工学硕士学位论文

面向多通道爬虫的 Web 信息抽取技术研究

RESEARCH ON MULTI-CHANNEL CRAWLER ORIENTED WEB INFORMATION EXTRACTION TECHNIQUES

马雪阳

哈尔滨工业大学

2016 年 6 月

国内图书分类号：TP399　　学校代码：10213
国际图书分类号：621.3　　密级：公开

工学硕士学位论文

面向多通道爬虫的 Web 信息抽取技术研究

硕 士 研 究 生：马雪阳

导　　师：张宏莉 教授

申 请 学 位：工学硕士

学　　科：计算机科学与技术

所 在 单 位：计算机科学与技术学院

答 辩 日 期：2016 年 6 月

授予学位单位：哈尔滨工业大学

Classified Index: TP399
U.D.C: 621.3

Dissertation for the Master's Degree in Engineering

RESEARCH ON MULTI-CHANNEL CRAWLER ORIENTED WEB INFORMATION EXTRACTION TECHNIQUES

Candidate: Xueyang Ma
Supervisor: Prof. Hongli Zhang
Academic Degree Applied for: Master of Engineering
Specialty: Computer Science and Technology
Affiliation: School of Computer Science and Technology
Date of Defence: June, 2016
Degree-Conferring-Institution: Harbin Institute of Technology

摘　要

随着 Internet 的迅猛发展，网络已经成为一个信息发布和消费的巨大平台。互联网具有快速传播和广泛覆盖的特性，对互联网舆情进行有效监控是必不可少的。由于网页固有的半结构性以及大量存在的与主题无关的噪声，研究如何从 Web 中抽取人们所需要的信息变得越来越重要。在一个聚焦于新闻、博客和论坛（它们都是很有代表性的信息传播渠道）的多通道爬虫系统中，我们面临如下挑战：1）大量网站需要监控；2）网站有不同的结构和布局；3）网站会不定期改版。这些挑战促使我们提出高度自动化的 Web 信息抽取技术，以减少系统的扩展和维护成本。

对于新闻、博客这种正文密集的网站，提出了一个模板无关的基于有效字符的内容抽取算法 CEVC（Content Extraction via Valid Characters）。为了验证该方法，从知名的中文新闻和博客网站上任意爬取了部分网页，构成测试数据集进行实验。实验结果表明 CEVC 能达到平均 95.8% 的 F_1-measure，效果优于之前的算法 CETR 和 CEPR，虽然抽取性能和 CETD 相当，但在预处理阶段依赖更小，适用性更强。

对于典型的论坛网站，利用帖子中普遍存在的发帖时间信息，提出了一个论坛帖子抽取算法 PEAN（Post Extraction via Anchor Nodes）。为了和同样利用发帖时间信息的帖子抽取算法 MiBAT 比较效果，我们从知名的中文论坛网站上采集网页进行实验。实验结果表明 PEAN 相比于 MiBAT 在召回率指标上有大幅度提升，平均 94.7% 的 F_1-measure 也优于 MiBAT。

为了验证本文提出的信息抽取算法的实际效果，我们针对实际需求设计并实现了一个 Web 新闻聚合系统。由于使用了模板无关的内容抽取算法，该爬虫能够在没有人工辅助的情况下爬取新的网站，大大减少了系统扩展和维护的成本。实际系统的运行情况表明，模板无关的内容抽取算法对多通道爬虫系统具有实际意义。

关键词：多通道；爬虫；信息抽取；模板无关的

Abstract

With the explosive growth of the Internet, the Web has become a large platform for information publishing and consuming. It is essential to supervise the Internet public opinion effectively due to the rapid dissemination and extensive coverage. As the inherent semi-structured characteristics and large part of topic irrelevant noises, effectively extracting main content and filtering these noises is necessary and challenging. In a multi-channel crawler system which focuses on news, blog and forum, which are all representative information channels, we face the following challenges: 1) enormous websites should be monitored; 2) websites have different structures and various layouts; 3) websites will change occasionally. These challenges motivated us to propose highly automated Web information extraction techniques to reduce the cost for system expansion and maintenance.

For information-intensive websites like Web news and blog, we propose a template independent content extraction approach based on valid characters (CEVC). To validate the approach, we conduct experiments by using onling news and blog files arbitrarily crawled from well-known Chinese news and blog websites. Experimental result shows that our method achieves 95.8% F_1-measure on average and outperforms previous methods CETR and CEPR. Although CEVC has almost equivalent extraction performance as CETD, CEVC has less dependence in the pre-processing stage thus more applicable.

For typical forum websites, we utilize the ubiquitous date information in forum posts and propose a forum post extraction method (PEAN). To compare the effectiveness with MiBAT, which also uses the date information to extract posts, we conduct experiments on various Chinese forums. Experimental result shows that our method achieves much higher recall than MiBAT, and the F_1-measure of 94.7% also outperforms MiBAT.

In order to verify the practicality of the methods, we design a framework for parallel multi-channel crawler system and implement a Web news crawler based on it. With the template independent content extraction approach, the crawler has the ability to crawl new websites without human efforts. The result shows that template independent content extraction methods have practical value on on multi-channel crawler system.

Keywords: multi-channel, crawler, information extraction, template independent

目录

摘要 I
Abstract II

第 1 章　绪论 1
1.1 课题背景与研究意义 1
1.2 国内外研究现状 2
1.2.1 Web 内容抽取 2
1.2.2 Web 数据记录抽取 5
1.3 研究内容与组织结构 7
第 2 章　基于有效字符的 Web 内容抽取 9
2.1 概述 9
2.2 CEVC 方法 11
2.2.1 DOM 11
2.2.2 有效字符 11
2.2.3 定位核心内容块 14
2.2.4 算法实现 17
2.3 对比算法及实现 18
2.3.1 CETR 18
2.3.2 CETD 19
2.3.3 CEPR 21
2.4 实验与分析 23
2.4.1 评价指标 23
2.4.2 数据集 23
2.4.3 参数调整 24
2.4.4 实验结果 25
2.4.5 结果分析 28
第 3 章　基于锚节点的论坛帖子抽取 31
3.1 概述 31

3.2 PEAN 方法 32
3.2.1 树匹配算法 32
3.2.2 锚节点 34
3.2.3 定位帖子节点 35
3.2.4 筛选候选帖子 37
3.2.5 算法实现 41
3.3 对比算法及实现 42
3.4 实验与分析 43
3.4.1 评价指标 43
3.4.2 数据集 43
3.4.3 实验结果 45
3.4.4 结果分析 46
第 4 章 应用实例——Web 新闻聚合系统 48
4.1 概述 48
4.2 系统架构和工作流程 48
4.2.1 系统架构 48
4.2.2 总体工作流程 48
4.2.3 新闻采集流程 50
4.3 系统模块和关键技术 50
4.3.1 新闻列表解析 50
4.3.2 新闻信息抽取 53
4.3.3 存储和检索 56
4.4 运行效果评估 57
结论 60
参考文献 61
攻读硕士学位期间发表的论文及其他成果 65
哈尔滨工业大学学位论文原创性声明 66
致谢 67

第 1 章 绪论

1.1 课题背景与研究意义

随着 Internet 的迅猛发展，互联网应用已经深入到我国的经济、社会、文化、教育以及娱乐等各个方面，成为人们生活中不可或缺的组成部分。由于互联网平台具有自由便捷的发布和获取方式、快速的信息传播能力、极为广泛的地理覆盖程度，Web 正逐渐成为信息发布和消费的巨大平台，被称为独立于传统的报纸、广播和电视的“第四媒体”，改变着信息在每个人中流动的方式。

在信息化技术高速发展的时代，网民人数不断增多，互联网信息量呈现指数型增长，成为传达社情民意的重要渠道。大众可以在互联网上自由、匿名地发表个人对社会现象、时事政治和热点事件的观点和态度，再由网络迅速传播，不同意见随之碰撞、发酵形成舆情，进而影响社会舆论。网络舆情可能包含危害国家安全和社会稳定的信息，为了维持社会稳定、减小一些舆情信息的负面影响，关注互联网舆情，及时掌握社会舆论动态和当前热点话题，对维护社会安定非常重要。

如今网络舆情的表达方式多种多样，新闻、博客和论坛是其中重要的信息传播渠道。阅读在线新闻是网民获取信息极为便捷的方式，而博客和论坛都为普通大众提供了发表意见、分享观点的平台，因而形成了一个庞大的网络社区。在一个聚焦于新闻、博客和论坛的多通道爬虫系统中，从下载的网页中抽取有用的信息，供后续处理和分析，是一个非常重要的环节。只抽取有用的信息而不是存储所有网页内容，能够节约存储和索引的空间，提升自然语言处理和文本挖掘的效果。

和传统媒体不同，互联网提供的信息爆炸式增长，如何从海量信息中找到人们所需要的信息，Web 信息抽取发挥着关键的作用。Web 信息抽取是以 Web 作为信息来源的信息抽取方式，从网页的无结构或半结构信息中抽取用户感兴趣的内容，转化为易于阅读和理解的格式，是信息能够被进一步分析和处理的基础。由于网页所固有的半结构化和大量存在的与主题无关的噪声，从中有效地抽取信息并不是一个简单的工作。针对一个多通道爬虫系统，我们还面临这些挑战：

- 大量网站需要关注
- 网站具有不同的页面组织结构和布局
- 网站会不定期升级改版

海量异构、持续变化的特点，给大范围舆情监控带来困难，迫切需要一种高度自动化的 Web 信息抽取方式，尽可能减少人工参与，以降低系统扩展和维护的成本。

综上所述，为了适应互联网舆情监控的需求，需要对多通道爬虫系统中的 Web 信息抽取技术做进一步研究。本文针对新闻、博客和论坛，研究了适应各自特点的 Web 信息抽取方式，在前人的基础上提出了改进的算法，并通过详细的实验比较验证了不同方法的效果。最后根据实际需求设计并实现了一个针对新闻的应用实例——Web 新闻聚合系统，以实际系统的运行效果，验证了本文提出的信息抽取技术的实际意义。

1.2 国内外研究现状

信息抽取（Information Extraction）最早可追溯到 20 世纪 60 年代中期，作为自然语言处理的一个分支，研究如何从自然语言中获取结构化信息。随着 Web 信息的爆炸式增长，Web 正成为最大的信息平台，Web 信息抽取逐渐成为新的研究热点。与传统的文本信息抽取不同，Web 信息抽取面对的是海量异构的语料，虽然以 HTML 标签作为分隔标志，但整体缺乏严格统一的语法和语义信息，传统的文本信息抽取技术不能直接利用。针对本文的研究内容，接下来主要从可用于新闻、博客的内容抽取技术，和可用于论坛的数据记录抽取技术两方面，分析国内外研究现状。

1.2.1 Web 内容抽取

网页内容抽取（Content Extraction）最早由 Rahman 等人于 2001 年提出，并给出了一个基本的内容抽取算法[1]。以新闻网页为例，典型的 Web 网页除了主要内容之外，还包含大量与主题内容无关的噪声信息，如导航栏、推荐链接、各种形式的广告等，如何从原始的网页中过滤噪声、抽取有用的信息，是 Web 内容抽取的研究方向。

1.2.1.1 包装器

最简单和直接的 Web 内容抽取方法是手工构建包装器（Wrapper），包装器是信息集成系统的一个独立模块，抽取网页数据并将其转换为结构化的数据。但这个过程容易出错也很耗费人力资源，随后一些研究致力于简化包装器构造的过程，降低专业知识门槛，例如 XWrap[2] 和 W4F[3]。

XWrap（XML-enabled Wrapper）通过和用户的交互，推导出信息抽取规则，交互式生成 XML 形式的包装器。W4F（World Wide Web Wrapper Factory）提供了一个更有表现力的领域专用语言描述复杂的抽取规则，并利用图形化工具辅助构造包装器。这些包装器使用的技术包括正则表达式、XPath、甚至特定领域的编程语

言，用来抽取嵌入在半结构化 HTML 网页中的文本。

虽然这些手工构造的包装器针对特定站点能够取得很高的抽取准确率，但它们的缺点也很明显，需要大量的人力成本，对大规模的内容抽取任务来说手工构造包装器是难以接受的。

1.2.1.2 基于机器学习

基于机器学习的方法主要利用网页的结构、语言学等特征，在人工标注的数据集上进行训练，根据训练出的分类模型，区分网页中的主要内容和噪声数据。

Kushmerick[4] 和 Davison[5] 提出使用机器学习的方法来识别网页中的广告、冗余和不相关的链接。Marek 等人[6] 提出了一种利用条件随机场（Conditional Random Fields，CRF）的方法来识别网页中的核心内容，首先通过 HTML 标签，将训练文件分成若干块，对于每一块提取 markup-based、content-based 和 document-related 的特征。依据这些特征，将内容块标记为若干类，训练条件随机场模型进行分类。FIASCO[7] 基于支持向量机（Support Vector Machine，SVM）来识别网页中的噪声数据。首先将网页文件解析成 DOM 树，对每一个目标节点提取语言学特征、结构特征和视觉特征，人工选择目标节点标注为 clean 或 dirty，然后使用 SVM 训练分类器。

然而这些方法难以推广使用，因为它们依赖大量人工标记的训练数据集和领域专业知识来生成分类规则。

1.2.1.3 基于视觉信息

2003 年微软亚洲研究院的蔡登提出了一种基于视觉信息的抽取算法 VIPS（VIsion based Page Segmentation）[8]。基于前人对于用户视觉心理的研究，一般情况下网页的核心内容处于网页的中间位置，而导航栏、广告和推广链接则处于网页的周边位置。这种方法利用网页的视觉信息，根据一些启发式规则，首先将网页分块，对不同的分块赋予相应的权重，然后进行删除、合并操作，最终确定网页的核心内容块。

随后宋瑞华[9] 在 VIPS 的基础上，使用人工标注的训练集，根据不同分块的位置特征和内容特征，用 SVM 和神经网络训练分类模型。Fernandes[10] 首先使用 VIPS 网页分块算法，将每个网页分成若干个由 DOM 树中的路径及其文本内容定义的块，在上述基础上使用向量空间模型（Vector Space Model，VSM）思想，计算其重要程度。

基于视觉信息分块的方法具有很好的通用性，因为不同网页的源代码结构可能差别很大，但经浏览器渲染之后的视觉表现却很接近。但这种方法必须对网页进行渲染才能利用视觉信息，非常消耗计算资源，而且有时候渲染所需的网页样

式文件不一定是可获取的。

1.2.1.4 模板检测

如今大部分网页都是使用固定模板动态生成的，网站后台程序从数据库中提取信息，与模板进行渲染就形成了最终的 HMTL 源文件。这样的模板在同一个网站被大量重复使用，就可以通过一组相同或相似模板生成的网页，逆向推导出共同的模板结构，以此提取网页核心内容，这就是模板检测算法。

Lin 等人[11] 提出一种基于当时 HTML 文档普遍存在的制表标签的方法。将网页划分为若干内容块，然后根据某些特征计算每个块的信息熵，由动态调整的阈值把内容块划分为核心内容块和噪声数据。Yi 等人[12] 在 DOM 树的基础上提出一个新型树结构，网站风格树（Site Style Tree，SST）。由于噪声块通常有相似的内容和表现的样式，如果相同内容和样式在多个网页中重复出现，表明其信息熵较低，可能是冗余内容。文章提出一种基于信息熵的度量方法来确定 SST 的哪部分代表噪声，哪部分代表主要内容。将网页 DOM 树匹配到网站的 SST 的过程，就能够检测并消除噪声部分，从而提取主要内容。

Bar-Yossef 等人[13] 提出利用经典的频繁项挖掘来检测网页的模板，类似的研究还有 [14]。

模板检测算法在模板生成后，实际提取网页内容的计算开销很小。但这类方法只能针对一个模板生成的网页进行处理，对于不同模板生成的网页，都需要重新构造模板。随着 Web 的发展，一个网站所使用的模板越来越丰富和多样，网站还会不定期改版，之前生成的模板就会失效，这些都给模板维护工作带来巨大负担。

1.2.1.5 基于统计规律

还有一类基于统计规律和启发式规则的内容提取算法，不需要一组网页进行训练因而与模板无关，而且自动化程度较高。

BTE（Body Text Extraction）[15] 将 HTML 网页处理为一个由词语和标签构成的序列，从中找到一个连续的区段，使得这个区段包含最多的词语，并且包含最少的标签。一般网页的正文包含较少的标签和相对密集的文字，这样的区段就认为是核心内容。FE（Feature Extractor）[16] 和 KFE（K-Feature Extractor）[17] 将网页划分为若干块，分析特定的文本、图片和标签等特征，从中找出出现次数较少，并且最符合期望特征的块。LQF（Link Quota Filters）[18] 利用网页分块中的链接比重来识别导航栏或相似的噪声内容。停止词和其它一些启发式规则也在一些工作中出现[19]。

CCB（Content Code Blurring）[20] 将网页 HTML 代码视为一个向量，生成二元的 Content Code Vector（CCV）。网页源文件中的文本内容在向量中为 1，HTML 代

码在向量中为 0，这样就把网页转换为一个二元向量。随后对向量进行处理，从中找出同质化格式的内容，即连续为 1 的部分。CETR（Content Extraction via Tag Ratios）[21] 对网页源文件一行一行处理，计算出每行的 Tag Ratios，即一行中的实际字符数除以 HTML 标签数。Tag Ratio 越高就越可能是主要内容，作者最后将计算结果映射到二维空间，利用聚类算法进行处理。这两种算法简单、高效不需要训练数据，也不需要解析 DOM 树或者渲染网页，但按行处理的方式没有利用 HTML 的结构化信息，对网页代码风格也较为敏感。

