正则语言推断综述

高俊涛[1],王 梅[1],徐光会[1],刘 聪[2]
(1. 东北石油大学计算机与信息技术学院,黑龙江大庆 163318;2. 山东理工大学计算机科学与技术学院,山东淄博 255000)

摘 要: 正则语言推断研究从语言的有限信息出发,通过归纳和推理得出正则语言模型.该技术在信息抽取、软件工程、模式识别等领域应用广泛.本文首先阐明了语言的可学习性概念和推断结果的评价准则.然后从推理策略、数据结构、算法复杂性等方面,对被动、主动和基于神经网络的学习算法进行分类归纳与对比,梳理各流派的技术发展脉络.接着分析推断产生的三种泛化效应.最后指出当前研究中不足,对未来研究方向进行展望.

关键词: 正则语言;归纳学习;正则推断;自动机学习;正则表达式学习;循环神经网络
中图分类号: TP181 **文献标识码:** A **文章编号:** 0372-2112(2021)12-2479-11
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.12263/DZXB.20200404

A Survey on Regular Language Inference

GAO Jun-tao[1], WANG Mei[1], XU Guang-hui[1], LIU Cong[2]
(1. *School of Computer and Information Technology, Northeast Petroleum University, Daqing, Heilongjiang* 163318, *China*;
2. *School of Computer Science and Technology, Shandong University of Technology, Zibo, Shandong* 255000, *China*)

Abstract: The regular language inference is devoted to the study of how to derive the definition of the language from the limited information. Regular language inference is popular in information extraction, software engineering, and schema learning, etc. The learnability models of regular languages and evaluation criteria are clarified. The passive, active, and neural network-based learning algorithms are respectively surveyed and compared from the aspects of reasoning strategies, data structures, algorithm complexity, etc, and the development trend is pointed out. Three kinds of generalization effects are analyzed. This paper points out the defects of the current researches and looks forward to future research directions.

Key words: regular languages; inductive inference; regular inference; automata learning; regular expression learning; recurrent neural network (RNN)

1 引言

正则语言推断是形式语言归纳学习的一个分支,研究如何从有限的语言信息推断正则语言模型,包括有限自动机、正则表达式和形式文法.与深度学习采用的神经网络相比,正则语言推断得到的语言模型结构清晰,能够更好地帮助人们理解潜藏在序列数据中的模式.目前,正则语言推断技术在信息抽取[1]、网络安全[2]、软件工程[3,4]等领域都得到了广泛的应用.

语言归纳学习研究起源于20世纪60年代.Gold在研究儿童学习自然语言的过程中提出了语言极限识认模型,首次对正则语言的可学习性进行阐述.后来,Angluin等学者在正则语言的学习模型、学习算法、学习效率等方面进一步研究,为正则语言推断打下了坚实的理论基础.早期研究工作侧重于理论研究,提出的算法具有坚实的理论基础,但应用条件比较苛刻.随着大数据时代的到来,序列数据的类型更加丰富、规模更加庞大,正则语言推断技术迎来新的挑战和机遇,更高效和实用的正则语言推断技术应运而生.

2 正则语言的可学习性

可学习性是正则语言推断首要解决的基础理论问题.Gold将极限思想引入语言可学习性研究,提出的语言极限识认模型(language identification in the limit)[5],为正则语言推断研究奠定了重要的理论基础.

定义1(极限可识认性) 假设在每个时刻t,学习者接收到一条关于目标语言M的语料信息并构建语言

收稿日期:2020-04-27;修回日期:2021-01-06;责任编辑:覃怀银
基金项目:国家自然科学基金(No. 51774090, No. 61902222);山东省泰山学者工程专项基金资助项目(No. tsqn201909109);大庆市指导性科技计划项目(No. zd-2019-22)

模型H_t. 若存在某个时刻T,后续构造的所有语言模型$H_t(t > T)$都稳定不变且正确描述语言M,则称语言M被极限识认.

在极限识认模型中,收敛性是语言极限识认的必要条件. 如果某个时刻的学习结果正确,但学习过程发散,仍不能称该语言被极限识认.

极限可识认性对语料信息非常敏感. 在提供正负样本的条件下,包括上下文相关文法在内的很多语言类都是可识认的,但是在只提供正样本的条件下,连简单的正则语言都是不可识认的. 这是因为正样本无法驳斥过度泛化的语言模型. Angluin提出了基于正样本的语言过度泛化判定方法[6],但并不是所有的语言都能计算出特征样本.

定义2(PAC可识认性[7]) 给定错误参数$\varepsilon \in [0,1]$和信度参数$\delta \in [0,1]$,学习者可PAC(ε,δ)识认目标语言M,当学习者根据样本数据推理得到的假设语言H,至少以概率$1-\delta$保证$P(s \in H \oplus M) \leqslant \varepsilon$. 其中$s$是测试集中任意字符序列.

起源于机器学习领域的PAC可识认性允许可能近似正确地推断正则语言模型.

因为正则语言全集的VC维度是无限大的,所以正则语言全集不能被PAC识认. 若将DFA(Deterministic Finite Automata)的状态数量限定在某个范围,VC维度转化为有限值. 我们用DFA_n表示状态数量不超n的所有DFA集合,可以证明DFA_n的VC维度为$O(n\log n)$[8]. 从信息论的观点看,DFA_n是PAC可识认的,而且结果自动机的规模越小,越可能接近于目标语言. 但是学习小规模自动机属于NP完全问题[9,10]. 如果允许主动地提出成员问题,最小DFA是PAC可识认的[11].

定义3(描述性泛化[12]) 称某个语言类中语言模型H是对目标语言M的描述性泛化,当且仅当在该语言类中不存在语言模型H'满足$M \subseteq L(H') \subset L(H)$. 函数$L$定义语言模型所描述的语言.

描述性泛化不要求学习结果准确定义目标语言,允许在给定的语言类中寻找最接近目标语言的语言模型. 描述性泛化与极限识认模型从不同角度为正则语言推断算法提供评判标准,主要用于定义正则语言子类的可学习性. 符合极限识认模型的学习算法并不一定达到描述性泛化要求.

3 模型质量评价准则

在提供正负样本的条件下的模型质量评价指标包括:Recall、Precision、F-measure、BCR(Balanced Classification Rate)和ε-similarity. 前三项指标比较常见,无需赘述. 下面给出后两项指标的定义.

(1) BCR[13]:敏感性和特异性的调和平均数,即,

$$\mathrm{BCR}=\frac{2PQ}{P+Q}$$

其中P代表敏感性,Q代表特异性.

