

分类号 TP3

密级

UDC

编号

中国科学院研究生院

硕士学位论文

Web 信息提取技术研究与应用

张玲

指导教师 高文 教授

中国科学院计算技术研究所

申请学位级别 工学硕士 学科专业名称 计算机应用

论文提交日期 2003.4 论文答辩日期 2003.6

培养单位 中国科学院计算技术研究所

学位授予单位 中国科学院研究生院

答辩委员会主席

声 明

本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。就我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

签名：

日期：

关于论文使用授权的说明

中国科学院计算技术研究所有权保留送交论文的复印件，允许论文被查阅和借阅；并可以公布论文的全部或部分内容，可以采用影印、缩印或其它复制手段保存该论文。

签名：

导师签名：

日期：

摘要

Web 信息提取是指从 Web 文档中自动提取感兴趣信息的过程。它主要用在元搜索、信息代理等场合。

本文首先介绍了 Web 信息提取出现的背景和发展历史,详细阐述了 Web 信息提取的特征选择、提取知识表达、主要学习算法、评价标准,并介绍了 Web 信息提取的典型系统。

对于“列表式”信息条目的提取,本文提出一种基于 HTML 结构树的提取算法.利用 HTML 标记出现的规律,生成 HTML 结构树,并在结构树中寻找目标信息所在的最大扇出子树,然后再对此子树利用两个启发式策略,进行记录的划分。本算法提取“个人主页”中“论文列表”信息,达到了 82.2%的准确率。

对于密集型信息提取任务,本文提出一种基于隐马尔可夫模型的信息提取算法。该算法主要解决隐马尔可夫模型用于密集型信息提取的结构学习问题。利用文法推断中的 Inferring Transducers 算法提取文本的符号特征,并提出一种状态合并途径生成隐马尔可夫模型的拓扑结构,然后利用最大似然方法学习隐马尔可夫模型的概率分布,在识别时,采用修改的 viterbi 算法。本算法对“个人主页”中单条的论文信息进行进一步解析,达到了 80%以上的准确率。

对于稀疏型信息提取,本文实现了一种基于关系学习的信息提取算法。本算法利用三种文本特征:简单 token 特征、关系特征和 HTML 结构特征,利用一个自顶向下的关系学习算法对样本进行归纳,生成用于提取的一阶逻辑规则。本算法用于“个人主页”的人名和 Email 的提取,达到了较好的效果。实验结果表明,该算法对于稀疏型信息提取是一种有效的方法,只需要很少的样本数就可以学习出有效规则,并且学习结果易于理解。另外,本文分析了该算法存在的缺陷以及改进方向。

关键词: Web 信息提取 文法推断 隐马尔可夫模型 HTML 结构树 关系学习

Abstract

Web information extraction is the process of extracting interesting information from Web documents. This technology is mainly used in meta-searching and information agent.

This paper introduced the background of information extraction and its history, and introduced the feature-selecting, extraction knowledge expression, learning algorithms and evaluation standards, and introduced the typical systems of Web information extraction.

For list-like information, this paper put forward a algorithm based on html structure tree. The algorithm build the structure tree by analyzing the model of HTML tag, found the biggest fan-out sub tree within which the target information occurred, and applied two heuristic strategies to the sub tree for dividing the records. In the experiment of extracting “publications” information from personal homepage, the precision was above 82.2%.

For dense information extraction, this paper put forward a HMM-based algorithm. This algorithm resolved the structure learning of HMM for dense information extraction. It used the Inferring Transducers algorithm in grammar inference to learn the text features, put forward a approach of state-merging for building the HMM, learned the probabilities distribution through max likelihood approach and extracted information through a variation of viterbi algorithm. In the experiment of deep parsing a single publication information, the precision was above 80%.

For sparse information extraction, this paper realized an algorithm based on relational learning. The features it used involved token features, relational features and html structure features. It used a top-down relational learning algorithm to induce the samples and got the first-order logic rules. In the experiments of extracting person name and email from personal homepage, the results were inspiring. The experiment results suggested that this algorithm is a practical method for sparse information extraction, it needs only a few train samples, and the results are human-readable. This paper analyzed the shortcomings of this algorithm too, and put forward the directions of improvement.

Keywords: Web information extraction grammar inference hidden
markov model HTML structure tree relational learning

目录

摘要	I
Abstract	II
第一章 绪论	1
1.1 引言	1
1.2 本文的主要工作	1
1.3 本文主要内容	3
第二章 Web 信息提取技术综述	4
2.1 Web 信息提取技术的发展历史	4
2.2 Web 信息提取的途径	5
2.3 文本特征提取	5
2.3.1 符号特征	5
2.3.2 关系特征	5
2.3.3 文本片段特征	6
2.3.4 文档结构特征	6
2.4 提取知识表达	6
2.4.1 有限状态自动机	7
2.4.2 一阶逻辑规则	7
2.5 Web 信息提取的评价标准	8
2.6 Web 信息提取算法	8
2.6.1 基于文法推断的信息提取算法	8
2.6.2 基于 HMM 的信息提取算法	10
2.6.3 基于关系学习 (Relational Learning) 的信息提取	10
2.7 典型系统(算法)	12
2.7.1 WIEN	12
2.7.2 Cora 计算机科学研究论文搜索引擎	13
2.7.3 SRV	13
2.8 小结	14
第三章 基于页面结构分析的信息提取	15
3.1 HTML 结构树	16
3.1.1 HTML 的简单介绍	16
3.1.2 HTML 结构树的概念	17
3.1.3 由 HTML 文档构造 HTML 结构树	17
3.2 基于 HTML 结构树的信息提取	18
3.2.1 信息所在区域的确定	18
3.2.2 最大扇出子树	18
3.2.3 记录划分策略	19
3.2.4 提取算法	19
3.3 实验结果	20
3.4 小结	20
第四章 基于隐马尔可夫模型的信息提取	22
4.1 基于符号特征提取的 HMM 结构学习	22
4.1.1 HMM 简介	22

4.1.2 基于符号特征提取的 HMM 的状态类型确定	23
4.1.3 HMM 结构学习	25
4.2 HMM 概率学习	25
4.3 基于 viterbi 算法的识别	26
4.4 实验结果及分析	26
4.5 小结	29
第五章 基于关系学习的信息提取	30
5.1 术语说明	30
5.2 样本空间	31
5.3 规则表达	31
5.4 文本特征	32
5.4.1 简单 token 特征	32
5.4.2 关系特征	32
5.4.3 HTML 结构特征	33
5.5 规则学习算法	33
5.6 基于规则的识别	37
5.7 实验结果及分析	37
5.8 讨论与小结	40
第六章 总结	41
6.1 主要研究内容	41
6.2 下一步研究内容	42
参考文献	44
致谢	47
作者简介及发表的学术论文	48

第一章 绪论

1.1 引言

随着 Internet 的普及和发展,人们越来越依赖于 Web 来获取信息。我们可以随时从 Web 上查找所需要的信息,Web 作为一个庞大的资源库,给我们的学习和生活带来了巨大的便利。

Web 上信息正在爆炸式的增长,这给信息检索带来了困难,由于信息的无序性,我们只能采用“全文检索”来查找所需信息,但“全文检索”会带来大量的无关信息。人们迫切需要更精确的检索技术。

另一方面,人们不再满足于自己去寻找信息,信息代理(information agent)正在解决信息的主动推送问题。信息代理从大量的资源网站收集资源,然后根据用户的需求或兴趣过滤和转换信息,再将处理后的信息发送给用户。但是,由于各个网站资源的异构性,很难准确地从大量的网页资源中发现用户需要的信息。

上述问题的出现是由 Web 信息描述方式引起的。现有的 Web 网页大部分还是由超文本标记语言(Hypertext Markup Language)描述的。HTML 没有严格的语法限制,也没有清晰的语义,再加上人们为了漂亮的显示效果而采用的 javascript 等脚本,使得 HTML 网页代码冗长、混乱,几乎淹没了有用信息。所以,现在的 Web 还停留在“人可理解”的层次,距“机器可理解”还有很长的距离。

可扩展标记语言(eXtensible Markup Language)的出现从一定程度上解决了上述两个问题。XML 格式要求严格、显示与内容分开等特点改善了 Web 描述方式,再加上各个行业 XML 描述标准(如 MathML)的出现,在小范围内已经可以实现 Web 信息的自动处理。但是,我们必须看到,XML 不是万能的,一方面它需要一个统一的资源描述方案(XML DTD 或 XML Schema 等),对于 Web 上各种各样的内容,要想制定一个统一的资源描述方案是不可能的;另一方面,Web 上已有的信息也不可能在短时间内都转化为 XML 格式。

Web 信息提取技术研究正是在这种背景下兴起的。信息提取(information extraction)是一种浅层次(shallow)的文档处理,它从文档中自动提取信息,并将其装入数据库中[1]。Web 信息提取处理对象是 Web 网页。Web 信息提取很好地解决我们提到的两个问题。比如,我们可以利用信息提取技术来实现元搜索,通过从全文检索(或关键字检索)返回的结果页中提取关键信息,来提供更精确的检索效果。在信息代理中也是相似的。Web 信息提取技术已经成为元搜索、信息代理等服务的核心技术。

1.2 本文的主要工作

本文的资助项目是《国家科学数字图书馆智能化网络信息搜索技术与机制研究》。该项目的目标是通过 Web 信息提取、网页分类、超链分析等技术为用户提供一个自动扩展的学科资源库,并在此基础上,提供更精确的检索服务。Web 信息提取是一个底层模块,主要用在学科知识库的自动构造上。它的任务是从特定的学科网页中提取学科信息,再填入数据库。在此基础上,可以直接提供基于关键词的检索服务,也可以将此知识库用来辅助分类。

在学科网络资源中, 研究人员的个人主页是一个重要的信息来源, 上面经常包含了研究人员的基本信息、研究兴趣、发表的论文等, 同时还可能包括一些重要的学术站点链接. 对于很多研究人员, 经常需要浏览同行的个人主页获得相关的学术资料. 基于这个原因, 本文选择大学教员的个人主页作为处理对象, 实现了特定类型网页的信息提取系统, 提取两类信息:

基本信息: 人名、EMAIL 等;

论文信息: 分为两个步骤, 首先要提取单独的论文条目, 然后将每个条目解析成更小的语义单元, 以方便多种方式的检索和统计分析.

所有的提取结果保存在数据库中, 并对外提供检索服务.

系统的逻辑结构如图 1.1:

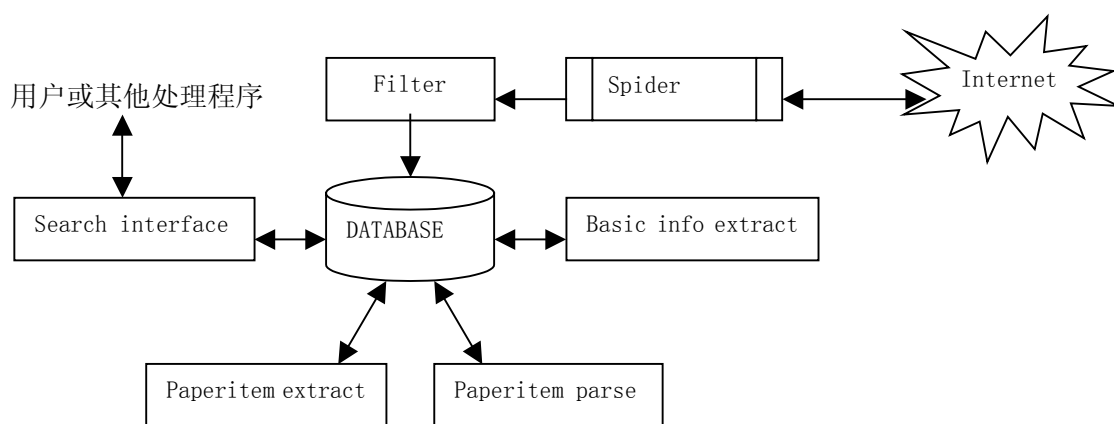


图 1.1

Spider: 网页抓取程序, 需要给定站点, 该程序将给定站点中的所有网页都抓取回来。本系统给定的站点是 89 所大学计算机系的主页;

Filter: 在抓取回来的网页中找出教员的个人主页, 网页的自动分类目前还是一个比较困难的问题, 所以本系统目前还是采用人工过滤的方式;

Basic info extract: 基本信息提取, 包括人名、EMAIL 地址;

Paperitem extract: 论文条目提取;

Paperitem parse: 将论文条目进一步解析成作者、标题、期刊名(会议名)、发表年等。

Search interface: 检索接口, 用户可以对本系统提取的信息进行各种检索。如按人名检索、论文标题检索、期刊名检索、会议名检索等, 检索接口返回数据库中所有相关信息。

在系统中, 个人主页中的信息提取划分成三个模块, 其划分的依据是三个模块处理的任务类型是不同的: Basic info extract 在网页中提取少量的信息, 可以称之为稀疏型信息提取; Paperitem extract 从网页中提取出论文信息, 论文列表在主页中是一个比较大的区域, 而且经常呈列表形式, 可以称之为列表式信息提取; Paperitem parse 将一个论文条目解析成多个域, 输入信息中含有较少的无关信息, 可以称之为密集型信息提取。三个模块处理的对象各有特点, 需要用不同的方法解决。

1.3 本文主要内容

本文主要讨论图 1.1 中的信息提取部分。本文利用了 HTML 结构分析、关系学习技术、隐马尔可夫模型等实现 Web 信息提取，主要内容包括：

- Web 信息提取技术综述. 从 Web 信息提取的发展历史、Web 信息提取的途径、Web 信息提取的文本特征、提取知识表达、主要学习算法、评价标准等方面对 Web 信息提取技术进行阐述。

- 基于 HTML 结构分析的 Web 信息提取. 从 HTML 结构出发，探讨 HTML 结构线索在信息提取中的应用，提出一种基于 HTML 结构树的信息提取算法，并对个人主页中的“论文列表”信息进行识别。

- 基于隐马尔可夫模型的信息提取。对密集型提取任务进行研究，提出一种基于 HMM 的信息提取算法，该算法利用文法推断的 Inferring Transducers[5]算法进行 token 抽象特征提取，生成 HMM 的状态结点，然后提出一种状态合并方法生成最终的 HMM，再利用最大似然方法学习 HMM 的概率分布，最后利用修改的 viterbi 算法进行识别。

- 基于关系学习技术的 Web 信息提取. 对稀疏型提取任务进行研究，实现了一种基于关系学习的信息提取算法。该算法沿用 SRV[5, 26]算法的主体框架和思路，对特征选择、样本生成和规则学习过程给出了具体的实现方法。然后利用该算法进行个人主页中人名、EMAIL 信息的提取。

第二章 Web 信息提取技术综述

信息提取是从自然语言文本中提取出特定信息的过程，传统的信息提取系统利用自然语言处理技术，使用基于语法或语义限制的提取模式，可以对自由文本进行处理，但是自然语言处理还没有达到理想的效果。