CETD（Content Extraction via Text Density）[22] 首先将 HTML 网页解析为 DOM 树，然后对每个节点计算 Text Density，即该 DOM 节点包含的字符数除以标签数，在考虑到链接与噪声的相关性后，提出 Composite Text Density，对文本密度做了一些修正。最后提出一种结合 DOM 树的 DensitySum 计算方式替换常用的平滑和阈值划分方法，最终确定核心内容块。

CEPR（Content Extraction via Path Ratios）[23] 认为网页内容布局与其 DOM 树的标签路径之间存在隐含的关联，针对每个节点计算文本标签路径比，即文本长度与标签路径出现次数的比值。文本标签路径比越高，表明这个路径模式承载的文本信息越多，越可能是主要内容。最后作者提出了基于路径编辑距离的加权高斯平滑方法，进行内容抽取。

1.2.2 Web 数据记录抽取

网页具有半结构化的特点，不同于 XML 或数据库记录等高度结构化的文档，不能被计算机程序直接读取。Web 数据记录抽取是从网页中抽取结构化数据，例如抽取、整合来自不同数据源的商品的各项元数据信息，抽取论坛中帖子的发帖时间、发帖内容等元数据信息。

1.2.2.1 半自动化方法

人工编写包装器来提取网页信息的方法需要大量人力投入，成本高也并不实用。于是研究人员实现了一系列工具来自动生成包装器，能够从用户的训练样本中归纳学习抽取规则，降低了包装器构造的成本。WIEN[24]，SoftMealy[25] 和 Stalker[26] 是这类半自动化方法的典型代表。

WIEN 基于多种不同的归纳学习技术，并提出了一个混合系统，训练阶段只需要少量人工参与。SoftMealy 基于有限状态转换器（Finite State Transducer，FST）和上下文规则，通过自底向上的归纳学习方法，由人工标注的样本文件生成包装器。它们的主要区别是抽取架构不同，WIEN 使用一遍处理的 LR 结构，Stalker 使用多

遍处理的层级结构。

国内孟小峰[27] 提出了一种基于预定义模式的方法来构造 HTML 包装器，并将它运用到 XWIS（基于 XML 的 Web 信息查询系统）中，由用户定义模式并给出模式与 HTML 页面的映射关系，随后推导出规则构造包装器。

另一类基于模板树匹配的方法，首先从标注数据生成模板树，并在模板树的相应位置标记数据记录、数据单元，随后对同一类型的网页，运行树匹配算法，即可抽取结构化数据。Chuang 等人[28] 提出一种模板树自动生成算法 TTAG，能够从少量的训练网页中学习模板的树型结构，适用于频繁更新的网站。Zheng 等人[29] 提出一种新的 Broom 结构来同时表示数据记录和生成的模板，能够有效抽取数据记录并识别内部的语义。

国内李效东[30] 提出一种归纳学习算法来半自动地生成提取规则，其原理是，将所需要提取的信息在 DOM 层次结构中的路径作为信息抽取的“坐标”。算法是一个典型的顺序覆盖算法，产生一个假设去覆盖集合中尽可能多的正例，再删除被覆盖的正例，循环直至所有的元素被覆盖。

这类半自动化方法需要人工标注的样本，这个过程依然费时费力，另外模板的生成和维护都需要很大的成本，不适用于互联网规模的网络信息抽取工作。

1.2.2.2 全自动化方法

为了克服依赖人工标注数据的缺陷，一部分研究工作聚焦于全自动化的信息抽取方法，这类方法更适合互联网环境下的大规模信息抽取工作。

Embley[31] 提出一种基于启发式规则的数据记录抽取方法，将网页文档的结构以嵌套标签树的形式刻画，然后定位出包含数据记录的子树，使用五种独立的启发式规则来识别候选的记录分隔符。Buttler 提出一个全自动化的对象抽取系统 Omini[32]，Omini 将网页解析为树型结构然后分两步抽取对象，定位子树和确定记录分隔符。它同样采用了多种启发式规则，主要贡献是自动化的规则学习算法。这部分方法过于依赖启发式规则，难以大规模扩展，而且如今的网页结构和布局也发生了很大变化。

Chang 等人提出一个能够从网页中自动发现抽取规则的系统 IEPAD[33]，它通过重复模式挖掘和多序列对齐来识别数据记录的边界。重复模式的挖掘通过一种新的数据结构 PAT 树来实现，并经过模式对齐进一步扩展以理解所有的记录实例。Wang 等人在此基础上提出了 DeLa[34]，该系统通过 HTML 表单提交查询，自动生成基于正则表达式的包装器从结果页中抽取数据对象。

上述单纯基于 HTML 标签序列模式特征的方法不能很好地利用网页的层次结构信息，部分研究人员开始在 DOM 树上识别重复相似子树，进而抽取数据记录，

MDR[35] 是其中的典型代表，它依据这样的假设：1）一组相似的数据记录通常位于连续的区域并具有相似的 HTML 标签结构；2）每个数据记录之下包含相同数目的兄弟子树。MDR 方法简单，抽取结果也由于 Omini 和 IEPAD。

Song 等人提出一种抽取用户生成数据（UGC）记录的方法 MiBAT[36]。论坛帖子、评论回复等用户生成数据格式自由，可能包含图片和格式化标签，前述方法主要针对高度格式化的数据，例如商品展示信息，从而忽略了 UGC 的影响。MiBAT 利用领域知识限制，即 UGC 中普遍存在的发布时间信息作为锚节点，来定位候选子树，降低了 UGC 标签的影响，取得了较好的效果。

1.3 研究内容与组织结构

本文的主要研究内容是面向多通道爬虫系统的 Web 信息抽取技术，具体分为以下几个方面：

（1）针对新闻、博客这类正文集中的网站，提出了一种基于有效字符的 Web 内容抽取方法 CEVC（Content Extraction via Valid Characters）。该方法主要基于这样的观察，网页中不属于链接并包含停止词的文本，更有可能是主要内容。定义这样的字符为有效字符，根据它们在 DOM 树中的分布，逐级确定正文区域，并最终提取正文。在知名的中文新闻、博客网站上采集网页进行实验，与之前的算法 CETR（基于文本标签比）、CETD（基于文本密度）和 CEPR（基于文本标签路径比）进行了比较。实验结果表明，CEVC 算法在各项评价指标上都优于 CETR 和 CEPR，虽然抽取性能和 CETD 相当，但在预处理阶段依赖更小，适用性更强。

（2）针对论坛网站，提出了一种论坛帖子抽取算法 PEAN（Post Extraction via Anchor Nodes）。该方法利用论坛帖子中普遍存在的发帖时间信息作为锚节点，根据它们在 DOM 树中的分布情况，定位论坛帖子集中的区域，并结合树匹配算法，在候选子树中过滤噪声，最终抽取出论坛帖子。为了和同样利用发帖时间信息的帖子抽取算法 MiBAT 比较效果，我们从知名的中文论坛网站上采集网页进行实验。实验结果表明，PEAN 相比于 MiBAT 在召回率指标上有大幅度提升，总体 F_1 指标也优于 MiBAT。

（3）为了验证本文提出的信息抽取算法的实际效果，根据实际需求设计并实现了一个针对新闻的应用实例——Web 新闻聚合系统。介绍了系统架构和工作流程，并对系统模块和关键技术做了详细阐述，最后对系统运行效果进行评估。由于使用了模板无关的内容抽取算法，该爬虫能够在没有人工辅助的情况下爬取新的网站，大大减少了系统扩展和维护的成本。

本文组织结构如下：

第一章为**绪论**，介绍了当前互联网舆情的发展形势，说明了自动化 Web 信息抽取技术在多通道爬虫系统中的关键作用，并从 Web 内容抽取和 Web 数据记录抽取两方面回顾了国内外研究现状，最后给出论文的研究内容与整体结构安排。

第二章为**基于有效字符的 Web 内容抽取**，详述针对新闻、博客网站的 Web 内容抽取方法 CEVC，并与 CETR、CETD 和 CEPR 进行了实验比较。

第三章为**基于锚节点的论坛帖子抽取**，详述了针对论坛网站的帖子抽取方法 PEAN，并与 MiBAT 进行了实验比较。

第四章为**应用实例——Web 新闻聚合系统**，设计并实现了一个针对新闻的应用实例——Web 新闻聚合系统。

第 2 章 基于有效字符的 Web 内容抽取

2.1 概述

在 Internet 时代，Web 新闻和博客都是很有代表性的信息发布和获取渠道。

Web 新闻区别于传统的报纸，在互联网上发布意味着它能够触及更广大的受众，扩大自己的影响力，对热点新闻的实时推送也提高了信息传播的即时性。阅读 Web 新闻了解实事动态已经成为广大网民的生活习惯，很多传统媒体也纷纷推出了自己的 Web 新闻。

博客（blog，weblog 的缩写，也叫网络日志）是发布在互联网上的文章集合，通常由个人撰写和管理。博客给普通大众提供了一个表达自己观点和态度的平台，人们还能在文章后发表评论，形成一个庞大的社区，促进信息的交流。

但除了新闻正文之外，一个典型的新闻网页还包含导航栏、广告、推荐链接和版权信息等，这些与正文无关的额外内容，通常称为噪声。Gibson 在 2005 年的研究 [37] 显示，网页模板在整个网页中占据了 40%–50% 的字节量，而且这个比例正以每年 6% 的速度增长。以一篇新浪新闻[1]为例，如图 2-1 所示，红色实线圈出的内容都与正文无关，它们占据了网页一半以上的篇幅。

博客和新闻类似，都属于正文集中的网站，博客网页的噪声主要有导航栏、广告和博客相关文章等。以一篇网易博客[2]为例，如图 2-2 所示，红色实线圈出的是与正文无关的内容。

从网页中抽取核心内容，过滤这些噪声，不仅能够给用户提供“干净”的内容，也能节约爬虫系统的存储开销，减小索引规模，提升后续自然语言处理的效果。由于新闻和博客网页布局与内容抽取目标的相似性，本章将两者合并处理，提出一种基于有效字符的 Web 内容抽取方法， 2.2 节详述该方法 CEVC（Content Extraction via Valid Characters），2.3 节简单介绍对比算法 CETR，CETD 和 CEPR 并给出算法实现细节， 2.4 节进行对比实验并分析结果。

[1] http://news.sina.com.cn

[2] http://blog.163.com

图 2-1 新浪新闻实例

图 2-2 网易博客实例

2.2 CEVC 方法

2.2.1 DOM

在网页内容抽取的研究工作中，为了利用 HTML 文档的结构信息，很多方法都是在 DOM 的基础上进行[12, 22, 23] 这里对 DOM 做简单介绍。

DOM（Document Object Model，文档对象模型）是一个跨平台语言无关的规范，用来表示和操作 HTML，XHTML 和 XML 的对象[3]。每个文档所包含的节点被组织为一个树型结构，称为 DOM 树。DOM 树节点所表示的对象可以通过一组方法来寻址或操作，这些接口以 API 的形式描述。通过这样一种规范，程序和脚本能够动态访问和更新文档的内容、结构以及风格，文档经过后续处理后，还能将结果更新到展示的页面上。

上世纪 90 年代，网景公司和微软公司相继推出自己的浏览器，和浏览器端的脚本语言 JavaScript。JavaScript 让 Web 开发者能够创造和用户交互的网页，检测用户产生的事件并更新 HTML 文档，这类第一代的语言被称为 Legacy DOM。随后 W3C（World Wide Web Consortium，万维网联盟）制定了第一版的 DOM 标准，之后相继发布了新版本。截止到 2005 年，W3C DOM 中的大部分特性都能被主流浏览器广泛支持。

HTML 文档可以解析为 DOM 树，每个 HTML 标签对应一个 DOM 树节点，父子节点对应 HTML 标签的嵌套关系，兄弟节点对应 HTML 标签的并列关系，这样就把序列化的 HTML 文档转换为层次化的对象模型，便于程序的访问和操作。

例 2.1 所示的 HTML 片段经过解析后形成一个 DOM 树，如图 2-3 所示。每个 HTML 标签都与一个 DOM 树节点对应，HTML 标签的属性仍然附属在 DOM 节点上。网页中能够显示的具体文本都位于 DOM 树的叶节点上，DOM 树的内部节点只负责结构控制和布局。HTML 的根节点是 `<html>`，一般包含两个节点 `<head>` 和 `<body>`。网页的可视区域都从 `<body>` 标签开始，因此我们的研究对象是 `<body>` 标签下的内容。

2.2.2 有效字符

回顾图 2-1 和 2-2 ，我们注意到网页正文一般是文本密集的区域，而噪声信息则包含高度格式化的、简短的文本。通过大量实例分析，我们发现噪声信息通常包

[3] http://www.w3.org/DOM/

例 2.1 一个简化的 HTML 代码片段

```
 1. <html>
 2.   <head> ... </head>
 3.   <body>
 4.     <div class="nav-wrap">
 5.       <h1>Small Vintage Plane Crashes in Hudson River</h1>
 6.     </div>
 7.     <div> ... </div>
 8.     <div class="ep-content">
 9.       <a>
10.         <span>From Stefan Holt</span>
11.       </a>
12.     <p>The plane was on a photo flight with two other planes ...</p>
13.       <p>They say they saw the pilot try to get out but ...</p>
14.     </div>
15.   </body>
16. </html>
```

图 2-3 一个简化的 DOM 树

含链接，无论是导航栏的网站板块链接，还是广告链接或相关文章链接，它们都符合这个特点，吸引用户点击来完成自己的职能。反之，一段不包含链接的文本更有可能是网页核心内容。

链接在 HTML 文档中由 `<a>` 标签组成，a 代表 anchor，所以链接内的文本也称为锚文本。DOM 节点的继承关系使得包裹在 `<a>` 标签内的所有文本都变成锚文本，例如图 2-3 中的 `From WWW` 虽然在 `<span>` 下，但其祖先节点是一个 `<a>` 标签，因此也变成了锚文本。

除此之外，噪声信息较为简短，甚至不能组成一个语义完整的句子，我们可以利用停止词来刻画这一特征。停止词是自然语言处理中过滤掉的词[4]，它们在一篇文章中大量出现，多为冠词、介词或连词，却对文章主体内容贡献很小。尽管停止词对文章没有多大意义，但一段话如果没有停止词，那么它就不是一个语义完整的句子，更有可能是一组关键词的堆砌，而不是网页的核心内容。例 2.2 节选自网页中的一段话，以下划线的形式标注了其中的停止词，可见停止词在正文中是普遍存在的。

例 2.2 网页中的停止词示例

买房<u>还是</u> 租房？<u>这是</u> 眼下很多<u>人</u> 要面对<u>的</u> 一道选择题。如果简单比较，一边<u>是</u>数额很大<u>的</u> 买房负担，一边<u>是</u> 相对<u>可</u> 接受<u>的</u> 租金价格，似乎租房应更受欢迎，现实<u>却</u> 非如此。山东某县一位购房者对笔者道出心声：“买房子，政府一平方米补贴 100 元，一套房子能省 1 万多元。租房子，<u>啥</u> 实惠没有，<u>还</u> 总<u>被</u> 赶来赶去，住<u>着不</u>踏实。”<u>他</u> 最终<u>还是</u> 选择<u>了</u> 勒紧裤腰带贷款买房。

结合上述链接和停止词与网页噪声的相关性，我们提出一个有效字符的概念：

定义 2.1 i 是 DOM 中的一个叶节点，如果 $\forall p \in ancestors(i)$ 都有 $tag(p) \neq a$，并且 $stopwords(i) > 0$，那么 i 所包含的文本就是有效字符。

定义 2.1 中，$ancestors(i)$ 表示 i 的所有祖先节点，$tag(p)$ 表示 p 的标签名，$stopwords(i)$ 表示 i 的停止词个数。有效字符代表网页中那些更有可能是正文的文本，忽略了链接噪声和太过松散的短语，有效字符越多，代表这个节点更有可能是正文。

为了利用 DOM 节点的有效字符分布信息，我们需要计算每个 DOM 节点下所包含的有效字符个数，可以将其作为一项属性，插入到 DOM 节点中，便于后续的

[4]https://en.wikipedia.org/wiki/Stop_words

处理。在计算之前，首先去除网页中的 `<script>`，`<style>` 和 `comment`。这些脚本、样式文件和注释对网页显示的文本内容没有贡献，却可能干扰计算结果。

如算法 2.1 所示，这是一个自顶向下的递归计算过程。ValidChars 被作为属性，存储在节点中。当前节点有子节点时，递归计算子节点，然后当前节点的有效字符数被累加到父节点上。当前节点是叶节点时，根据定义 2.1 判断它包含的文本是否是有效字符，如果是，则将非空白字符数累加到父节点上。这里忽略了空白字符，是为了排除网页排版风格带来的影响，让计算结果更加稳定。

算法 2.1 的主要操作就是遍历 DOM 树，所以它的运行时间与 DOM 树中的节点数成正比，时间复杂度为 $O(n)$，n 是 DOM 树中的节点数。为了在内存中存储解析后的 DOM 树，其空间复杂度也是 $O(n)$。

```
Input: DOM node N
Output: DOM node N
if N has child nodes then
    for C ∈ N.children() do
        countValidChars(C) ;
    end
    N.parent.validChars += N.validChars ;
else
    // N is a leaf text node
    if N consists of valid characters then
        length = getNonSpaceLength(N) ;
        N.parent.validChars += length ;
    end
end
```