(2) ε-similarity[14]:考虑自动机A_0和数据集D_0. 其中,D_0由可以用A_0生成的正样例集和不能用A_0生成的负样例集组成. 我们称自动机A_1与A_0在D_0上具有ε-similarity,若从D_0中随机选取任意样例,A_1以概率$1-\varepsilon$给出与A_0一致的分类结果(正样例或负样例),也就是说,D_0中A_1与A_0分类结果相同的样例占比为$1-\varepsilon$.

在只提供正样本的应用场景中,上述指标是无法计算的,可采用语言泛化程度或编码复杂程度衡量语言模型的优劣.

(1) 语言泛化程度

语言泛化程度指正则语言包含样本以外字符序列的数量. 无限正则语言的泛化程度可以通过限定字符序列长度或衡量增长率方法[15]近似量化语言规模. 包含关系可定性描述语言的泛化程度. 正则语言包含关系的判定可利用有限自动机的逻辑运算实现.

(2) 编码复杂程度

MML(Minimum Message Length)评价准则[16]的本质是基于给定样本的语言模型后验概率最大化. J. Rissanen提出的MDL(Minimum Description Length)准则[17]与MML准则类似,也选择编码长度作为评价准则,但二者的理论基础不同. MML准则以贝叶斯理论为基础,而MDL准则源自柯尔莫戈洛夫复杂度理论.

4 正则语言推断算法分析

正则语言推断方法取决于所能获取的语言信息和信息获取方式. 为了方便读者定位自己感兴趣的算法,本节将语言信息类型和获取方式作为提纲组织正则语言学习算法,将现有学习算法划分为正负样本学习、正样本学习、主动学习和基于神经网络的四个大类,从语言模型、推理机制、数据结构、推理策略、算法复杂度几个维度对比,分析各个学术流派的特点和发展趋势.

4.1 正负样本学习

根据Gold极限识认理论,正则语言在正负样本充足的条件下是可学习的. 早期的正负样本学习算法的研究主要采用有限自动机作为语言模型,近些年面向正则表达式的正负样本学习算法才开始出现并应用于信息抽取和信息分类等应用领域.

虽然识认与样本一致的最小DFA属于NP完全问题[18],但当样本满足某些性质时,学习算法可以在多项式时间内获取最小DFA. TB算法是最早的最小DFA学习算法[19],首先根据一致完整性样本(包含长度小于某个常数的所有样例)构建前缀树自动机PTA(Prefix

Tree Acceptor),然后迭代地选择状态两两合并,直到无状态可合并为止. 同时期的Gold算法用证据表(evidence table)组织样本,若样本是目标自动机的特征样本(目标自动机的所有状态在样本中都是明显可区分的),就可以通过识别明显可区分(obviously distinguishable)状态进行状态分裂,最终生成最小DFA. 虽然TB算法与Gold算法表述方式不同,但基本逻辑相同,因此被合称为TBG算法.

最小DFA推断所需的样本性质的探索经历了漫长的过程. 早期的TBG算法研究结果表明提供一致完整性样本或语言的特征集样本是生成最小DFA是充分条件[18,19],否则可能得出与样本不一致的自动机. 后续研究发现比一致完整性宽松的样本合理完整性[20]也能保证获取最小DFA. 2020年提出的样本语义完整性[21]进一步放宽了最小DFA推断的充分条件,并提出相应的推断算法.

尽管如此,理论上有效的样本特性在实际应用中仍然很难满足. 工程实践需要更健壮的推断算法,根据任意样本进行推断. RPNI(Regular Positive and Negative Inference)算法[22]采用贪婪状态合并策略,具有较强的泛化能力,从任意样本都能推断出一致的DFA. 同时期提出的Traxbar算法[23]与RPNI算法表示方式不同,但基本思想是一样的.

虽然RPNI算法能够保证结果自动机与样本的一致性,但是其按字典序搜索可合并状态的策略比较死板,在样本不是特征集的条件下不能保证生成的DFA是最小的. 针对相同的前缀树自动机,如果按不同的顺序合并状态,生成的自动机规模可能存在很大差异. 为提高结果自动机的质量,后续研究对状态合并顺序的优化给予了极大的关注,并产生众多的状态合并算法,其中代表性算法包括三类:启发式状态合并算法[24]、搜索式状态合并算法[25,26]、约束可满足性问题[27~29]的编码算法.

如表1所示,学习算法通常始于仅接受样本数据的有限语言,采用PTA、APTA(Augmented Prefix Tree Acceptor)、PTMM(Prefix Tree Moore Machine)或证据表描述. 其中,PTA是只包含正样本的前缀树自动机;APTA在PTA基础上增加了拒绝态,可描述正样本和负样本;PTMM采用摩尔型有限状态机定义前缀树,支持多分类样本的描述. 证据表采用表格的形式描述自动机的结构,支持自动机的逐步细化.

表1　正负样本学习算法列表

算法名称	数据结构	样本条件	语言模型	算法效率
TB算法[19]	APTA	一致完整性样本	最小DFA	运行时间上限:mn^2
		非一致完整性样本	PTA	
Gold算法[18]	状态特征矩阵,也称为证据表	特征样本	最小DFA	时间复杂度:$O(n^2 \cdot \|\|S\|\|)$
		非特征样本	PTA	空间复杂度:$O(\|\|S\|\| \cdot n)$
RPNI算法[22]	PTA	特征样本	最小DFA	时间复杂度:$O((\|\|S^+\|\| + \|\|S^-\|\|) \cdot \|\|S^+\|\|^2)$
		非特征样本	DFA	运行时间上限:mn^2
EDSM算法[24]	APTA	特征样本	最小DFA	运行时间上限:$(m^3n, m^4n), \ll m^4n$
		非特征样本	DFA	
Blue-Fringe 算法[24]	PTMM	特征样本	最小摩尔DFA	运行时间上限:mn^3
		非特征样本	摩尔DFA	
Zhang算法[21]	红蓝集合	语义完整性样本	最小DFA	时间复杂度:$O(\|\|S\|\|^3)$
		非语义完整性样本	DFA	空间复杂度:$O(\|\|S\|\|^2)$
Exbar算法[25]	APTA	任意样本	最小DFA	–
QSM算法[30]	PTA	特征样本	最小DFA	时间复杂度:$O((\|\|S^+\|\| + \|\|S^-\|\|) \cdot \|\|S^+\|\|^2)$
		非特征样本	DFA	
DeLeTe2算法[31]	PTMM	任意样本	RFSA	时间复杂度:$O(\|\|S\|\|^4)$
Grinchtein算法[32]	布尔表达式	任意样本	最小DFA	编码长度:$O(mn\log_2 n + n\|\|S^+\|\| \cdot \|\|S^-\|\|)$
dfasat算法[28]	布尔表达式	任意样本	最小DFA	编码长度:$O(n^3k + mn^2)$
BFS算法[33]	布尔表达式	任意样本	最小DFA	编码长度:$O(n^3 + n^2k^2)$
DFA-Inductor2[27]	等式及布尔表达式	任意样本	最小DFA	编码长度:$O(n^2k)$

注:$\|\|S^+\|\|$:正样本S^+的长度总和;$\|\|S^-\|\|$:负样本S^-的长度总和;$\|\|S\|\|$:样本S的长度总和;n:结果自动机的状态数目;m:初始前缀树接受器;k:字符集规模.算法效率中"–"表示算法复杂度未知.