Web 文档是一种半结构化的文本。这种文本具有两个显著特点：(1) 文本中含有大量的标记和超链；(2) 文本中很少出现完整的句子。由于有这两个特点，自然语言处理技术不完全适用于 Web 信息提取。研究人员从 Web 文本特征、提取知识表达、学习算法等方面做了大量的研究工作。本章将从这几方面全面解析 Web 信息提取技术，另外，本章还将介绍 Web 信息提取的评价标准和典型系统等。

2.1 Web 信息提取技术的发展历史

有关“信息提取 (IE)”的研究起源于 20 世纪 90 年代初，主要是由 TIPSTER 的消息理解会议 (MUC) 发起的。IE 的前身是文本理解[2]，在 IE 出现之前，已经有大量的关于自然语言处理的研究和系统。但这些系统通常只能处理一个很狭窄领域的文本，而且很难移植到新的领域[3]。

TIPSTER Text Program 是一个美国国防部领导的行动，它开始于 1991 年，其目的是提高文本处理的技术发展水平。TIPSTER 研究共分为 3 个阶段，在第 1 阶段，TIPSTER 通过消息理解会议，在信息提取算法方面取得了很大进展，在自动识别命名实体（如人名、组织名等信息）方面取得了巨大进步。在第 2 个阶段，TIPSTER 主要研究软件体系结构，使得不同的 TIPSTER 成员之间可以共享软件。第 3 个阶段，TIPSTER 增加了几个新的领域，如自动文本摘要等。由于缺乏资金，这项研究计划于 1998 年正式结束。

随着 Web 的出现和繁荣，IE 研究人员逐渐将兴趣转移到 Web 信息提取的研究上，涌现了许多算法和系统。其中最知名的研究项目是卡耐基-梅隆大学“自动学习和发现中心 (Center for Automated Learning and Discovery)”的“Web 挖掘 (Mining the World Wide Web)”项目。该项目的目标是通过自动的从 Web 中提取事实，来创建大型的、结构化的有用事实的数据库。他们的技术途径是研究机器学习算法，通过训练，能够自动提出信息。用户首先定义要被提取的类（比如公司、产品、雇员）和关系（比如“被雇佣”），并通过 Web 提供训练样本，系统然后使用这些训练数据学习通用的信息提取步骤，然后按照这个步骤从其他 Web 页面中提取信息。他们已经开发了许多学习算法，包括：(1) first-order 规则学习算法；(2) 文法推断算法 (Grammar Inference)。他们已经证明，这些方法能够提取关于大学教员、学生、课程和研究项目的信息，达到大约 70% 的精确度和 30% 的查全率。

最近几年，研究人员借鉴其他领域成功的模型或方法，推动了 Web 信息提取技术的进展。隐马尔可夫模型、归纳逻辑编程 (ILP) 等在信息提取中得到初步应用，取得了较大的成功。

2.2 Web 信息提取的途径

用于 Web 信息提取的软件系统也称为包装器(wrapper), wrapper 是一种软件构件, 负责将隐含在 HTML 文档中的信息提取出来, 并且转换成能够被进一步处理的以某种数据结构存储的数据[7]。

Wrapper 的构造有两种途径: 一种是基于知识工程的途径, 即由领域专家构造提取规则, 这种系统完全依赖于领域专家的技巧。由于 Web 上网页形式的多样化和不断更新, 使得手工构造的途径是不现实的。

现在一般采用机器学习途径自动构造 wrapper. 只需要提供标注好的样本, 机器学习算法自动学习提取知识, 并以适当的模型存储, 当遇到新的网页时, 提取模型与网页进行匹配, 如果匹配, 则提取出相应信息片段。

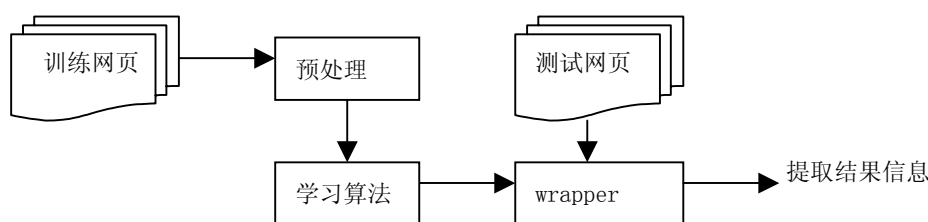


图 2.1 基于机器学习的 wrapper 构造和识别过程

基于机器学习的 Web 信息提取过程见图 2.1。预处理步骤主要完成文本特征提取。学习算法在文本特征基础上学习提取模型, 并保存在 wrapper 中。识别时, wrapper 比较文本是否满足提取模型, 如果满足, 则获得目标信息。

2.3 文本特征提取

由于 Web 文本的特点, Web 信息提取算法的特征选择一般不采用自然语言处理中的语法和语义特征, 而使用下列特征:

2.3.1 符号特征

HTML 文档中的符号包括标记符号和文本中的符号。对于英文文本, 文本的符号化很简单, 可以用非数字字母符号将文本字符序列隔开成字符串, 这些隔开的字符串称之为 token. 对于标记符号 token, 特征有标记名、属性等。对于文本符号 token, 特征有类型(数字、单词等)、值(文本实际内容)等。

例如, 文本片段**Pentium 90**可以转化为:
`[token(type=html, tag=b), token(type=word, txt= "Pentium"),
 token(type=int, val=90), token(type=html_end, tag=b)]`.

2.3.2 关系特征

除符号特征外, 符号之间的关系也可能是信息提取的重要线索。关系特征考虑符号之间的关系。如在图 2.2 所示的个人主页中, 如果要提取电话号码, 那么

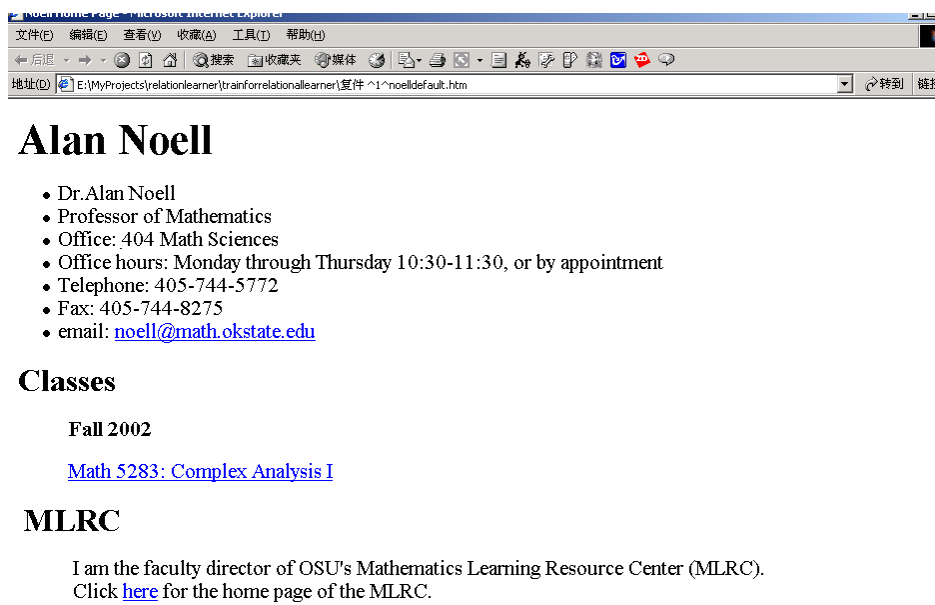


图 2.2 网页示例 1

“405”和“Telephone:”之间的前后关系是一个很好的线索:如果一个 token 的前面有 token 串“Telephone:”,那么该 token 很可能是电话号码的起始 token.

最简单的关系特征是 token 的直接前后继关系,如:

$$\text{next_token}(\text{token1})=\text{token2}$$

表示 token1 后面紧挨着 token2.

更复杂的关系特征还可以考虑 token 之间的距离,如 $\text{distance}(\text{token1}, \text{token2}, 3)$ 表示 token1 与 token2 之间相距 3 个 token.

2.3.3 文本片段特征

单纯从 token 层次来描述文本特征是不够的,要提取的信息一般是由很多个 token 组成的文本片段,因此从文本片段层次上来描述特征是必要的。文本片段特征如

$\text{length}(=, 5)$, 表示文本片段的长度为 5;

$\text{has_token}(@)$, 表示文本片段中包含字符 '@'.

2.3.4 文档结构特征

由于 HTML 文档中含有大量的标记信息,而标记之间是层层嵌套的。大多数的文本都处于标记之内,有些类型网页的标记出现有一定规律可循,另外有少数标记具有语义线索,如 $\langle \text{title} \rangle$, $\langle \text{head} \rangle$, $\langle \text{h1} \rangle$ 标记中的内容一般对整个文档的语义起概括作用。可以利用的文档结构特征有符号所在的标记、父标记等。

2.4 提取知识表达

提取知识是指由样本学习获得的对目标信息提取的知识。在现有的 Web 信息提取系统中,提取知识一般有两种表示方式:(1)有限状态自动机(或文法),包括随机有限

状态自动机；(2) 以一阶逻辑表达的提取规则。

2.4.1 有限状态自动机

以有限状态自动机表示的提取知识是待识别的文本片段的文法模型。如果文本片段能使有限状态自动机达到终态，则该文本片段是目标信息。有限状态自动机的输入符号是特征化后的文本符号序列。

隐马尔可夫模型 (Hidden Markov Model) 是最近几年应用最广泛的提取知识表达模型。它是一种随机的有限状态自动机。由于 HMM 有成熟的学习算法和坚实的统计基础，所以在信息提取中是一种成功的模型。图 2.3 是一个用于引文信息提取的简单 HMM：

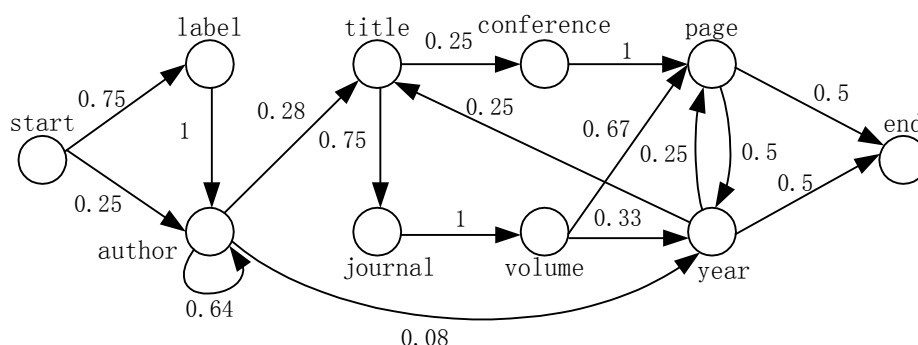


图 2.3: 一个用于引文信息提取的简单 HMM

与有限状态自动机相对应的是各种类型的形式文法。也可以用各种形式文法（如上下文无关文法、正则文法等）来表示提取知识。

2.4.2 一阶逻辑规则

一阶逻辑语言比起命题逻辑语言（零阶逻辑），表达能力大增，它引入了量词、变元，能处理命题逻辑不能处理的推理。在命题逻辑中，简单命题的逻辑形式是不可分析的，这样，命题逻辑不能包括所有的逻辑规律。如下面的推理：

前提：所有自然数有大于它的素数；

前提： 2^{100} 是自然数

结论： 2^{100} 有大于它的素数。

这个推理是由两个不同的简单命题推出另一个不同的简单命题，这在命题逻辑中是不成立的。但这个推理却是正确的，它可以由一阶逻辑推理实现。详见[4]。

利用一阶逻辑，可以很方便的表达基于 Web 文档特征的提取知识。如在 SRV 算法中，用了 5 种形式的规则，其中，有一条是：

$\text{some}(\text{Var}, \text{Path}, \text{Feat}, \text{Value})$

Var 是一个变元，Path 是关系特征，Feat 是符号特征，Value 是 Feat 的一个合法值。

比如：

$\text{some}(\text{?A}, [], \text{all_lower_case}, \text{true})$

用一阶逻辑表达式可以表示为：

$\exists A \in F \wedge \text{all_lower_case}(A) = \text{true}$

其含义是文本片段包含某个 token A, A 中只包含小写字母字符。

2.5 Web 信息提取的评价标准

对信息提取效果，一般采用下面三个标准[2]：

$$\text{正确率: } Precision = \frac{\#correct\ answers}{\#answers\ produced}$$

$$\text{查全率: } Recall = \frac{\#correct\ answers}{\#total\ possible\ corrects}$$

$$\text{综合评价指标: } F = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

#correct answers 是指提取结果中正确的信息条数，#answers produced 是指提取得到的信息条数，#total possible corrects 是指测试样本中的信息条数。 F 是一个综合评价指标， β 越大， $Recall$ 对 F 的影响越大，一般取 $\beta=1$ ，这时 P 和 R 有同等的权重。

2.6 Web 信息提取算法

对应提取知识表达方式，Web 信息提取算法大致可以分为三类：（1）基于文法推断的提取算法；（2）基于隐马尔可夫模型的信息提取算法；（3）基于关系学习的信息提取算法。

2.6.1 基于文法推断的信息提取算法

文法推断[22]是一类归纳推理算法，它的目标是给定一个有限的样本集，能为未知的目标语言推断出一个一致的文法。当待学习语言的最终定义以不同的形式表示时，其归纳学习过程也有所不同。严格地说，文法推断是指待学习语言的定义用文法表示时的归纳学习过程[6]。文法推断算法基于“要提取的信息具有某种文法形式”的前提。比如，外国人的名字有“Freitag D.”等形式。在这个前提下，可以根据提取域的文法形式来发现目标信息。由于人工去描述提取域的文法形式可能不完整，所以，必须要采取机器学习的办法。

文法推断的经典方法由 Gold 于 1969 年提出。他介绍了语言的极限识认模型[6]：在任何时刻 t ，向推断设备输入信息单元 i_t ，推断设备输出一个猜测（即推断结果）。如果在有限时间之后，推断设备输出的猜测不因提供更多的信息单元而改变，且该猜测是关于待学习语言的正确描述，则可以说推断设备使用的是一个成功的推断算法，这个过程也称“语言的极限识认”。

Gold 同时证明，语言的极限识认方法如果在只提供正例的条件下，连正规语言都不能识认。而在很多情况下，反例信息很难提供。Angluin 提出一种交互式学习模型，通过提问回答生成文法。最近几年，PAC (Probably approximately correct learning) 模型得到广泛应用。

文法推断包括两个步骤：猜测生成和猜测选择。所谓猜测是指结果文法。猜测方法又分为枚举法和构造法。枚举法对所考虑的文法类中的文法进行枚举。构造法对样本进行系统化的分析，从样本实例中所包含的结构信息，构造出语法规则，通过枚举法或构造法生成猜测空间后，根据一定的标准对猜测空间进行搜索。搜索标准有简单性、通用性等。

[10-13]利用了文法推断来实现信息提取。这里用[11]中的 wrapper 构造过程说明文法推断在信息提取中的应用和实现。在[11]中，文法的描述对象是整个网页。文法的构造过程是：

- HTML 文本转换为抽象字符串后得到，将标记内内容转换为标记名，标记之间的文字由抽象字符串” text” 代表。如：

```
<hr><a href=mailto:sales@a.com>A-1 Realtors</a>
<hr><a href=mailto:help@b.com>Bee Estate Agents</a>
都转换成抽象字符串 hr a text /a.
```

- 样本的抽象字符串对应一个初始的文法，如：

```
S->html head ... table tr td b text /b br text ... /html|html head ... table
tr td img b text /b br.../html|...
```

- 在初始文法的基础上，通过对产生式的各种操作，进行文法推断。这些操作包括：

替代：由产生式 $X_1 \rightarrow asb, x_2 \rightarrow csd$, 得到一条新的产生式 $Y \rightarrow s$

分裂：由产生式 $X_1 \rightarrow asb, X_2 \rightarrow atb$ 得到一条新的产生式 $Y \rightarrow s|t$

...

