中 文 信 息 学 报
第 17 卷 第 4 期 JOURNAL OF CHINESE INFORMATION PROCESSING Vol. 17 No. 4

文章编号:1003 - 0077(2003)04 - 0045 - 07

中文语音合成系统中的文本标准化方法①

陈志刚[1],胡国平[2],王熙法[1]

(1. 中国科学技术大学 计算机科学与技术系; 2. 中国科学技术大学 电子工程与信息科学系,安徽 合肥 230027)

摘要:文本标准化是对输入文本进行分析,生成其中非汉字符号的拼音、节奏等信息的过程。本文提出了一种层次化的、基于外部规则的标准化方法,通过规则匹配识别这些符号,并给出各种正确信息。本文首先介绍了分析树的概念,其次给出构造规则的步骤,利用权值控制规则的匹配顺序,最后给出实验结果。实验结果表明:这种方法具有很好的易维护性和可扩展性,开放测试的正确率达到 99.76 %。

关键词:计算机应用;中文信息处理;文本标准化;特殊符号;外部规则

中图分类号:TP391.42 **文献标识码**:A

Text Normalization In Chinese Text-To-Speech System

CHEN Zhi-gang, HU Guo-ping, WANG Xi-fa

(Dept. of Computer Science and Technology & Dept. of Electronic Eng. and Information Science, USTC, Hefei 230027, China)

Abstract: Text normalization is a procedure to generate information, such as pronunciation, rhythm and so on, for special symbols correctly. In this paper, a method based on hierarchical, external rules is presented. By matching rules, we can recognize normal special symbols and generate correct information. This paper introduces the concept of analysis tree firstly, then shows the steps of constructing rules and presents the experiment results. The results show that we can achieve easy-maintainability and easy-expandability, and the correct rate of open test is 99.76 %.

Key words: computer application; Chinese information processing; text normalization; special symbols; external rules

一、引言

语音合成系统,又称文语转换系统,目的是把各种形式的文本转化为自然语音。如何获得高质量的自然语音,一方面取决于语音层处理如何由恰当的韵律知识和适当的生成算法产生自然流畅的语音;另一方面取决于对目标语言自身的了解,包括具体环境下特定文本的内部语法结构、韵律层结构等相应信息。后者,又称文本分析,很大程度上决定了得到的合成语音是否具有人类说话的自然度。

文本标准化(Text Normalization,又称特殊符号处理)是指对输入文本进行分析,把输入文本中的数字、符号等字符转化为规范的文本,并给出相应节奏和轻重读等韵律信息的过程。文本标准化是文本分析过程中一个重要处理环节,其处理效果直接影响着文本的拼音信息的正

① **收稿日期**:2002 - 12 - 03
基金项目:国家自然科学基金资助(69975018)
作者简介:陈志刚(1977 —),男,硕士研究生,研究方向为智能计算.

确性,而且对文本韵律也有一定的影响。

文本标准化过程的困难是因为计算机目前不可能理解输入文本的准确语义,无法通过句子或词语的意思来指导计算机识别特殊符号并给出拼音信息。所以我们必须从特殊符号出发,提取有用、足够的上下文信息,归纳出在特定环境下的各种处理策略。

本文中将中文字符以外的字符均称为特殊符号,主要包括英文字符、数字字符和符号字符。扩展缩写词和数字字符正确识别是文本标准化中两个最有代表性的内容,由于中文自身的特点,中文标准化过程和其他文字的标准化过程有很大不同。如今世界信息交流十分频繁,汉语在发展过程中产生了很多与其他语言结合的新词,这些词也属于中文标准化的处理范畴。中文缩写词不常见到,主要还是中英文混读的句子中的英文词的缩写,如"他的成绩总是NO.1"。数字的中文读法通常有很多种,可以按值读,如"863名工人",也可按串读,如"863计划"等,除此之外,还有多种读法。

特殊符号有着和传统的统计语言学不同的规律,所以不适合采用统计方法,另外关于特殊符号国内外都较少有大规模的标注语料供计算机训练,所以特殊符号处理原则上都是采取基于规则的方法。现有特殊符号处理方法有:借助于一些外部的规则描述符通过程序代码中实现[3],通过代码规定每一种特殊符号如何处理,这种处理的不足是不易维护、可扩展性差;也有用可扩展标记语言SSML根据上下文环境对特殊符号进行标记[4],然后转化SSML标记为已经定义的词语类别,在以后根据词语类别对特殊符号进行处理。因为特殊符号处理常有例外现象发现,所以规则体系的易维护和可扩展性显得尤为重要。故本文则给出一种基于外部规则的层次化匹配方法,特殊符号的识别完全由外部规则来控制,这样易于维护和扩展。

二、外部规则库的构造

2.1 特殊符号的分类

特殊符号处理的输入就是经过自动分词后的词(节点)序列,节点类型有中文H、英文Y、数字S和符号F,对于中文标准化来说,主要目的就是处理这些英文串、数字串和符号串节点,给出它们的汉字、拼音以及节奏等韵律信息。节点类型在规则命名时使用频繁。

2.2 分析树

输入的节点序列在规则的匹配过程,逐一处理拼接,最终得到一棵树(或林),本文把它称为分析树。对于分析树上的每一个节点来说,节点类型就定义为其字符串的类型。"占有23.31%"分词后可以得到这样的节点序列:H1(占有)、S1(23)、F1(.)、S2(31)和F2(%),其中节点名是由字符串类型加一个数字组成。