状态合并算法具有成熟的理论基础,但学习对象只限于自动机,很难移植到其他类型语言模型.基于遗传编程的正则语言推断算法,也称为正则文法进化算法(regular grammatical evolution),为正负样本学习提供了一种新计算范型.文法进化算法具有较强的普适性,可推断自动机[34,35]、正则表达式[36]以及CFG(Context Free Grammar)等多种文法.

4.2 正样本学习

虽然充足的正负样本可以在理论上保证正则语言的极限可识认性,但负样本在实际应用中获取成本非常高昂.所以正样本学习一直是文法推断领域的研究热点.

目前,正样本学习技术主要有两种,特征式学习和启发式学习.特征式学习通过给学习任务附加限制条件保证学习过程符合极限识认模型.这类算法通常将目标语言限定到正则语言的某个子类,并证明该子类是基于正样本可极限识认的,因此特征式学习方法也称为受限式学习.目标语言的限制条件相当于为推理算法提供某种附加信息,如使用面向SORE的RWR算法相当于告知算法"样本数据来自一种SORE表达式".特征式学习的优点是具有严格的理论基础,可以事先估计推理结果的性质;缺点是应用场景较为苛刻.启发式学习不考虑语言的可识认性,根据启发式规则寻找符合应用要求的正则语言.其缺点是不能事先判定学习结果是否符合要求,可能导致过度泛化;优点是运行条件宽松,在工程实践中应用较为灵活.

4.2.1 受限式学习

20世纪80年代,Angluin首先发现正则语言的子类K-可逆语言是基于正样本可极限识认的,并提出ZR算法和k-RI算法分别用于推断0-可逆语言和K-可逆语言.该研究开启了受限式学习的先河.此后,一系列正则语言子类的正样本极限可识认性得到研究,包括TDRL语言[37]、K-TSSI语言[38]、LTLSS语言[39]、(k,h)-contextual语言[40]、SRL语言[41]、CRL语言[42]、UTR语言[43]、SL语言[44]等.这类研究的重点是对语言在正样本上极限可识认性的理论分析,提出的学习算法具有坚实的理论基础,但缺少实际应用场景.其原因是早期受限式学习算法为保证语言子类的正样本可学习性,往往将目标语言划分到一个较小的语言子类,很难适应实际应用中复杂的语言建模需求.因此,在很长一段时间里受限式学习的研究都停留在理论阶段.直到2010年,Bex发现SORE语言可用于定义XML文档的DTD模式,并从理论上证明了SORE语言、CHARE语言及k-ORE语言在正样本上的极限可识认性[45,46],提高了受限式学习的实用性.此后,以XML数据模式挖掘为应用背景的受限式学习成为正则语言推断的一个研究热点.

4.2.2 启发式学习

Feldman与Biermann在尼罗德等价关系的基础上定义了K尾状态等价关系,提出了K尾算法[47].K尾算法采用的启发式规则是"如果系统在某两个状态下的短期行为相同,那么它们的长期行为应该也相同",其中系统短期行为由长度小于或等于整数K的后缀序列集合定义,长期行为由长度大于整数K的后缀序列集合定义.

正则表达式提取算法通过分析样本的公共模式构造正则表达式.常用的分析手段包括字符序列前缀/后缀提取、频繁子序列提取、全局/局部对齐等操作.这类方法通常遵循"对齐部分可相互替换"的假设[48].例如,Min采用"先分组、再对齐"策略提出的正则表达式抽取算法[49]、Galassi采用局部对齐算法提出的带噪声正则表达式抽取算法[50].Bell实验室研发的XTRACT系统[51]采用MDL作为目标函数,将正则表达式学习问题转换为组合优化问题.由于正则表达式的搜索空间非常庞大,执行效率是这类算法应用的瓶颈.超过1000个字符序列就可能导致XTRACT系统崩溃.本课题组提出了编码成本感知的表达式构建算法,有效提高了正则表达式候选集的质量,在真实数据集的实验中表现出优于XTRACT系统的性能[52].

4.3 主动学习

主动学习也称在线学习,学习者主动选择样本向教师提问.Angluin率先提出了L*算法,利用多项式个成员问题和等价问题极限识认最小DFA[53].该算法包括自动机构造和自动机确认两个阶段,在自动机构造阶段利用成员问题获取DFA构建所需信息,在自动机确认阶段利用等价问题判断构造的自动机是否正确.大部分主动学习算法都遵循L*算法的框架.如表2所示,主动学习算法的发展总体呈现出下面两个趋势.

4.3.1 算法效率的提高

在数据结构方面,早期主动学习算法采用观察表(observation table)存储教师提供的自动机信息,描述字符序列前缀与后缀组成的所有字符序列,空间复杂度比较高.后来,Kearns提出了更加精简的辨别树(discrimination tree),只存储具有区分能力的后缀,不仅降低了算法空间复杂度,还提高了算法的执行效率[54].目前性能最优的TTT算法就是采用辨别树作为数据结构[55].

由于自动机的构造过程是反例驱动的,反例的处理策略对学习效率起着决定性作用.目前,主动学习算法采用的反例处理策略包括前缀策略和后缀策略两类.

(1)前缀策略将反例的前缀加入数据结构的前缀集合,使观察表进入不一致状态.