在候选的产生式中选择下一个产生式是由一个文法复杂度函数来控制，对于一个形如：

$$X_1 \rightarrow w_{11} | w_{12} | \dots | w_{1,m1} \quad [P_{11}, P_{12}, \dots, P_{1,m1}]$$

$$X_2 \rightarrow w_{21} | w_{22} | \dots | w_{2,m2} \quad [P_{21}, P_{22}, \dots, P_{2,m2}]$$

.

.

.

$$X_n \rightarrow w_{n1} | w_{n2} | \dots | w_{n,mn} \quad [P_{n1}, P_{n2}, \dots, P_{n,mn}]$$

的文法 $(P_{11}, \dots, P_{n,mn})$ 代表在产生式中各项对应的概率)

其复杂度定义为：

$$C(G) = \sum_{i=1}^n \sum_{j=1}^{mn} -\log P_{ij} + c(w_{ij}),$$

其中， $c(w) = (K+1)\log(K+1) - \sum_{i=1}^r k_i \log k_i$

K 是 w 的长度， w 有 r 个不同的符号，分别出现 K_1, k_2, \dots, k_r 次。

- 在前面三个步骤之后，得到最终的文法，该文法中某个特定的非终结符对应要提取的文本，[11]最终得到的文法为：

```
S->html head title text /title /head body h1 text /h1 table T /table p address
text /address /body /html
```

```
T->TT|tr td U b text /b br text br text /td /tr
```

$U \rightarrow e | \text{img} | \text{br}$

其中, T 对应于待提取的文本。

到目前为止, 文法推断的研究取得了很大的进展, 但在信息提取中的应用还刚刚起步, 其优点是直观、完备, 缺点是要对整个文本建立文法, 而对于信息提取, 并不是所有文本都需要考虑, 只有部分文本是值得考察的线索。

2.6.2 基于 HMM 的信息提取算法

HMM 是一种随机的有限状态自动机, 已成功应用于语音识别、手写体识别等领域。HMM 最近几年被引入到信息提取中, 已经得到了成功的应用。HMM 有非常适合自然语言处理的统计基础, 对于新数据的处理有良好的鲁棒性, 并且有成熟的学习算法[20], 对信息提取是一个成功的模型。

对于 HMM, 有三个基本问题[23]:

- 识别问题: 给定一个输出序列和模型, 模型可能创建的序列概率是什么?
- 序列问题: 给定一个输出序列和模型, 什么最可能的状态序列可以创建输出序列?
- 训练问题: 给定一个输出序列和拓扑结构, 怎样调整模型参数, 包括状态转移和输出的概率分布, 使模型创建的输出序列具有最大概率?

在信息提取中, 涉及后两个问题。“序列问题”即是由模型对样本进行识别的过程, 而“训练问题”正是各种基于 HMM 的信息提取所要解决的问题。最初的 HMM 构造是通过观察样本手工构造。或者一个状态对应一个域[21], 或用几个状态对应一个域[17, 19], 然后通过调整概率来获得较好的效果, 如在[17]中, 提出一种 shrinkage 概率估算方法。现在的 HMM 构造一般是采用自动构造[15, 16, 18, 20]。[18]提出一种随机优化方法, 通过不断搜索可能的 HMM 结构, 使之达到最好的提取效果, [15]是对它的一个改进, [16]通过对句子成分的分析(如名词、名词短语等)构造 HMM,。这三种方法都应用于“稀疏型”提取任务, 即待提取文本中含有大量的无关信息。[18]对“讲座通知”中的地点、演讲人、演讲标题等信息进行提取, [15]对旅馆信息的旅馆名、电话号码等信息进行提取。

基于 HMM 的信息提取算法的优点是充分利用统计学的理论, 对于新数据的适应性强, 并且训练耗时短。其缺点是需要大量的训练数据, 并且还缺乏通用的模型构造方法。

2.6.3 基于关系学习 (Relational Learning) 的信息提取

关系学习是从结构化样本中学习结构化概念定义的问题, 关系表达使用一阶模型来描述问题领域(规则)和以对象和对象关系形式给出的样本[30]。

关系学习的实现途径是归纳逻辑程序设计 (Inductive Logic Programming, 简称 ILP), ILP 被定义成归纳学习 (Inductive Learning) 和逻辑程序设计 (Logic Programming) 的交集[42], 因此它能够同时利用机器学习和逻辑程序设计的技术。ILP 继承了归纳机器学习的目标: 即从样本中归纳出假设。同时它利用计算逻辑 (computational logic) 作为假设和样本的表述机制, 能够克服传统的机器学习的两个固有缺陷: 有限的知识表述机制, 通常是命题形式; 难以利用背景知识。这使得传统机器学习不能解决很多领域的学习问题, 在这些领域, 专家知识只能表述成一阶逻辑的形式, 另外, 大量的背景知识需要加入到归纳过程中。

ILP 继承了计算逻辑的表述机制和相关技术，与其他归纳学习不同的是，ILP 着重于推理规则的属性、算法的收敛性和计算的复杂性。它与计算逻辑的区别在于它延伸了计算逻辑的理论和实践，它研究基于归纳而不是基于演绎的推理。

按规则的构造过程分，关系学习算法可分为两类：

自顶向下：算法由最一般的常识性规则开始，通过引入反例，修改规则，使规则逐步精确。

自底向上：算法首先选择一个初始的训练集，然后产生能够覆盖这些训练集的规则，然后逐步对这个规则进行修改，使之能够覆盖其他的训练实例。

已经出现了许多关系学习的算法，其中 FOIL[9] 和 GOLEM[24] 是最知名的算法。FOIL 是许多信息提取算法的基石。

FOIL 是一个典型的自顶向下的 ILP 算法，它的输入包括标注好的正例和反例集、候选子句集，通过一个收益函数控制子句的逐步添加，直到样本中所有正例被覆盖，而没有反例被覆盖。

FOIL 基本覆盖算法(摘自[14])

Initialization

Definition=null; //Definition 是归纳结果

Remaining=all positive example; //Remaining 是正例集

While Remaining is not empty

//找到一个子句，能够覆盖 Remaining 中的样本，但不覆盖反例

Find a clause, C, that covers some example in Remaining,

But no negative examples.

Remove examples covered by C from Remaining.

Add C to Definition.

End while

FOIL 的寻找一个子句的过程是通过一个从一般到特殊的爬山搜索实现，不断添加前件(antecedents)到当前正在形成的子句。在每一步，算法衡量每个可能添加的文字(literal)，并且选择使信息增益最大的 literal。FOIL 的寻找一个子句的算法如下(摘自[14])：

Initialize C to R(V₁, V₂, ..., V_k) //R 是 k 元目标谓词

Initialize T to contain the positive tuples in positive-to-cover and all the negative tuples. //T 为包含当前还未覆盖的正例(positive-to-cover)和所有反例的样本集

While T contains negative tuples

Find the best literal L to add to the clause //找到最好的文字(literal)添加到子句中

Form a new training set T' containing for each tuple t in T that satisfies L, all tuples of the form t.b(t and b concatenated) where b is a set of bings for the new variables introduced by L such that the literal is satisfied(i.e, matches a tuple in the extensional definition of it's predicate) //形成一个新的训练集，它包含满足当前子句的的样本

Replace T by T' .

下面举例说明 FOIL 怎样用于信息提取规则学习。假设有一个文档 D 已被转换为特征序列 word(Pos, Token), Pos 是 token 所在的位置. 假设我们已经有了一个文字集合用来测试 token 特征(包括简单特征和关系特征), 如 fragment(length, 7) 该文字检查片段的长度是否为 7。然后，通过 FOIL 的归纳过程，得到文字的运算，形成子句(规则)。如，我们要提取 Email 信息。第一步，通过 FOIL 的收益函数，添加第 1 个文字

fragment(length, 7), 还剩余有正例未被覆盖, 于是接着添加文字, 得到第二个文字 has_token(@), 依次类推。最后, 得到一条规则如下:

fragment(length, 7) ^ has_token(“@”) ^ last_token(“cn”)

其含义是片段的长度为 7, 且含有字符 ‘@’, 且最后一个 token 是字符串 “cn”。

由这个例子可以看出基于关系学习的信息提取的关键是样本的特征的选取和规则的表述。

关系学习非常适用于信息提取问题[5], 已经有一些系统[25-30]利用关系学习来产生信息提取规则。在 RAPIER[25]系统中, 使用三种类型的规则表达, 分别考虑提取目标的前缀、后缀和目标本身, 文本特征有词、词性和语义标签。SRV[5, 26]算法用五种类型的规则表达, 它考虑的特征是单个 token 的特征。[27]实现了一种基于 rlgs (relative least-general generalizations) 构建的自底向上算法, 通过计算随机选择的正例的 rlgs 来创建规则。[28]同时利用了自底向上和自顶向下技术。[29]通过为一个随机的种子样本构造更特别的子句和 A*搜索来寻找最简单的一致概括。

基于关系学习的信息提取算法最大的优点在于它对样本和规则的表述模式, 能够学习复杂的理论, 充分考虑上下文关系, 而且比起文法推断算法对整个文档(或文本片段)建模, 关系学习只考虑对识别有用的 token. 这样学习得到的规则可读性好, 在此基础上可进行人工的纠正。

2.7 典型系统(算法)

2.7.1 WIEN

WIEN[31, 32]是一个构造信息提取系统的工具。主要处理具有“HLRT”结构的网页。在这类网页中, 有一个头部的定界符, 对提取的每个条目有一个左定界符和右定界符。WIEN 为每个提取域寻找统一的定界符, 其途径是通过归纳学习。归纳学习算法的输入是标注好的样本, 学习算法搜索所有可能的定界符, 直到找到的定界符与样本一致。另外, 它用一个基于计算学习理论的模型预测需要输入多少样本来保证学习结果的可用性。

WIEN 还研究了样本的自动标注。通过输入特定领域的启发式规则自动标注样本。

WIEN 能处理自动产生的网页(如通过查询数据库产生的网页)。这类网页具有固定的结构。比如图 2.4 的网页。



图 2.4: 网页示例 1, 摘自[1]

其中，(a)是浏览器中的显示结果，(b)是HTML代码，(c)是要提取的内容。

2.7.2 Cora 计算机科学研究论文搜索引擎

Cora[20, 33]搜索引擎中，运用了HMM来提取论文的头部的信息，如作者、标题、email、关键字等。HMM的构造方法是从最特殊的结构开始，通过状态合并“neighbor-merging”、“V-merging”逐步生成通用的拓扑结构。

2.7.3 SRV

SRV是一个基于FOIL的规则学习算法。它应用的特征有简单token特征、关系特征，它有5种文字形式，分别描述文本片段的长度、文本片段中某一个token的特征、文本片段所有token的特征，文本片段某一个token的位置、文本片段某两个token的相对位置。

SRV的候选文字是在扫描样本过程中产生的，向规则中添加文字的标准是FOIL的收益函数：

$$I(S) = -\log_2(P(S)/(P(S)+N(S)))$$

$P(S)$ 和 $N(S)$ 分别是样本集合 S 中的正例和反例数目。

$GAIN(S) = P(S_A) (I(S) - I(S_A))$ 。 S_A 是添加 A 到规则中后规则覆盖的样本集合。

下面是SRV算法用于“讲座通知”中讲座开始时间信息提取产生的一条规则，摘自[5]：

```
some(?A, [], single_digit_p, true),
position(?A, fromfirst, <1),
some(?B, [prev_token], word, “:”);
length(<, 5),
some(?A, [prev_token, prev_token], quadrupletonp, true),
some(?C, [], doubletonp, true),
some(?A, [prev_token, prev_token, prev_token], all_lower_case, false)
等价于一阶逻辑表达式：
```

$$\begin{aligned} &stime(F) \leftarrow length(F) < 5 \\ &\wedge (\exists A, B, C \in F. A \neq B \neq C \\ &\wedge single_digit_p(A) = true \\ &\wedge quadrupletonp(prev_token(A)) = true \\ &\wedge all_lower_case(prev_token(prev_token(prev_token(A)))) = false \\ &\wedge dist_from_first(A, F) < 1 \\ &\wedge word(prev_token(B)) = ':' \\ &\wedge doubletonp(C) = true \end{aligned}$$

用自然语言来描述这条规则是：如果一个文本片段满足下面条件，那么这个文本片段是讲座开始时间：

- (1) 该文本片段的长度小于5；
- (2) 该文本片段中存在三个token A、B、C，分别满足：

- A 是一个单一的数字字符 token;
- A 是该文本片段的第 1 个 token;
- B 前面的 token 是” :” ;
- A 前面距离为 1 的 token 有 4 个字符长;
- C 的长度为 2;
- A 前面距离为 3 的 token 不完全由小写字母构成;

2.8 小结

本章从 Web 信息提取的途径、文本特征提取、提取知识表达、学习算法等方面剖析了 Web 信息提取技术，并介绍了 Web 信息提取的典型系统。

Web 信息提取有两种途径，一种是知识工程途径，一种是机器学习途径，实用的途径是机器学习途径。

Web 信息提取考虑的特征有符号特征、关系特征、文本片段特征、Web 结构特征。

Web 信息提取知识表达模型分为有限状态自动机和一阶逻辑规则。与此相对应，Web 信息提取的学习算法分为基于文法推断的算法、基于 HMM 的算法、基于关系学习的算法。

与很多其他应用领域一样，Web 信息提取的评价也采用准确率、查全率、综合评价指标三个标准。

第三章 基于页面结构分析的信息提取

通过观察可以发现，90%以上的个人主页中的“论文列表”都具有相似的显示布局，一般布局为：论文列表由若干个段落组成；每个段落由段落标题和多个格式相似的论文条目构成；多个段落之间显示格式是相似的。如图 3.1 所示是一个典型的“论文列表”的显示布局。该论文列表只包括一个段落“Selected Publication”，该段落由 5 条论文纪录组成。

Selected Publications

- **The Task of the Referee** *IEEE Computer*, Vol. 23, No. 4, April 1990, pp. 65-73.
- **Machine Characterization Based on an Abstract High Level Language Machine** (with R. Saavedra-Barrera and E. Miya), *IEEE Trans. Computers*, Special Issue on Performance Evaluation, Vol. 38, No. 12, December 1989, pp. 1659-1679.
- **The Fairchild CLIPPER: Instruction Set Architecture and Processor Implementation** (with W. Hollingsworth and H. Sachs), *Communications of the ACM*, Vol. 32, No. 2, February 1989, pp. 200-219.
- **Line (Block) Size Selection in CPU Cache Memories** *IEEE Trans. Computers*, Vol. C-36, No. 9, September 1987, pp. 1063-1075.
- **Cache Memories** *Computing Surveys*, Vol. 14, No. 3, September 1982, pp. 473-530.