算法 2.1: countValidChars(N)

经过算法 2.1 的处理，我们可以得到一个标注了各个节点 ValidChars 的 HTML 页面，如图 2-4 所示。

2.2.3 定位核心内容块

经过大量实例分析，我们发现对于新闻、博客这类正文信息集中的网站，核心内容块都位于同一个父节点之下，通常表现为文本的段落。图 2-4 中的新闻正文位

```
<html>
▶ <head>…</head>
▼ <body valid_chars="4161">
  ▶ <div id="BAIDU_DUP_fp_wrapper" style="position: absolute; left: -1px; bottom: -1px; z-index: 0;
  visibility: hidden; display: none;">…</div>
  ▶ <div class="ntes_nav_wrap" valid_chars="4">…</div>
  ▶ <div class="ep-header" valid_chars="5">…</div>
  ▼ <div class="ep-content" id="js-epContent" style="position: relative;" valid_chars="4102">
    ▼ <div class="ep-content-bg clearfix" valid_chars="4102">
      ▼ <div class="ep-content-main" id="epContentLeft" valid_chars="3912">
          <!-- <a href="http://1.mail.163.com/?from=portal163" target="_blank"><img width="635" heig
          src="http://img1.126.net/channel4/other/63540_0730.jpg"></a>
          <span class="blank9"></span> -->
          <h1 class="ep-h1" id="h1title" valid_chars="15">习近平振兴中国经济的战略和战术</h1>
        ▶ <div class="clearfix" valid_chars="30">…</div>
          <div class="otitle" valid_chars="21">（原标题：习近平振兴中国经济的战略和战术）</div>
          <style>.otitle{font-size: 14px;text-indent: 2em;margin: 20px 0 0;color:#252525;}</style>
        ▼ <div class="end-text" id="endText" valid_chars="3685">
          ▶ <p class="f_center" valid_chars="61">…</p>
          ▶ <p class="f_center" valid_chars="61">…</p>
          ▶ <p valid_chars="131">…</p>
          ▶ <p valid_chars="10">…</p>
          ▶ <p valid_chars="266">…</p>
          ▶ <p valid_chars="408">…</p>
          ▶ <p valid_chars="329">…</p>
          ▶ <p valid_chars="505">…</p>
          ▶ <p valid_chars="316">…</p>
          ▶ <p valid_chars="10">…</p>
          ▶ <p valid_chars="428">…</p>
          ▶ <p valid_chars="360">…</p>
          ▶ <p valid_chars="158">…</p>
          ▶ <p valid_chars="0">…</p>
          ▶ <div class="gg200x300" valid_chars="0">…</div>
          ▶ <p valid_chars="475">…</p>
          ▶ <p valid_chars="142">…</p>
          ▶ <div class="ep-source cDGray" valid_chars="25">…</div>
          </div>
          <!-- 分页 -->
```

图 2-4 有效字符示例

于 `<div#endText>` 节点之下，是由 `<p>` 标签构成的段落。

这里 `div#endText` 的记号是 CSS 选择符，可以指定 CSS 样式规则作用在哪些 DOM 节点上，简单的用法是 `div#endText` 选择 `<div id='endText'>` 这样的标签，而 `div.text` 选择 `<div class='text'>` 这样的标签。和正则表达式、XPath 一样，CSS 选择符也可以用来在 HTML 文档中描述提取规则。

现在我们看到每个 DOM 节点上都标注了有效字符的个数，有效字符越多表明这个节点越可能是核心内容。但这个比较必须在同一个层次的兄弟节点间才有意义，因为有效字符数是嵌套包含的，毫无疑问 `<body>` 的有效字符数最多。如果从 `<body>` 节点开始，每次都选择有效字符数最多的子节点深入，就能定位出一条从 `<body>` 通往核心内容块的正确路径，如图 2-4 中红线标示的节点。

每次沿着有效字符数最大的子节点不断深入，意味着我们每一步放弃了信息量较小的兄弟节点。这个过程不能一直持续下去，否则会陷入到叶节点中，而错失最终的父节点。如图 2-4 所示，最后一步进入了一个 `<p>` 标签，而错失了真正的核心内容块 `<div#endText>`。我们需要定义一个指标，来判断何时停止向下深入

的过程。

定义 2.2 i 是 DOM 树中的一个节点，i 的最大子节点字符占比 MCR（Max Chars Ratio）如下所示，其中 C_i 表示 i 的有效字符数，j 是 i 的子节点：

$$MCR_i = max_{j \in children(i)} \frac{C_j}{C_i} \tag{2-1}$$

在 DOM 树中每次向下深入一步，我们认为最大子节点包含了父节点的绝大部分文本信息。当定义 2.2 中的 MCR 低于一个阈值时，表明最大子节点的有效字符数并不足以在父节点中占据统治地位，即最大子节点的文本不能在可接受的损失范围内代表父节点，这时候放弃兄弟节点是不合理的，因此向下深入的过程应立即终止，当前节点就是核心内容块。

例 2.3 MCR 计算过程

```
1. <body> valid_chars=4161 MCR=0.99
2. <div#js-epContent> valid_chars=4102 MCR=1.0
3. <div.ep-content-bg> valid_chars=4102 MCR=0.95
4. <div#epContentLeft> valid_chars=3912 MCR=0.94
5. <div#endText> valid_chars=3685 MCR=0.14 核心内容块
6. <p> valid_chars=505
```

如例 2.3 所示，每一行显示的都是图 2-4 中各层的最大子节点。在到达 `<body#endText>` 之前，MCR 一直保持 0.9 以上的高比率，而 `<body#endText>` 的 MCR 锐减到 0.14，因为此时已深入到正文的一个段落中，这个段落无法代表正文的内容，`<body#endText>` 就是最终的核心内容块。

上述过程如算法 2.2 所示。在经过算法 2.2 `countValidChars` 预处理后的 DOM 树上，先遍历子节点，找出有效字符数最大的子节点，并累加所有字符数。然后选定最大子节点 `maxNode`，如果 MCR 小于一个阈值 α 就停止并输出当前节点，否则在 MaxNode 上继续递归调用算法 `stepInto`。`output` 过程负责把当前节点的所有有效字符输出，作为核心内容抽取的结果返回。注意第 19 行，当遇到一种极端情况，正文仅仅包含一个段落，那么 StepInto 深入到最底层仍然会有很高的 MCR，导致无法终止直到进入叶节点。这时候采取一种简单的策略，认为当前节点的父节点就是核心内容块。后续实验表明，这是一个简单也可以接受的方案。

通过在 DOM 树中不断选择最有代表性的子节点不断深入，算法 2.2 至多被递

归调用 h 次，h 是 DOM 树的高度，所以其时间复杂度为 $O(h)$，能够很快运行结束。

```
Input: DOM node N
Output: main content
maxNode = null ;
maxValidChars = 0 ;
totalValidChars = 0 ;
for C ∈ N.children() do
    totalValidChars += C.validChars ;
    if C.validChars > maxValidChars then
        maxValidChars = C.validChars ;
        maxNode = C ;
    end
end
if maxNode is not null then
    MCR = maxValidChars / totalValidChars ;
    if MCR < α then
        return output(N) ;
    else
        stepInto(maxNode) ;
    end
else
    // Step into the leaf node
    return output(N.parent) ;
end
```

算法 2.2: stepInto(N)

2.2.4 算法实现

本文中 CEVC 算法使用 Python 2.7 实现。

面对 HTML 网页中普遍存在的代码错误，例如缺失结束标签、使用非标准的

标签和错误的嵌套等，这里使用 Python 第三方库 BeautifulSoup[5] 来尽可能纠正这些错误。BeautifulSoup 是一个可以从 HTML 或 XML 文件中提取数据的 Python 库，它为操作 DOM 提供了一套方便的接口，底层调用 lxml 等具体的解析器，能够处理有代码错误的网页。

CEVC 算法中需要识别停止词，这里使用了 jieba[6]中的停止词模块。jieba（结巴中文分词）是一个开源的 Python 中文分词组件，MIT 协议授权，支持多种分词模式。

2.3 对比算法及实现

2.3.1 CETR

Weninger 等人于 2010 年提出 CETR（Content Extraction via Tag Ratios）算法[21]。

CETR 的基本概念是定义了 Tag Ratio。它以“行”为单位读取 HTML 网页并处理，分别计算每一行的 Tag Ratio，即非 HTML 标签字符个数与 HTML 标签个数的比值。如果遇到 HTML 标签个数为零的特殊情况，该行的 Tag Ratio 就是所有字符的个数。作者认为 Tag Ratio 越高，反映了文本密集程度越高，更有可能是网页的核心内容。

通过计算可以得到每一行 Tag Ratio 的直方图，最直接的方法就是找到一个阈值，所有 Tag Ratio 高于此阈值的行被认为是有效内容，所有 Tag Ratio 低于此阈值的行被认为是噪声内容。如何找到这样一个合适的阈值，就成为了文章的主要研究内容。

为了提升直方图的区分度，作者对数据进行了高斯平滑。标准的高斯平滑算法主要用来做图像处理，是应对连续数据的，作者重新实现为一维的离散操作。公式（ 2-2 ）是高斯核函数的构造过程，给定窗口大小为 $2(\lceil\sigma\rceil)+1$。

$$k_i = \sum_{j=-\lceil\sigma\rceil}^{\lceil\sigma\rceil} e^{\frac{-j^2}{2\sigma^2}}, 0 \le i \le 2(\lceil\sigma\rceil) \tag{2-2}$$

公式（ 2-3 ）中高斯核函数 k 被处理为 k'。

[5]http://www.crummy.com/software/BeautifulSoup/

[6]https://github.com/fxsjy/jieba

$$k'_i = \frac{k_i}{\sum_{j=0}^{\lceil\sigma\rceil} k_j}, \lceil\sigma\rceil \le i \le 2(\lceil\sigma\rceil) \tag{2-3}$$

最后高斯核函数 k' 和直方图 T 一起做卷积运算，形成平滑后的直方图 T'，如公式（2-4）所示。

$$T'_i = \sum_{j=-\lceil\sigma\rceil}^{\lceil\sigma\rceil} k'_{j+\lceil\sigma\rceil} T_{i-j}, \lceil\sigma\rceil \le i \le len(T) - \lceil\sigma\rceil \tag{2-4}$$

上述以阈值划分直方图的方法，其缺陷是将 Tag Ratio 仅仅视为一组数据，而不是有序的序列，为了利用有序的信息，作者提出了一个二维模型。

$$f'(T'_i) = G_i = \frac{\sum_{j=0}^{\alpha} T'_{i+j}}{\alpha} - T'_i, 0 \le i < len(T') - \alpha \tag{2-5}$$

注意到 $len(G) \neq len(T')$，实际上 $len(G) = len(T') - 1$ 因为 G 是一组数据的差异值。接着使用公式（2-4）平滑 G 得到 G'。最后计算 $\hat{G}$ 使得 $\hat{G}_i = |G'_i|$，对于 G' 中的每一个 i。

将平滑后的 Tag Ratio T' 和 T' 的 Absolute Smoothed Derivatives $\hat{G}$ 联合组成二维模型，相比于一维数据，更加具有区分度。

随后作者使用 K-means 聚类算法，将所有的样本点进行聚类，这里使用 $k = 3$。与一般的 K-means 聚类不同，将其中一个簇的中心点固定为二维坐标系的原点，这样可以使得剩余簇远离原点，并且很容易划分最终的结果，即中心点固定在原点的簇是噪声内容，而剩余簇代表有效内容。

实现细节

作者一共提出了三种算法，记为 CETR-TM、CETR-KM 和 CETR。CETR-TM 是阈值算法，CETR-KM 是一维数据上的 K-means 聚类算法，CETR 是二维数据上的 K-means 聚类算法。根据作者的实验对比，CETR 的抽取性能最佳，也是本章中参与对比的算法。

作者提供了 Java 实现的算法源代码[7]，我们对其进行了简单的包装，以 Python 接口调用并参与后续的评价指标计算。

2.3.2 CETD

Sun 等人于 2011 年提出 CETD（Content Extraction via Text Density）算法 [22]。

CETD 算法的核心概念是 Text Density，它在 DOM 树的基础上定义，而不像

[7]http://www3.nd.edu/ tweninge/cetr/

CETR 那样以行为单位处理，更能利用 HTML 的结构信息。Text Density 的基础定义如 2.3 所示。

定义 2.3 如果 i 是一个标签（对应 DOM 树中的一个节点），那么 i 的 Text Density（TD_i）是其字符个数和标签个数的比值：

$$TD_i = \frac{C_i}{T_i} \tag{2-6}$$

在进一步研究后，作者发现很多网页噪声由超链接组成，基于这样的观察，对每个节点计算了两个额外的统计指标：

1. *LinkCharNumber*：子树中所有链接字符的个数
2. *LinkTagNumber*：子树中所有链接标签的个数

在加入了链接的统计信息后，定义了 Composite Text Density：

定义 2.4 如果 i 是一个标签（对应 DOM 树中的一个节点），那么 i 的 Composite Text Density（CTD_i）是：

$$CTD_i = \frac{C_i}{T_i} \log_{ln(\frac{C_i}{\neg LC_i} LC_i + \frac{LC_b}{C_b} C_i + e)} (\frac{C_i}{LC_i} \frac{T_i}{LT_i}) \tag{2-7}$$

定义 2.4 中，C_i 表示 i 节点下的所有字符个数，T_i 表示 i 节点下的所有标签个数，LC_i 表示链接字符个数，$\neg LC_i$ 表示非链接字符个数，LT_i 表示链接标签个数，LC_b 表示 `<body>` 节点下的所有链接字符个数，C_b 表示 `<body>` 节点下的所有字符个数。遇到分母为零的情况，直接将分母设为 1。

为了从 Text Density 中区分噪音节点和内容节点，最直接的方法是以 `<body>` 的 Text Density 作为划分阈值。因为 `<body>` 比噪音节点包含更多的文本内容，又比内容节点包含更多的超链接，所以它是一个足够区分二者的中间值。

数据平滑可以削峰填谷，增加区域内部的内聚性和区域之间的差异性，虽然一般来说数据平滑可以取得较好的结果，它仍然会丢失一些低文本密度的节点。为了改进传统的平滑、基于阈值划分的方法，作者提出了 DensitySum 的方法。

DensitySum 基于这样的观察，网页的内容块在 DOM 树结构中都属于一个祖先节点，并且内容节点的文本密度显著高于噪声节点。如果把一个节点的所有子节点的文本密度累加，称为 DensitySum，那么 DensitySum 就会在内容节点上取得一个极高值。其定义如下：

定义 2.5 如果 N 是一个标签（对应 DOM 树中的一个节点），i 是 N 的一个子节点，那么 N 的 DensitySum 就是它所有子节点文本密度之和：

$$DensitySum_N = \sum_{i \in C} TextDensity_i \tag{2-8}$$

实现细节

作者一共提出了三种算法 CETD-DS，CECTD-S 和 CECTD-DS。CETD-DS 是基于 Text Density 和 DensitySum 的算法，CECTD-S 是基于 Composite Text Density 和数据平滑的算法，CECTD-DS 是基于 Composite Text Density 和 DensitySum 的算法。根据作者的实验对比，CECTD-DS 的抽取性能最佳，也是本章中参与对比的算法。如无特殊说明，以后简称 CECTD-DS 为 CETD 算法。

根据 [22] 给出的算法伪代码，我们用 Python 2.7 重新实现了 CETD。为了处理 HTML 文档中的错误，同样使用了 BeautifulSoup 尽可能纠正。并通过 BeautifulSoup，将 HTML 文档转化为内存中的 DOM 树，在进行计算和操作之前去除了所有 `<script>`，`<style>` 和 `comment`。

2.3.3 CEPR

Wu 等人于 2013 年提出了 CEPR（Content Extraction via Path Ratios）算法 [23]。

CEPR 算法的核心概念是 **T**ext to Tag **P**ath **R**atio，或简称为 TPR。作者首先在 DOM 树的基础上，提出 Tag Path 的概念，即从根节点到文本叶节点[8] 的一条路径，由 HTML 标签构成。例如 `div.div.div.h1` 就是一条 Tag Path。在 Tag Path 的基础上，提出 Text to Tag Path Ratio：

定义 2.6 假设 p 是一条 Tag Path，那么 p 的 Text to Tag Path Ratio（TPR）就是：

$$TPR(p) = \frac{\sum_{v \in accNodes(p)} length(c(v))}{|accNodes(p)|} \tag{2-9}$$

定义 2.6 中，$accNodes(p)$ 是路径 p 能够访问的文本叶节点集合，$c(v)$ 是文本叶节点 v 的文本内容。TPR 既是每条 Tag Path 的度量值，也是每个文本叶节点的度量值。在一般网页上，通常有这样的观察：

1. 内容节点通常有相似的 Tag Path，而噪声节点也有相似的 Tag Path；
2. 内容节点通常包含更多的文本，而噪声节点通常包含较少的文本；
3. 所有文本节点都是叶节点。

显然，内容节点的 TPR 值较高而噪声节点的 TPR 值较低，可以用来抽取核心内容。TPR 的阈值定为 $\tau = \lambda\sigma(H')$，λ 是调节参数，$\sigma(H')$ 是直方图的标准差。

进一步的研究表明，网页内容会包含标点符号，而噪声则没有。除此之外，文本长度和标点符号个数在内容节点中的分布差异很大，在噪声节点中却很少变化。作者进一步提出了 **E**xtended **T**ext to Tag **P**ath **R**atio（ETPR）：