(2)后缀策略将反例的后缀加入数据结构的后缀

表2　主动学习算法列表

算法名称	语言模型	数据结构	反例处理	算法复杂度
L*算法[53]	DFA	OT表	增加反例的所有前缀到数据结构中	$N(M)=O(kmn^2)$ $N(E)=O(n)$ 空间复杂度:$O(kn(n+m))$
基于采样的L*算法[53]	DFA	OT表	增加反例的所有前缀到数据结构中	–
Rivest&Schapire算法[60]	DFA	OT表	只增加反例的一条后缀到数据结构中	$N(M)=O(kn^2+n\log_2 m)$ 空间复杂度:$O(kn^2+nm)$
Kearns&Vazirani算法[54]	DFA	DT树	只增加反例的一条前缀到数据结构中	空间复杂度:$O(kn+nm)$
TTT算法[55]	DFA	DT树	只增加反例的一条后缀到数据结构中	$N(M)=O(kn^2+n\log_2 m)$ $N(E)=n$ 空间复杂度:$O(kn)$
NL*算法[57]	RNFA	OT表	增加反例的所有后缀到数据结构中	$N(M)=O(mn^3)$ $N(E)=O(n^2)$
AL*算法[58]、UL*算法[58]	AFA、UFA	OT树	增加反例的部分后缀到数据结构中	$N(E)=O(l)$ $N(M)=O(mnl)$
AL**算法[59]	Residual AFA	OT表	增加反例的所有后缀到数据结构中	$N(E)=O(n)$ $N(M)=O(nl(1+k)m)$

注:$N(M)$:成员查询的次数;$N(E)$:等价查询的次数;k:字符集规模;m:最长反例的字符数量;n:最小DFA状态数;l:完整OT表的列索引,也就是能够接受目标语言的逆语言的最小DFA的状态数;算法效率中"–"表示算法复杂度未知.

集合,使观察表进入不封闭状态.

4.3.2　NFA学习

L*算法采用DFA描述目标语言,面对复杂应用问题时往往生成规模庞大的DFA. 众所周知,NFA(Nondeterministic Finite Automata)比DFA更加简洁,在描述相同语言时可用的最小NFA比最小DFA包含的状态更少,这种差距会随语言的复杂度增加呈指数级扩大趋势. 然而,NFA不具备L*算法所需的残余性质,所以很难将L*算法直接改造成NFA学习算法. 后来,Denis等提出的RNFA(Residual NFA)[56]是一类具有残余性质的NFA,开启了面向NFA主动学习的研究进程[57~59].

4.4　基于神经网络的学习算法

RNN(循环神经网络)是一种分析序列数据的重要神经网络,在自然语言学习方面表现突出. 将循环神经网络应用于形式化语言学习为正则语言学习提供新思路. 目前可用于学习自动机的循环神经网络主要包括一阶循环网络[61]、二阶循环网络[62]、LSTM(Long-Short Term Memory)和GRU(Gated Recurrent Unit). 其中LSTM和GRU在学习过程中收敛速度较快. 但是循环神经网络可解释性差,这在一定程度上限制了神经网络在形式化语言学习中的应用.

神经网络的规则抽取方法[63, 64]假设循环网络的状态能够模拟自动机的状态,通过将循环神经网络内部的状态空间离散化,抽取有限自动机的状态及转移关系. 循环神经网络的内部状态空间是连续的,将其离散化是抽取有限自动机的关键. 目前的状态空间离散化方法主要有网格法[65]和聚类法[66]. 网格法的缺点是容易造成状态爆炸,很难扩展到规模网络. 聚类方法比网格法更容易扩展到大规模网络,但对超参数值的设定比较敏感,例如应用k-means算法聚类时,如果k值设定得太小,会对抽取结果产生较大影响.

基于谱学习算法抽取方法利用谱学习算法从RNN直接抽取带权重自动机[67]. 其优点是适用范围广,可以将任何处理序列数据的神经网络模型作为抽取对象;缺点是抽取结果依赖于基础矩阵的规模,规模越大抽取结果越好,需要的算力也越高.

此外,Gail将RNN与主动学习经典算法L*相结合,研究以RNN模型为教师的主动学习算法[68]. 由于L*算法的多项式时间复杂度和对噪声的敏感性,该方法在处理行为复杂的RNN网络时性能较差.

将RNN转换为DFA对抽取方法具有一定依赖性. 对相同的RNN采用不同的抽取方法,结果自动机可能属于完全不同的语言类.

4.5　软件逆向工程中学习算法推荐

本节从多个技术维度对比和分析了正则语言学习的主流算法,对具体应用场景的算法选择具有一定的指导意义. 下面以软件工程领域的软件状态模型逆向生成问题为例,提出自动机学习算法的应用条件和算法推荐列表.

如表3所示,若目标系统允许进行试探性交互操作,可优先考虑主动学习模型. 当等价性问题可被直接回答时,选择高效的TTT算法;否则选择基于采样的L*

表3 在软件模型推断中学习算法推荐列表

序号	算法	应用条件	
1	TTT算法	目标系统允许试探性操作	能回答等价性问题
2	采样L*算法		允许近似推断
3	dfasat算法	目标系统不允许试探性操作	样本规模较小
4	DFA-Inductor2算法		
5	Exbar算法		
6	Blue-Fringe算法		样本规模较大
7	ESDM算法		
8	RPNI算法		
9	增量式SAT推断算法[69]		样本数据较长

算法或其他主动PAC学习算法.

主动学习算法可以利用软件自动测试技术实现成员问题的自动回答,但是所需的问题数量通常比较大,有时可达到几千万条. 中型软件系统的模型生成可能要花几个小时,甚至几天的时间. 如何针对具体应用减少问题数量,是决定主动学习算法能否成功应用于软件模型推断的重要问题.

若目标系统用于支撑重要生产经营活动,而且该系统也不能复制,采用主动学习算法可能影响生产经营活动的正常进行. 此时应考虑被动学习算法,因为分析系统日志不会对系统本身造成影响. 当日志规模较小时,选用精确推断算法,如dfasat算法、DFA-Inductor2算法、Exbar算法等;当日志规模较大时,选用高效的Blue-Fringe算法、EDSM算法、RPNI算法;当日志数据普遍较长时,传统学习算法的复杂度增长较快,可选择增量式SAT推断算法[69].

在应用被动学习算法生成软件系统的状态模型时经常遇到以下问题:

(1)正负样本不均衡问题. 由于系统故障或异常情况不经常出现,所以真实日志中负样本数量较少,造成正负样本不均衡问题. 负样本限定了被动学习的泛化程度,样本不均衡容易造成过度泛化.

(2)理论正确的推断不符合实际情况. 经典算法执行的状态合并可能出现与实际不符的推断结果. 例如某软件系统的日志中包含 $abccc$ 和 $accc$ 两条轨迹,采用dfasat算法学习时,会认为前缀 a 和 ab 的后续行为是类似的,并合并相应状态,产生自身转移环. 虽然在理论上该合并是正确的,但是它允许事件 b 出现无限多次,与实际情况相悖[70].