图 3.1

对 HTML 文件，文档结构和显示效果之间有直接的关系，如 HTML 中有列表、表格等元素，其显示效果直接表示为列表和表格。图 3.2 是图 3.1 的 HTML 片段(黑体字部分是实际的显示内容)：

```
<H2>Selected Publications</H2>
<UL>
  <LI>
    <P><STRONG>The Task of the Referee</STRONG> <I>IEEE Computer</I>, Vol. 23, No.
    4, April 1990, pp. 65-73. </P>
  </LI>
  <LI>
    <P><STRONG>Machine Characterization Based on an Abstract High Level Language
    Machine</STRONG> (with R. Saavedra-Barrera and E. Miya), <I>IEEE Trans.
    Computers</I>, <I>Special Issue on Performance Evaluation</I>, Vol. 38, No. 12,
    December 1989, pp. 1659-1679. </P>
  </LI>
  <LI>
    <P><STRONG>The Fairchild CLIPPER: Instruction Set Architecture and Processor
    Implementation</STRONG> (with W. Hollingsworth and H. Sachs),
```

```

<I>Communications of the ACM, </I>Vol. 32, No. 2, February 1989, pp. 200-219.
</P>
</LI>
<LI>
<P><STRONG>Line (Block) Size Selection in CPU Cache Memories</STRONG> <I>IEEE
Trans. Computers</I>, Vol. C-36, No. 9, September 1987, pp. 1063-1075. </P>
</LI>
<LI>
<P><STRONG>Cache Memories</STRONG> <I>Computing Surveys</I>, Vol. 14, No. 3,
September 1982, pp. 473-530. </P></LI></UL>

```

图 3.2

在图 3.2 中，图 3.1 的两个段落标题位于标记<H2>中，论文纪录位于多个标记中。

HTML 显示内容和文档结构的这种对应启发了 web 信息提取中对 HTML 文档结构的研究。[31, 32]主要处理具有“HLRT”结构的网页，在这类网页中，有一个头部的定界符，对提取的每个条目有一个左定界符和右定界符，通过归纳学习为每个提取域寻找统一的定界符；[34]提出了一种 HTML 结构树的概念，并将 HTML 结构树用在其他的提取算法中，试验结果表明改进了提取效果；[7]利用 HTML DOM(文档对象模型)和归纳学习生成提取规则；[35, 36]主要探讨了在 HTML 标记树中利用各种启发式规则进行记录的划分。

本章要解决的问题是从个人主页中提取出单条的论文纪录，由图 3.1 可以看出，一个最直接的解决思路是：先找到“论文列表”所在段落，然后在该段落所属的范围划分出单条的论文纪录。遗憾的是，HTML 中并没有固定的段落标记和记录的标记。[31, 32]的方法直接寻找纪录的左右定界符，比较适用于由数据库检索产生的网页，而本章要处理的个人主页是由单个的主页所有者创建的，所以创作工具各异，最终的 HTML 代码也是各不相同的；[7, 35, 36]用到了 HTML DOM 树和标记树，但是 HTML DOM 树和标记树只能反映 HTML 标记间的嵌套关系，而 HTML 标记的嵌套和显示的段落层次不是一一对应的；[34]提出的 HTML 结构树考虑了显示的段落层次，但并不完整。

本章在现有方法的基础上，提出一种基于 HTML 结构树的信息提取算法，探索 HTML 结构线索在 Web 信息提取中的应用。

本章首先介绍了 HTML 结构树的概念，接着描述了从 HTML 代码到 HTML 树的转换，然后详细分析了如何根据 HTML 结构树提取信息。

最后，根据本章提出的方法，实验了对个人主页中的“论文列表”的提取，取得了较好的效果。

3.1 HTML 结构树

3.1.1 HTML 的简单介绍

HTML 的全称是超文本标记语言 (Hypertext Markup Language), 它是目前 Web 上通用的描述语言。它用描述性的标记符 (称为标记) 来指明文档的不同内容。标记是区分文本各个组成部分的分界符, 用来把 HTML 文档划分成不同的逻辑结构, 如段落、标题和表格等。HTML 标记一般有两个作用, 首先是定义文档结构, 以便浏览器可以显示该文

档。其次是提供各种路标,把 Web 客户搜索器程序(如 spider)引导到该文档的关键区域。

HTML 的标记间具有一定的嵌套结构,通过 tidy 等工具,可以将其转换成 XML 格式的文件,然后生成一颗 DOM 树,在此基础上,可以利用 DOM 接口完成多种操作。但另一方面,HTML 又不是完全结构化的,它的许多元素只具有显示效果(如<I>,,等标记),另外,HTML 文件中经常夹杂着大段的脚本(javascript,vbscript 等)代码,这些代码完全是为了显示效果而增加的。

3.1.2 HTML 结构树的概念

Dan DiPasquo[34]提出了 HTML 结构树的概念。

定义: HTML 结构树[34]

HTML 结构树是一个 n 叉树,它的每个结点对应 HTML 页面中的一个 HTML 标记,树中的父子关系意味着孩子结点的内容包含于父结点的范围之内。

[34]对于结点父子关系提出了两条原则:

- (1) 如果结点 b 嵌套在结点 a 之内,那么 b 是 a 的孩子结点;
- (2) 否则,如果 a 是 b 最近的逻辑标题(如 $h1-h6, title$),那么 b 是 a 的孩子结点。

如图 3.2 对应的 HTML 结构树如图 3.3 所示:

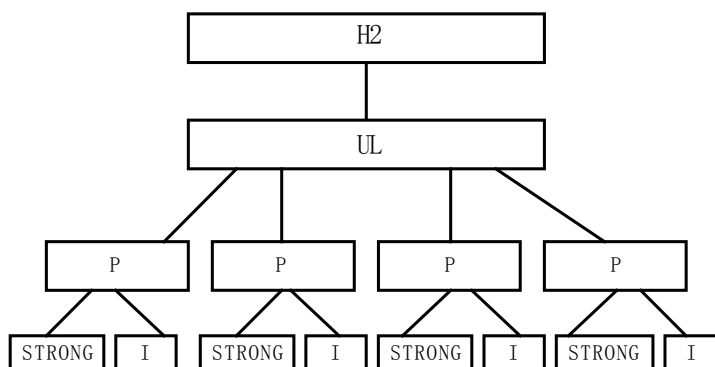


图 3.3 HTML 结构树

在图 3.3 中, $H2$ 和 UL 之间的父子关系是由原则(2)得到的,树中的其他父子关系是由原则(1)得到的。

3.1.3 由 HTML 文档构造 HTML 结构树

HTML 结构树的构造可以直接利用 HTML 标记之间的嵌套关系。另外,[34]中还提出了 HTML 标记的分级的概念,将 HTML 标记分为 5 个级别,级别的次序由高到低为 root、title、 $h1-h6$ 、other、dd,如果 $tag2$ 在 $tag1$ 的后面,并且 $tag1$ 的级别比 $tag2$ 高,那么 $tag1$ 是 $tag2$ 的父结点(如图 3.3 中 $h2$ 并没有直接包含 UL ,但它的级别比 UL 高,故它是 UL 的父结点)。

但另一方面,HTML 标记间的嵌套结构和[34]的 5 个级别并不足以描述 HTML 文档的结构。如

Preprints (dvi)

- The binary code generated by quadrics in a finite projective space (with N. S. N. Sastry), J. Comb. Theory A 94, 1-14 (2001) [Online at IDEAL](#)
- The permutation module of a symplectic vector space over a field of prime order, J. Algebra, 241, 578-591 (2001) [dvi file](#)

图 3.4 网页示例 3

对应的源代码为

```
<b>Preprints (dvi)</b>
<ul type="disc">
<li>The binary code generated by quadrics in a finite projective
space (with N. S. N. Sastry), J. Comb. Theory A 94, 1-14 (2001)
<a
href="http://www.idealibrary.com/links/doi/10.1006/jcta.2000.3118">
Online at IDEAL</a></li>
<li>The permutation module of a symplectic vector space over a
field of prime order, J. Algebra, 241, 578-591 (2001) <a
href="http://www.math.ufl.edu/~sin/preprints/symp.dvi">dvi
file</a></li>
```

图 3.5 网页示例 3 对应的 HTML 代码

其中标记中的内容起到了标题的作用，但按[34]的方法，并不能将作为的孩子结点。所以，本文为构造 HTML 结构树增加了一条原则：

如果 tag2 在 tag1 的后面，tag1 属于加重显示标记(用大的字体或加粗、居中，具体考虑了, <big>, <center>三个元素)，并且 tag1 与 tag2 之间没有别的加重显示标记，那么 tag1 是 tag2 的父结点。

还需要说明的是，HTML 文档并不是格式良好的，即它并没有完全遵循层层嵌套的格式，可能会出现开始标记和结束标记并不对应的情形，这种格式在浏览器能够得到纠正，但要构造 HTML 结构树，必须先对 HTML 标记进行清理。本系统采用 sun 公司的 jtidy 组件工具完成这一步骤。

3.2 基于 HTML 结构树的信息提取

3.2.1 信息所在区域的确定

在个人主页中，“论文列表”一般有三种出现方式：第一种是直接主页中出现完整的论文列表；第二种是论文列表单独出现在一个页面中，在主页中有超链链向它；第三种情况是主页中包含一部分，同时存在一个单独的论文列表页面。所以，算法在处理时，首先需要检查主页的所有超链标记，如果存在“Publication”或“Paper”等超链，那么在处理主页的同时，还需要处理这些链接指向的其他页面。

在构造好页面的 HTML 结构树后，首先利用关键字搜索(如文章列表的标题中通常含有“Publication”、“Paper”等关键字)找到目标信息所在子树，然后搜索该子树的最大扇出子树[35]，接着再用[36]中提出的启发式规则划分记录，得到单条的论文信息。

3.2.2 最大扇出子树

定义：最大扇出子树(highest-fan-out-subtree)

最大扇出子树：在一棵树中，以具有最大出度的结点为根结点形成的子树。如果有

两个结点的出度相同，则取树结点更多的子树。

如图 3.3 所示的树，以结点 UL 为根的子树是最大扇出子树。

最大扇出子树可以由树的遍历得到。

3.2.3 记录划分策略

在最大扇出子树中，利用[36]中的 MA(Most Appearance)和 SD(Standard Deviation)策略进行记录的划分：

MA：出现最频繁的标记作为记录的分割标记。

SD：每个候选标记，计算每一次出现所包含文字的长度，然后对长度求标准差，具有最小标准差的标记即为记录的分割标记。

在寻找记录分割标记时，首先对最大扇出子树应用 MA 策略，如果只找到一个标记，那么该标记就是分割标记，否则，对找到的多个标记应用 SD 策略，具有最小长度标准差的标记是分割标记。

3.2.4 提取算法

```

1  for each file f in directory
2  {
3      filequeue.in(f); //当前文件入队列
4      while filequeue is not empty //队列中还有文件
5      {
6          currentfile=filequeue.out(); //取得队列中下一个未处理页面
7          tidy.clear(currentfile); //清理当前页面
8          HTMLtree=counstructtree(currentfile); //构造 HTML 树
9          for each tag1 in HTMLtree //对 HTMLtree 中的每个标记
10         {
11             if((tag1.type==a) and (gettext(tag1) include keyword)) //如果 tag1 是超链标记并包含
                关键词
12             {
13                 targeturl=geturl(tag1); //取得 tag1 指向的链接
14                 filequeue.in(targeturl); //将 targeturl 加入到待处理队列中
15             }
16             else if (iscandidateroot(tag1)) //是论文列表所在区域子树的根
17             {
18                 HFOS=find_HFOS(tag1); //在以 tag1 为根的子树中查找最大扇出子树的根
19                 Divisiontag=find_divisiontag(HFOS); //在以 HFOS 为根的子树中找分割标记
20                 paperitemextract(HFOS, Divisiontag); //根据分割标记在最大扇出子树中提取信息
21             }
22         }
23     }
24 }
```

算法对每个待处理的主页，先用 tidy 清理其标记(line 7)，然后构造 HTML 结构树

(line 8), 对 HTML 结构树中的结点(line 9), 判断其是否是链向其他论文列表页的超链, 如果是, 将该超链的目标地址加入到队列中(line 11-13), 否则判断它是否是目标区域的根, 如果是, 在该区域内查找最大扇出子树, 然后在最大扇出子树中找分割标记, 最后根据分割标记在最大扇出子树中提取信息(line 16-21).

3.3 实验结果

实验数据包括 2070 个页面, 其中 1152 个页面有论文列表。算法找到论文列表的页面有 618 个, 正确的页面有 508 个, 所以如果从页面层次计算, 正确率为 82.2%, 查全率为 44.1%。从记录层次来看, 实验提取的论文条目有 8703 条, 其中正确的有 7241 条, 正确率为 83.2%。

从实验结果来看, 查全率不高, 主要的原因是对于论文列表所在区域的确定比较困难。本算法只考虑了最常见的情况, 即有关键字作为段落标题或锚文字的情况, 实际中还有一些例外: 没有锚文字, 而直接用图片指向论文页面; 锚文字是 “here, detail” 等词, 如 “see my publications, click [here](#)”; 主页是分帧的页面; 关键词与论文条目之间并没有显示差别, 即虽然有关键词, 但并不是段落标题。

影响正确率的一个主要因素是实验采用的 tidy 工具在清理 HTML 页面时, 有时会破坏 HTML 树的结构, 比如如果原有的 HTML 代码是:

```
<p>J. Solworth and C. Orji, "Write-Only Disk Cache Models on Multisurface Disks,"
```

```
<p>J. Solworth and C. Orji, "Write-Only Disk Cache Experiments on Multisurface Disks,"
```

两条论文记录用<p>隔开, 但没有相应的结束标记。理想的转换结果应该是:

```
<p>J. Solworth and C. Orji, "Write-Only Disk Cache Models on Multisurface Disks,"
```

```
</p>
```

```
<p>J. Solworth and C. Orji, "Write-Only Disk Cache Experiments on Multisurface Disks,"
```

```
</p>
```

但 tidy 会将其转换为:

```
<p>J. Solworth and C. Orji, "Write-Only Disk Cache Models on Multisurface Disks,"
```

```
<p>J. Solworth and C. Orji, "Write-Only Disk Cache Experiments on Multisurface Disks,"
```

```
</p>
```

```
</p>
```

这样, 两条记录之间的兄弟关系变成了父子关系, 破坏了树的结构。

3.4 小结

本章从 HTML 文档的结构出发, 介绍了 HTML 结构树及最大扇出子树的概念和构造方法, 并在此结构基础上, 应用 MA 和 SD 策略进行信息提取, 实验结果表明, 本章

提出的方法用于论文条目的提取具有较高的准确率。本方法的特点是从标记的布局出发,并且根据标记的布局结构构造 HTML 结构树,在此基础上利用启发式规则寻找记录的分割标记,由于本方法忽略具体的标记名,使其可以适用于由多种工具制作的多种风格的个人主页信息提取。

第四章 基于隐马尔可夫模型的信息提取

本章要解决的问题是对第三章的提取结果做进一步的解析。把“论文列表”中的每个论文条目信息划分成：作者、标题、会议、期刊、年等信息，以便机器的自动处理。如将下面的文章条目信息

Smillie, Keith. "Using J with software packages." Gimme Arrays, 1994, vol. 2, no. 3, pp. 10 - 11.