[8] 文本全部分布在 DOM 树的叶节点上，见 2.2.1 节

定义 2.7 假设 p 是一条 Tag Path，那么 p 的 Extended Text to Tag Path Ratio（ETPR）就是：

$$ETPR(p) = TPR(p) \cdot \frac{\sum_{v \in accNodes(p)} puncNum(c(v))}{|accNodes(p)|} \cdot \sigma_{cs} \cdot \sigma_{ps} \tag{2-10}$$

定义 2.7 中 $puncNum(c(v))$ 是 v 中标点符号的个数，$\sigma_c s$ 和 $\sigma_p s$ 分别是路径 p 能访问的文本叶节点集合中文本长度和标点符号个数的标准差。

在 [21] 的高斯平滑算法基础上，作者提出以编辑距离作为权重，改善平滑效果。即公式（ 2-4 ）的卷积运算修改为公式（ 2-11 ）。

$$H'_i = \sum_{j=-r}^{r} w_{ij} k'_j H_{i-j}, r \leq i \leq len(H) - r \tag{2-11}$$

其中权重定义为：

$$w_{ij} = \begin{cases} 1, p_i = p_j \\ \frac{1}{(dist(p_i, p_j))^{\alpha}}, p_i \neq p_j \end{cases} \tag{2-12}$$

公式（ 2-12 ）中的 $dist(p_i, p_j)$ 是路径 p_i 和 p_j 的编辑距离[38]，α 是调节编辑距离对平滑贡献的参数。

实现细节

作者一共提出了三种算法 CEPR-TPR，CEPR-ETRP 和 CEPR-SETPR。CEPR-TPR 是基于 TPR 的算法，CEPR-ETPR 是基于扩展的 ETPR 的算法，CEPR-SETPR 是基于 ETPR 并进行加权高斯平滑处理后的算法。根据作者的实验对比，CEPR-SETPR 的抽取性能最佳，也是本章中参与对比的算法。如无特殊说明，以后简称 CEPR-SETPR 为 CEPR 算法。

根据 [23] 给出的算法伪代码，我们用 Python 2.7 重新实现了 CEPR。同样使用了 BeautifulSoup 来处理 HTML 文档，转化为 DOM 树结构。CEPR 算法需要判断标点符号，这里使用了 Python 标准库的 `string.punctuation` 来定义英文标点符号，并补充了一个中文标点符号列表。CEPR 算法需要计算路径的编辑距离，这里以 Tag 的尺度计算，而不是直接计算两个 Tag Path 字符串的编辑距离。

我们在小规模数据集上进行了参数调整实验，确定了和 [23] 中一致的参数配置：$\lambda = 0.8$，$\alpha = 1$ 和 $r = 1$。

2.4 实验与分析

2.4.1 评价指标

为了验证算法的抽取效果，我们定义标准评价指标精确率 Precision、召回率 Recall 和 F_1。通过比较算法抽取的文本 e 和金标准的文本 g，Precision、Recall 和 F_1 可以如下计算：

$$P = \frac{LCS(e,g).length}{e.length}, R = \frac{LCS(e,g).length}{g.length} \tag{2-13}$$

$$F_1 = \frac{2 \times P \times R}{P + R} \tag{2-14}$$

对于中文文档，视为字符的序列而不进行分词处理，因为中文分词需要额外的依赖，而且 CEVC 算法重在对有效字符的统计而不需要词的信息。$LCS(e,g)$ 是 e 和 g 的最长公共子序列，文档中两个相同的字符如果位置顺序不同，也会被 LCS 的计算区分开来。相比于 [21] 中使用的计算方法，把两个文档的词视为两个集合，该方法利用了不同词的位置信息而更加准确。

除了标准评价指标外，这里给出另一个常用的评价方法 Score。Score 在 [22] 和 [23] 中都提到过，是为了替换 CleanEval 中运算开销很大的 Levenshtein Distance[38]（对齐两个字符串所需要的插入、删除操作次数，不允许替换），而使用最长公共子序列计算产生的。其定义如下：

$$Score(e,g) = \frac{LCS(e,g).length}{e.length + g.length - LCS(e,g).length} = \frac{1}{\frac{1}{P} + \frac{1}{R} - 1} \tag{2-15}$$

2.4.2 数据集

为了在验证内容抽取算法对当今中文新闻、博客网站的实际抽取效果，我们从知名的网址导航网站 hao123[9] 和 360 导航[10] 中任意选取了 30 个知名的新闻网站和博客网站作为测试对象。网站详情和网页数量见表 2-1 和表 2-2 。

对每个网站，我们分别编写爬虫脚本下载原始 HTML 网页，然后通过人工编写的包装器抽取网页正文作为金标准，并存储为 UTF-8 编码的文本文件。为了确保数据集的质量，还对包装器抽取的文本进行了人工审核，排除了抽取错误的网

[9] https://www.hao123.com/

[10] https://hao.360.cn/

页和正文不足 100 字的网页。15 个新闻网站和 15 个博客网站总计含有 1270 个有效网页。

表 2-1 新闻数据集

网站	网址	数量
网易新闻	news.163.com	47
央视网新闻	news.cctv.com	45
凤凰资讯	news.ifeng.com	47
腾讯新闻	news.qq.com	53
新浪新闻	news.sina.com.cn	39
搜狐新闻	news.sohu.com	34
中国军网	www.81.cn	48
参考消息	www.cankaoxiaoxi.com	40
中国信息网	www.chinanews.com	46
央广网	www.cnr.cn	44
环球网	www.huanqiu.com	43
人民网	www.people.com.cn	39
南方网	www.southcn.com	50
新华网	www.xinhuanet.com	39
联合早报	www.zaobao.com	36
总计	-	650

表 2-2 博客数据集

网站	网址	数量
网易博客	blog.163.com	48
财经网博客	blog.caijing.com.cn	45
中金博客	blog.cnfol.com	31
东方财富博客	blog.eastmoney.com	46
和讯博客	blog.hexun.com	45
凤凰博客	blog.ifeng.com	45
瑞丽博客	blog.rayli.com.cn	39
科学网博客	blog.sciencenet.cn	47
新浪博客	blog.sina.com.cn	42
搜狐博客	blog.sohu.com	29
天涯博客	blog.tianya.cn	50
博客大巴	www.blogbus.com	13
博客中国	www.blogchina.com	45
博客网	www.bokee.com	46
博客日报	www.bokerb.com	49
总计	-	620

2.4.3 参数调整

CEVC 算法中包含一个阈值参数 α，它决定了什么时候终止算法 2.2 中的向下深入过程。α 的取值从 0 到 1，如果它太低的话，终止条件就会很苛刻，算法会因过于深入 DOM 树底层而返回一小段正文中的文本。这种情况下因为损失了其它核心内容，召回率会下降。另一方面，如果为 α 设定一个过高的值，算法过早终止，会返回不相关的内容，从而影响最终的精确率。

为了试验抽取性能和参数 α 的关系，我们从各个网站中抽取少量网页进行了测试，具体结果如图 2-5 所示。召回率随着 α 的升高而升高，精确率随着 α 的升高而降低，总体趋势符合之前的期望。从图中可以看到，作为一个综合性指标，F_1 在 α 从 0.3 增长到 0.6 的过程中都很稳定，保持在 0.95 以上。CEVC 算法对参数的变化并不敏感，因此在后续的实验中，设置 α 为 0.5。

图 2-5 Precision, Recall 和 F_1 随参数 α 的变化情况

2.4.4 实验结果

首先对 CEVC 算法的抽取性能进行实验评估，表 2-3 给出了 CEVC 在新闻和博客数据集上的实验结果，包含了 Precision、Recall、F_1 和 Score 四项指标。

我们可以看到 CEVC 算法在 30 个不同的新闻、博客网站上都取得了良好的内容抽取效果。对新闻网站的总体抽取效果达到了 97.4% 的 F_1，对博客网站的总体抽取效果达到了 94.4% 的 F_1，在所有测试集上，F_1 达到了 95.8%。表明 CEVC 算法能够适应新闻、博客类网站的核心内容抽取需求。

然后在同样的数据集上分别测试 CETR、CETD 和 CEPR，并和 CEVC 算法做比较，如表 2-4 所示。为了节省篇幅，这里只列出了两个综合性指标 F_1 和 Score，以 F_1/Score 的形式给出。

表 2-4 中每一行代表一个网站的测试结果，粗体标示的是 CETR、CETD、CEPR 和 CEVC 四种不同算法在 F_1 和 Score 指标上的最优者。在大约三分之二的测试网站上，CEVC 算法都能取得最优的内容抽取结果，CETD 和 CEPR 在剩余三分之一网站上抽取效果最优。在总体 F_1 指标上，CEVC 达到 95.8%，和 CETD 的 95.5% 大致相当，随后是 CEPR 的 92.5%，大幅度超越了 CETR 的 75.2%。

虽然新闻、博客类网站结构相近，能够使用同一种内容抽取算法处理，但从表 2-4 中的结果可以看出，不同算法在博客上的抽取效果都不同程度地低于新闻。因此有必要对新闻和博客分类计算各项指标，为了更直观地显示，绘制成表 2-6 所示的折线图，四种算法分别以四种不同的折线表示。

在各项指标上，四种算法抽取博客网站的效果都较新闻网站有不同程度的下

表 2-3 CEVC 在新闻和博客数据集上的实验结果

网站	网址	类型	Precision	Recall	F_1	Score
环球网	www.huanqiu.com	news	91.1%	96.4%	93.7%	88.1%
人民网	www.people.com.cn	news	96.2%	99.0%	97.6%	95.3%
凤凰资讯	news.ifeng.com	news	99.5%	99.9%	99.7%	99.4%
央视网新闻	news.cctv.com	news	93.7%	99.9%	96.7%	93.7%
联合早报	www.zaobao.com	news	97.5%	99.1%	98.3%	96.7%
新华网	www.xinhuanet.com	news	98.6%	99.9%	99.2%	98.5%
中国信息网	www.chinanews.com	news	98.1%	99.9%	99.0%	98.0%
中国军网	www.81.cn	news	97.6%	100.0%	98.8%	97.6%
网易新闻	news.163.com	news	94.7%	98.8%	96.7%	93.7%
腾讯新闻	news.qq.com	news	93.5%	99.9%	96.6%	93.4%
央广网	www.cnr.cn	news	97.7%	99.5%	98.6%	97.2%
参考消息	www.cankaoxiaoxi.com	news	95.3%	95.4%	95.3%	91.1%
搜狐新闻	news.sohu.com	news	97.5%	97.8%	97.6%	95.4%
南方网	www.southcn.com	news	91.5%	99.3%	95.2%	90.9%
新浪新闻	news.sina.com.cn	news	96.3%	100.0%	98.1%	96.3%
新闻	-	-	95.8%	99.1%	97.4%	95.0%
和讯博客	blog.hexun.com	blog	96.4%	97.3%	96.8%	93.9%
瑞丽博客	blog.rayli.com.cn	blog	86.6%	97.1%	91.6%	84.5%
凤凰博客	blog.ifeng.com	blog	96.9%	93.9%	95.3%	91.1%
中金博客	blog.cnfol.com	blog	86.8%	89.2%	88.0%	78.5%
财经网博客	blog.caijing.com.cn	blog	96.3%	95.9%	96.1%	92.5%
博客大巴	www.blogbus.com	blog	91.8%	100.0%	95.7%	91.8%
新浪博客	blog.sina.com.cn	blog	94.3%	86.1%	90.0%	81.9%
博客日报	www.bokerb.com	blog	94.3%	97.0%	95.7%	91.7%
科学网博客	blog.sciencenet.cn	blog	94.8%	95.0%	94.9%	90.3%
天涯博客	blog.tianya.cn	blog	93.8%	97.8%	95.8%	91.9%
博客中国	www.blogchina.com	blog	98.4%	99.6%	99.0%	98.0%
搜狐博客	blog.sohu.com	blog	94.4%	92.4%	93.4%	87.6%
东方财富博客	blog.eastmoney.com	blog	93.3%	97.1%	95.2%	90.8%
网易博客	blog.163.com	blog	81.7%	92.5%	86.7%	76.6%
博客网	www.bokee.com	blog	93.5%	100.0%	96.6%	93.5%
博客	-	-	93.4%	95.5%	94.4%	89.4%
总体	-	-	94.5%	97.2%	95.8%	92.0%

表 2-4 CETR、CETD、CEPR 和 CEVC 的对比结果

网站	CETR	CETD	CEPR	CEVC
网易新闻	68.3%/51.9%	96.9%/93.9%	**97.9%/95.9%**	96.7%/93.7%
央视网新闻	85.3%/74.4%	95.8%/91.9%	**99.1%/98.3%**	96.7%/93.7%
凤凰资讯	53.8%/36.8%	99.5%/99.1%	95.6%/91.6%	**99.7%/99.4%**
腾讯新闻	80.4%/67.2%	96.4%/93.0%	**96.9%/93.9%**	96.6%/93.4%
新浪新闻	81.4%/68.7%	**98.1%**/96.2%	93.6%/88.0%	**98.1%/96.3%**
搜狐新闻	73.7%/58.3%	**98.6%/97.3%**	96.0%/92.3%	97.6%/95.4%
中国军网	89.6%/81.2%	98.6%/97.3%	97.1%/94.3%	**98.8%/97.6%**
参考消息	46.5%/30.3%	**99.8%/99.7%**	92.6%/86.3%	95.3%/91.1%
中国信息网	86.5%/76.3%	**99.0%/98.1%**	98.9%/97.8%	**99.0%**/98.0%
央广网	85.8%/75.2%	98.5%/97.1%	92.0%/85.2%	**98.6%/97.2%**
环球网	79.6%/66.1%	92.1%/85.3%	**97.0%/94.3%**	93.7%/88.1%
人民网	87.8%/78.3%	96.9%/93.9%	89.6%/81.2%	**97.6%/95.3%**
南方网	87.8%/78.3%	**95.5%/91.4%**	93.3%/87.5%	95.2%/90.9%
新华网	97.1%/94.3%	**99.2%/98.5%**	99.0%/98.0%	**99.2%/98.5%**
联合早报	86.5%/76.3%	96.6%/93.3%	91.3%/84.0%	**98.3%/96.7%**
新闻	76.6%/62.1%	**97.4%**/94.9%	95.6%/91.6%	**97.4%/95.0%**
网易博客	58.3%/41.2%	**97.1%/94.4%**	92.1%/85.3%	86.7%/76.6%
财经网博客	85.5%/74.7%	93.7%/88.2%	94.0%/88.6%	**96.1%/92.5%**
中金博客	59.4%/42.3%	87.0%/76.9%	62.1%/45.0%	**88.0%/78.5%**
东方财富博客	66.5%/49.8%	87.9%/78.4%	92.1%/85.3%	**95.2%/90.8%**
和讯博客	55.6%/38.5%	88.2%/78.8%	82.7%/70.4%	**96.8%/93.9%**
凤凰博客	67.6%/51.0%	**96.4%/93.1%**	89.3%/80.7%	95.3%/91.1%
瑞丽博客	61.4%/44.3%	88.4%/79.2%	88.8%/79.9%	**91.6%/84.5%**
科学网博客	67.6%/51.0%	89.4%/80.8%	91.0%/83.6%	**94.9%/90.3%**
新浪博客	82.4%/70.1%	**90.0%**/81.8%	86.4%/76.1%	**90.0%/81.9%**
搜狐博客	78.5%/64.7%	93.1%/87.0%	86.0%/75.4%	**93.4%/87.6%**
天涯博客	91.0%/83.5%	**98.9%/97.8%**	91.3%/83.9%	95.8%/91.9%
博客大巴	87.1%/77.2%	95.4%/91.2%	**97.1%/94.4%**	95.7%/91.8%
博客中国	90.8%/83.2%	98.3%/96.7%	93.6%/88.0%	**99.0%/98.0%**
博客网	80.8%/67.8%	95.6%/91.5%	86.9%/76.8%	**96.6%/93.5%**
博客日报	85.5%/74.7%	**97.2%/94.5%**	96.5%/93.2%	95.7%/91.7%
博客	73.9%/58.6%	93.8%/88.2%	89.6%/81.1%	**94.4%/89.4%**
总体	75.2%/60.2%	95.5%/91.3%	92.5%/86.0%	**95.8%/92.0%**

降。CEVC 的下降幅度最小，而 CEPR 和 CETR 的召回率下降幅度最大，从对新闻的 99% 左右下降到对博客的 88% 左右。CEVC 对新闻和博客的抽取效果更为稳定，优于对比算法。

图 2-6 CETR、CETD、CEPR 和 CEVC 在新闻、博客数据集上的各项指标

2.4.5 结果分析

相对于职业编辑撰写的新闻，博客的内容由用户编辑产生，形式更加自由，所使用的 HTML 标签和样式也更丰富，所以不同方法对博客内容的抽取效果都有所下降。新闻正文一般由 <p> 标签组成的段落构成，格式简洁也很扁平化，而博客内容可以由富文本编辑器产生，带有字体、颜色、大小等控制标签，结构上层次也比较深，给准确抽取内容带来了挑战。

CETR 算法对新闻和博客的整体抽取 F_1 只有 75.2%，虽然召回率 93% 较高，但大幅度牺牲了精确率 63.1%。

CETR 算法以“行”为单位处理 HTML 文档，针对每一行计算 Tag Ratio，根据

Tag Ratio 的高低来划分内容或噪声。虽然算法经过了平滑处理，也结合序列信息转化为二维数据处理，但从实验结果来看，依然将一部分噪声标记为内容，例如推荐文章、免责声明等。CETR 对每一行都平等对待，没有结合 DOM 的结构信息，局部的文本密集也被识别为有效内容，这就导致最终的结果精确率很低。