(3)缺失异常转移发生频率. 软件工程师在软件系统逆向工程中最感兴趣的信息是系统异常情况. 被动学习算法可以帮助工程师识别异常轨迹,但不能提供异常出现的频率.

4.6 小结

虽然学习算法各自采用不同的形式描述正则语言,但在学习机制上存在相通之处. 例如,自动机学习算法采用的典型前缀树自动机的构造过程与正则表达式提取算法采用的前缀提取运算过程类似. 自动机的状态合并本质上就是寻找公共后缀. 产生式集合中非终结符合并与自动机的状态合并类似. 然而,由于采用的语言模型不同,每类算法呈现出不同的特点. 面向产生式的学习算法容易扩展到CFG学习,而前缀自动机无法定义CFG,所以自动机学习算法很难扩展到CFG学习. 另外,这几类算法出现的时间和发展趋势也不相同. 如图1所示,产生式推断算法和自动机学习算法出现得比较早,而且自动机学习算法在正则语言推断的研究中占有很大比例. 从20世纪90年代开始正则表达式提取算法开始流行,逐渐成为正则语言推断研究的一个重要方向.

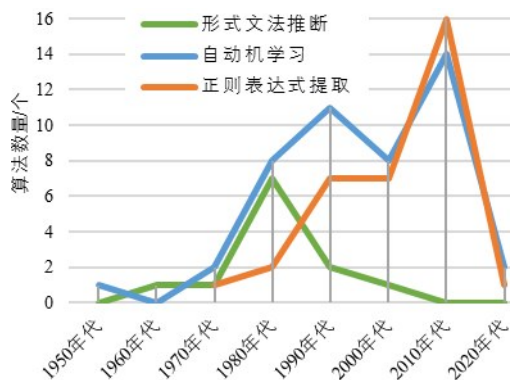

图1 正则语言推断算法发展趋势

5 泛化效应分析

学习算法生成的语言超出样本的现象称为泛化. 泛化是正则语言推断过程中必然存在的效应,如果生成的正则语言只能匹配样本数据,那么该语言只是对样本数据进行信息压缩,而没有实现真正的学习. 但是过度泛化会导致正则语言包含错误数据. 正则语言推断主要包括以下三种泛化效应.

5.1 循环次数泛化

循环次数泛化指正则语言允许子序列重复出现的

次数超出其在样本中出现次数的现象. 可连续出现的子序列称为循环体[44]. 简单的循环体只包含单个字符,复杂的循环体允许包括多个字符[45,51,71]. 学习算法引入循环次数泛化的条件会存在很大差异. XTRACT通过设定阈值r定义泛化条件,若某段子序列在某个样本中连续出现次数超过r,则该子序列可能出现任意多次. 文献[49]采用先分组后合并的方式定义循环泛化规则. 例如,若abc和bc为两组相邻子序列,则合并为$(a)(bc)\{1,2\}$. 循环体bc的循环次数估计值为最小1次,最大2次. 文献[44]通过分析循环次数组成数列的规律,对循环次数进行更为精确的逼近.

上述算法通过识别样本中连续出现的子序列作为循环体的识别方法,主要在单个样例推理阶段引入循环次数泛化. 在整个样本集合学习阶段也可能引入循环次数泛化. 例如K值设为2的K尾算法对正样本{1001, 10}的学习结果为10(01)*. 虽然K尾算法没有明确的循环次数泛化规则,但K值较小容易产生循环泛化的效应. XTRACT采用窗口扫描算法实现包括不同字符的复杂循环体识别[51]. Feldman算法通过合并残余非终结符,引入循环次数泛化[72]. Feldman改进算法针对Feldman算法容易过拟合的问题引入更为宽松的非终结符合并条件[73],使得相同的样本更容易训练出带克林尼闭包的文法.

我们对正负样本学习中RPNI、EDSM、Blue-Fringe等算法进行分析,发现状态合并类算法在推断得到最小自动机的过程中表现出的循环次数泛化与上述正样本学习算法的泛化规则不同. 例如,当$S_+=\{a\}$且$S_-=\{a^4\}$时,a的所有奇数次循环序列都被自动机接受;当$S_+=\{a\}$且$S_-=\{a^5\}$时,a的$\{1+3\times n|n\in\mathbb{N}\}$次循环序列都被自动机接受,其中$\mathbb{N}$代表自然数集合;当提供的字符序列样本为$S_+=\{a,aa\}$且$S_-=\{a^5\}$时,结果自动机接受字符$a$的$\{1,2\times n|n\in\mathbb{N}\}$次循环序列. 状态合并类算法泛化效果是由最小自动机的状态数决定的. 根据Gold理论,给定样本判断最小自动机规模是NP完全问题,因此状态合并算法中的循环泛化规则无法用数学解析式描述.

5.2　可替换项泛化

可替换项泛化指正则语言通过给某段子序列定义多个可替换项的方式引入的泛化现象. 可替换项泛化的正则表达式通常包含选择运算,但选择运算本身不代表正则表达式一定对样本进行了泛化. 例如基于形式微商的学习算法通常会产生带选择运算的正则表达式,但形式微商并没有对样本进行泛化[74]. 学习算法引入可替换项泛化的条件各不相同. K尾算法可以理解为按前缀对字符序列分组,若两组字符序列的长度小于等于K的后缀子序列集相同,则这两组字符序列的前缀可互换. 若这两组序列存在长度大于K的后缀子序列则发生替换泛化[75]. 状态合并类自动机学习算法也会产生替换泛化,泛化条件随着状态等价性定义的不同而变化. 所有基于对齐机制的学习算法都是允许对齐部分相互替换的[48,51,76],也就是样本对齐部分的不同子序列允许任意组合构成新字符序列.

5.3　区域性泛化

区域性泛化规则指忽略某段子序列字符间的顺序,允许字符在该区域以任意顺序出现任意多次的现象. 区域性泛化规则基于以下直觉:如果字符序列s_i中字符$a_1,a_2,\cdots,a_m$近距离多次重复出现,则认为字符序列s_i很可能源自正则表达式$(a_1|a_2|\cdots|a_m)^*$[51]. XTRACT将输入样例s分割成许多短小的子序列$s_1,s_2,\cdots,s_n$,使得s_i中任何字符a,在其他的子序列s_j中都不会出现. 即使出现相同字符,它在序列S中的位置离s_i中该字符的位置也大于局部阈值d. 然后,将每个s_i替换成形如$(a_1|a_2|\cdots|a_m)^*$的正则表达式. 其中,$a_1,a_2,\cdots,a_m$是在s_i中出现的不重复的字符. 举例,给定一个输入序列$abcbac$,若参数$d=2$,则首先将s分隔为a、bcb、a和c,最终生成正则表达式$a(b|c)^*ac$. 若参数$d=3$,则首先将s分隔为a和$bcbac$. 最终生成正则表达式$a(a|b|c)^*$.