进一步解析成：

<author>Smillie</author><author>Keith</author><title>using J with software package</title><journal>Gimme Arrays </journal> <year>1994 </year>2, 3, pp. 10-11.

对于这种类型的信息，可以采用“启发式”方法进行提取，但效果不太理想，题目的准确率为 80.2%，作者的准确率为 82.1%，页号的准确率为 44.2%[37]。Ying Ding[38]等人提出了基于模板的方法，取得了较好的效果，但这种方法需要手工书写模板，且适应性较差。

近几年，开始了将 HMM 应用于信息提取的研究。HMM 可以有效地体现时域或空域上的随机概率过程，已经成功地应用于语音识别和手写体识别。在信息提取中，HMM 用状态对应提取域，状态的词表对应每个域出现的符号，用状态之间的转换对应各个域之间的位置关系。最初的 HMM 构造是通过观察样本手工构造。或者一个状态对应一个域[21]，或用几个状态对应一个域[17, 19]，然后通过对概率的调整来获得较好的效果，如在[17]中，提出一种 shrinkage 概率估算方法。现在的 HMM 构造一般是采用自动构造[15, 16, 18, 20]，与本文的提取任务相似的是“Cora 计算机科学研究论文搜索引擎”[20]，在 Cora 中，利用 HMM 提取每篇论文的头部信息，包括标题、作者、关键词等。

现有的 HMM 结构学习算法虽然取得了一定的效果，但是并没有充分考虑文本的特征，因此还有较大的改进空间。针对本章所述信息提取问题，本文提出一种基于符号特征提取的 HMM 结构学习方法，并通过修改的 Viterbi 算法[39]进行论文信息的解析，并与[20]的 HMM 结构学习算法进行了对比。实验结果表明，本文的算法提取效果要优于该算法。

4.1 基于符号特征提取的 HMM 结构学习

4.1.1 HMM 简介

一个 HMM 由以下几部分构成[23]：

N =模型中的状态数；

M =模型中的输出符号数；

$S = \{s_1, s_2, \dots, s_N\}$ 模型中的状态；

$A = \{a_{ij}\}$, $a_{ij} = P(s_j \text{ at } t+1 | s_i \text{ at } t)$, 状态转移的概率分布；

$B = \{b_j(k)\}$, $b_j(k) = P(v_k \text{ at } t | s_j \text{ at } t)$, 在状态 s_j 输出的符号概率分布, 其中 $\{v_k\}$ 是输出符号集；

$\pi = \{\pi_i\}$, $\pi_i = P(s_i \text{ at } t=0)$, 初始状态分布。

用 HMM 来解决信息提取的一般途径是：每个域（提取的每个语义项）对应一个或多个状态，原始文本中的符号作为状态的输出符号，如果模型给定，那么信息提取过程就是搜索最可能创建符号序列的状态序列。这个问题可以由 Viterbi 算法解决。

4.1.2 基于符号特征提取的 HMM 的状态类型确定

简单的 HMM 每个状态对应一个域，这种模型可根据域个数自动生成，其特点是实现简单，但不能很好的刻画每个域内部的特征。

我们知道，每个域都具有一定的线索，如年代由两位或四位数字组成，而形如“I. Muslea”的字符串肯定是代表人名（“I”、“.”、“Muslea”这样的被非数字字母符号隔开的字符串称之为 token），显然，我们不是由这些 token 的具体内容推断出这是一个人名，而是根据这些 token 的形式得出这是一个人名的结论。如果我们能把这些形式特征提取出来，把得到的每一个特征对应 HMM 的一个状态类型，这样就能刻画域内部的特征。

我们借用了文法推断中的 Inferring Transducers 算法[5]进行符号特征提取。在[5]中，该算法用于域的文法模型构造，它将原始的域字符串转换成抽象的特征串，以便控制字母表的规模。该算法是单域的特征提取算法，我们将它修改为多域的特征提取算法。算法自动学习待识别域内 token 的合理的抽象特征，如 Dr. Kolta 的符号特征为 “Dr” + “.” + “capi- talized”。

算法的输入包括标注好的训练样本和一个预定义的特征列表，输出是对应每个域一个用于 token 特征产生的决策列表（decision list）。

```

^begin
<label>2</label>. <author>R. Agrawal</author>and<author>R. Srikant</author>. <title>Mining sequential patterns</title>. In Proc. of Int. Conf. on
<conference>Data Engineering </conference>, Tasipei, Taiwan, Mar. <year>1995
</year>.
^End
...

```

图 4.1: 训练样本

```

single_letter      single_lower_letter  multi_letter
multi_digitletter  single_digit         multi_digit
capitalized        multi_upper_case    multi_lower_case
four_digit         two_digit            three_digit         word
single_upper_letter

```

图 4.2: 预定义的特征列表

```

If                      Emit
Single_upper_letter=true  single_upper_letter
Word= “.”               word+ “.”
Capitalized=true        capitalized
...

```

图 4.3: 对应于“作者”的决策列表

算法的基本思路是：对每一个域，正例为域包含的 token，反例为样本中包含的其他 token。算法的目的是“贪心”地将特征逐步添加到决策列表中。选择特征的标准是该特征能够最好的区分正例和反例。每添加一个特征，从训练样本中删去已被本特征覆盖的正例。直到样本中的正例数小于某个预定值。

算法说明：

1. `positivefeathash[j]`和 `anyfeathash` 是映射表，分别记录每个特征在剩余样本第 j 个域的 token 上为真的次数和在所有的 token 上为真的次数。

2. 函数 `fieldtokenuncoveredcount()` 计算未被决策表中特征覆盖的剩余样本第 j 个域的 token 数目。函数 `anytokenuncoveredcount()` 计算未被决策表中特征覆盖的剩余样本所有 token 数目。

3. `DopositiveAccounting()` 计算每个特征对剩余样本第 j 个域的 token 为真的次数，`DoAnyAccounting()` 计算每个特征对剩余样本所有 token 为真的次数。

4. 特征选择标准是使 $Gain(w)$ 最大的特征 w ，由函数 `findbestfv()` 完成每一步的特征搜索。

$$Gain(w) = \frac{f_w + m/n}{n_w + m}$$

n 是保留在样本中的所有的 token 数目， f_w 是匹配 w 的正例 token 数目， n_w 是匹配 w 的所有 token 的数目，实验中， m 取 3。

Inferring Transducers 算法

构造 token 表；

构造预定义特征表；

for $i=1$ to `labelnumber`

 初始化决策列表 `decisiontable[i]`;

for $j=1$ to `labelnumber`

{

`field=labeltable[j]`;

`tcount=fieldtokenuncoveredcount(decisiontable[j], field)`;

`acount=anytokenuncoveredcount(decisiontable[j])`;

 while `tcount`>预定值

 {

 set all feature-value entries in hash tables to 0;

`dopositiveaccounting(positivefeathash[j])`;

`doanyaccounting(anyfeathash)`;

`temppair=findbestfv(tcount, acount, positivefeathash[j], anyfeathash, field)`; //temppair 是选出特征

`addtodecisionlist(decisiontable[j], temppair)`;

`tcount=fieldtokenuncoveredcount(decisiontable[j], field)`;

`acount=anytokenuncoveredcount(decisiontable[j])`;

 }

}

通过 `Inferring Transducers` 算法，可以确定每个域的内部状态类型，对于两个域之间的 token，由于这些 token 是域之间分界的重要线索(如“Aiken, Alexander, and Alexandru

Nicolau ” 中含有 3 个作者名, 3 个人名之间的分界标记是 “,” , “,” 和 “and”) , 所以保留它们的原始形式, 用一个状态类型 “word” 表示。

4.1.3 HMM 结构学习

通过 Inferring Transducers 算法, 确定了 HMM 中的状态类型, 然后本文提出了一种状态合并方法生成 HMM:

Step1 对剩余样本 a 的 token 串, 找到决策列表中第 1 个匹配的特征, 将 a 转换成特征串 a' ;

Step2 对 a' 中的每个特征, 生成一个状态结点, 状态结点对应三项: $label$ =域标签; $type$ =特征; 词表= a 中对应的 token, 并根据 a' 中特征的出现顺序, 生成状态转换弧;

Step3 对于现有模型中的任何两个状态结点 s_a, s_b 考查是否满足以下条件:

I。 $s_a.type = s_b.type, s_a.label = s_b.label$

II。 对应 s_a, s_b , 存在两条入弧或出弧 $e1, e2$, $e1$ 的另一个顶点 s_{e1} 和 $e2$ 的另一个顶点 s_{e2} , 有 $s_{e1}.type = s_{e2}.type, s_{e1}.label = s_{e2}.label$

如果满足这两个条件, 则将 s_a 和 s_b 合并;

Step4 重复 Step1、Step2、Step3 直到处理完所有的样本。

4.2.HMM 概率学习

在 HMM 结构确定的前提下, 模型需要确定的参数有状态转换概率 A 和每个状态的符号概率分布 B 。由于在我们的 HMM 中, 对每个样本都存在一条唯一的路径, 我们直接用 “最大似然” 方法来计算 A 和 B :

$$a_{ij} = \frac{transition_{ij}}{\sum_{1 \leq j \leq N} transition_{ij}}, \quad b_j(k) = \frac{emit_{jk}}{\sum_{1 \leq r \leq M} emit_{jr}}$$

其中, $transition_{ij}$ 是状态 i 到状态 j 的转换次数, $emit_{jk}$ 是在状态 j 发出第 k 个符号的次数。

为了避免训练数据不完整带来的 “零概率” 问题, 需要对概率进行平滑。传统的概率平滑方法是 laplace 平滑, 但根据 [40] 中分析, absolute discounting 方法更适合信息提取问题。对于 $b_j(k)$ 具体处理方法是:

(1) 样本中每个出现符号的概率 = $b_j(k) - x$;

(2) 每个未出现符号的概率 = $\frac{m_j x}{m - m_j}$

m_j 是状态 j 中出现的符号数, m 是词表大小, x 的取法没有统一的标准, 在实验中我们取 $x = 1/(m_j + m)$, 对于 a_{ij} , 采取相同的处理方式。

对于 “状态合并” 途径生成的 HMM, A 和 B 可以在合并过程中得到。

4.3 基于 viterbi 算法的识别

基于 HMM 的信息提取过程可以等同于：给定 HMM 和符号序列，寻找能使产生该符号序列的概率最大的状态序列，解决这个问题最经典的方法是 viterbi 算法。但由于在每一个状态增加了一个 *type* 参数，所以修改 viterbi 算法如下：

假设输入的符号序列为 $o_1 o_2 \cdots o_T$, $V_t(i)$ 是在状态 i 已经发出 o_1, o_2, \cdots, o_t 的最大概率, $Q_t(i)$ 是使 $V_t(i)$ 最大的 $t-1$ 时刻状态。

1. 初始化:

对 $1 \leq i \leq N$,

如果 $s_i.type$ 对于 o_1 为真 则

$$v_1(i) = \pi_i b_i(o_1)$$

$$Q_1(i) = 0$$

否则

$$v_1(i) = 0$$

2. 递推计算:

对 $1 \leq i \leq N, 2 \leq s \leq T$

如果 $s_i.type$ 对于 o_s 为真 则

$$v_s(i) = \max_{1 \leq r \leq N} \{a_{ri} v_{s-1}(r) b_i(o_s)\}$$

$$Q_s(i) = \arg \max_{1 \leq r \leq N} \{a_{ri} v_{s-1}(r)\}$$

否则

$$v_s(i) = 0$$

3. 终态:

$$q_T^* = \arg \max_{1 \leq r \leq N} \{a_{r(T+1)} v_T(r)\}$$

其中, 状态 $T+1$ 表示结束状态。

4. 回溯得到最优状态序列:

对 $t = T-1, T-2, \cdots, 1$

$$q_t^* = Q_{t+1}(q_{t+1}^*)$$

$q_1^*, q_2^*, \cdots, q_T^*$ 即为所求的状态序列。

4.4 实验结果及分析

对每一个域, 通常采用下面三个评价标准: 准确率 (Precision)、查全率 (Recall)、综合评价指标 (F). 由于本章解决的是多域的信息提取, 为了综合评价信息提取系统对各个域的平均性能, 这里引入文本分类中的两个评价指标: 微 (Micro) 平均和宏 (Macro)

平均[41]来评价实验结果:

对于一个多域的信息提取系统,

$$\text{Marco } P = \frac{\sum_{1 \leq i \leq y} P_i}{y}, \quad \text{Marco } R = \frac{\sum_{1 \leq i \leq y} R_i}{y}$$

$$\text{Micro } P = \frac{\sum_{1 \leq i \leq y} \# \text{correct answers}_i}{\sum_{1 \leq i \leq y} \# \text{answers produced}_i}, \quad \text{Micro } R = \frac{\sum_{1 \leq i \leq y} \# \text{correct answers}_i}{\sum_{1 \leq i \leq y} \# \text{total possible corrects}_i}$$

y 是域的个数。为统计的方便, 这里用 token 的层次上进行统计。*#correct answers* 是指提取结果中本域正确的 token 数, *#answers produced* 是指提取结果中本域总的 token 数, *#total possible corrects* 是指样本中本域的 token 数。相应的, 可计算 *Micro F* 和 *Marco F*。微平均偏向于反映出现频繁的域的提取效果, 而宏平均偏向于反映出现次数较少的域的提取效果。

实验数据集是第三章实验结果中正确的 1200 条论文记录, 对该集合进行 4 分交叉测试, 即每次取 900 条作为训练集, 300 条作为测试集, 重复四次。提取的域包括:

author: 作者;

title: 标题;

journal: 论文发表的期刊;

year: 出版年份;

conference: 论文发表的会议。

在同样的数据集上, 进行了两种方法的对比实验, 表 4.1-4.4 是实验结果。

方法 1: 利用[20]中的方法生成的 HMM:

- 先建立初始模型: 训练集中的每个 token 对应一个状态, 状态的标签是该 token 所属的域, 状态之间的转换对应 token 之间的前后关系。
- 在初始模型的基础上, 实现两种类型的状态合并:
neighbor-merging: 如果状态 a 有到 b 的转换弧, 并且 a、b 的标签相同, 则合并 a, b.
v-merging: 如果 a、b 的标签相同, 并且有来自或指向同一个状态的两条弧分别与之关联, 那么 a, b 合并。
用标准的 viterbi 算法进行识别。

方法 2: 利用本章提出的方法生成的 HMM, 用修改后的 viterbi 算法进行识别。

表 4.1 第 1 组数据

	author		title		conference		year		journal	
	方法 1	方法 2	方法 1	方法 2	方法 1	方法 2	方法 1	方法 2	方法 1	方法 2
#ANSWERS PRODUCED	2611	1857	2475	2181	1183	802	254	224	483	492
#CORRECT ANSWERS	1918	1741	1994	1794	969	650	249	223	237	345
#TOTAL POSSIBLE CORRECTS	2413		2323		1081		251		447	