CETD 算法的整体抽取性能和 CEVC 基本相当，对新闻的 F_1 都是 97.4%，总体 F_1 分别是 95.5% 和 95.8%，几乎没有差别。

这是由于两者的核心评估指标相似，CETD 基于 DOM 节点的文本密度，CEVC 基于 DOM 节点的有效字符数量，也都考虑了链接与网页噪声之间的相关性。CETD 基于 DensitySum 的方式，选取子节点文本密度之和最大的节点标记为内容，而 CEVC 则从 `<body>` 开始，逐层选择有效字符最多的子节点不断深入，二者的输出都是一个特定 DOM 节点下的文本内容，所以整体抽取效果很接近。

实验早期 CETD 算法对博客网站的 F_1 不足 90%，分析具体网页后发现，部分博客网站包含 `<textarea>` 这样的标签，标签内部包含大量代码，导致 CETD 对文本密度的计算向 `<textarea>` 倾斜，最终错误地选择 `<textarea>` 作为正文内容。为了更好地比较 CETD 算法的性能，在预处理阶段，除了 `<script>` 和 `<style>` 之外，还去除了 `<textarea>` 标签。最终的实验结果如表 2-4 所示，对博客网站的 F_1 提升到 93.8%。这表明虽然 CETD 和 CEVC 的抽取性能相当，但其预处理阶段需要处理更多的特殊标签，而 CEVC 算法加入了对停止词的识别，从而能够有效区分文本内容，不受这些标签的干扰。

CEPR 算法的整体 F_1 为 92.5%，精确率和召回率都在 92% 左右，对精确率和召回率都没有偏倚。

CEPR 的计算过程和 CETR 有相似之处，CETR 以“行”为单位，而 CEPR 以路径模式为单位，对每个文本叶节点计算出一个指标。二者也都进行了高斯平滑，只不过 CEPR 的平滑过程加入了路径编辑距离的权重。CEPR 的抽取效果较 CETR 提升很多，但分散的文本叶节点之间，只靠路径模式的相似性关联，容易被遗漏或误报，不如 CETD 和 CEVC 基于 DOM 内容节点整体来做决策。

CEVC 算法对正文内容较短的网页，抽取效果较差。如表 2-4 所示，参考消息和环球网包含部分短讯，新闻主体内容很短，导致算法错误地将推荐文章、新闻评价板块标记为核心内容，从而降低了 F_1。这也是 CEVC 基于文本聚集程度来划分核心内容的一个缺陷。

以图 2-7 的一篇科学网博客[11]为例，直观地比较不同算法的抽取区域。图中标

[11] http://blog.sciencenet.cn/

记了 A、B、C、D 四个区域，A、B、C 为博客正文，而 D 为博客的相关推荐文章。

• CETR 算法抽取结果为 A、B、C、D，虽然没有遗漏正文，却包含了噪声部分 D。

• CETD 算法抽取结果为 B，这是因为 B 区域是一个 <backquote> 标签，包含了许多 <p> 标签，其子节点的文本密度之和最大，从而遗漏了 A 和 C。

• CEPR 算法根据路径模式划分内容，丢弃了 Tag Path Ratio 较低的 A 部分，选择了 B 和 C。

• CEVC 算法根据有效字符在 DOM 树中的分布，正确地选择了 A、B、C 作为核心内容。

图 2-7 不同算法的抽取区域比较

第 3 章 基于锚节点的论坛帖子抽取

3.1 概述

随着互联网技术的快速发展，我们已经进入 Web 2.0 的时代[39]。不同于早期 Web 1.0 的网络应用，Web 2.0 更强调用户生成内容（User-Generated Content，UGC），可用性和互操作性，开始以用户为中心，体现资源平等的特点。用户能够在社交网络中相互交流和协作，越来越多的人向虚拟社区贡献着自己产生的内容，不同的观点和意见在这里交融碰撞，而不是在早期网站上那样被动地接收有限的内容。网络论坛就是 Web 2.0 中的典型代表，用户可以在论坛上发帖、回帖进行讨论，是一种交互性强、内容丰富而及时的网络应用。

一个典型的网络论坛组织结构如图 3-1 所示，论坛一般按照话题分为若干个板块（Board）；板块内部用户围绕具体事件进行的一系列发帖、回帖统称为 Thread，一般按照发帖时间线性排列；在一个 Thread 内部，用户的每一次发帖是 Post。以网易论坛[1]的一篇帖子为例，如图 3-2 所示，红色实线圈出的是每一个 Post，它们按照时间顺序排列，包含发帖人、发帖时间、发帖内容等元信息，共同组成了一篇帖子的核心内容。

图 3-1 论坛典型结构

由于 HTML 网页所固有的半结构性，被设计为用户能够方便阅读，却不能被机器有效读取的形式。虽然用户能够很容易区分不同的帖子，但面对结构样式各异的论坛网站，如何从论坛网页中自动化地抽取帖子，转化为结构化的数据，是一个

[1] http://bbs.163.com/

图 3-2 论坛帖子实例

很有挑战性的工作，也是舆情爬虫系统的重要环节。本章利用论坛帖子中普遍存在的发帖时间信息作为锚节点，提出一种基于锚节点的帖子抽取方法，3.2 节详述该方法 PEAN（Post Extraction via Anchor Nodes），2.3 节简单介绍对比算法 MiBAT 并给出算法实现细节，3.4 节进行对比实验并分析结果。

3.2 PEAN 方法

3.2.1 树匹配算法

在数据记录抽取的研究工作中，很多方法[35, 36] 都依赖树匹配算法来检测模板或衡量相似度，这里对树匹配问题给出形式化定义和简单介绍。

在 2.2.1 节中已经提到，网页可以解析为 DOM 树从而描述其结构信息。树是广泛使用的数据结构，形式化称为有向无环图，我们关心的 DOM 树是一种带标记的有序有根树（labeled ordered rooted tree）。有序有根树意味着根节点固定，子节点之间的相对顺序也是固定的。

树由节点集合 V 和边集合 T 构成的二元组，$T = (V, E)$。为了描述方便，记两个树 T_1 和 T_2 的节点集合分别为 V_1 和 V_2。在两个树的节点集合间寻找一个最优

的一一映射，这就是树的比较或匹配。树匹配[40] 的形式化定义如下：

定义 3.1 从 T_1 到 T_2 的匹配 M，是一个有序二元组的集合 (u, v)，$u \in V_1$，$v \in V_2$，并对所有的 $(u_1, v_1), (u_2, v_2) \in M$，满足以下条件：

1. $u_1 = u_2$，当且仅当 $v_1 = v_2$；
2. u_1 在 u_2 的左边，当且仅当 v_1 在 v_2 的左边；
3. u_1 是 u_2 的祖先，当且仅当 v_1 是 v_2 的祖先。

在定义 3.1 中，第一条保证了在一个匹配中每个节点最多出现一次，第二条保证了节点间的兄弟关系在匹配中被保留，第三条保证了节点间的父子关系在匹配中被保留。直观上看，树匹配描述了把一个树转换为另一个树所需要的一系列操作，而忽略这些操作之间的顺序。因此树匹配问题可以转化为树的编辑距离问题，找到最优的匹配就是找到代价最小的编辑操作。

树匹配问题的计算很消耗时间，解决它都需要平方复杂度以上的算法。不过在具体的应用领域，可以使用一个受限制的树匹配描述，自顶向下（top-down）匹配[41]，它在很多 Web 相关的研究中被成功运用。其定义如下：

定义 3.2 从 T_1 到 T_2 的匹配 M 如果满足以下条件就是自顶向下匹配：对所有非根节点 $u \in V_1$，$v \in V_2$，如果 $(u, v) \in M$，那么 $(parent(u), parent(v)) \in M$，其中 $parent(v)$ 是 v 的父节点。

图 3-3 自顶向下的树匹配

如图 3-3 所示，虚线所连接的就是这两个树之间的自顶向下匹配，灰色的节点是不能匹配的节点。可以注意到，左树的 G 节点和右树的 H 节点处于同一位置，但由于标记不同而不匹配。父节点不匹配后，虽然 G 节点和 H 节点都有相同的 I 节点，也不能匹配，这就是自顶向下限制的体现。

在自顶向下的限制之下，[42] 给出了一个动态规划的树匹配算法，时间复杂度为 $O(n^2)$。如算法 3.1 所示，树匹配过程和通过动态规划求两个字符串的最长公共子序列有相似之处。在两个根节点 A 和 B 匹配成功（标记相同）后，递归寻找 A 和 B 的第一级子树之间的最大匹配节点数，构成一个 $m \times n$ 的矩阵，m 和 n 分别是 A 和 B 第一级子树的个数。通过这个矩阵，使用动态规划的方法计算出 A 与 B 之间的最大匹配节点数。

Input: Tree A and B
Output: Number of maximal matched nodes

```
if the roots of A and B contain distinct labels then
    return 0 ;
end
m = the number of first-level subtrees of A ;
n = the number of first-level subtrees of B ;
Initialize a m × n matrix M with all zeros ;
for i = 1 to m do
    for j = 1 to n do
        W[i, j] = treeMatch(A_i, B_j) ;
        /* A_i and B_j are the ith and jth first-level subtrees of A
           and B respectively                                        */
        M[i, j] = max(M[i, j − 1], M[i − 1, j], M[i − 1, j − 1] + W[i, j]) ;
    end
end
return M[m, n] + 1 ;
```

算法 3.1: treeMatch(A, B)

3.2.2 锚节点

使用锚节点来辅助信息抽取，在 [36] 中就已经提出。锚节点具有这样的特点：

- 作为关键的数据单元出现在每一个数据记录的结构化模板中；
- 本身的文本具有一定的格式，容易抽取和识别。

发帖时间就是符合这样要求的一种锚节点。首先论坛帖子中几乎都存在发帖时间，在图 3-2 中，3 个不同帖子的楼层关系，正是靠发帖时间排序的。其次发帖

时间还是结构化模板中的一部分，格式固定，比如都位于帖子的右上角。最后，形如“2014-05-13 20:07:23”这样的时间格式，使用简单的正则表达式就能有效识别。

图 3-4 所示的就是本文使用的，用来识别发帖时间的正则表达式，能够识别多种不同的时间格式：

- `2014-06-12 10:10:20`
- `2014/06/12 10:10`
- `2014/6/12 10:10:20`

图 3-4 识别发帖时间的正则表达式

如果以发帖时间作为锚节点（Pivot），那么在 DOM 树中包含发帖时间的最底层的节点，就定义为锚节点：

定义 3.3 i 是 DOM 中的一个节点，如果 $|D_i| > 0$ 且 $\forall j \in descendant(i)$ 都有 $|D_j| = 0$，那么 i 就是锚节点

定义 3.3 中，$|D_i|$ 表示节点 i 的文本中发帖时间的个数，$descendant(i)$ 表示 i 的子孙节点集合。

我们可以计算出当前节点所包含的锚节点个数 Pivots，通过 Pivots 来指导程序发现帖子存在的区域。类似于算法 2.1 ，这是一个自顶向下的递归计算过程，Pivots 被作为属性，存储在相应节点中。同样在处理之前，需要从 DOM 树中移除 `<script>`，`<style>` 和 `comment`。经过算法 countPivots 处理之后，我们可以得到一个标注了各个节点 Pivots 的 HTML 页面，如图 3-5 所示。

3.2.3 定位帖子节点

经过大量实例分析，我们发现论坛帖子有一个共同的特点，它们都位于 DOM 树中同一个父节点之下。图 3-5 的帖子形式如 `<div#post_30073401>`，都位于父节点 `<div#postlist>` 之下。因为帖子之间是相似的，它们在逻辑地位上是平等的，论坛网站的服务端程序从数据库中取出数据记录，经过前端模板的渲染最终形成网页，为了统一对它们进行调整和控制，隶属于同一个父节点是很自然的。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transit
<html class="widthauto" date_pivots="22" xmlns="http://www.w3.org/1999/xhtml">
▶ <head>…</head>
▼ <body class="pg_viewthread" date_pivots="22" id="nv_forum" onkeydown="if(event.keyCode==27) return false;">
    <div class="rtzyw960 rtzybg40" date_pivots="0">

    </div>
  ▶ <div date_pivots="0" style="width:100%; clear:both">…</div>
    <div date_pivots="0" id="append_parent"></div>
    <div date_pivots="0" id="ajaxwaitid"></div>
  ▶ <div date_pivots="0" id="hd">…</div>
  ▼ <div class="wp" date_pivots="22" id="wp">
    ▶ <div class="wp a_t jgbj_q">…</div>
    ▶ <div class="bm cl" date_pivots="0" id="pt">…</div>
    ▶ <div class="wp" date_pivots="0">…</div>
      <a date_pivots="0" name="posts"></a>
    ▼ <div class="wp cl" date_pivots="22" id="ct">
      ▶ <div class="pgs mbm cl" date_pivots="0" id="pgt">…</div>
      ▼ <div class="pl bm bmw" date_pivots="22" id="postlist">
        ▶ <table cellpadding="0" cellspacing="0" class="plh" date_pivots="0">…</table>
        ▶ <div date_pivots="1" id="post_30073401">…</div>
        ▶ <div date_pivots="3" id="post_30073602">…</div>
        ▶ <div date_pivots="3" id="post_30073757">…</div>
        ▶ <div date_pivots="2" id="post_30073796">…</div>
        ▶ <div date_pivots="3" id="post_30073898">…</div>
        ▶ <div date_pivots="2" id="post_30073948">…</div>
        ▶ <div date_pivots="2" id="post_30073983">…</div>
        ▶ <div date_pivots="3" id="post_30074017">…</div>
        ▶ <div date_pivots="1" id="post_30074027">…</div>
        ▶ <div date_pivots="2" id="post_30074040">…</div>
        ▶ <div class="pl" date_pivots="0" id="postlistreply">…</div>
        ▶ <table cellpadding="0" cellspacing="0" date_pivots="0">…</table>
        </div>
      ▶ <form autocomplete="off" date_pivots="0" id="modactions" method="post" name="modactions">…</form>
```

图 3-5 时间锚节点示例

从图 3-5 中可以看到，每个帖子都至少包含一个时间锚节点，直观地看，时间锚节点密集的区域就是帖子存在的区域。如果从 <body> 节点开始，每次都选择时间锚节点最多的子节点深入，就能定位出一条从 <body> 通往帖子节点的正确路径，如图 3-5 中红线标示的节点。为了描述方便，将这条路径在例 3.1 中列出，其中 RMD 和 MPR 在下文中详述。

例 3.1 RMD 和 MPR 计算过程

```
1. <body> Pivots=22 RMD=0 MPR=1
2. <div#wp> Pivots=22 RMD=0 MPR=1
3. <div#ct> Pivots=22 RMD=0 MPR=1
4. <div#postlist> Pivots=22 RMD=0.29 MPR=0.14 帖子的父节点
5. <div#post_30073602> Pivots=3 RMD=0 MPR=1
6. ...
```

每次沿着 Pivots 最大的子节点不断深入，这个过程不能一直持续下去，否则会陷入叶节点中，而错失最终的目标节点。我们定义两个指标，来判断何时停止向下深入的过程。

定义 3.4 相对平均偏差（Relative Mean Deviation，RMD）是样本的平均偏差除以样本均值的绝对值：

$$RMD = \frac{\frac{1}{n}\sum_{i=1}^{n}|x_i - \bar{x}|}{|\bar{x}|} \tag{3-1}$$

把同一个父节点下的子节点的时间锚节点个数组成一组样本，去除那些锚节点数为零的样本，然后计算相对平均偏差 RMD，来衡量锚节点在子节点中分布的均匀程度。

因为锚节点属于格式化模板中的一部分，在帖子中分布是较为均匀的，所以当 RMD 小于某个阈值时，表明锚节点分布均匀，当前节点下很可能包含一组帖子。当父节点只有一个子节点包含锚节点时，相对平均偏差为零，这种特殊情况下毫无疑问应当继续深入。在例 3.1 中，1-3 行的 RMD 都为零，时间锚节点高度集中在一个子节点中。

定义 3.5 i 是 DOM 树中的一个节点，i 的最大子节点锚节点占比 MPR（Max Pivots Ratio）如下所示，其中 P_i 表示 i 的锚节点个数，j 是 i 的子节点：

$$MPR_i = max_{j \in children(i)} \frac{P_j}{P_i} \tag{3-2}$$

在 DOM 树中每次向下深入一步，我们认为最大子节点包含了父节点的绝大部分锚节点。当定义 3.5 中的 MPR 低于一个阈值时，表明最大子节点的锚节点并不足以在父节点中占据统治地位，继续向下深入可能会损失信息。

通过 RMD 和 MPR 联合确定停止向下深入的条件，即 RMD 和 MPR 都小于各自的阈值时，向下深入的过程立即终止，当前节点被标记为帖子的父节点。此时，extract 过程负责从当前节点的子节点中筛选出真正的帖子。这个过程详见算法 3.2 。

第 12 行，在比较 RMD 和 MPR 时，还需要确定 pivots 集合的大小，即包含锚节点的子节点不能只有一个。在这种情况下单一样本的 RMD 一定为零，却并不意味着子节点中锚节点分布均匀，而是锚节点全部集中在一个子节点中，必须继续向下深入。

算法 3.2 能够很快运行结束，因为它至多被递归调用 h 次，h 是 DOM 树的深度，时间复杂度为 $O(h)$。

3.2.4 筛选候选帖子

在定位了帖子的父节点后，筛选真正的帖子并不是一件简单的工作，因为帖子的兄弟节点中可能包含很多噪声部分，例如广告、装饰性内容等。以一篇西陆论