6　总结与展望

正则语言推断研究的理论与方法与机器学习具有一定相似性,但是二者也存在明显差别. 例如,机器学习主要解决分类和预测问题,而正则语言推断的目标是语言的文法结构;机器学习主要处理带标注样本数据,而正则语言推断主要采用无标注的字符序列数据作为样本;机器学习以随机模型为主,正则语言推断则起源于非随机语言模型. 因此,正则语言推断作为一个独立的研究方向,在大数据时代的背景下,其独特的理论和技术受到越来越多的关注,同时也存在如下问题亟待解决.

(1)序列样本的噪声处理. 应用环境的复杂性极易产生噪声数据. 噪声数据的模式一般与正常数据有显著差别,如不进行处理,会导致学习得到的正则表达式过度泛化. 为获得更加准确有效的正则语言模型,必须对样本数据进行数据清洗.

(2)等价问题的回答. 等价问题是主动学习算法是否终止的判定条件,但工程实践中往往很难找到等价问题的准确答案,这也导致早期的主动学习研究主要停留在学术层面. 根据应用问题的特点,研究等价问题的近似解答技术对提高主动学习的实用性具有重要意义. 随机游走、主被动结合等策略为高效可用的近似回答技术研究提供了新的思路.

(3)增量式正则语言推断算法. 由于正则语言推断

的复杂性,学习算法的时间复杂度比较高.如果每次有新样本数据加入都要对整个数据集重新学习,必然造成算力的巨大浪费.目前针对增量式学习算法的研究还比较少,有必要研究正则语言的增量学习方法,以减少重复计算,更好地满足实际应用的需求.

(4)与深度学习技术结合.深度学习技术在自然语言处理领域已取得令人瞩目的成绩,但在形式语言推断领域的研究还比较少.将深度学习得到的网络模型作为主动学习的教师角色,回答学习算法提出的问题,可以为正则语言推断提供新的思路.将主动学习算法与深度学习技术结合可以将深度神经网络这种黑盒子模型转变为可解释的正则表达式.即可以有效解决主动学习缺少教师系统的问题,又可以弥补神经网络模型不可解释的缺陷.

致谢 感谢Sicco Verwer先生给本文提出的参考意见.

参考文献

[1] BEX G J, NEVEN F, SCHWENTICK T, et al. Inference of concise regular expressions and DTDs[J]. ACM Trans Database Syst, 2010, 35(2): 1 – 47.

[2] XIE Y, YU F, ACHAN K, et al. Spamming botnets: signatures and characteristics[J]. SIGCOMM Comput Commun Rev, 2008, 38(4): 171 – 182.

[3] CHIVILIKHIN D, PATIL S, CHUKHAREV K, et al. Automatic state machine reconstruction from legacy programmable logic controller using data collection and sat solver[J]. IEEE Transactions on Industrial Informatics, 2020, 16(12): 7821 – 7831.

[4] CHEN Y-F, HSIEH C, LENG L O R, et al. PAC learning-based verification and model synthesis[A]. DILLON L. Proceedings of the 38th International Conference on Software Engineering[C]. New York, United States: ACM, 2016. 714 – 724.

[5] GOLD E M. Language identification in the limit[J]. Information and Control, 1967, 10(5): 447 – 474.

[6] ANGLUIN D. Inductive inference of formal languages from positive data[J]. Information and Control, 1980, 45(2): 117 – 135.

[7] VALIANT L G, VALIANT L. A theory of the learnable[J]. Communications of ACM, 1984, 27(11): 1134 – 1142.

[8] ISHIGAMI Y, TANI S I. VC-dimensions of finite automata and commutative finite automata with k letters and n states[J]. Discrete Applied Mathematics, 1997, 74(3): 229 – 240.

[9] GOLD E M. Complexity of automation identification and control[J]. Information and Control, 1978, 37(4): 302 – 320.

[10] ANGLUIN D. On the complexity of minimum inference of regular sets[J]. Information and Control, 1978, 39(3): 337 – 350.

[11] ANGLUIN D. Queries and concept learning[J]. Machine Learning, 1988, 2(3): 319 – 342.

[12] FREYDENBERGER D, REIDENBACH D. Inferring descriptive generalizations of formal languages[J]. Journal of Computer and System Sciences, 2012, 79(5): 622 – 638.

[13] WALKINSHAW N, BOGDANOV K, DAMAS C, et al. A framework for the competitive evaluation of model inference techniques[A]. GROZ R, LI K. Proceedings of the First International Workshop on Model Inference In Testing[C]. New York, United States: ACM, 2010. 1 – 9.

[14] GRACHEV P, BEZBORODOV R, SMETANNIKOV I, et al. Exploring the relationship between the structural and the actual similarities of automata[A]. Proceedings of the 3rd International Conference on Machine Learning and Soft Computing[C]. New York, United States: ACM, 2019. 81 – 86.

[15] CHAN M GC, AND R. RASTOGI. RE-tree: an efficient index structure for regular expressions[J]. the VLDB Journal, 2003, 12(2): 102 – 118.

[16] GRUNWALD P D. The Minimum Description Length Principle[M]. Dordrecht, Netherlands: Springer, 2007.

[17] RISSANEN J. Modeling by shortest data description[J]. Automatica, 1978, 14(5): 465 – 471.

[18] GOLD E M. Complexity of automaton identification from given data[J]. Information and Control, 1978, 37(3): 302 – 320.

[19] TRAKHTENBROT B A, BARZDIN Y M. Finite Automata: Behavior and Synthesis[M]. North-Holland, Netherlands: Elsevier, 1973.

[20] ZQUEZ DE PARGA MV, GARC A P, L PEZ D. Minimal consistent DFA revisited[J]. Theoretical Computer Science, 2016, 647: 43 – 49.

[21] ZHANG C. Minimal consistent DFA from sample strings[J]. Acta Informatica, 2020, 57(3): 657 – 670.

[22] ONCINA J, GARCIA P. Inferring regular languages in polynomial update time[A]. SANFELIU A, BLANCA N P D L, VIDAL E. Pattern Recognition and Image Analysis[C]. Singapore: World Scientific, 1992. 49

– 61.