表 4.2 第 2 组数据

	author		title		conference		year		journal	
	方法 1	方法 2	方法 1	方法 2	方法 1	方法 2	方法 1	方法 2	方法 1	方法 2
#ANSWERS PRODUCED	2289	2619	3543	3426	1231	1238	257	215	526	516
#CORRECT ANSWERS	1827	2381	2740	2861	1005	1036	251	212	448	426
#TOTAL POSSIBLE CORRECTS	2587		2927		1091		260		620	

表 4.3 第 3 组数据

	author		title		conference		year		journal	
	方法 1	方法 2	方法 1	方法 2	方法 1	方法 2	方法 1	方法 2	方法 1	方法 2
#ANSWERS PRODUCED	2153	2546	2724	3450	1562	1313	273	218	882	377
#CORRECT ANSWERS	1224	1888	1818	2574	1090	1034	269	216	500	325
#TOTAL POSSIBLE CORRECTS	2319		2713		1205		270		508	

表 4.4 第 4 组数据

	author		title		conference		year		journal	
	方法 1	方法 2	方法 1	方法 2	方法 1	方法 2	方法 1	方法 2	方法 1	方法 2
#ANSWERS PRODUCED	1886	1841	2038	2755	1125	743	272	241	941	854
#CORRECT ANSWERS	840	1482	1561	2507	501	496	262	238	771	769
#TOTAL POSSIBLE CORRECTS	1771		2545		563		270		962	

两种方法的宏平均和微平均见图（取四组数据的均值）：

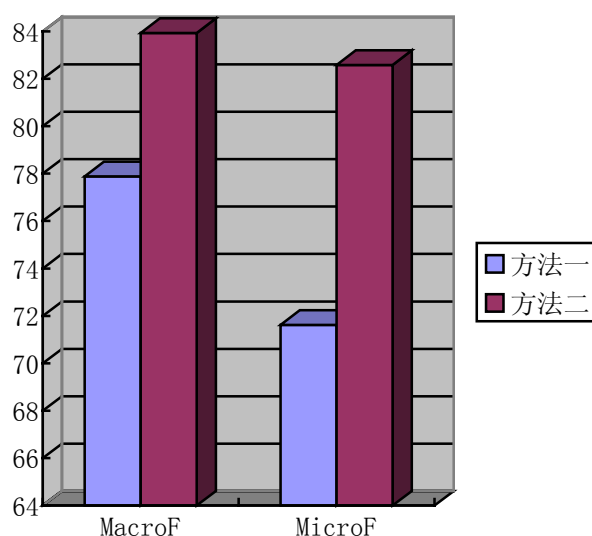


图 4.4: 宏平均与微平均对比图(前缀 mi 表示微平均, ma 表示宏平均)

- 结论一: HMM 对于密集型信息提取(要提取的域在位置上很靠近, 并且无关文本较少)是一种合适的模型。这是因为论文信息从本质上讲是一种半结构化的文本, 它的各个语义项的出现顺序有一定规律, 而且各项之间有一些线索, 而 HMM 的状态转移概率 a_{ij} 和符号发出概率 $b_j(k)$ 能表达这种规律, 同时有 viterbi 算法根据 a_{ij} 和 $b_j(k)$ 进行识别, 所以能够取得较好的效果;
- 结论二: 加入了特征提取后效果有一定的提高。这是因为在论文信息中, 如人名、年代等都有某些固定的模式, 用特征提取可以自动学习这些模式。

4.5 小结

本章提出了一种用于密集信息提取的 HMM 结构学习方法, 并用于个人主页中论文信息的提取。实现途径是用特征提取算法学习论文信息 token 的抽象特征, 并由此确定 HMM 的状态类型。在此基础上, 本章提出一种状态合并方法生成 HMM。

由于论文信息提取是一个多域的信息提取问题, 所以本章在 Precision、Recall 和 F 三个常用评价标准基础上, 引入了文本分类中的宏平均和微平均两个评价标准。实验结果表明, HMM 对于密集信息提取是一种有效模型, 而利用特征提取来学习 token 的抽象特征提高了提取效果。

第五章 基于关系学习的信息提取

第四章研究了密集型提取任务的算法，本章将解决稀疏型提取问题，即待处理的文档中含有大量的无关文本。本章的提取任务是提取个人主页中的个人信息（人名、Email）。

基于HMM的信息提取算法中，HMM的状态要考虑整个文本，这对于含有较少无关 token 的文本是合适的，否则会造成 HMM 结点个数过多，一方面会增加训练时间，另一方面大量无关 token 会降低 HMM 建模的有效性。而个人主页的差异性较大，必然含有大量无规律的 token，所以用 HMM 来从主页中提取人名这样的少量信息不是很合适。

在没有更适用的数学模型的前提下，“if then”规则是对学习假设的一种常用表示方法。通过观察可以发现，在个人主页中，人名或 EMAIL 的出现是有一定规律可循的，比如主页标题的一种模式是“***’s home page”，其中***是人名。直觉上，我们可以构造出这样一条 if-then 规则：*if* 一个 token 串出现标题中，并且在“‘s home page”之前，*then* 该 token 串是人名。遗憾的是，主页标题的模式不止这一种，要想人工构造出所有模式的 if-then 规则比较困难，所以有必要利用学习规则集合的方法。学习规则集合的经典方法是先学习决策树，再将其转化为等价的规则。但决策树学习需要实现给定足够的有用属性，而且决策树学习的规则一般是析取形式的命题逻辑表达式，其表征能力有限。对于稀疏型信息提取，需要一种表达能力更强的规则形式，关系学习是一种合适的途径，因为关系学习的样本和规则都可以用一阶子句表达，一阶子句中引入的变量和量词可以描述目标信息所在的上下文，另一方面又不必对整个文本建模，可以只考虑对于提取有用的 token。

在现有的基于关系学习的信息提取算法中，SRV 和 Rapier[14]是两个最好的算法。Rapier 使用三种类型的规则表达，分别考虑提取目标的前缀、后缀和目标本身，文本特征有词、词性和语义标签。而 SRV 考虑的是单个 token 的形式特征，不涉及词性，语义等，这对于 HTML 文本更为适合，所以，本章利用 SRV 的主体算法实现稀疏信息的提取。

SRV 是一个基于 FOIL 的信息提取算法，它基于两类特征进行归纳：一类是简单特征，将 token 映射到一个任意值（通常是布尔值）；一类是关系特征，将一个 token 映射到同一文档中的另一个 token。SRV 通过一个自顶向下的序列覆盖算法覆盖整个样本：在学习一条规则的过程中，它考虑的样本是所有的负例和还未被当前规则覆盖的正例，贪心地添加文字到规则的前件，直到当前规则不再覆盖反例或找不到下一个文字。每生成一条规则，移去其覆盖的正例，然后生成下一条规则。

5.1 术语说明

为方便算法说明，先介绍关系学习中的常见术语。

表达式：每个良构的表达式由常量（如 Mary）、变量（如 x）、谓词（如在 Female(Mary) 中的 Female）和函数（age(Mary) 中的 age）组成。

项 (term)：为任意常量、任意变量或任意应用到项集合上的函数。

文字 (literal)：是应用到项集合上的任意谓词或其否定。

基本文字 (ground literal)：是不包含任何变量的文字。

负文字 (negative literal)：是包含否定谓词的文字。

正文字 (positive literal)：是不包含否定符号的文字。

子句 (clause): 是多个文字析取式;

Horn 子句: 是一个如下形式的表达式:

$$H \leftarrow (L_1 \wedge \dots \wedge L_n)$$

其中 H , L_1 , L_n 为正文字。 H 被称为 Horn 子句的头(head)或推论(consequent)。文字合取式 $L_1 \wedge \dots \wedge L_n$ 被称为 Horn 子句的体(body)或者前件(antecedents)。

SRV 学习的规则类似于 Horn 子句。

说明: 本节内容主要摘自[43]。

5.2 样本空间

SRV 的样本空间是文本片段集合。对每个目标域, 对应一个最小长度 \min 和最大长度 \max , 这两个值可以事先设定, 也可以通过扫描训练网页得到。对每一个长度在 \min 和 \max 之间的片段, 都是合法的样本, 如果该片段是目标域的实例, 那么这个片段就是正例, 否则是反例。

训练样本通过一个预处理程序自动生成。预处理程序的输入是对目标域进行标注的网页, 输出是该网页对应的所有合法文本片段。

仅由长度限制来生成样本, 数目过于庞大, 如对一个仅含 100 个 token 的 HTML 文档, 如果长度限制为 2-4, 就会产生 294 个合法的样本。而且大多数的 HTML 文档所含的 token 数远远超过 100。在本系统中, 为了尽可能减少训练样本, 只考虑处于同一标记的文本片段, 即合法的文本片段必须满足长度限制, 同时文本片段内的所有 token 必须处于同一个 HTML 标记之内。

5.3 规则表达

SRV 的五种文字形式是:

(1) length(Relop, N)

Relop 的取值有 “<”、“=”、“>”。N 是一个整数。这种文字限制目标域的长度。比如 Length(<, 4) 表示目标域的长度小于 4。

(2) some(Var, Path, Feat, Value)

Var 是一个变量名, Path 的取值为关系特征集合, Feat 包括简单 token 特征和 HTML 结构特征集合。Value 为 Feat 的值。some 能够同时表达目标域内部和上下文的特征。比如: some(?A, [], Captialized, true) 表示在目标域内部, 存在一个 token 是 Captialized 的; some(?B, next, Captialized, true) 表示目标域中存在一个 token, 它的后继 token (不论是否在目标域中) 是 Captialized 的。

在 SRV 的规则中, 不同的变量绑定到不同的 token。

(3) every(Feat, Value)

every 文字对片段中的每 1 个 token 进行测试。比如, every(single_digit, false) 表示片段中的每个 token 都不能是 single_digit 的。

(4) position(Var, From, Relop, N)

position 对片段中特定 token 的位置进行限制, From 的取值有 fromfirst, fromlast 两种, 分别表示从比较的基准是片段头部还是尾部。比如: position(?A, fromfirst, <, 2)

表示 A 所绑定的 token 与片段头部的距离不超过 2 (单位为 token).

(5) relpos (Var1, Var2, Reop, N)

relpos 对片段内部的两个特定 token 之间的距离进行限制. 比如: relpos (?A, ?B, =2) 表示 A 所绑定的 token 在 B 所绑定的 token 前面, 且距离为 2.

5.4 文本特征

系统选取的特征包括三类: 简单 token 特征、关系特征和 HTML 结构特征。

5.4.1 简单 token 特征

简单特征主要考虑目标域内部的 token 特征. 简单特征将 token 映射到一个任意值, 这个任意值通常是二值的。

系统利用的 token 特征有 14 个:

```
single_digit      single_char  double_digit  three_digit
four_digit two_char  three_char    four_char    long_digit long_char
capitalized      all_upper_case
all_lower_case   word
```

其中, 前 13 个特征是二值的, 分别从 token 的形式和长度考虑, “word” 特征是多值的, 考虑 token 的内容。

我们注意到, 仅由 5.3 所述的 5 种文字和这些简单特征组合, 并不能完全表述文本片段的属性. 比如要表示“文本片段中不包含 home 这个词”, 由上面的 every 文字和 word 特征难以表达. every(word, wordvalue) 只能表示文本片段的每个 token 都是 wordvalue. 但如果要添加一个文字 every(word, wordvalue, true|false), 将会大大增加问题的规模, 这意味着要扫描所有的正例和反例, 获得在正例中包含而反例中没有的 token (数目巨大), 并对这些 token 按照某种策略进行优先级排序. 为了降低问题的难度, 我们通过分析样本, 手工添加了 9 个特征 “home”, “Dr” 等, 现在可以用 every(home, false) 来表示文本片段中不含有 “home”.

5.4.2 关系特征

关系特征考虑目标域的 token 与上下文之间的关系. 比如在个人主页中, Email 的前面一般会有类似 “Email:” 这样的文本片段. 系统利用的初始关系特征包括三个: “previous”, “next” 和 “current”. 它们将一个 token 映射到另外一个 (或同一个) token.

假设在网页文本中, token1 在 token2 前面, 那么

```
previous(token2)=token1
```

```
next(token1)=token2
```

```
current(token1)=token1
```

如果 token1 为第 1 个 token, 那么 previous(token1)=null; 如果 token2 为最后一个 token, 那么 next(token2)=null.

在规则学习过程中, 关系特征会自动扩展, 由考虑直接前后继到考虑距离为 2, 3, ...

的 token, 理论上可以考虑任意的前后继关系。

5.4.3 HTML 结构特征

由于网页中的文本都是处于特定的 HTML 标记之中, HTML 标记有时是信息提取的一个很好的线索, 比如在有些个人主页中, 人名会出现在<title>标记中。本文利用的 HTML 结构特征有 14 个:

in_title	in_a	in_h1	in_h2	in_h3	in_li
in_b	in_l	in_font	in_strong	in_em	
in_center	in_emphatic	in_center			

5.5 规则学习算法

对每个目标域, 可能有多条提取规则, 规则之间是“或”的关系, 即只要满足其中一条, 就识别为目标域。每条规则由多个文字(literal)组成, 文字之间是“且”的关系, 即一个片段必须满足一条规则的所有文字才算测试成功。

算法 5.1 是对一个目标域的规则学习函数。算法 5.2 是对目标域的一条规则的学习函数, 算法 5.3 是规则的第 1 个文字的学习函数, 算法 5.4 是下一个文字的学习函数。

算法 5.1

```
Function Growrule(fieldname, traindocs, simple_feats, relational_feats)
{
    rulelist=empty; //规则列表初始化
    while(have_not_matching_positive(fieldname, traindocs, rulelist)) //如果还有未匹配的正例, 则继续生成规则
    {
        rule=growonerule(fieldname, traindocs, simple_feats, relational_feats, rulelist); //生成一条规则
        add_rule_to_rulelist(rule, rulelist); //将规则添加到规则列表中
    }
    return(rulelist);
}
```

函数说明:

1. growrule 的参数 fieldname 是提取域的名称, traindocs 是训练文本所在文件夹; simple_feats 包括简单 token 特征和 HTML 结构特征; relational_feats 是关系特征集。
2. 本函数学习一个提取域的所有规则, 当样本集合中还存在正例时, 继续学习下一条规则, 否则结束。

算法 5.2[5]

```
function growonerule(fieldname, traindocs, simple_feats, relational_feats, rulelist)
{
    rule=empty; //当前规则初始化
    candidate_paths=relational_feats_to_path(relational_feats); //将关系特征转化为路径
```

```

literal=firstliteral(fieldname, traindocs, simple_feats, candidate_paths, rulelist); //生成第1个文字
add_literal_to_rule(literal, rule); //将文字添加到规则中
path=pathofsomeliteral(literal); //取得“some”文字的路径参数
candidate_paths=augmentcandidatepaths(candidate_paths, path); //增长现有的路径集合
while matchingnegativeexamples(fieldname, traindocs, rule) //当前规则匹配反例
{
    literal=nextliteral(rule, fieldname, docs, simple_feats, candidate_paths, rulelist) //生成下一个
文字
    if literal=null //已经找不到下一个文字
        break;
    add_literal_to_rule(literal, rule);
    if literaltype(literal)=some
    {
        path=pathofsomeliteral(literal);
        candidate_paths=augmentcandidatepaths(candidate_paths, path);
    }
}
return(rule);
}

```

函数说明：本函数生成一条规则。值得说明的是，本函数中，路径参数是动态增长的。每生成一个 some 文字，就将该文字中的路径附加到现有路径集合中的每一条路径，形成新的路径集合。规则产生过程终止的条件是当前规则还匹配有反例或找不到下一个文字。

算法 5.3：第 1 个文字的寻找算法

```

function firstliteral(fieldname, traindocs, simple_feats, candidate_paths, rulelist)
{
    hash=empty;
    for doc in traindocs
        for feat in simple_feats
            {
                values=allvalueindoc(doc, feat)
                for value in values
                    for path in candidate_paths
                        {
                            tokens=tokenwithpathvalue(doc, path, feat, value);
                            for token in tokens
                                {
                                    nonoverlapnegativecount(&pcount, &ncount, fieldname, doc, rulelist, token, tokens)
                                    literal=“some(?A, path, feat, value)”
                                    add_literal_to_hash(hash, literal, pcount, ncount)
                                }
                            }
                        }
            }
}

```

```

    return(findbestliteral(hash));
}

```

函数说明:

本函数生成规则的第 1 个文字。为缩小搜索空间, 假设第 1 个文字形式为 some.

