汉语复句关系词自动标识中规则引擎的研究

杨进才[1] 谢 芳[2] 胡金柱[1]
(华中师范大学计算机学院 武汉 430079)[1] (湖北工业大学计算机学院 武汉 430068)[2]

摘 要 近年来规则引擎的研究取得了丰硕的成果,将其成果应用到各个方面为汉语复句处理带来了全新的思路与途径。将规则引擎用于复句关系词自动标识中,设计了规则引擎的结构,提出了关系搭配集的模式匹配策略、消除冲突规则的"消除包含最大化策略"以及最终结果集的"正覆盖"策略。在规则引擎中,3种策略的应用提高了复句关系词识别的效率与准确率。

关键词 自动标识,规则引擎,模式匹配,规则冲突
中图法分类号 TP391 **文献标识码** A

Research on Rule Engine for Automatic Identification of Relational Words in Chinese Complex Sentences

YANG Jin-cai[1] XIE Fang[2] HU Jin-zhu[1]
(School of Computer Science, Central China Normal University, Wuhan 430079, China)[1]
(School of Computer Science, Hubei University of Technology, Wuhan 430068, China)[2]

Abstract In recent years, the study of the rules engine has achieved fruitful results. It brings new ways and ideas to the processing of Chinese complex sentences. In this paper, we designed a rule engine for automatic identification of relational words in Chinese complex sentences. A pattern-matching strategy to select relation words set, a'eliminate contains maximizing strategy' to eliminate the conflict rules, and a 'front cover strategy' to select final result were designed. By using the three strategies in rule engine, the efficiency and accuracy of identification of relational words in Chinese complex sentences are improved.

Keywords Automatic identification, Rule engine, Pattern matching, Rule conflicts

1 引言

汉语中多数句子是复句,它表达的语义信息丰富而复杂,承载的信息量远大于单句[1,2]。因而复句的研究有更重要的价值。

在汉语语言学界,对复句的研究系统而深入,成果很多。而从中文信息处理角度对复句和复句关系词自动识别的研究成果相对较少[3,4]。较为系统地对复句关系词进行自动标识的研究有:李艳翠等利用清华树库,从中标识复句关系词并标注复句的类别,但由于依靠已有的树库,关系词判别的准确率以树库的正确标识为基础,如果树库的分词有误,关系词无从识别;胡金柱从关系词出发,对汉语复句进行了系列的研究,并建立了"基于关系词自动识别的规则库",在此基础上进行关系词的自动标识。基于规则的标识方法的不足在于规则库中规则有限,对关系词的使用情形无法完全概括,规则的匹配以及匹配中冲突没有得到很好的处理。上述研究的共同不足在于追求关系词识别的正确率时忽视了系统的效率,研究的系统性有待提高。

复句关系词及搭配使用非常灵活、组合形式多样,给规则的匹配带来了困难。近年来中文智能搜索引擎技术的广泛运用大大提高了互联网检索的准确率与效率。本文将探讨汉语复句关系词自动标识中规则引擎的设计与实现,通过规则引擎,提高汉语复句关系词识别的准确率与效率。

2 规则引擎与匹配算法

2.1 规则引擎的结构

规则引擎作为复句关系词自动标识的核心,包含4大模块:模式匹配器、冲突消解器、结果评估器以及执行引擎,结构图如图1所示。

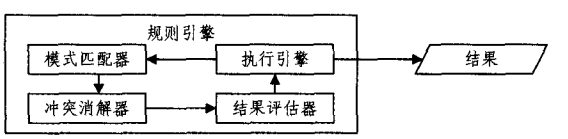

图1 规则引擎的结构

规则引擎在复句中的执行过程为:首先,复句特征分析器获取关系词序列传输给模式匹配器,经模式匹配算法得到关系搭配集;其次,关系搭配集传输给冲突消解器,判断关系搭配集中每个元素对应规则库中的规则是否存在冲突,若存在冲突则进行冲突消解,得到无冲突规则集;接着,无冲突规则集传输给结果评估器,判断无冲突规则集中的每条规则,若规