```
Input: DOM node N
Output: Posts
pivots = ∅ ;
maxNode = null ;
for C ∈ N.children() do
    if C.pivots > max(pivots) then
        maxNode = C ;
    end
    pivots.append(C.pivots) ;
end
if maxNode is not null then
    RMD = computeRMD(pivots) ;
    MPR = max(pivots) / sum(pivots) ;
    if pivots.length > 1 and RMD < α and MPR < β then
        return extract(N) ;
    else
        stepInto(maxNode) ;
    end
end
```

算法 3.2: stepInto(N)

坛[2]的帖子为例，如图 3-6 所示，红色实线标示的是真正的帖子，第一个为楼主发布的主帖，剩下 10 个为回帖。

这些帖子都位于父节点 `<body>` 之下，但 `<body>` 之下还有很多非帖子的节点，`<div#BAIDU_UNION>` 是百度联盟提供的广告，`<div#d_main>` 是主帖前后的推荐信息。如何准确筛选真正的帖子，需要利用帖子和帖子之间 DOM 树结构的相似性。

在 3.2.1 节的树匹配算法的基础上，可以给出树的相似度定义。

定义 3.6 已知 M 是两个树 T_1 和 T_2 的匹配，则 T_1 和 T_2 的相似度是：

$$TreeSim(T_1, T_2) = \frac{|M|}{(|V_1| + |V_2|)/2} \tag{3-3}$$

其中 $|M|$ 是 M 中匹配节点的对数，$|V_1|$ 和 $|V_2|$ 分别是 T_1 和 T_2 节点的个数。

[2]http://club.xilu.com/

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loc
<html>
▶ <head style="width:98%;">…</head>
▼ <body style="width:98%;">
  ▶ <div id="BAIDU_DUP_fp_wrapper" style="position: absolute; left: -1px; bottom: -1px; z-inde
    <div id="OnlineNotifyGroup"></div>
  ▶ <div id="cnzz_float_101553" style="position: fixed; right: 5px; bottom: 5px; z-index: 2147
  ▶ <div id="BAIDU_UNION__wrapper_u1149318_0_right" style="box-sizing: content-box; width: 131
  ▶ <div id="BAIDU_UNION__wrapper_u1149318_0_left" style="box-sizing: content-box; width: 131p
  ▶ <div class="center">…</div>
  ▶ <div class="center">…</div>
  ▶ <div class="center">…</div>
  ▶ <div class="center">…</div>
    <a name="UP" id="UP"></a>
    <!--上下主题开始-->
  ▶ <div id="d_main">…</div>
    <!--上下主题结束-->
  ▶ <div id="d_main">…</div>
  ▶ <div id="d_main" align="left">…</div>
    <!--for mingyan start-->
    <script src="/style/js/mingyan_float.js" type="text/javascript" charset="utf-8"></script>
  ▶ <script type="text/javascript">…</script>
    <!--for mingyan end-->
  ▶ <div class="center">…</div>
    <!--上下主题开始-->
  ▶ <div id="d_main">…</div>
    <!--上下主题结束-->
  ▶ <div class="center">…</div>
  ▶ <div class="center">…</div>
  ▶ <div class="center">…</div>
  ▶ <div class="center">…</div>
  ▶ <div class="center">…</div>
  ▶ <div class="center">…</div>
  ▶ <div class="center">…</div>
  ▶ <div class="center">…</div>
  ▶ <div class="center">…</div>
  ▶ <div class="center">…</div>
  ▶ <div style="MARGIN: 0px auto; WIDTH: 920px" align="center">…</div>
  ▶ <div class="center">…</div>
```

图 3-6 帖子中的噪声

从定义 3.6 中可知，树的相似度受树节点规模的影响，在 $|M|$ 相同的情况下，如果 $|V_1|$ 或 $|V_2|$ 增大就会稀释整个相似度。

我们从图 3-7 中的例子分析，这两个子树代表两个帖子，虚线连接的是相互匹配的节点。帖子第一层子节点中，代表发帖时间和发帖内容的节点都能够相互匹配，但灰色节点标志的用户生成内容却不能匹配。论坛帖子有别于商品展示信息，它的内容由用户生成（UGC），具有很高的自由度，因此体现出高度差异化和多样性。左边帖子直接包含文本和换行标签 `<br>`，右边帖子则通过 `<p>` 包裹文本，并提供了一张图片 `<img>`。

如果直接使用定义 3.6 的树相似度计算，就会被高度自由的 UGC 节点干扰，使两个帖子原本应该很高的相似度降低，最后被认为不相似而遗漏。和字符串的最长公共子序列类似，最大匹配节点数反映了两个树之间“重合”部分的大小。虽然 UGC 内容带来的结构、形式差异会导致帖子间相似度参差不齐，但相互之间能够

图 3-7 帖子的匹配

匹配的最大节点数却比较稳定，可以利用这个“重合”部分来度量相似性。

从候选子节点中筛选真正帖子的过程 extract，如算法 3.3 所示。算法 3.2 中选定的 maxNode 在这里被当做基准节点，其它至少包含一个锚节点的兄弟节点一一与之进行树匹配计算，然后以节点作为 key、最大匹配节点数作为 value 存储在一个 map 数据结构中。

基准节点 maxNode 被认定为帖子，output 过程将其内容输出。其它节点按照和基准节点之间重合部分的大小，从高到低排列，虽然它们彼此的相似度可能参差不齐，但重合度却比较稳定。按照这个顺序遍历最大匹配节点个数，如果发现某个最大匹配节点数低于前者的一半，就认为筛选结束，该节点之前的都属于目标帖子，而放弃之后重合度太低的节点。

记候选帖子数量为 m，每个帖子的平均节点数为 n。算法 3.3 的运算分为两部分：

1. 首先是基准节点和候选帖子节点之间的树匹配过程，因为 treeMatch 的复杂度为 $O(n^2)$，所以其复杂度为 $O(mn^2)$。

2. 另一部分是对候选帖子的排序、遍历过程，排序复杂度为 $O(m \log m)$，所以其复杂度为 $O(m \log m)$。

一般情况下，由于论坛结构和 UGC 内容日趋丰富，再加上分页对单篇网页帖子数量的限制，候选帖子数量 m 小于帖子的平均节点数 n，自然有 $n^2 > \log m$，算法 3.3 extract 的总体时间复杂度为 $O(mn^2)$。

```
Input: DOM node N, maxNode
Output: Posts
mappings = ∅ ;
for C ∈ N.children() do
    if C ≠ maxNode and C.pivots > 0 then
        mappings[C] = treeMatch(C, maxNode) ;
    end
end
output(maxNode) ;
lastNum = 0 ;
for C ∈ mappings ordered from high to low do
    if mappings[C] < lastNum/2 then
        break ;
    end
    lastNum = mappings[C] ;
    output(C) ;
end
```

算法 3.3: extract(N, maxNode)

3.2.5 算法实现

本文中 PEAN 算法使用 Python 2.7 实现。

面对 HTML 网页中普遍存在的代码错误，例如缺失结束标签、使用非标准的标签和错误的嵌套等，这里使用 Python 第三方库 BeautifulSoup 来尽可能纠正这些错误。BeautifulSoup 是一个可以从 HTML 或 XML 文件中提取数据的 Python 库，它为操作 DOM 提供了一套方便的接口，底层调用 lxml 等具体的解析器，能够处理有代码错误的网页。

PEAN 需要从 HTML 文档中抽取发帖时间，所使用的正则表达式在 3.2.2 节已经描述过。

3.3 对比算法及实现

本章的对比算法为 MiBAT[36]（**Mi**ning data records **B**ased on **A**nchor **T**rees），该算法也通过 Python 2.7 实现。

面对论坛帖子中 UGC 内容格式自由、复杂多变而干扰相似度计算的问题（图 3-7），MiBAT 的主要解决方法是，在树相似度计算过程中，尽可能选择隶属于格式化模板的一部分节点进行比较，而不是将所有节点全部考虑进去。这里相互比较的节点子集，称为 tree fragments。树相似度定义扩展为：

定义 3.7 已知 M 是两个树 T_1 和 T_2 的匹配，f 是 tree fragment 选择函数，则 T_1 和 T_2 在 f 作用下的相似度是：

$$TreeSim_f(T_1, T_2) = \frac{M \cap (f(V_1) \times f(V_2))}{(|f(V_1)| + |f(V_2)|)/2} \tag{3-4}$$

定义 3.7 中，$f(V_1) \times f(V_2) = (u, v)|u \in f(V_1), v \in f(V_2)$。最佳情况是，tree fragment 选择函数直接选取了树的模板部分，但这难以做到。[36] 定义了几种 tree fragment 选择函数，并最终使用了 Pivot and Siblings（PS）选择函数。

定义 3.8 **Pivot and Siblings (PS)** $f_{PS}(V) = \{v|v \in V, parentv = parentp\}$ 其中 p 是 T 的一个锚节点，$parent(v)$ 是 v 的父节点。

如定义 3.8 所示，简单来说，就是用锚节点和锚节点的兄弟节点作为模板部分的近似，从而规避 UGC 节点对相似度计算的干扰。

为了从 DOM 中定位候选 Anchor Trees，在兄弟节点之间横向进行相似度比较，具体过程是：通过正则表达式匹配获得候选锚节点后，可以用它们来鉴别新的 Anchor Trees，一旦新的 Anchor Tree 加入到集合中，就可以更新候选锚节点集合。这个过程一直持续到没有新的 Anchor Tree 加入即终止。

实现细节

本文中 MiBAT 算法使用 Python 2.7 实现。

为了处理 HTML 文档中的错误，同样使用了 BeautifulSoup 尽可能纠正。并通过 BeautifulSoup，将 HTML 文档转化为内存中的 DOM 树，在进行计算和操作之前去除了所有 `<script>`，`<style>` 和 `comment`。

MiBAT 需要从 HTML 文档中抽取发帖时间，所使用的正则表达式在 3.2.2 节已经描述过。

3.4 实验与分析

3.4.1 评价指标

为了验证算法的抽取效果，我们定义标准评价指标准确率 Precision、召回率 Recall 和 F_1。通过比较算法抽取的帖子集合 e 和金标准的帖子集合 g，Precision、Recall 和 F_1 可以如下计算：

$$P = \frac{e \cap g}{|e|}, R = \frac{e \cap g}{|g|} \tag{3-5}$$

$$F_1 = \frac{2 \times P \times R}{P + R} \tag{3-6}$$

3.4.2 数据集

为了验证帖子抽取算法在实际中文论坛上的抽取效果，我们从知名的网址导航网站 hao123[3] 和 360 导航[4] 中任意选取了 20 个知名的论坛网站作为测试对象，如表 3-1 所示。从采集的网页中，去除失效的、没有回帖的网页，总计有效网页 973 个，平均每个网站 50 个以内。

数据的采集和标注都需要消耗很多的精力，为了节省人力成本，我们通过构建特定论坛的包装器来解决这个问题。如例 3.2 所示，FORUM 是 Python 的字典结构，以 key-value 对的形式存储各个论坛的详细配置。

以网易论坛为例，采集程序从板块地址`http://bbs.news.163.com/`入手，得到某个板块的网页，然后通过 thread 的 CSS 选择符定位每个 thread 的 url。选择符 `ul.rankList a[href]` 的含义是，选择所有 `<ul class='rankList'>` 标签下的，有 `href` 属性的 `<a>` 标签。同理，通过帖子的 CSS 选择符可以标注真正的帖子，供后续实验分析。

这里的 CSS 选择符可以是一个列表，包含不止一个选择符，以应对更负责的情况。例如在凯迪社区的配置中，帖子既可能是 `div.reply-box`，也可能是 `div.posted-box-add`，对应主帖和后续跟帖。

[3]https://www.hao123.com/

[4]https://hao.360.cn/

表 3-1 论坛数据集

编号	论坛名称	论坛代码	网址
1	网易论坛	163	http://bbs.163.com/
2	360 安全社区	360safe	http://bbs.360safe.com/
3	55BBS 论坛	55bbs	http://bbs.55bbs.com/
4	腾讯论坛	qq	http://bbs.ent.qq.com/
5	环球论坛	huanqiu	http://bbs.huanqiu.com/
6	春秋社区	ichunqiu	http://bbs.ichunqiu.com/
7	卡饭论坛	kafan	http://bbs.kafan.cn/
8	远景论坛	pcbeta	http://bbs.pcbeta.com/
9	看雪学院	pediy	http://bbs.pediy.com/
10	瑞丽论坛	rayli	http://bbs.rayli.com.cn/
11	天涯社区	tianya	http://bbs.tianya.cn/
12	铁血论坛	tiexue	http://bbs.tiexue.net/
13	精睿	vc52	http://bbs.vc52.cn/
14	华声论坛	voc	http://bbs.voc.com.cn/
15	中华网论坛	china	http://club.china.com/
16	新浪论坛	sina	http://club.history.sina.com.cn/
17	凯迪社区	kdnet	http://club.kdnet.net/
18	宽带山	pchome	http://club.pchome.net/
19	西陆论坛	xilu	http://club.xilu.com/
20	泡泡俱乐部	pcpop	http://pop.pcpop.com/

例 3.2 论坛包装器示例

```
FORUM = {
  '163': {
    'name': '网易论坛',                    # 论坛名称
    'home': 'http://bbs.163.com/',         # 论坛主页地址
    'board': 'http://bbs.news.163.com/',   # 论坛板块地址
    'thread': ['ul.rankList a[href]'],     # 论坛thread的CSS选择符
    'post': ['div.tie-item'],              # 论坛帖子的CSS选择符
  },
}
```

3.4.3 实验结果

在 20 个论坛构成的数据集上对 MiBAT 和 PEAN 算法分别进行评估，实验结果如表 3-2 所示，列出了精确率、召回率和 F_1 三项指标。每一行代表一个网站的测试结果，粗体标示的是 MiBAT 和 PEAN 在 F_1 上的优胜者。

从表 3-2 中可以看出，在不同论坛网站上，两种算法各有优劣。MiBAT 的总体精确率较高，达到了 99.6%，高于 PEAN 的 98.3%，但却牺牲了召回率，导致总体 F_1 只有 84.6%，比 PEAN 的 94.7% 低了很多。实验结果表明，PEAN 相比于 MiBAT 在召回率指标上有大幅度提升，总体 F_1 指标也优于 MiBAT。

表 3-2 MiBAT 和 PEAN 在论坛数据集上的对比结果

论坛名称	MiBAT-P	MiBAT-R	MiBAT-F_1	PEAN-P	PEAN-R	PEAN-F_1
网易论坛	100.0%	97.0%	98.5%	100.0%	100.0%	**100.0%**
360 安全社区	100.0%	63.4%	77.6%	100.0%	97.5%	**98.7%**
环球论坛	100.0%	48.8%	65.6%	100.0%	97.7%	**98.8%**
春秋社区	100.0%	62.3%	76.8%	100.0%	100.0%	**100.0%**
卡饭论坛	100.0%	78.9%	88.2%	100.0%	95.3%	**97.6%**
远景论坛	100.0%	99.6%	**99.8%**	100.0%	97.9%	98.9%
看雪学院	100.0%	98.8%	**99.4%**	100.0%	97.5%	98.8%
腾讯论坛	100.0%	39.3%	56.5%	100.0%	92.0%	**95.8%**
瑞丽论坛	100.0%	100.0%	**100.0%**	62.5%	67.7%	65.0%
新浪论坛	100.0%	99.7%	**99.8%**	100.0%	92.9%	96.3%
天涯社区	100.0%	98.6%	99.3%	100.0%	100.0%	**100.0%**
铁血论坛	83.1%	7.7%	14.0%	100.0%	89.3%	**94.3%**
精睿	100.0%	86.4%	92.7%	100.0%	98.6%	**99.3%**
华声论坛	100.0%	94.7%	**97.3%**	100.0%	89.4%	94.4%
中华网论坛	100.0%	88.5%	**93.9%**	93.5%	13.9%	24.2%
凯迪社区	100.0%	87.5%	**93.3%**	100.0%	83.0%	90.7%
宽带山	99.8%	98.3%	**99.1%**	99.8%	98.0%	98.9%
西陆论坛	100.0%	61.0%	**75.8%**	100.0%	53.5%	69.7%
泡泡俱乐部	100.0%	76.4%	86.6%	100.0%	95.0%	**97.4%**
55BBS	100.0%	93.0%	**96.4%**	92.6%	90.7%	91.6%
总计	99.6%	73.5%	84.6%	98.3%	91.3%	**94.7%**

3.4.4 结果分析

MiBAT 在树相似度计算过程中，尽可能选择隶属于格式化模板的一部分节点进行比较，而不是将所有节点全部考虑进去。最佳情况是，tree fragment 选择函数直接选取了树的模板部分，但这难以做到。作者给出了一个近似，即锚节点和锚节点的兄弟节点，认为这部分节点在不同帖子中结构相对固定，有利于相似度的计算。

但在实验结果中发现，随着近年来论坛页面复杂程度的提高，帖子和帖子之间的差异性越来越大。许多帖子在锚节点和锚节点的兄弟节点这个层面上出现了不同，虽然不匹配的节点**绝对数量**并不多，但在相似度计算中真正起关键作用的**相对比例**却足够大，所以导致整体相似度大幅度降低。

