用的维度，极大减少了相似度计算的时间损耗。而在在分类精度角度，本文的改进重点在于在相似度计算上引入属性熵值来提高对分类指导能力较强的特征的作用，在 KNN 在 K 个邻居的决策时不仅考虑类别出现次数，更考虑了类别的平均相似度，以及输入样本对各类别的隶属程度，使得新的 KNN 方法在 K 个邻居的决策中不再片面、武断，而综合考虑了各方面因素。

在基于上述这些改进后，本系统利用个 3578 个真实网站内容作为测试集对系统进行了性能测试，最终的成绩是分类精度达到 85.05%，平均一个网页的分类速度是 0.88 秒。通过对实验结果进行分析，得出本文提出的新的 KNN 分类器在测试集数据的环境下达到了高速分类和分类正确率远高出传统方法的结论。本文提出的新的高效 KNN 算法作为网站分类器比原有的 KNN 分类方法和加权 KNN 方法有更快的速度，同时比两者有更高的分类精度。

总而言之，本文设计的 LCX-KNN 分类器是一个高效、高精度的网站分类器，达到了最初设计的构想。

参考文献

[1] Ceci, M., Malerba, D. Hierarchical Classification of HTML Documents with WebClassII, Lecture Notes ub Cimputer Science, 2003, pp. 57-72.

[2] Weston J, Watkins C. Muti-class support vector machines. Royal Holloway College, Tech Rep:CSK-TR-98-04,1998.

[3] Xie D X, Xia W F. Design and Implementation of the Topic-Focused Crawler Based on Scrapy[J]. Advanced Materials Research, 2014, 850: 487-490.

[4] G.Guo, H.Wang and K.Greer. An kNN model-based approach and its application in text categorization, 5th Int. Conf., CICLing Springer, Seoul, Korea, 2004, pp. 559-570

[5] Kwon O W, Lee J H. Web page classification based on k-nearest neighbor approach[C]. Proceedings of the fifth international workshop on on Information retrieval with Asian languages. ACM, 2000: 9-15.

[6] 奉国和, 吴敬学. Knn 分类算法改进研究进展[J]. 信息技术, 2012, 56(21).

[7] 卢志茂, 于洪霞, 范冬梅, 等. 基于改进 CHI 算法的垃圾邮件过滤方法[J]. 2010.

[8] 吴春颖, 王士同. 一种改进的 KNN Web 文本分类方法[J]. 计算机应用研究, 2008, 25(11): 3275-3277.

[9] 张玲珠, 周忠眉. 结合属性值贡献度与平均相似度的KNN改进算法[J]. 计算机工程与应用, 2010, 46(18): 130-131.

[10] 刘海峰, 苏展, 刘守生. 一种基于词频信息的改进 CHI 文本特征选择[J]. Computer Engineering and Applications, 2013, 49(22): 110-114.

[11] 严晓明. 基于类别平均距离的加权 KNN 分类算法[J].

[12] 李连, 朱爱红, 苏涛. 一种改进的基于向量空间文本相似度算法的研究与实现[J]. 计算机应用与软件, 2012, 2: 282-284.

[13] 奉国和, 吴敬学. KNN 分类算法改进研究进展[J]. 信息技术, 2012, 56(21).

[14] 周靖, 刘晋胜. 基于特征熵相关度差异的 KNN 算法[J]. Computer Engineering, 2011, 37(17).

[15] 童先群. 基于属性值信息熵的 KNN 算法改进研究[D]. 漳州师范学院, 2010.

[16] Cover T, Hart P. Nearest neighbor pattern classification[J]. Information Theory, IEEE Transactions on, 1967, 13(1): 21-27.

[17] Lewis D D, Schapire R E, Callan J P, et al. Training algorithms for linear text classifiers[C]//Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 1996: 298-306.

[18] Wu X, Kumar V, Quinlan J R, et al. Top 10 algorithms in data mining[J]. Knowledge and Information Systems, 2008, 14(1): 1-37.
[19] Li S, Mnatsakanov R M, Andrew M E. k-Nearest neighbor based consistent entropy estimation for hyperspherical distributions[J]. Entropy, 2011, 13(3): 650-667.
[20] Soucy P, Mineau G W. A simple KNN algorithm for text categorization[C]//Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on. IEEE, 2001: 647-648.

哈尔滨工业大学本科毕业设计（论文）原创性声明

本人郑重声明：在哈尔滨工业大学攻读学士学位期间，所提交的毕业设计（论文）《基于特征熵值分析的网站分类系统实现》，是本人在导师指导下独立进行研究工作所取得的成果。对本文的研究工作做出重要贡献的个人和集体，均已在文中以明确方式注明，其它未注明部分不包含他人已发表或撰写过的研究成果，不存在购买、由他人代写、剽窃和伪造数据等作假行为。

本人愿为此声明承担法律责任。

作者签名：　　日期：　　年　　月　　日

致　谢

岁月如梭，时光荏苒。那个站在哈工大校门口的懵懂而胆怯的少年，终究还是站在毕业的门槛上了。四年来，哈工大带给我欢笑，带给我泪水，在这里我遇到了志同道合的好朋友，遇到了学术造诣高深的老师，在这里我学习了大量的计算机科学知识，第一次认识到了自己真正想要的梦想，真正追求的未来。

在这里我首先要感谢我的导师张宏莉老师，虽然她很忙，但是在我的开题和资料查找上，她给了我很多指点与帮助。还要感谢我的校外导师沈智杰，他身为互联网公司副总裁每天事务缠身，但他仍然对我的毕业设计进行了多次指点和评价，还经常跟我分享他在母校时候的学习状态。他让我明白一个道理，知识真的可以改变命运，真正的人才从不会在学习知识的痛苦过程中退缩。

另外，我还要衷心的感谢所有教过我课程的老师，是你们的培养使我真正产生了学习计算机技术的兴趣。

感谢我的好朋友们，我们在一起竞争和互相讨论中互相成长；感谢所有1003104班的同学，快乐的大学时光就是你们的热情。

最后，感谢我的父母与亲人，你们给予我的关怀和鼓励让我前进的时候斗志十足。