[23] LANG K J. Random DFA's can be approximately learned from sparse uniform examples[A]. HAUSSLER D. Proceedings of the Fifth Annual Workshop on Computational Learning Theory [C]. New York, United States: ACM, 1992. 45 – 52.

[24] LANG K J, PEARLMUTTER B A, PRICE R A. Results of the abbadingo one DFA learning competition and a new evidence-driven state merging algorithm[A]. HONAVAR V G, SLUTZKI G. Proceedings of the 4th International Colloquium on Grammatical Inference[C]. Berlin, German: Springer, 1998. 1 – 12.

[25] LANG K. Faster Algorithms for Finding Minimal Consistent DFAs[R]. Kovilpatti, India: National Engineering College, 1999.

[26] OLIVEIRA A L, SILVA J P M. Efficient algorithms for the inference of minimum size DFAs[J]. Machine Learning, 2001, 44(1-2): 93 – 119.

[27] ZAKIRZYANOV I, MORGADO A, IGNATIEV A, et al. Efficient symmetry breaking for SAT-based minimum DFA inference[A]. MART N-VIDE C, OKHOTIN A, SHAPIRA D. Language and Automata Theory and Applications [C]. Cham, Switzerland: Springer, 2019. 159 – 173.

[28] HEULE M J H, VERWER S. Exact DFA identification using SAT solvers[A]. SEMPERE J M, GARC A P. Grammatical Inference: Theoretical Results and Applications[C]. Berlin, German: Springer, 2010. 66 – 79.

[29] SMETSERS R, FITERĂU-BROŞTEAN P, VAANDRAGER F. Model learning as a satisfiability modulo theories problem[A]. KLEIN S T, MART N-VIDE C, SHAPIRA D. Language and Automata Theory and Applications [C]. Cham, Switzerland: Springer, 2018. 182 – 194.

[30] DUPONT P, LAMBEAU B, DAMAS C, et al. The QSM algorithm and its application to software behavior model induction [J]. Applied Artificial Intelligence, 2008, 22(1-2): 77 – 115.

[31] DENIS F, LEMAY A, TERLUTTE A. Learning regular languages using non deterministic finite automata [A]. Proceedings of the 5th International Colloquium on Grammatical Inference: Algorithms and Applications[C]. Berlin, Germany: Springer, 39 – 50.

[32] GRINCHTEIN O, LEUCKER M, PITERMAN N. Inferring network invariants automatically [A]. FURBACH U, SHANKAR N. Automated Reasoning [C]. Berlin, Germany: Springer, 2006. 483 – 497.

[33] ZAKIRZYANOV I, SHALYTO A, ULYANTSEV V. Finding all minimum-size dfa consistent with given examples: SAT-based approach [A]. CERONE A, ROVERI M. Software Engineering and Formal Methods [C]. Cham, Switzerland: Springer, 2018. 117 – 131.

[34] GRACHEV P. Reputational genetic model for regular inference[A]. Proceedings of the 3rd International Conference on Advances in Image Processing[C]. New York, United States: ACM, 2019. 185 – 189.

[35] GRACHEV P. Grammar inference with multiparameter genetic model [A]. Proceedings of the 3rd International Conference on Advances in Image Processing [C]. New York, United States: ACM, 2019. 160 – 164.

[36] LIU J, BAI R, LU Z, et al. Data-driven regular expressions evolution for medical text classification using genetic programming[A]. 2020 IEEE Congress on Evolutionary Computation [C]. Piscataway, United States: IEEE, 2020. 1 – 8.

[37] RADHAKRISHNAN V, NAGARAJA G. Inference of regular grammars via skeletons[J]. IEEE Transactions on Systems, Man, and Cybernetics, 1987, 17(6): 982 – 992.

[38] GARCIA P, VIDAL E. Inference of k-testable languges in the strict sense and application to syntactic pattern recognition[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1990, 12(9): 920 – 925.

[39] GARCIA P, VIDAL E, ONCINA J. Learning locally testable languages in the strict sense[A]. ARIKAWA S, GOTO S, OHSUGA S, et al. Proceedings of ALT'90 [C]. Tokyo, Japan: JSAI, 1990. 325 – 338.

[40] AHONEN H. Generating Grammars for Structured Documents Using Grammatical Inference Methods[D]. Helsinki, Finland: Univeristy of Helsinki, 1996.

[41] TANIDA N, T.YOKOMORI. Polynomial time identification of strictly regular languages in the limit[J]. IEICE Transaction on Information and Systems, 1992, E75−D (6): 125 – 132.

[42] EMERALD J D, SUBRAMANIAN K G, THOMAS D G. Learning code regular and code linear languages[A]. MICLET L, HIGUERA C D L. Grammatical Interference: Learning Syntax from Sentences[C]. Berlin, Germany: Springer, 1996. 211 – 221.

[43] MAKINEN E. Inferring uniquely terminating regular languages from positive data[J]. Information Processing Letters, 1997, 62(2): 57 – 60.

[44] FERNAU H. Algorithms for learning regular expressions

from positive data [J]. Information and Computation, 2009, 207(4): 521 – 541.

[45] BEX G J, GELADE W, NEVEN F, et al. Learning deterministic regular expressions for the inference of schemas from XML data [J]. ACM Transactions on The Web, 2010, 4(4): 1 – 32.

[46] 冯晓强, 郑黎晓, 陈海明. 一类受限正则表达式的推断算法[J]. 计算机科学, 2014, 41(4): 178 – 183.
FENG Xiao-qiang, ZHENG Li-xiao, CHEN Hai-ming. Inferring algorithm for a subclass of restricted regular expressions [J]. Computer Science, 2014, 41(4): 178 – 183. (in Chinese)

[47] BIERMANN A W, FELDMAN J A. On the synthesis of finite-state machines from samples of their behavior [J]. IEEE Transactions on Computers, 1972, C-21(6): 592 – 597.

[48] ZAANEN M M V. Bootstrapping Structure into Language: Alignment-Based Learning [D]. Mahikeng, South Africa: North-West University, 2001.

[49] MIN J-K, J-YAHN, C-WCHUNG. Efficient extraction of schemas for XML documents [J]. Information Processing Letters, 2003, 85(1): 7 – 12.

[50] GALASSI U, GIORDANA A. Learning regular expressions from noisy sequences [A]. ZUCKER J-D, SAITTA L. Abstraction, Reformulation and Approximation [C]. Berlin, Germany: Springer, 2005. 92 – 106.

[51] GAROFALAKIS M, GIONIS A, RASTOGI R, et al. XTRACT: learning document type descriptors from XML document collections [J]. Data Mining and Knowledge Discovery, 2003, 7(1): 23 – 56.