本文受国家教育部人文社科基金(13YJAZH117),国家社科基金(11BYY052)资助。
杨进才(1967—),男,教授,硕士生导师,主要研究领域为现代信息系统、中文信息处理;谢 芳(1981—),女,博士生,主要研究领域为中文信息处理、软件工程;胡金柱(1947—),男,教授,博士生导师,主要研究领域为中文信息处理、软件工程。

则约束条件满足，则规则保留，否则丢弃；最后，若存在多条不同搭配的规则，按照规则表的优先顺序决定唯一搭配方案，最先调用的规则作为最终的解决方案，最终解决方案存在多条规则，采用正覆盖策略对复句的关系标记判定赋值。

2.2 模式匹配算法

模式匹配器的核心是模式匹配算法，其中 RETE 算法是最经典、效率很高的一种多模式/多对象匹配算法。RETE 算法分为两部分：规则编译和运行时执行。编译算法描述规则如何产生一个有效的辨识网络，其核心思想就是通过数据在网络中的传播来过滤数据。运行执行过程为：当一个应用调用一个对象后，规则引擎就将数据传递给根节点处，并向下传递；当对象与一个节点条件匹配时，节点就将它记录到相应的内存中。此种方法虽然获得了更快的执行速度，但是也造成大量内存的消耗。特别是关系词自动识别中复句中存在大量的关系标记，关系标记搭配方案的多样性和规则的多样性，使得 RETE 算法并不适合复句关系词自动标识中规则引擎中的模式匹配。本文结合 RETE 网络结构，提出适合该系统中规则引擎的模式匹配策略。

2.2.1 算法思想

例如：关系词“不仅/而且/还”，形式化表示为 keymarks =“A/C/E”(规则中用“/”来分隔连用的关系词)。一个复句中的关系标记为：A1-B-A2-C-D-E(A1 和 A2 后缀的数字 1 或 2 表示复句中该关系标记的重复分词数)。匹配过程如图 2 所示。

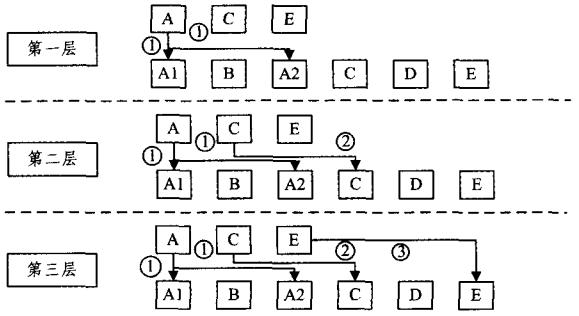

图 2 匹配过程示例

对规则库中的一条规则的 keymarks=“A/C/E”，将 keymarks 中的每个分词依次与关系标记序列中的每个关系标记进行匹配，如果匹配成功，将匹配成功的关系标记进行存储；直到所有的关系标记都经过了匹配验证。

通过模式匹配算法，最终确定 2 个关系标记序列即“A1/C/E”和“A2/C/E”；得到了关系标记序列集后，把每一个关系标记序列传送给规则解析器，进行规则解析。规则解析器的主要功能就是把关系标记序列与规则库中的规则进行匹配，根据判断约束条件的匹配与否选择正确的关系标记序列，也称关系词搭配集。

2.2.2 算法描述

基于 RETE 网络的关系标记匹配算法如下：

Step1 关系标记序列传递给 RootNode 结点(简称根节点)，关系标记序列分别存储在下一结点 TyepeNode(简称 TN，分别以 TN1、TN2 等表示不同结点)。

Step2 将 TN1(A1)与其余关系标记进行搭配，与规则表中的规则匹配，如果 A1 与 B 连用，则存储 B，否则丢弃 B，继续与下一关系标记搭配然后再与规则匹配；同理，如果 A1 与 A2 连用，则存储 A2，否则丢弃 A2，直到 A1 与其他关系标记匹配完毕。

Step3 如果 A1 与 C 关系连用，则对 C 与其后的关系标记进行搭配，再与规则表中的规则匹配，如果 C 与 D 连用，或者 A1/C/D 连用，则存储 D，否则丢弃 D，继续与下一关系标记搭配然后再与规则匹配。