例如图 3-8 的一篇 360 安全社区[5]所示，红色矩形标示的是 MiBAT 所抽取的帖子，网页中的第一个帖子被遗漏了。分析原因发现，蓝色矩形标示的是锚节点，但后面的两个帖子中出现了对发帖的引用（红色椭圆），干扰了锚节点和兄弟节点结构的稳定性，算法倾向于选择相似度更高的结果。

图 3-8 360 安全社区

[5]http://bbs.360safe.com

还有一种情况，在论坛页面的楼主发帖（第一个帖子）中，形式结构和后续跟帖在锚节点层面上有较大不同。如图 3-9 的一篇天涯社区[6]所示，蓝色矩形标示的锚节点中，楼主发帖的锚节点包含额外的点击、回复信息，导致依赖锚节点及其兄弟节点的相似度匹配失效。

PEAN 算法直接计算两个候选帖子中最大匹配节点个数，来衡量“重合”部分的大小，避免了片面选择 Tree Fragment 带来的不稳定性，在实验中取得了更好的效果。

图 3-9 天涯论坛

[6]http://bbs.tianya.cn

第 4 章 应用实例——Web 新闻聚合系统

4.1 概述

为了验证本文提出的模板无关的内容抽取技术在爬虫系统中的实际效果，设计并实现了一个 Web 新闻聚合系统。实时采集多种渠道来源的 Web 新闻，通过模板无关的内容抽取技术抽取新闻正文，并结合一些领域规则抽取标题、日期等元信息，封装成结构化信息转储在后端的数据仓库中，提供面向中文的全文检索支持，也为后续的数据分析处理提供了接口。

在这样一个网络信息爆炸的时代，网络舆论对社会的影响越来越重要，Web 新闻聚合系统对关注网络舆情，研究热点事件的产生、传播规律，构建语料库都具有实际意义。

本章 4.2 节介绍 Web 新闻聚合系统的总体架构，4.3 节详细阐述系统各个模块的设计和关键技术，4.4 节对系统的运行效果进行评估。

4.2 系统架构和工作流程

4.2.1 系统架构

面对海量异构、持续变化的新闻网站，大数据的存储压力，在设计 Web 新闻聚合系统时，需要考虑以下几个关键问题：

- 关键模块的可扩展能力
- 存储与检索性能
- 系统的稳定可用性

Web 新闻聚合系统的总体架构如图 4-1 所示，标示了系统的关键组件和数据流动情况。

4.2.2 总体工作流程

Web 新闻聚合系统的核心工作流程分为以下几个部分：

（1）任务分解和下发

MySQL 数据库中存储了各个新闻网站的配置，例如 RSS 源、新闻网页的 URL 模式等。Controller 作为调度模块，定时从数据库中加载配置，解析并分解为相应

图 4-1 Web 新闻聚合系统整体架构

任务，发送到消息队列中，每一条消息都完整包含了完成一个特定任务需要的信息，等待 Worker 消费并执行。同时 MySQL 中还维护了相应的任务表，记录每个任务的基本信息和当前状态。用户对整个系统的操作都集中在配置中，包括添加新的新闻网站，设定不同的抓取优先级和频率等，给用户提供了简单统一的接口。

（2）网页下载和信息抽取

网页下载和信息抽取由一组幂等的 Worker 构成，每个 Worker 是一个独立的进程，可以部署在一台机器上，也可以部署在多台机器上进行水平扩展。Worker 监听消息队列，接收任务并执行，根据需要更新数据库或者转发消息给下一个模块。

在新闻采集系统中，任务主要分为两类，新闻列表解析和新闻信息抽取。新闻列表解析就是从 RSS 源或包含新闻列表的网页中，抽取每一篇新闻的 URL，供后续信息抽取。然后新闻信息抽取模块根据指定的 URL，抽取新闻正文和标题等元信息，然后转储在数据仓库中。为了防止重复抓取同一个页面而造成资源浪费，还需要一个全局的 URL 过滤组件，判断当前 URL 是否已经重复，这里使用 Redis 实现。

（3）存储与检索

MySQL 作为传统的关系型数据库，不擅长全文检索，因此选用了 Elasticsearch 作为存储和检索引擎，将需要检索支持的结构化信息导入 Elasticsearch，而把传统的配置管理、任务管理等交给 MySQL 处理，充分发挥各自的优势。Elasticsearch 具有很强的扩展能力，多机部署构成集群，面向用户提供透明的 HTTP 接口，能够满

足存储和检索的分布式扩展需求。

4.2.3 新闻采集流程

Web 新闻从多种信息来源采集，对每一个新闻来源，系统都有与之对应的新闻列表解析模块，从不同渠道的来源中解析新闻页面 URL，转交给信息抽取模块。这种架构的优势在于，通过消息队列的中转，新闻列表解析和新闻信息抽取相互解耦，屏蔽底层细节，为信息抽取提供统一的接口，也利于计算资源的水平扩展。

图 4-1 表现了 Controller 和 Worker 的一般关系，为了详细阐述新闻采集过程，将 Worker 细分为负责新闻列表解析的 Parser，以及负责新闻信息抽取的 Extractor，形成图 4-2 。其中 1–6 标示的是各个流程：

1. Controller 以定时任务或其它方式触发，从 MySQL 数据库加载配置；
2. Controller 分解任务，封装为消息投放到消息队列中，消息还可以设定相应的优先级和延迟执行时间，从而灵活配置计算资源；
3. Parser 接收任务后从指定的新闻源解析得到一系列新闻 URL；
4. Parser 将新闻 URL 封装成消息投放到消息队列中；
5. Extractor 接收任务后从 URL 指定的新闻页面中，抽取正文内容及其它元信息；
6. Extractor 将新闻正文内容及其它元信息转储到 Elasticsearch 中；

4.3 系统模块和关键技术

4.3.1 新闻列表解析

Web 新闻采集的来源包括 RSS、元搜索和一般新闻网站三类。RSS 是被广泛接受的标准信息聚合接口，这是系统优先考虑的采集来源，但部分新闻网站可能不提供 RSS 接口，所以需要针对一般新闻网站的列表解析模块。元搜索利用现有搜索引擎对新闻的聚合能力，是对热点新闻聚合的一种补充方式。

4.3.1.1 RSS

RSS 可以是以下三个解释的其中一个：

- Really Simple Syndication
- RDF (Resource Description Framework) Site Summary
- Rich Site Summary

RSS 使用一组标准的 Web 信息流格式来发布频繁更新的信息，如博客、新闻

图 4-2 新闻采集流程

头条和音视频等。RSS 是互联网上被广泛采用的内容包装和投递协议，搭建了一个信息迅速传播的技术平台，使得每个人都成为潜在的信息提供者。

一个 RSS 文档包含摘要内容和一些元信息，例如发布日期和作者的名字。RSS 是 XML 格式的普通文本，简单的格式使得它既能被程序自动化解析也能被普通人理解，一个 RSS 如例 4.1 所示[1]。

在新闻列表解析模块，我们最关心的是新闻页面的 URL，即 `<link>`。通过简单的正则表达式匹配，就能抽取出一组新闻页面的 URL，然后提交给新闻信息抽取模块。

4.3.1.2 元搜索

为了利用现有搜索引擎对新闻的聚合能力，系统还采用元搜索的方式获得新闻。元件搜索引擎又称集合型搜索引擎，将多个单一搜索引擎集成在一起，将用户的检索提问同时提交给多个独立的搜索引擎，同时检索多个数据库。并根据多个独立搜索引擎的检索结果进行二次加工，如对检索结果去重、排序，然后输出给自己。

元搜索根据关键词提交查询请求，系统通过维护热点关键词列表，能够及时获得热点新闻，这对于热点新闻的聚合是一种有效的补充方式。Web 新闻聚合系

[1]https://en.wikipedia.org/wiki/RSS

例 4.1 RSS 示例

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
<channel>
  <title>RSS Title</title>
  <description>This is an example of an RSS feed</description>
  <link>http://www.example.com/main.html</link>
  <lastBuildDate>Mon, 06 Sep 2010 00:01:00 +0000 </lastBuildDate>
  <pubDate>Sun, 06 Sep 2009 16:20:00 +0000</pubDate>
  <ttl>1800</ttl>

  <item>
    <title>Example entry</title>
    <description>Here is an interesting description.</description>
    <link>http://www.example.com/blog/post/1</link>
    <guid isPermaLink="true">7bd204c6-1655-4c27</guid>
    <pubDate>Sun, 06 Sep 2009 16:20:00 +0000</pubDate>
 </item>
</channel>
</rss>
```

统面向国内舆情热点，选取了知名的中文搜索引擎百度[2]和 360[3] 作为元搜索入口：

- http://news.baidu.com/ns?word=**keyword**&tn=news&from=news
- http://news.so.com/ns?q=**keyword**&src=newhome

URL 中粗体标示的 **keyword** 可以替换为搜索关键词，搜索引擎会返回与之相关的新闻。如图 4-3 所示的百度新闻搜索，包含了新闻题目、摘要、发布时间和链接等信息。通过人工编写的包装器，可以抽取出一组新闻页面的 URL，由于搜索引擎较少，编写包装器的开销并不大。

[2]http://www.baidu.com/

[3]https://www.so.com/

图 4-3 百度新闻搜索

4.3.1.3 一般新闻网站

RSS 基于 XML 提供了一种统一的信息发布格式，大大降低了聚合信息的代价，也是系统优先考虑的采集来源。但仍存在部分新闻网站，不提供 RSS 接口，我们采用编写包装器的方式解析新闻列表。因为我们关心的是新闻页面的 URL，所以包装器主要包含新闻页面 URL 的正则表达式。

如表 4-1 所示，这是一组新闻页面 URL 的正则表达式，其中既包含绝对地址（`http://\w+.cnr.cn/\w+/\w+/\d{8}/\w+.shtml`），也包含相对地址（`/news/\w+/story\d+-\d+`）。在新闻列表的 HTML 页面上匹配相应的 URL 模式，再根据相对地址、绝对地址进行转换，就能抽取出一组新闻页面的 URL。

4.3.2 新闻信息抽取

新闻信息抽取模块的输入是新闻页面的 URL，然后返回抽取的发布日期、标题和新闻正文三元组。新闻信息抽取流程如图 4-4 所示，在抽取各项信息之前，还需要检测新闻网页的编码格式，将 HTML 文档在内存中解析为 DOM 树。在信息抽取结束后，还需要判断抽取的三元组是否有效，无效则放弃。

表 4-1 新闻页面 URL 模式

编号	新闻网站	URL 模式
1	联合早报	`/news/\w+/story\d+-\d+`
2	中国信息网	`/\w+/\d\d\d\d/\d\d-\d\d/\d+.shtml`
3	南方网	`http://\w+.southcn.com/\w+/\d\d\d\d-\d\d/\d\d/content_\d+.htm`
4	央广网	`http://\w+.cnr.cn/\w+/\w+/\d{8}/\w+.shtml`
5	中国军网	`\d\d\d\d-\d\d/\d\d/content_\d+.htm`

图 4-4 新闻信息抽取流程图

4.3.2.1 发布日期

发布日期是构成新闻必不可少的元素，新闻的发布日期一般在标题之下。通过 3.2.2 节中提供的日期正则表达式，可以从 HTML 文档中抽取日期，从中进行筛选得到最终的发布日期。发布日期的抽取过程如算法 4.1 所示，只选择大于某个特定时期（如 2016 年 1 月 1 日），并且在当前时间之前的日期作为候选，最后从其中选择最近的日期，作为这篇新闻的发布日期。

```
Input: HTML Text Src
Output: Publication date
candidate_dates = ∅ ;
for every date string s extracted from Src via regular expression do
    Build date structure d from date string s ;
    if d > 2016/1/1 and d < now() then
        candidate_dates.add(d) ;
    end
end
if candidate_dates is not null then
    return maximum(candidate_dates) ;
else
    return null ;
end
```

算法 4.1: extractPubDate(Src)

4.3.2.2 新闻标题

HTML 文档中 `<title>` 标签表示网页的标题，但它一般不能直接作为新闻的标题。以 2.1 节图 2-1 为例，这篇新闻的 `<title>` 为 **国务院参事:“建 10 个类似北京超大城市”是误解 | 仇保兴 | 超大城市 _ 新浪新闻**，但其中包含了分隔符、关键词、新闻网站名称，需要从中过滤这些额外信息。通过观察发现，新闻的标题一般也会出现在 HTML 文档的 `<h1>` 标签中，`<h1>` 表示一级标题。在计算 `<titie>` 和 `<h1>` 的最长公共子串后，这些额外信息就能被过滤。

为了新闻标题抽取的准确性，我们通过以下步骤确定最终的标题：

1. 抽取 `<title>` 和 `<h1>`；
2. 计算 `<titie>` 和 `<h1>` 的最长公共子串；

3. 如果最长公共子串长度小于 5，选取 `<titie>` 为标题；
4. 如果 `<titie>` 为空，选取 `<h1>` 为标题。

4.3.2.3 新闻正文

根据 2.4 节的对比结果，我们选择本文提出的 CEVC（Content Extraction via Valid Characters）算法来抽取新闻正文。CEVC 易于实现，预处理阶段依赖小，并且在多个数据集上得到了优于其它对比算法的抽取结果。

4.3.3 存储和检索

由于传统的关系型数据库 MySQL 对中文全文检索支持不足，且扩展性差，除了配置管理、任务管理等传统业务存储于 MySQL 中，未来增长较快、关系结构简单、需要全文检索的数据则存储于其它数据仓库中。

在对比了 Mongodb[4]，Solr[5]和 Elasticsearch[6]后，我们选择了 Elasticsearch 作为数据存储和检索的一体化平台。

Elasticsearch 是一个实时的分布式搜索和分析引擎，它建立在全文搜索引擎框架 Apache Lucence[7] 的基础之上，其诞生伊始就完全基于分布式架构。Elasticsearch 可以用于全文搜索、结构化搜索和分析，在当前互联网上已经有许多成功使用的案例：

- Github 使用 Elasticsearch 搜索 20TB 的数据，包括 13 亿的文件和 1300 亿行的代码[43]；
- Stackoverflow 使用 Elasticsearch 代替 SQL 全排索引，因为其良好的扩展性和低成本；
- 维基百科使用 Elasticsearch 进行全文检索并高亮显示搜索关键词；

Elasticsearch 为了提供一个扩展性强同时易用的分布式系统，做了许多自动化工作，例如文档划分到不同容器或者分片中，集群中索引和搜索的跨节点负载均衡，数据分片的自动冗余备份，无缝扩展或者恢复整个集群。

Elasticsearch 是面向文档的分布式搜索引擎，文档可以代表一个对象。新闻对象在 Elasticsearch 中的字段映射如例 4.2 所示，其中定义了 `pubdate` 为日期类型，`title` 和 `content` 为字符串，`smartcn` 是所使用的中文分词组件。

[4]https://www.mongodb.com/

[5]http://lucene.apache.org/solr/

[6]https://www.elastic.co/

[7]http://lucene.apache.org/

例 4.2 新闻对象的字段映射