1. allvalueindoc(doc, feat) 得到文档 doc 中特征 feat 的所有取值。
2. Tokenwithpathvalue(doc, path, feat, value) 得到 doc 中所有具有路径 path 和特征 feat 取值为 value 的 token.
3. Nonoverlapnegativecount (&pcount, &ncount, fieldname, doc, rulelist, token, tokens) 对包含 token 的片段进行记数, pcount 中记录文档 doc 中包含 token 的正例片段个数, ncount 记录文档 doc 中包含 token 的反例片段个数。为了避免对同一 fragment 的重复记数, 在记数时, 对含有当前 token, 并且有后继 token 出现在 tokens 的片段不进行记数。需要说明的是, Nonoverlapnegativecount () 对已经被当前规则列表(规则列表中可能已有多条规则)所覆盖的片段不进行记数。
4. 对每 1 个 path, feat, value 组合所形成的文字, 将文字和相应的覆盖正例、反例的个数存放在 hash 表中。

从 hash 表中选择一个最好的文字的标准是使下式取最大值的文字 A:

$$I(S) = -\log_2(P(S) / (P(S) + N(S)))$$

P(S) 和 N(S) 分别是样本集合 S 中的正例和反例数目。

$$GAIN(S) = P(S_A) (I(S) - I(S_A)) . S_A \text{ 是添加 A 到规则中后规则覆盖的样本集合。}$$

算法 5.4[5]: 下一个文字的生成算法

```

function nextliteral(currentrule, char* fieldname, char* docs, char* candidate_path[])
{
    varnumber=getvar(currentrule, vars); //得到当前规则中已经使用的变量名
    getnewvar(vars, newvar); //得到新的变量名
    for each file in docs
    {
        fragmentlist=getfragment(file); //获得所有的合法文本片段
        for each fragment in fragmentlist //对每个文本片段
            if is_matching_fragment(currentrule) //只考虑被正在形成的规则所覆盖的文本片段
                {
                    //是否是正例
                    if fragment.fieldname=fieldname
                    {
                        is_fieldinstance=true;
                    }
                    else
                    {
                        is_fieldinstance=false;
                    }
                    //生成 length 文字
                    search_for_length_lits(fragment, is_fieldinstance);
                    //生成对现有变量进行限制的 some 文字
                    for (k=0;k<varnumber;k++)

```

```

        search_for_some_specializations(currentrule, fieldname, candidate_path,
vars[k], fragment, is_fieldinstance);
        //生成含新变量的 some 文字
        search_for_new_some_lits(currentrule, fieldname, vars,
candidate_path, fragment, newvar, is_fieldinstance);
        //生成 every 文字
        search_for_every_lits(fragment, is_fieldinstance);
        //生成 position 文字
        for (k=0;k<varnumber;k++)
            search_for_position_lits(currentrule, vars[k], fragment, is_fieldinstance);
        //生成 relpos 文字
        for (k=0;k<varnumber;k++)
            for (l=0;l<varnumber;l++)
                if (vars[k]!=vars[l])
                    search_for_relpos_lits(currentrule, vars, varnumber, vars[k],
vars[l], fragment, is_fieldinstance);
    }
}
return (findbestliteral(hash));
}

```

函数说明:

本函数由样本生成新的候选文字集, 在该集合中寻找下一个文字, 下面是函数几个主要部分在本系统中的实现方法:

(1) search_for_length_lits():

由样本生成 length 文字, 对每一个样本 fragment, 生成三个 literal, 分别为 length(=, n), length(>, n-1), length(<, n+1), n 是 fragment 的长度。

(2) search_for_some_specializations(···, var, ···)对规则中的 var 进一步限制. 如假设现有规则中已有 some(?A, next, word, “@”), 再生成一条文字 some(?A, previous, word, “:”).

在寻找这样的文字时, 必须先将 var 绑定到 fragment 中的某个 token 上, 然后根据该 token 的特征对 var 进一步限制. 可能生成多个 literal.

(3) search_for_new_some_lits()生成含有新变量的 literal.

实现时, 只对不能绑定到当前变量集合的 token 进行处理. 根据这些 token 的特征, 生成新的 literal. 可能生成多个 literal.

(4) search_for_every_lits()考虑 fragment 中所有 token 的共性.

如果所有 token 都具有某一特征, 则生成新的 literal, 如果所有的 token 有多个共同特征, 则生成多个 literal.

(5) search_for_position_lits(···, var, ···)

对规则中 var 的位置限制, 首先必须将 var 绑定到 fragment 的某一个 token, 根据该 token 在 fragment 中的位置生成 6 个 literal:

position(var, fromfirst, <, k),	position(var, fromlast, <, n-k+1)
position(var, fromfirst, =, k-1)	position(var, fromlast, =, n-k)
position(var, fromfirst, >, k-2)	position(var, fromlast, >, n-k-1)

这里，假设 fragment 的长度为 n ，该 token 是 fragment 的第 k 个 token。

(6) `search_for_repos_lits(..., var1, var2, ...)`

对规则中两个变量关系进行限制的 literal。首先必须将 `var1` 和 `var2` 绑定到 fragment 的两个 token `a` 和 `b`，根据 `a` 和 `b` 的相对位置生成 3 个 literal。

`Relpos(var1, var2, <, j-i+1)`

`Relpos(var1, var2, =, j-i)`

`Relpos(var1, var2, >j-i-1)`

假设 `a` 是 fragment 的第 i 个 token，`b` 是 fragment 的第 j 个 token，且 $i < j$ ，否则，生成相对应的 3 个 literal。

在生成上述的 literal 的同时，对其进行计数。如果 hash 表中已有这个 literal，根据 fragment 是正例还是反例改变其计数，如果 hash 表中没有这个 literal，则将其添加到 hash 表中，根据 fragment 设置 literal 覆盖的正例和反例数。

需要特别说明的是，对于含有参数 “<”，”>” 的 literal 不能在生成时进行计数。因为样本的顺序是随机的，如果立即进行计数，可能不准确。举例来说，如果先遇到一个长度为 2 的样本 `fragment1`，那么会生成 `literal1:length(<, 3)`，然后遇到一个长度为 4 的样本 `fragment2`，那么生成 `literal2:length(<, 5)`，显然先前遇到的 `fragment1` 也是满足 `literal2` 的，但当前已经不知道前面遇到了多少满足 `literal2` 的样本。所以系统采用了下面的策略：

生成完所有的 literal 后，根据相应的含参数 “=” 的 literal 对含有参数 “<”，”>” 的 literal 重新计数。如 `length(<, 3)` 的两个计数 `positive`、`negative` 由下列公式计算：

$$positive = \sum_i literal_i.positive \quad negative = \sum_i literal_i.negative$$

这里，`literali` 是形如 `length(=, n)` 的 literal， $n < 3$ 。

5.6 基于规则的认识

在训练结束后，对每个域，一般会生成多条规则，SRV 算法对每条规则赋予一个可信度 (`m-estimate`):

$$c = \frac{\text{number of positive covered} + m / \text{number of samples}}{\text{number of covered} + m}$$

`number of positive covered` 是规则覆盖的正例数，`number of covered` 是规则覆盖的样本数，`number of sample` 是样本总数。

对于系统所给出的每个识别结果，也有一个可信度：

$$C = 1 - \prod_{c \in C} (1 - c)$$

`c` 是匹配的每条规则的可信度。

在识别时，可以通过调节 `C` 来平衡识别的准确率和查全率。

5.7 实验结果及分析

系统的实验数据包括 400 个来自于各大学网站的个人主页。分为训练集和测试集两个部分。提取信息为人名和 Email 地址。

下面是学习获得的提取人名的第一条规则（训练集包括 60 个网页）：

```
some(?A, next, in_h1, true)
length(=, 2)
some(?A, previous, in_h1, false)
every(all_lower_case, false)
some(?A, [], Page, false)
some(?A, [], in_h1, false)
```

这条规则覆盖 22 个正例，5 个反例。

提取 Email 的第一条规则（训练集包括 100 个网页）：

```
some(?A, [], word, "edu")
position(?A, fromfirst, =, 6)
length(<, 8)
some(?B, next, word, "@")
relpos(?A, ?B, =6)
```

这条规则覆盖 118 个正例，0 个反例。

查全率与准确率随可信度变化如图 5.1, 5.2。由图 5.1 可看出，随着可信度的上升，准确率越高，而查全率越低。而在图 5.2 中，可信度在 0.2-0.7 之间，查全率和准确率都没有变化，这是因为提取 Email 的规则数很少，通过学习只获得 4 条规则。

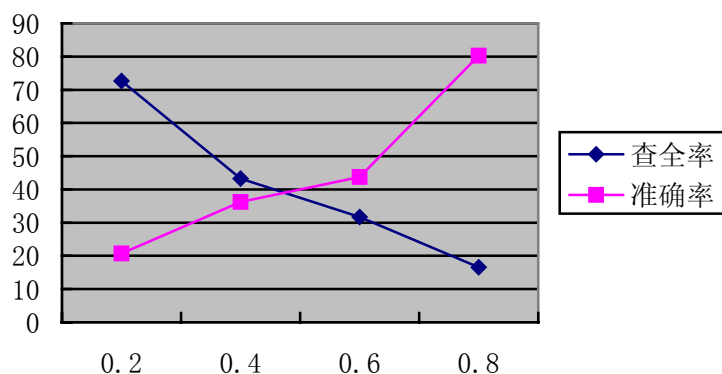


图 5.1: 提取人名的准确率和查全率随可信度变化折线图，训练样本 60 个，测试样本 340 个。横轴为可信度，竖轴为百分比。

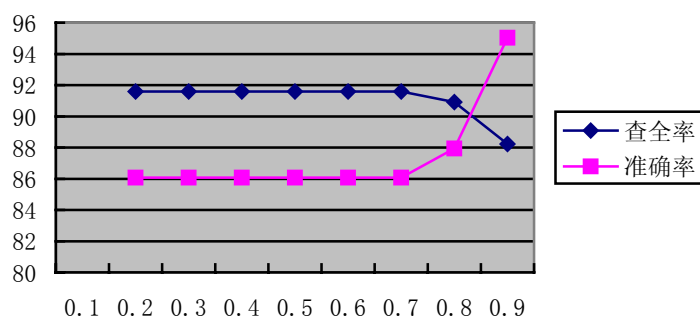


图 5.2: 提取 Email 的准确率和查全率随可信度变化折线图，训练样本 100 个，测试样本 300 个。横轴为可信度，竖轴为百分比。

图 5.3 和 5.4 是训练样本规模对提取效果的影响折线图。值得注意的是，算法

在很少的样本基础上就能学习出有效的规则。如图 5.3, 在 10 个样本情况下, 就能达到 80% 以上的查全率。另外一个现象是提取效果并没有简单的随着样本数的增加而提高。由于在实验中各次所取的样本是随机选择的, 所以样本的规模对于提取效果没有绝对的影响, 只取决于样本的具体模式。

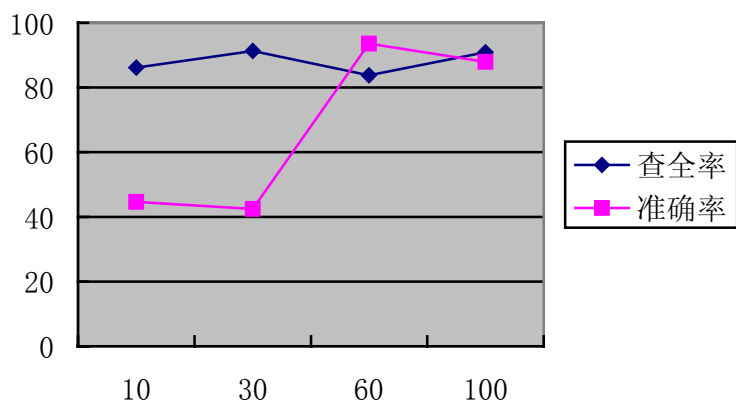


图 5.3: 提取 Email 的训练样本规模对提取结果的影响折线图(可信度为 0.5), 横轴为样本个数, 竖轴为百分比。

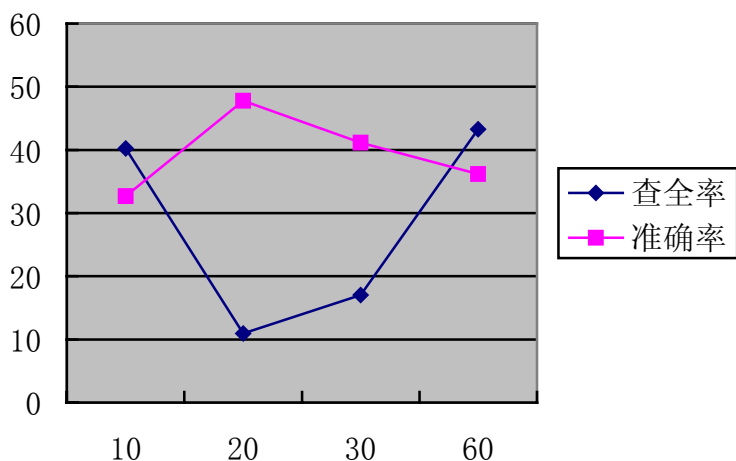


图 5.4: 提取人名的训练样本规模对提取结果的影响折线图(可信度为 0.4), 横轴为样本个数, 竖轴为百分比。

- 结论一: 关系学习对于稀疏型信息提取是一种有效的方法。尤其是对于从 Web 中提取人名这种困难的任务, 由于其内容没有什么规律可行, 而且又不能借助于句子的分析来学习规则, 只能通过它出现的上下文, 如所在的标记、临近的词等, 这种要借助于对象间的关系学习规则的问题, 最适合用关系学习来解决, 可以说, 还没有另外的途径可以与之媲美。
- 结论二: 关系学习只需要很少的样本数就能迅速的学习出规则, 因为它的提取知识不是蕴涵在统计概率中, 而是以一阶逻辑语言来描述。极端地说, 对单独的一个样本就能学习出一条或多条规则。
- 结论三: 关系学习的学习结果容易理解。如前面学习得到的两条规则, 可以立即翻译为自然语言。所以, 可以方便地对学习结果进行验证和修改。