Step4 当关系标记序列都进行了搭配再匹配之后，最终得到一个关系标记搭配集。

3 消除包含最大化策略

规则冲突是指在同一个标记序列中存在多条规则，而且这些规则的条件存在包含现象，而规则的结果存在不包含现象。当一个关系词触发多条规则时，选取最优规则非常重要。

一条规则包含约束条件部分和结果部分。现设 R_m 和 R_n 是某个标记序列的两条规则，而且 $R_m=C_m+A_m$，$R_n=C_n+A_n$。其中 C_m 表示规则 R_m 的约束条件部分，即前件，A_m 表示规则 R_m 的结果部分，即后件。规则冲突表明：$C_m \subset C_n \Rightarrow A_m \not\subset A_n$。其中，$C_m=\bigcap_{k=1}^{i} c_{mk}$，$A_m=\bigcap_{k=1}^{j} a_{mk}$，$c_{mk}$ 为条件项，a_{mk} 为结果处理项。例如在表 2 中，规则 ID=124，可知 c_{11} =“D(首先，然后)=1”，c_{12} =“D(然后，最后)>3”，a_{11} =“ R(首先)=true”，a_{12} =“R(然后)=true”，a_{13} =“R(最后)=false”。

例如，当关系标记“首先/然后/最后”触发表 2 中的规则时，由于有多个匹配项，系统不知道需要提取哪一条规则。而用户需要的是一个明确的表示结果，因此提出一种选择最大化的消解策略。

其原理为，对于相同的规则 R_m、R_n，其中，$R_m=C_m+A_m$，$R_n=C_n+A_n$，若 $C_m \subset C_n$，则 $FA=A_m$，$FR=R_n$。其中，FA 称为最终结果，FR 称为有效规则。

简而言之，当触发到多条相同的规则时，如果两条规则之间约束条件存在包含关系，那么就选择约束条件包含其他规则的规则约束条件的结果项作为最终的匹配结果，而相对应的规则作为该关系标记的使用规则，即有效规则。最大化策略可以比较有效地消除规则冲突。

3.1 基本原理

首先，对 R_m 中的每条规则中所对应的约束条件集进行分析，提取约束条件个数和每个约束条件元素；然后，简单地从约束条件个数来判断冲突规则之间是否存在约束条件的包含性，如果存在个数的包含，再根据约束条件元素进行每条匹配，判定约束条件是否存在包含关系，如果存在包含关系，就调用具有包含另一条规则中的约束条件的那条规则，然后依次进行匹配判断，返回最后所得的那条规则。如果还存在规则之间没有包含关系的，那么就对最后剩余的规则的约束条件进行匹配验证，最后选择匹配成功的规则传送给规则引擎。这里需要注意的是，对于相同的关系标记所触发的规则集 R_m，若约束条件不满足，则该条规则不激活。

3.2 算法描述

Max_digestion(最大化消解)算法实现如下：

输入：规则冲突集 R_m

输出：无冲突规则集 UR_m

方法：

(1)int Num[6];//规则中的约束条件个数

(2)for(规则冲突集中所有元素)

```
(3)   for each 规则
(4)     Num[i]=eachR_C(Ri,i);//获取每个规则中的约束条件数
(5)for(规则冲突集中所有元素)
(6)   for(规则冲突集中所有元素个数-1){
(7)     int m=0;
(8)    if(Num[i]<Num[j]{ //i 为冲突集元素下标,j 为循环变量
(9)       for(前一规则的所有约束条件)
(10)        for(后一规则的所有约束条件)
(11)          if(RCj[t]与 RC[j+1][k]匹配)
(12)            m++;
(13)    if(m==Num[i])//前一规则的约束条件都包含在后一规则的约束条件中
(14)       URm=delRule(Rm,i)//删除前一规则
(15)    if(Num[i]>Num[j])
(16)      for(后一规则的约束条件个数)
(17)        for(前一规则的约束条件个数)
(18)          if(RC[j+1][t] 与 RCj[k]匹配)
(19)            m++;
(20)    }//for end
(21)returan URm;
```