```
"news": {
  "properties": {
    "url": {"type": "string"},
    "src": {"type": "string"},
    "pubdate": {"type": "date", "format": "yyyy-MM-dd HH:mm:ss"},
    "title": {"type": "string", "analyzer": "smartcn"},
    "content": {"type": "string", "analyzer": "smartcn"}
  }
}
```

4.4 运行效果评估

Web 新闻聚合系统的开发环境是一台 MacBook Pro（2 GHz Intel Core i7 处理器，8G 内存，256G SSD），系统功能的实现得益于许多开源项目的支持：

- 编程语言：Python 2.7
- 存储与检索：MySQL，Elasticsearch，Redis
- 消息队列：Beanstalkd
- Web：Flask Web Framework，JQuery，Bootstrap

图 4-5 新闻网站的采集情况

系统从 2016 年 6 月 8 日至 14 日，平均每天采集 2000 条新闻，涉及政治、经济和时事等各个方面。新闻数量 TOP 5 的网站分布如图 4-5 所示，新闻采集量最多的网站依次是中国新闻网、环球网、南方网、新华网和凤凰资讯。

为了评估新闻信息抽取模块对标题、发布日期抽取的准确率，我们从系统采集的新闻中随机抽取 200 篇，人工审核标题、发布日期抽取是否正确，最终结果如表 4-2 所示。新闻标题通过 `<title>` 和 `<h1>` 联合确定，在新闻网页中基本都符合这个规律，所以在样本中准确率达到了 100%。而发布日期的判断不够严谨，可能被网页中的其它日期干扰，影响了最终的准确率。

需要注意的是，新闻信息抽取模块在抽取标题、发布日期和新闻内容三元组后，进行了简单的有效性判断（标题多于 5 个字，发布日期有效，新闻内容多于 100 个字）。这使得系统在采集过程中，丢弃了部分新闻，可能是由于正当的原因（URL 失效，非新闻网页），也可能由于信息抽取的失败，所以表 4-2 中的结果要优于实际情况。

表 4-2 新闻标题和发布日期抽取效果评估

抽取项目	总数量	正确数量	准确率
标题	200	200	100%
发布日期	200	191	95.5%
标题和发布日期	200	191	95.5%

新闻检索界面如图 4-6 所示，在检索框输入关键词“奥兰多”，系统返回与之关联的新闻，并对关键词进行高亮处理。表格中只给出了新闻内容的部分片段，点击新闻标题，能够进入一个独立的页面，详细展示全部新闻内容。借助于 Elasticsearch，系统能够从数据仓库中匹配标题或正文包含搜索关键词的新闻，并按照相关性的高低排序返回结果。Elasticsearch 使用了高效的索引结构，系统对关键词查询的响应时间在 0.5 秒以内，能够适应实时检索的要求。

Web 新闻聚合系统使用了模板无关的内容抽取算法 CEVC，并结合一些领域规则抽取标题、发布日期，不需要对新闻页面单独构造包装器，大大降低了系统扩展和维护的成本。

系统从 RSS、元搜索和一般新闻网站三个信息来源采集新闻，通过模板无关的抽取技术整合了各个渠道的差异，使人工成本限制在新闻列表解析中。借助于 RSS 的统一协议和元搜索技术，新闻列表解析中人工配置和包装器构造的成本能够接受。系统的后续工作将集中于如何自动扩展新闻采集来源，进一步降低人工

参与的成本，实现系统的高度自动化。

新闻检索

Show 10 entries　　Search: 奥兰多

标题	来源	发布日期	新闻内容
奥兰多夜店发生疑似恐怖袭击事件 致50人遇难	中国信息网	2016-06-12 23:22:52	华盛顿6月12日电 美国佛罗里达州奥兰多市当地时间12日凌晨发生一起枪击事件，造成50人遇难，53人受伤。枪手已被警方击毙。警方表示，这可能是一起恐怖袭击事件。 奥兰多警察局长约翰·米那12日在记者会上表示，当天凌晨2时左右，一名枪手携带一支来福枪和一把手枪现身当地一家夜店，并与酒吧安保发生交火。奥兰多警方在接到报警后迅速赶往现场，与枪手展开枪战。枪手随后退回至夜店内，劫持了300多名人
枪声骤起、子弹横飞，50人死于美国奥兰多枪击案	腾讯新闻	2016-06-13 06:56:00	自动播放 正在加载... 据新华社消息，美国佛罗里达州奥兰多市一家同性恋夜店当地时间6月12日凌晨2点左右发生大规模枪击事件。一名男性枪手闯入夜店开枪扫射并劫持人质，与警方对峙。警方动用装甲车和炸药才攻入夜店，击毙枪手，救出数十名人质。这起事件造成50人死亡、53人受伤。美国联邦调查局已经把这起枪击
特朗普就奥兰多恐袭要求奥巴马下台 希拉里退选	腾讯新闻	2016-06-13 13:01:00	资料图：特朗普原标题：特朗普就奥兰多恐袭要求奥巴马下台 希拉里退选【环球网综合报道】美国伤亡惨重的枪击案——奥兰多枪击案发生后，共和党总候选人特朗普发布声明，指责现任总统、民主党人奥巴马在讲话中闭口不提“极端伊斯兰”，叫板民主党候选人希拉里敢不敢说这两个词，敢不敢挑明此次枪击与伊斯兰教极端恐怖主义有关。当地时间12日，美国奥兰多市一家同性恋酒吧遭枪袭，造成至少50人死亡，53人受伤。奥巴马称，这是
习近平就美国奥兰多枪击事件向奥巴马致慰问电	搜狐新闻	2016-06-13 12:06:54	http://news.sohu.com/20160613/n454114741.shtml news.sohu.com false 央视网 http://m.news.cctv.com/2016/06/13/ARTIunlt4JoMAdDY080USTjm160613.shtml report 264 2016年6月13日，国家主席习近平致电美国总统奥巴马，就美国佛罗里达州奥兰多市发生枪击事
美奥兰多枪击案凶手：恐同、家暴、被FBI两次调查	搜狐新闻	2016-06-12 23:37:27	枪手身份确认为奥马尔·撒廷（Omar Sateen） 文｜张梦圆 当地时间12日凌晨，美国佛罗里达州奥兰多市一间同性恋夜总会遭枪手袭击，造成至少50死亡，53人受伤，目前枪手已被击毙。这是美国历史上伤亡最严重的枪击案。枪手身份确认为奥马尔·撒廷（Omar Sateen），是一名29岁出生在纽约的美国人，其父母均来自阿富汗。 效忠IS、恐同、家暴倾向 据《纽约每
奥兰多枪击案遇难者人数下调至49人 不包括枪手	搜狐新闻	2016-06-13 21:17:23	中新网6月13日电 据外媒报道，当地时间13日，美国当局将奥兰多夜总会枪击案的遇难人数下调至49人，称原先枪手本人被计算在了遇难者之内。 美国联邦调查局(FBI)特工保罗·威斯帕在新闻发布会上称，“我们不把枪手算在受害者之中”。他表示，大约一半的受害者家属已经接到了通知。 此外，此次枪击事件还导致超过50人受伤。 当地时间6月12日晚，数百名美国民众聚集在白宫外举行

图 4-6 新闻检索界面

结　论

互联网具有快速传播和广泛覆盖的特性，对互联网舆情进行有效监控是必不可少的。在一个聚焦于新闻、博客和论坛的多通道爬虫系统中，海量异构、持续变化的特点，给大范围舆情监控带来困难，迫切需要一种高度自动化的 Web 信息抽取方式，以降低系统扩展和维护的成本。

本文的主要工作成果如下：

（1）针对新闻、博客这类正文集中的网站，提出了一种基于有效字符的 Web 内容抽取方法 CEVC（Content Extraction via Valid Characters）。该方法主要基于这样的观察，网页中不属于链接并包含停止词的文本，更有可能是主要内容。定义这样的字符为有效字符，根据它们在 DOM 树中的分布，逐级确定正文区域，并最终提取正文。与之前的算法 CETR（基于文本标签比）、CETD（基于文本密度）和 CEPR（基于文本标签路径比）进行了比较。实验结果表明，CEVC 算法在各项评价指标上都优于 CETR 和 CEPR，虽然抽取性能和 CETD 相当，但在预处理阶段依赖更小，适用性更强。

（2）针对论坛网站，提出了一种论坛帖子抽取算法 PEAN（Post Extraction via Anchor Nodes）。该方法利用论坛帖子中普遍存在的发帖时间信息作为锚节点，根据它们在 DOM 树中的分布情况，定位论坛帖子集中的区域，并结合树匹配算法，在候选子树中过滤噪声，最终抽取出论坛帖子。实验结果表明，PEAN 相比于 MiBAT 在召回率指标上有大幅度提升，总体 F_1 指标也优于 MiBAT。

（3）为了验证本文提出的信息抽取算法的实际效果，根据实际需求设计并实现了一个针对新闻的应用实例——Web 新闻聚合系统。介绍了系统架构和工作流程，并对系统模块和关键技术做了详细阐述，最后对系统运行效果进行评估。系统从 RSS、元搜索和一般新闻网站三个信息来源采集新闻，通过模板无关的抽取技术整合了各个渠道的差异，使人工成本限制在新闻列表解析中。

为了进一步提高多通道爬虫系统的自动化采集能力，在以下几个方面还值得继续深入研究：

（1）研究新闻、博客采集源的自动扩展，进一步减少人工参与。

（2）在抽取论坛帖子的基础上，进一步研究作者、发帖内容等元信息的抽取。

（3）研究通用论坛的爬行策略，从论坛中自动解析板块入口和帖子入口。

参考文献

[1] Rahman A, Alam H, Hartono R. Content extraction from HTML documents[C] // 1st Int. Workshop on Web Document Analysis (WDA2001). 2001 : 1 – 4.

[2] Liu L, Pu C, Han W. XWRAP: An XML-enabled wrapper construction system for web information sources[C] // Data Engineering, 2000. Proceedings. 16th International Conference on. 2000 : 611 – 621.

[3] Sahuguet A, Azavant F. Building intelligent web applications using lightweight wrappers[J]. Data & Knowledge Engineering, 2001, 36(3) : 283 – 316.

[4] Kushmerick N. Learning to remove internet advertisements[C] // Proceedings of the third annual conference on Autonomous Agents. 1999 : 175 – 181.

[5] Davison B D. Recognizing nepotistic links on the web[J]. Artificial Intelligence for Web Search, 2000 : 23 – 28.

[6] Marek M, Pecina P, Spousta M. Web page cleaning with conditional random fields[C] // Building and Exploring Web Corpora: Proceedings of the Fifth Web as Corpus Workshop, Incorporationg CleanEval (WAC3), Belgium. 2007 : 155 – 162.

[7] Bauer D, Degen J, Deng X, et al. FIASCO: Filtering the Internet by Automatic Subtree Classification, Osnabruck[C] // Building and Exploring Web Corpora: Proceedings of the 3rd Web as Corpus Workshop, incorporating Cleaneval : Vol 4. 2007 : 111 – 121.

[8] Cai D, Yu S, Wen J-R, et al. VIPS: a visionbased page segmentation algorithm[R]. [S.l.] : Microsoft technical report, MSR-TR-2003-79, 2003.

[9] Song R, Liu H, Wen J-R, et al. Learning block importance models for web pages[C] // Proceedings of the 13th international conference on World Wide Web. 2004 : 203 – 211.

[10] Fernandes D, de Moura E S, Ribeiro-Neto B, et al. Computing block importance for searching on web sites[C] // Proceedings of the sixteenth ACM conference on Conference on information and knowledge management. 2007 : 165 – 174.

[11] Lin S-H, Ho J-M. Discovering informative content blocks from Web documents[C] // Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. 2002 : 588 – 593.

[12] Yi L, Liu B, Li X. Eliminating noisy information in web pages for data mining[C] // Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. 2003 : 296 – 305.

[13] Bar-Yossef Z, Rajagopalan S. Template detection via data mining and its applications[C] // Proceedings of the 11th international conference on World Wide Web. 2002 : 580 – 591.

[14] Chen L, Ye S, Li X. Template detection for large scale search engines[C] // Proceedings of the 2006 ACM symposium on Applied computing. 2006 : 1094 – 1098.

[15] Finn A, Kushmerick N, Smyth B. Fact or fiction: Content classification for digital libraries[J], 2001.

[16] Debnath S, Mitra P, Giles C L. Automatic extraction of informative blocks from webpages[C] // Proceedings of the 2005 ACM symposium on Applied computing. 2005 : 1722 – 1726.

[17] Debnath S, Mitra P, Giles C L. Identifying content blocks from web documents[G] // Foundations of Intelligent Systems. [S.l.] : Springer, 2005 : 285 – 293.

[18] Mantratzis C, Orgun M, Cassidy S. Separating XHTML content from navigation clutter using DOM-structure block analysis[C] // Proceedings of the sixteenth ACM conference on Hypertext and hypermedia. 2005 : 145 – 147.

[19] Gottron T. Combining content extraction heuristics: the CombinE system[C] // Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services. 2008 : 591 – 595.

[20] Gottron T. Content code blurring: A new approach to content extraction[C] // Database and Expert Systems Application, 2008. DEXA'08. 19th International Workshop on. 2008 : 29 – 33.

[21] Weninger T, Hsu W H, Han J. CETR: content extraction via tag ratios[C] // Proceedings of the 19th international conference on World wide web. 2010 : 971 – 980.

[22] Sun F, Song D, Liao L. Dom based content extraction via text density[C] // Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval. 2011 : 245 – 254.

[23] Wu G, Li L, Hu X, et al. Web news extraction via path ratios[C] // Proceedings of the 22nd ACM international conference on Conference on information & knowledge management. 2013 : 2059 – 2068.

[24] Kushmerick N. Wrapper induction for information extraction[D]. [S.l.] : University of Washington, 1997.

[25] Hsu C-N, Dung M-T. Generating finite-state transducers for semi-structured data extraction from the web[J]. Information systems, 1998, 23(8) : 521 – 538.

[26] Muslea I, Minton S, Knoblock C. A hierarchical approach to wrapper induction[C] // Proceedings of the third annual conference on Autonomous Agents. 1999 : 190 – 197.

[27] 孟小峰, 王海燕, 谷明哲, et al. XWIS 中基于预定义模式的包装器 [J]. 计算机应用, 2001, 21(9) : 1 – 3.

[28] Chuang S-L, Hsu J Y-J. Tree-structured template generation for web pages[C] // Web Intelligence, 2004. WI 2004. Proceedings. IEEE/WIC/ACM International Conference on. 2004 : 327 – 333.

[29] Zheng S, Song R, Wen J-R, et al. Efficient record-level wrapper induction[C] // Proceedings of the 18th ACM conference on Information and knowledge management. 2009 : 47 – 56.

[30] 李效东, 顾毓清. 基于 DOM 的 Web 信息提取 [J]. 计算机学报, 2002, 25(5) : 526 – 533.

[31] Embley D W, Jiang Y, Ng Y-K. Record-boundary discovery in Web documents[C] // ACM SIGMOD Record : Vol 28. 1999 : 467 – 478.

[32] Buttler D, Liu L, Pu C. A fully automated object extraction system for the World Wide Web[C] // Distributed Computing Systems, 2001. 21st International Conference on.. 2001 : 361 – 370.

[33] Chang C-H, Lui S-C. IEPAD: information extraction based on pattern discovery[C] // Proceedings of the 10th international conference on World Wide Web. 2001 : 681 – 688.

[34] Wang J, Lochovsky F H. Data extraction and label assignment for web databases[C] // Proceedings of the 12th international conference on World Wide Web. 2003 : 187 – 196.

[35] Liu B, Grossman R, Zhai Y. Mining data records in Web pages[C] // Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. 2003 : 601 – 606.

[36] Song X, Liu J, Cao Y, et al. Automatic extraction of web data records containing user-generated content[C] // Proceedings of the 19th ACM international conference on Information and knowledge management. 2010 : 39 – 48.

[37] Gibson D, Punera K, Tomkins A. The volume and evolution of web page templates[C] // Special interest tracks and posters of the 14th international conference on World Wide Web. 2005 : 830 – 839.

[38] Levenshtein V I. Binary codes capable of correcting deletions, insertions, and reversals[C] // Soviet physics doklady : Vol 10. 1966 : 707 – 710.

[39] O'reilly T. What is Web 2.0: Design patterns and business models for the next generation of software[J]. Communications & strategies, 2007(1) : 17.

[40] Tai K-C. The tree-to-tree correction problem[J]. Journal of the ACM (JACM), 1979, 26(3) : 422 – 433.

[41] Selkow S M. The tree-to-tree editing problem[J]. Information processing letters, 1977, 6(6) : 184 – 186.

[42] Yang W. Identifying syntactic differences between two programs[J]. Software: Practice and Experience, 1991, 21(7) : 739 – 755.

[43] Paro A. ElasticSearch cookbook[M]. [S.l.] : Packt Publishing Ltd, 2015.

攻读硕士学位期间发表的论文及其他成果

（一）发表的学术论文

[1] Xueyang Ma(马雪阳), Hongli Zhang, Xiangzhan Yu, Yingjun Li. A Template Independent Approach for Web News and Blog Content Extraction[C] //3rd International Conference on Information Science and Control Engineering (ICISCE), Beijing, China, July 8-10, 2016（录用待发表）

哈尔滨工业大学学位论文原创性声明

学位论文原创性声明

本人郑重声明：此处所提交的学位论文《面向多通道爬虫的 Web 信息抽取技术研究》，是本人在导师指导下，在哈尔滨工业大学攻读学位期间独立进行研究工作所取得的成果，且学位论文中除已标注引用文献的部分外不包含他人完成或已发表的研究成果。对本学位论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。

作者签名：　　日期：　　年　　月　　日

致　谢

在哈工大计算机学院度过的三年硕士研究生生活中，我得到了许多无私的帮助，也感受了网安实验室家一般的温暖。值此论文完成之际，向所有关心、支持和帮助我的老师、同学和亲人们致以衷心感谢。

首先衷心感谢我的导师张宏莉教授。张老师严谨认真的治学态度，高屋建瓴的学术视角，平易近人的为人风格让我受益匪浅，从毕业论文的选题到研究工作中的困难，从实验室的项目到未来的工作，张老师都给我宝贵的建议和指导，让我收获了知识和能力，更得到了难得的锻炼。

感谢余翔湛老师几年来对我的耐心指导和帮助。余老师为人宽厚谦和，对我的每次请教都耐心细致地讲解，无论是学习中的问题还是工作中的问题，都给了我很多帮助。余老师渊博的知识、严谨的作风、科学的态度深深地感染和激励着我，让我受益良多。

感谢叶麟师兄、王星师兄，你们在实验室的工程项目上给予我很多指导，作为实验室的青年教师，你们勇于承担责任、勤勤恳恳兢兢业业的作风潜移默化地影响了我，怀念与你们共同奋斗的日日夜夜！

感谢实验室的张伟哲老师、何慧老师、翟健宏老师、张宇老师、张羽老师、张玥老师等，是这些老师们的专业素养营造了实验室良好的学术研究氛围。感谢实验室的秘书左冬梅老师、赵老师，你们处处为学生着想，为实验室营造了舒适的学习工作环境。

感谢牛林华、商兴奇、吴钧超、尹志敏师兄，感谢夏重达、王岳、徐洋、王松、李肖强、赵尚杰、谢虎成、孟媛媛以及实验室的其他同学，感谢李英俊、詹东阳、丛小亮、赵卫晨、姚崇崇、韩硕、何泽宇、淼爷、宋、叁姐、云超和其他师弟师妹们，是你们的陪伴让我度过了充满回忆的硕士生涯。

感谢主席、泽神、晓亮、大镕哥，认识你们是我人生的宝贵财富，也祝愿你们早日毕业、前程似锦！

最后感谢我的父母，是你们的辛勤付出让我在近二十载的求学生涯中能够全力以赴，今天的成果离不开你们的鼓励与支持！还要感谢我的姐姐，家庭永远是我最坚实的后盾！