5.8 讨论与小结

本章实现了一个基于关系学习的信息提取算法，该算法沿用了 SRV 算法的主题框架和思路，在此基础上，对特征选择、样本生成、规则学习过程提出了具体的实现方法。

实验结果表明，该算法对于稀疏型信息提取效果较好，主要体现在：

- (1) 需要的训练样本很少；
- (2) 准确率较高；
- (3) 规则具有良好的可读性。
- (4) 可以通过调节可信度来平衡查全率与准确率。

另一方面，该算法也有一定的缺陷。最突出的一个弱点是时间复杂度太高，它的规则学习是在规则空间内进行穷举式搜索，所以一次训练经常耗费数小时的时间。这也是许多基于关系学习的信息提取算法的共性。另一个问题是样本空间太大，虽然本系统对样本空间进行了进一步收缩，只考虑同一个标记内的文本片段，但其反例的数量还是惊人的。在实现时，往往不能一次读入内存，需要来回地和硬盘交换数据，又大大增加了训练时间。

针对这两个问题，该算法有两个改进方向：

(1) 选择性取样。可以分为两个层次：HTML 文档层次和文本片段层次。HTML 文档层次是指通过某种策略，判断文档是否对规则的学习起作用，如果没有，则抛弃该文档，否则将其加入到训练集中。文本片段层次是指进一步限制反例的空间。

(2) 增量式学习。由于一次训练时间很长，因此训练结果是十分宝贵的。但如果要添加新的训练样本，就需要重新训练。如果能在已有规则的基础上对新样本生成新规则而原有规则仍然有效，就能节约训练时间。

该算法的另外一个问题是特征的选择。初始特征如果选得过多，就会造成规则空间过大，增加训练时间，如果选得过少，又会造成规则的表达能力不够充分，影响提取效果，因此一个有效的自动的特征选取算法也是一个值得研究的问题。

第六章 总结

Web 信息提取技术是知识发现的一项重要技术，其典型应用如元搜索、信息代理等。同时，Web 信息提取又是一个新兴的研究领域，它从出现发展至今不足十年的时间，还有许多问题值得研究。

6.1 主要研究内容

本文第一章介绍了 Web 信息提取出现的背景。Web 上信息爆炸式增长，人们对信息检索的准确性和信息访问的方便性要求提高，促进了对 Web 信息提取的研究。

本文第二章从 Web 信息提取的发展历史、Web 信息提取的途径、Web 信息提取的文本特征、提取知识表达、主要学习算法、评价标准等方面对 Web 信息提取技术进行阐述。Web 信息提取的前身是信息提取，它是由美国国防部领导的 TIPSTER Text Program 所发起，该行动通过消息理解会议推动了对于文本处理的研究。Web 信息提取与传统的信息提取有所不同，因为 Web 文档的特点，所以需要在传统信息提取的基础上，研究新的方法。Web 信息提取的特征有 token 特征、关系特征、片段特征、HTML 结构特征等几种。提取知识的表达有有限自动机和一阶逻辑规则两种，主要学习算法分别是：基于文法推断的信息提取算法；基于 HMM 的信息提取算法，基于关系学习的信息提取算法。Web 信息提取的评价标准有准确率、查全率、综合评价指标三种。另外，第二章还介绍了典型的 Web 信息提取系统。

本文第三章从 HTML 结构出发，探讨 HTML 结构线索在信息提取中的应用，提出一种基于 HTML 结构树的信息提取算法，并对个人主页中的“文章列表”信息进行识别，取得了 82.2% 以上的准确率。

本文第四章对密集型提取任务进行研究，提出一种基于 HMM 的信息提取算法，该算法利用文法推断的 Inferring Transducers 算法进行 token 抽象特征提取，生成 HMM 的状态结点，然后提出一种状态合并方法生成最终的 HMM，再利用最大似然方法学习 HMM 的概率分布，最后利用修改的 viterbi 算法进行识别。在对个人主页中单篇文章信息进行识别时，取得了 80% 以上的准确率。

本文第五章对稀疏型提取任务进行研究，实现了一种基于关系学习的信息提取算法。该算法沿用 SRV 算法的主体框架和思路，对特征选择、样本生成和规则学习过程给出了具体的实现方法。然后利用该算法进行个人主页中人名、EMAIL 信息的提取。实验表明，利用该算法，在样本数很少的情况下，就可以取得很高的准确率。该算法还可以根据规则的可信度来平衡查全率和准确率。

本文所有工作在“国家科学数字图书馆智能化网络信息搜索技术与机制研究”项目中开展，初步实现了该项目中的 Web 信息提取模块，该模块的主要作用在于利用 Web 资源自动构建学科资源库。该资源库可以用于直接查询和辅助分类等。

图 6.1 是演示系统的查询界面，图 6.2 是查询结果页面。

计算机学科研究人员查询系统

根据人名查找：

人名：

根据发表的论文信息进行查找：

论文标题：

期刊名：

会议名：

图 6.1

计算机学科研究人员查询系统

查询结果

人名: *M. Betke*

主页地址: <http://www.cs.bu.edu/faculty/betke/>

EMAIL: betke@cs.bu.edu

论文列表:

1. Multiple Vehicle Detection and Tracking in Hard Real Time
其他作者: E. Haritaoglu, L. Davis 时间: 1996
会议: IEEE Symposium on Intelligent Vehicles
下载地址: <http://www.cs.bu.edu/faculty/betke/papers/betke-ha-da-96.pdf>
2. Information-Conserving Object Recognition

图 6.2

6.2 下一步研究内容

前面已经提到, Web 信息提取是一个新兴的研究热点, 在算法方面还没有十分成熟, 而且由于 Web 信息的多样性, Web 信息提取还面临很多挑战。本文对 Web 信息提取技术做了初步研究, 对特定类型网页的特定信息的提取做了一些工作。下一步将继续在 Web

信息提取的特征的自动选择、增量式学习算法等方面做进一步研究。

参考文献

- [1] Nicholas Kushmerick, Bernd Thomas, “Adaptive information extraction: Core technologies for information agents”, In Intelligent Information Agents R&D In Europe: An AgentLink Perspective, 2002.
- [2] Line Eikvil, “Information Extraction from World Wide Web”, Survey Report, 1999.
- [3] K. Zechner. “A Literature Survey on Information Extraction and Text Summarization”. Term paper, Carnegie Mellon University, 1997.
- [4] 陆钟万, “面向计算机科学的数理逻辑”, 科学出版社, 1998.
- [5] Freitag, D. Maching. “Learning for information extraction in informal domains”. Ph.D. thesis, Carnegie Mellon University, 1998.
- [6] 张瑞岭, “文法推断研究的历史和现状”, 第 10 卷, 第 8 期, 软件学报, 1999.
- [7] 李效东, 顾毓清, “基于 DOM 的 Web 信息提取”, 第 25 卷, 第 5 期, 计算机学报, 2002.
- [8] Peter Clark, Tim Niblett, “The CN2 Induction Algorithm”, Machine Learning, vol. 3, 1989.
- [9] Quinlan, J. R. 1990. “Learning logical definition from relations”. Machine Learning, vol. 5, 1990.
- [10] Dayne Freitag, “Using Grammar Inference to Improve Precision in information Extraction”, in ICML '97 Workshop on Automata Induction, Grammatical Inference, and Language Acquisition, Nashville, TN, USA. 1997.
- [11] T. W. Hong and K. L. Clark. “Using grammar inference to automate information extraction from the Web”, In Principles of Data Mining and Knowledge Discovery, pages 216–227, 2001.
- [12] C. -N. Hsu and M. -T. Dung. “Generating finite-state transducers for semi-structured data extraction from the Web”, Information Systems, 23(8): 512–538, 1998
- [13] B. Chidlovskii, J. Ragetli, and M. de Rijke. “Wrapper generation via grammar induction”. In 11th European Conference on Machine Learning, ECML' 00, pages 96–108, 2000.
- [14] Mary Elaine Califf, “Relational Learning Techniques for Natural Language Information Extraction”, Ph.D. thesis, the university of Texas at Austin, 1998.
- [15] Nancy R. Zhang, “Hidden Markov Models for Information Extraction”, June, 2001.
- [16] Soumya Ray, Mark Craven, “Representing Sentence Structure in Hidden Markov Models for Information Extraction”, Proceedings of the 17th International Joint Conference on Artificial Intelligence, Seattle, WA. Morgan Kaufmann.
- [17] Dayne Freitag, Andrew Kachites MaCallum, “Information Extraction with HMMs and Shrinkage”, AAAI99.
- [18] Dayne Freitag and Andrew McCallum, “Information Extraction with HMM Structures Learned by Stochastic Optimization”, Proceedings of AAAI-2000.

- [19]T. R. Leek, “Information extraction using hidden Markov models.” Master’ s thesis, UC San Diego, 1997.
- [20] Kristie Seymore, Andrew McCallum, Ronal Rosenfel, “Learning Hidden Markov Model Structure for Information Extraction”, AAI’ 99 Workshop on Machine Learning for Information Extraction.
- [21]J. Connan, C. W. Omlin, “Bibliography Extraction with Hidden Markov Models”, Technical Report US-CS-TR-00-6, 24 February 2000, Department of Computer Science, University of Stellenbosch.
- [22]Y. Sakakibara, “Recent advances of grammatical inference”, Theoretical Computer Science 185, 14-45, 1997.
- [23]史忠植, “知识发现”, 清华大学出版社, 2002。
- [24]S. Muggleton, C. Feng. “Efficient Induction of Logic Programs”. Proceedings of the First Conference on Algorithmic Learning Theory. New York, 1990.
- [25]Mary Elaine Califf, Raymond J. Mooney, “Relational Learning of Pattern-Match Rules For Information Extraction”, Working Notes of AAI Spring Symposium on Applying Machine Learning to Discourse Processing, 1997.
- [26]Dayne Freitag, “Information Extraction from HTML: Application of a General Matching Learning Approach”, Proceedings of the Fifteenth Conference on Artificial Intelligence AAI-98 (1998), 517--523.
- [27]Muggleton, S., and Feng, C. 1992. “Efficient induction of logic programs”. In Muggleton, S., ed., Inductive Logic Programming. New York: Academic Press.
- [28]Zelle, J. M., and Mooney, R. J. 1994. “Combining top-down and bottom-up methods in inductive logic programming”. In Proceedings of the Eleventh International Conference on Machine Learning.
- [29]Muggleton, S. 1995. “inverse entailment and Prolog”. New Generation Computing Journal 13:245-286.
- [30]Dan Roth, Wen-tau Yih, “Relational Learning via Propositional Algorithms: An Information Extraction Case Study”, Proc. of the International Joint Conference on Artificial Intelligence, 2001.
- [31]N. Kushmerick, D. S. Weld, R. Doorenbos. “Wrapper Induction for information Extraction”. 15th International Joint Conference on Artificial Intelligence (IJCAI-97), Nagoya, August 1997.
- [32]N. Kushmerick. “Wrapper Induction for Information Extraction”. Ph. D. thesis, University of Washington. Technical Report UW-CES-97-11-04, 1997.
- [33]<http://www.cora.justresearch.com>
- [34]Dan Dipasquo, “Using HTML Formatting to Aid in Natural Language Processing on the World Wide Web”, Senior Honors Thesis, School of Computer Science, Carnegie Mellon University, June, 1998.
- [35]D. W. Embley, Y. Jiang, Y. -K. Ng, “Record-Boundary Discovery in Web Documents”, SIGMOD’ 99.
- [36]D. W. Embley, D. M. Campbell, T. S. Jiang, etc, “A Conceptual-Modeling Approach to Extracting Data from the Web”, In Proceedings of the 17th International Conference on Conceptual Modeling (ER’ 98), Singapore, November 1998.

- [37]C. Lee Giles, Kurt D. Bollacker, Steve Lawrence. “CiteSeer:An Automatic Citation Indexing System” ,Digital Libraries 98.
- [38]Ying Ding, Gobinda Chowdhury, Schubert Foo. “Template mining for the extraction of citation from digital documents” Proc. Second Asian Digital Library Conference (Taiwan) 47-62, Nov 8-9, 1999.
- [39]Rakesh Degad, U. B. Desai, “A TUTORIAL ON HIDDEN MARKOV MODELS” , Technical Report No. : SPANN-96. 1, Department of Electrical Engineering , Indian Institute of Technology.
- [40]Vinayak Borkar, Kaustubh Deshmukh, Sunita Sarawagi. “Automatic segmentation of text into structured records” ,Proceedings of the 2001 ACM SIGMOD international conference on Management of data 2001, Santa Barbara, California, United States.
- [41]John M. Pierre, “Practical Issues for Automated Categorization of Web Sites” , ECDL 2000 Workshop on the Semantic Web.
- [42]STEPHEN MUGGLETON AND LUC DE RAEDT, “Inductive Logic Programming :Theory and Methods” , Journal of Logic Programming, 19, 20, 1994.
- [43]Tom M. Mitchell 著, 曾华军 张银奎等译, “机器学习”, 第 10 章 “学习规则集合” , 机械工业出版社, 2003. 1。

致谢

衷心感谢我的导师高文教授，高老师开阔的视野、严谨的治学态度和强烈的事业心对我产生了深远的影响。在我学习和完成论文期间，始终得到了他的悉心指导。高老师的言传身教将使我终生受益。

感谢黄铁军老师在研究和生活上给我的帮助，他为我的研究和论文指出了具体的方向。

感谢中科院研究生院网络多媒体研究中心知识引擎研究组的各位成员，他们协助我完成了很多工作。感谢中科院计算技术研究所的老师 and 同学们给我关心和支持！

由衷感谢尊敬的学术界前辈在百忙之中评阅本文。

谨将本文给我的父母亲，以报答他们的养育之恩。

作者简介及发表的学术论文

张玲，女，1978 年生于四川省自贡市，1995 年毕业于四川省荣县中学，进入四川师范大学计算机科学系学习，1999 年毕业并获计算机科学教育专业理学学士学位，同年考入中科院计算技术研究所攻读硕士学位，师从高文教授，主要研究方向为：智能搜索引擎技术、Web 信息提取、机器学习。在读期间参加过的主要项目有：

- “中美百万册书数字图书馆”工程。负责电子书描述标准确定、电子书制作工具和电子书浏览器开发；
- “国家科学数字图书馆智能化网络信息搜索技术与机制研究”，负责 Web 信息提取技术算法研究与实现。

在读期间发表的学术论文：

1. 张玲，黄铁军，张小明，“一种基于 xml 的电子图书描述与处理工具”，第七届计算机科学与技术研究生学术研讨会，四川广元，2002。
2. 张玲，黄铁军，高文，“基于隐马尔可夫模型的引文信息提取”，《计算机工程》，已录用。