4 正覆盖策略

当规则引擎对冲突规则消解后,将获得一个结果规则集,接着规则引擎调用规则解析器进行匹配,返回满足规则约束条件的规则,并存储在一个数据链表 Result 中。如果 Result 集中存在多条规则,同样需要从中选择一个最佳结果。

4.1 基本原理

按照规则解析器调用规则库中 4 张表(连用句式表→连用规则表→规则句式表→普通规则表)的优先顺序,连用句式表中的规则应优先调用。结果规则集传送给执行引擎后,首先剔除优先级低的规则,保留最高优先级的结果集项。如果最高优先级的规则满足约束条件且存在多条,那么,调用所有最高优先级的规则的结果项,对关系标记搭配中的分词进行赋值;如果多条规则中相同的关系标记给出不同的赋值结果,则采用分词已经标识为关系词,再次标识为非关系词的,不进行覆盖赋值;若分词已经标识为非关系词,再次标识为关系词的,则进行覆盖赋值。在这里称其为'正覆盖'策略。

4.2 算法描述

FrontCover 算法实现如下:

```
输入:结果规则集 Result
输出:最优结果规则集
方法:
(1)for(结果集 Result 中的元素)
(2)  optimal_result =dealwith(SA;RSi;Result[i]);//根据结果值对复句 SA 进行特征信息赋值;RSi 为关系标记搭配
(3) Rmax=Result[0];//设结果规则集中第一个规则的优先级最大并赋值给规则 Rmax
(4)for(结果集 Result 中的元素)//Result 集中元素以变量 i 表示
(5)   if(Rmax的优先级小于 Result[i]的优先级)//
(6)     Rmax=Result[i];//优先级大的赋值给 Rmax
(7)optimal_result =dealwith(SA;RSi;Rmax);//最优规则 Rmax对复句 SA 特征赋值
(8)return optimal_result;
   dealwith(SA;RSi;rule)
(1)r=getC(Rule. result);//提取规则的结果项
(2)for(规则中结果项中元素) {
(3)   if(R(m)==true)//m 为关系标记的分词,R(m)=true,表示 m 标记为关系词
(4)     if(SA 中关系标记 m 为 false 或 null)
(5)       m 标记为 true;
(6)   if(R(m)==false)
(7)     if(SA 中关系标记 m 为 null)
(8)       m 标记为 false;
(9) }
(10)return SA;
```

5 算法实例

下面以一个例子来说明上述 3 大算法连用的执行过程。

例句(1):"经过充分讨论,学校决定首先考虑教师,然后是工人、一般干部和中层干部,校级领导放在最后,比例最少。"(《人民日报》1980 年 11 月 27 日)

该条复句(例句(1))通过规则引擎直接传送到复句特征分析器进行分词:经过/p 充分/ad 讨论/v,/w 学校/n 决定/v 首先/d 考虑/v 教师/n,/w 然后/c 是/v 工人/n、/w 一般/a 干部/a 和/c 中层/f 干部/n,/w 校/ng 级/q 领导/n 放在/v 最后/f,/w 比例/n 最/d 少/a。/w

提取关系词标记序列:MS={首先,然后,是,最后}。

通过模式匹配获得关系词标记搭配集:RPS{首先/最后(ordinary 表中),首先/然后/最后(sentence 表中)}。

"首先/最后"关系标记搭配在 ordinary 表中存在一条规则,如表 1 所列,而关系标记搭配"首先/然后/最后"在 sentence 表中存在多条规则,如表 2 所列,则存在冲突,首先对这 6 个冲突集进行最大化策略冲突消解,因为每条规则的约束条件没有包含关系,所以 7 个规则判定为无冲突的规则集,再匹配这 7 个规则的约束条件来判定。根据表 2 所列每条约束条件可知 D(首先,然后)=1,6 条规则都满足,而且复句中"首先"与"然后"所在分句相差为 1,即约束条件满足;接着判断:

1)ID=78 的规则,D(首先,最后)>1,在复句中,"首先"与"最后"所在分句相差为 2,该约束条件满足。

2)ID=124 的规则,D(然后,最后)>3,在复句中,"然后"与"最后"所在分句相差为 1,该约束条件不满足。

3)ID=125 的规则,backword(最后)='-',依照复句可知分词"最后"的后一个分词不是'-',该约束条件不满足。

4)ID=126 的规则,End(clause(最后))='最后',依照复句可知该函数表示"最后"居于所在分句的末尾,该约束条件满足。

5)ID=127 的规则,pos(最后>)=vp,分词"最后"的右连接词没有,所以不是 vp.,约束条件不满足。注:">"表示分词最后的右连接词。

6)ID=128 的规则,numclause(然后)=numclause(最后),依照复句可知分词"然后"和"最后"不在同一个分句中,该约束条件不满足。

7)ID=129 的规则,D(然后,最后)=1,依照复句可知,分词"然后"和"最后"在分句相差为 1,该约束条件满足。

表 1 关系标记"首先/最后"的规则

ID	Keymarks	priority	constraintType	constraints	result
78	首先/最后		2	D(首先,最后)>1	R(首先)=false,R(最后)=false

表 2　关系标记“首先/然后/最后”的规则

ID	keymarks	prior	constraint	constraints	result
124	首先/然后/最后		2	D(首先,然后)=1^D(然后,最后)>3	R(首先)=true,R(然后)=true,R(最后)=false
125	首先/然后/最后		2+1	D(首先,然后)=1^backword(最后)='—'	R(首先)=true,R(然后)=true,R(最后)=false
126	首先/然后/最后		2+1	D(首先,然后)=1^End(clause(最后))= '最后'	R(首先)=true,R(然后)=true,R(最后)=false
127	首先/然后/最后		2+1	D(首先,然后)=1^pos(最后))=VP	R(首先)=true,R(然后)=true,R(最后)=false
128	首先/然后/最后		2+3	D(首先,然后)=1^Numclause(然后)= Numclause(最后)	R(首先)=true,R(然后)=true,R(最后)=false
129	首先/然后/最后		2	D(首先,然后)=1^D(然后,最后)=1	R(首先)=true,R(然后)=true,R(最后)=true

所以可知 ID＝78、ID＝126 和 ID＝127 的规则满足匹配，又可知，根据规则解析器调用 4 张规则表的顺序，ID＝126 和 ID＝127 的规则优先采用，ID＝78 的规则就丢弃；ID＝126 的规则结果“首先”和“然后”充当关系词，“最后”判定为不充当关系词，而 ID＝127 的规则结果“首先”和“然后”充当关系词，“最后”判定为充当关系词，所以根据正覆盖策略，当一个关系标记已标记为 false，再判定标记为 true 时，要对先前的赋值标记进行重新赋值，最终的结果选择搭配关系为“首先/然后/最后”，复句中的分词“首先”、“然后”和“最后”都充当关系词。

结束语　本文设计了适用于复句关系词自动标识的规则引擎的框架结构，并提出 3 大实现策略，以解决关系标记搭配的匹配问题、规则冲突的消解问题，以及最优结果的选取问题。

本文对提出的算法，用大量复句实例进行了测试。实验结果表明，本文提出的策略优化了关系标记搭配的匹配的推理过程，大大提高了系统的可维护性，提高了系统的效率，使汉语复句标识变得方便快捷。同时，由于解决了关系词匹配过程的问题，关系词识别的正确率高达 90.5%。

参 考 文 献

[1]　邢福义. 汉语复句研究[M]. 北京：商务印书馆，2001

[2]　王跃龙，姬东鸿. 汉语树库综述[J]. 当代语言学，2009，12(1)：47-55

[3]　姚双云. 复句关系标记的搭配研究[M]. 武汉：华中师范大学出版社，2008

[4]　李文翔，晏蒲柳，张滨. 基于语料库的关联词识别方法[J]. 计算机工程与应用，2004(7)

[5]　刘云. 复句关系词语离析度考察[J]. 语言教学与研究，2008(6)

[6]　吴锋文. 基于关系标记的汉语复句分类研究[J]. 汉语学报，2011(3)：63-73

[7]　李艳翠，孙静，周国栋，等. 基于清华汉语树库的复句关系词识别与分类研究[J]. 北京大学学报，2013(12)

[8]　胡金柱，舒江波，等. 面向中文信息处理的复句关系词提取算法[J]. 计算机工程与科学，2009(10)