[52] GAO J, ZHANG Y. Regular expression learning from positive examples based on integer programming [J]. International Journal of Software Engineering and Knowledge Engineering, 2020, 30(10): 1 – 37.

[53] ANGLUIN D. Learning regular sets from queries and counterexamples [J]. Information and Control, 1988, 75(2): 87 – 106.

[54] KEARNS M J, VAZIRANI U. An Introduction to Computational Learning Theory [M]. Cambridge, United States: MIT, 1994.

[55] ISBERNER M, HOWAR F, STEFFEN B. The TTT algorithm: a redundancy-free approach to active automata learning [A]. BONAKDARPOUR B, SMOLKA S A. Runtime Verification [C]. Cham, Switzerland: Springer, 2014. 307 – 322.

[56] DENIS F, LEMAY A, TERLUTTE A. Residual finite state automata [A]. FERREIRA A, REICHEL H. Proceedings of the 18th Annual Symposium on Theoretical Aspects of Computer Science [C]. Berlin, Germany: Springer, 2001. 144 – 157.

[57] BOLLIG B, HABERMEHL P, KERN C, et al. Angluin-style learning of NFA [A]. KITANO H. Proceedings of the 21st International Joint Conference on Artifical intelligence [C]. San Francisco, United States: Morgan Kaufmann. 1004 – 1009.

[58] ANGLUIN D, EISENSTAT S, FISMAN D. Learning regular languages via alternating automata [A]. YANG Q, WOOLDRIDGE M. Proceedings of the 24th International Conference on Artificial Intelligence [C]. Palo Alto, United States: AAAI Press, 2015. 3308 – 3314.

[59] BERNDT S, LIŚKIEWICZ M, LUTTER M, et al. Learning residual alternating automata [A]. Proceedings of the 31st AAAI Conference on Artificial Intelligence [C]. Palo Alto, United States: AAAI Press, 2017. 1749 – 1755.

[60] RIVEST R L, SCHAPIRE R E. Inference of finite futomata using homing sequences [A]. JOHNSON D S. Proceedings of the 21st Annual ACM Symposium on Theory of Computing [C]. New York, United States: ACM, 1989. 411 – 420.

[61] ELMAN J L. Finding structure in time [J]. Cognitive Science, 1990, 14(2): 179 – 211.

[62] WATROUS R L, KUHN G M. Induction of finite-state automata using second-order recurrent networks [J]. NIPS, 1991, 4(3): 309 – 316.

[63] GRACHEV P, LOBANOV I, SMETANNIKOV I, et al. Neural network for synthesizing deterministic finite automata [A]. KLIMOVA A, BILYATDINOVA A, KORTELAINEN J, et al. Proceedings of the 6th International Young Scientist Conference on Computational Science [C]. North-Holland, Netherlands: Elsevier, 2017. 73 – 82.

[64] MICHALENKO J J, SHAH A, VERMA A, et al. Representing formal languages: a comparison between finite automata and recurrent neural networks [EB/OL]. https://openreview. net/pdf? id=H1zeHnA9KX, 2019-05-06/2020-04-27.

[65] OMLIN C W A GC. L. Extraction of rules from discrete-time recurrent neural networks [J]. Neural Networks, 1996, 9(1): 41 – 52.

[66] CECHIN A L, SIMON D. R. P., AND STERTZ, K. State automata extraction from recurrent neural nets using

k-means and fuzzy clustering[A]. Proceedings of the XXIII International Conference of the Chilean Computer Science Society[C]. Washington, United States: IEEE Computer Society, 2003. 73 – 78.

[67] AYACHE S, EYRAUD R, GOUDIAN N. Explaining black boxes on sequential data using weighted automata [A]. UNOLD O, DYRKA W, WIECZOREK W. Proceedings of Machine Learning Research 93[C]. Cambridge, United States: MIT, 2018. 81 – 103.

[68] WEISS G, GOLDBERG Y, YAHAV E. Extracting automata from recurrent neural networks using queries and counterexamples[A]. JENNIFER D, ANDREAS K. Proceedings of the 35th International Conference on Machine Learning[C]. Cambridge, United States: JMLR, 2018. 5247 – 5256.

[69] AVELLANEDA F, PETRENKO A. FSM inference from long traces[A]. HAVELUND K, PELESKA J, ROSCOE B, et al. Formal Methods[C]. Cham, Switzerland: Springer, 2018. 93 – 109.

[70] WIEMAN R, ANICHE M, LOBBEZOO W, et al. An experience report on applying passive learning in a large-scale payment company[A]. Proceedings of IEEE International Conference on Software Maintenance and Evolution[C]. Piscataway, United States: IEEE, 2017. 564 – 573.

[71] BRĀZMA A. Efficient identification of regular expressions from representative examples[A]. PITT L. Proceedings of the 6th Annual Conference on Computational Learning Theory[C]. New York, United States: ACM, 1993. 236 – 242.

[72] FELDMAN J A, GIPS J, HORNING J J, et al. Grammatical Complexity and Inference[R]. Stanford, United States: Stanford University, 1969.

[73] ITOGA S Y. A new heuristic for inferring regular grammars[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1981, PAMI-3(2): 191 – 197.

[74] FU K-S, BOOTH T L. Grammatical inference: introduction and survey-part I[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986, PAMI-8(3): 343 – 359.

[75] GGEMANN-KLEIN ABR. Regular expressions into finite automata[J]. Theoretical Computer Science, 1993, 120(2): 197 – 213.

[76] RULOT H, VIDAL E. Modelling (Sub) String Length based Constraints Through a Grammatical Inference Method[A]. DEVIJVER P A, KITTLER J. Pattern Recognition Theory and Applications[C]. Berlin, Heidelberg: Springer, 1987. 451 – 459.

作者简介

高俊涛　男,1979年出生于黑龙江哈尔滨.现为东北石油大学副教授、硕士生导师.研究方向包括软件工程、过程建模、自动机学习等.
E-mail:gjt@nepu.edu.cn

徐光会　女,1995年出生于山东济宁.硕士研究生.主要研究方向为自动机学习.
E-mail:xuguanghui6688@163.com

王　梅　女,1976年出生于河北安国.现为东北石油大学教授,研究方向包括机器学习、模型选择和核方法.
E-mail:wangmei@nepu.edu.cn

刘　聪　男,1990年出生于山东淄博.现为山东理工大学教授,研究方向包括业务过程管理、过程挖掘、数据分析、Petri网理论与应用.
E-mail:liucongchina@sdust.edu.cn