[9]　胡金柱，陈江曼，杨进才. 基于规则的连用关系标记自动标识研究[J]. 计算机科学，2012(5)：190-194

[10]　胡金柱，雷利利，杨进才. 多重复句关系标记搭配的求解模型研究[J]. 计算机工程与科学，2011(11)：177-182

[11]　姚双云，胡金柱，等. 关联词搭配的自动发现[J]. 计算机应用研究，2011(12)

[12]　胡金柱，舒江波，等. 基于 VML 的复句关系层次树的可视化研究[J]. 计算机应用研究，2010(1)

[13]　姚双云，胡金柱. 篇章连贯语义关系的自动标注方法[J]. 计算机工程，2012(7)

[14]　邓沌华，胡金柱，等. 面向现代汉语复句信息工程的语料仓库构建研究[J]. 信息系统工程，2013(9)

[15]　宋震，郭福顺，李莲治. IMPR：一种优于 RETE 算法的多模式/多对象匹配算法[J]. 小型微型计算机系统，2002(2)

（上接第 24 页）

智能 MIDI 音乐系统。实验表明，我们将贝叶斯网与计算机自动作曲结合，更能反映音乐不确定性这一特性，并能生成符合音乐乐理规则的音乐。

从贝叶斯网与 MIDI 音乐库中定义并提取更多独立的证据，使音乐推理更加地灵活、丰富，是我们将要进行研究的工作。

参 考 文 献

[1]　冯寅，周昌乐. 算法作曲的研究进展[J]. 软件学报，2006，17(2)：209-215

[2]　Tromp M，Bod R，Honingh A. Learning Symbolic Music through Hidden Markov Model Induction[J]. 2012

[3]　Fernández J D，Vico F. AI methods in algorithmic composition：a comprehensive survey[J]. Journal of Artificial Intelligence Research，2013，48(1)：513-82

[4]　Bell C. Algorithmic music composition using dynamic Markov chains and genetic algorithms[J]. Journal of Computing Sciences in Colleges，2011，27(2)：99-107

[5]　Salas H A G，Gelbukh A，Calvo H. Music composition based on linguistic approach[M] // Advances in Artificial Intelligence. Springer，2010：117-128

[6]　Boutsinas C，Ha B. Automatic interactive music improvisation based on data mining[J]. International Journal on Artificial Intelligence Tools，2012，21(4)：1-24

[7]　Khan A H. Artificial intelligence approaches to music composition[D]. Northern Kentucky University，2013

[8]　Edwards M. Algorithmic composition：computational thinking in music[J]. Communications of the ACM，2011，54(7)：58-67

[9]　Donnelly P，Sheppard J. Evolving four-part harmony using genetic algorithms[M] // Applications of Evolutionary Computation. Springer，2011：273-82

[10]　Morone A，Manzolli J，Von Zuben F，et al. Evolutionary Computation applied to Algorithmic Composition[J]. NICS Reports，2013(3)：49-53

[11]　Matić D. A genetic algorithm for composing music[J]. Yugoslav Journal of Operations Research，2013，20(1)：2334-6043

[12]　Kirke A，Miranda E R. A survey of computer systems for expressive music performance [J]. ACM Computing Surveys (CSUR)，2009，42(1)：3

[13]　刘惟一，李维华，岳昆. 智能数据分析[M]. 北京：科学出版社，2007

[14]　Raphael C，et al. A Bayesian Network for Real-Time Musical Accompaniment[M] // Advances in Neural Information Processing Systems. 2002

[15]　Oyen D，Lane T，et al. Leveraging Domain Knowledge in Multitask Bayesian Network Structure Learning[C] // AAAI. 2012

[16]　李金鑫. 计算机自动混音系统设计——面向自动作曲系统 MIDI 表达的实现[J]. 电脑编程技巧与维护，2010(14)：104-1055

[17]　曹西征，毛文涛，乔锟，等. 基于音高旋律元的柔和乐曲的自动作曲算法[J]. 自动化学报，2012，38(10)：1627-1638

[18]　Povel D-J. Melody generator：A device for algorithmic music construction[J]. Journal of Software Engineering and Applications，2010，3(7)：683