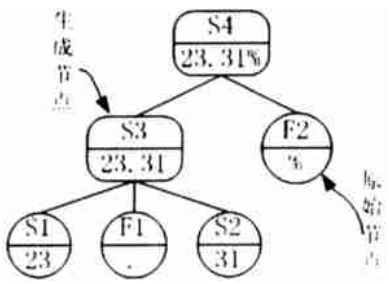

图1 "23.31%"的分析树

2.2.1 原始节点与生成节点

分析树上每一个节点可以分为原始节点和生成节点。原始节点是基于词库分词后得到的节点;生成节点是原始节点在拼接后得到的节点或者在处理时引入的新节点。

图1为"23.31%"的分析树,圆形节点为原始节点,方形节点为生成节点:

2.2.2 节点的属性

所有的节点,都有基本属性和扩展属性两类属性。其中节点类型、汉字信息,拼音信息和韵律信息为基本属性,因为这是所有节点共有的必要属性。对于原始节点,其基本属性是在词库基础上分词时就已经得到了;基本属性之外的属性都属于

扩展属性,常用的扩展属性有词类、长度,对于数字类型的节点还有数值等。

识别文本中特殊符号就是利用节点的类别和属性,确定节点的拼音韵律信息。很多场合,节点本身的属性难以确定节点正确的读法和韵律,这时可以把节点前后文的信息作为节点属性引入,如前后汉字、前后节点词性等等。所以,节点的属性集是开放的,可以在总结规律后根据需要引入新的属性。

2.2.3 节点的 L 属性

节点的 L 属性是表示节点的语义的一个扩展属性,是规则层次化匹配的一个重要属性。原始节点的 L 属性是空的,随着规则的匹配,节点的 L 属性被设为代表各种含义的值,可以根据需要定义 L 属性的值。在我们系统中定义了"无符号整数"、"小数"、"百分数"、"时间"、"比分"等值。

首先构造一些基本的非终结符的产生式,然后利用这些基本的、简单的非终结符又可以方便地构造出更为复杂的非终结符,依此类推,最终形成整个特殊符号处理的规则体系。

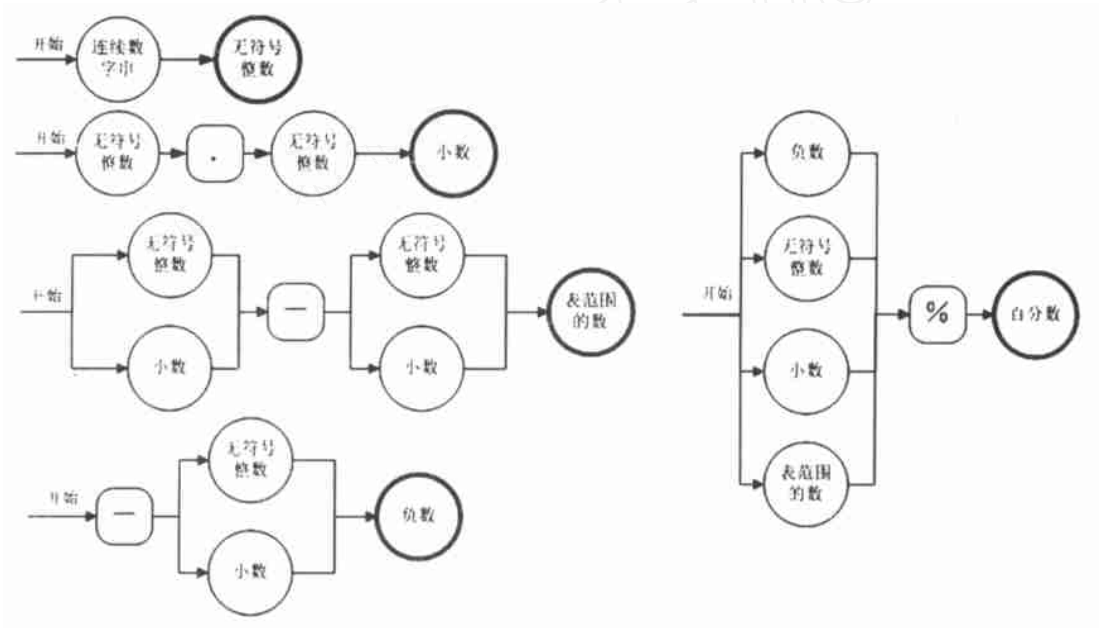

图 2 特殊符号处理的层次化规则体系(部分)

根据这个思想,可以利用节点的 L 属性构造带嵌套结构的规则体系,如图 2 所示,其中粗框节点为规则的拼接节点,普通框节点是规则的条件节点,圆节点为数字节点,方节点为符号节点。可以看出,右边的百分数规则是建立在左边规则的基础上。这样可以使得小数和百分数等在不同的层次上得到处理,就不必为每种百分数情况建立规则。有了左边规则的支持,右边的百分数规则能处理类似"23 %"、" - 23.1 %"和"23 - 23.1 %"等多种情况,从而减少了规则的数量,提高了规则匹配的效率。

2.3 规则的构造

2.3.1 规则的形式

本文介绍的方法基于外部规则。外部规则库保存着一系列的规则,每条规则 R 均由三个部分来描述:条件部分 Q、处理部分 P 和合并部分 C。

条件部分 Q 可以用节点名序列来描述,每个节点均有各自的条件,所有节点的条件加上

节点类型的序列组成了规则的整个条件。

处理部分 P 是对条件部分 Q 定义的节点作各种处理,如改变汉字和拼音信息。可以引入新节点对文本含义进行扩充,对于不包括结果信息的节点可以删除。总之,处理部分的节点序列中每个节点均是最终文本含义的组成部分。

合并部分 C 是对处理部分节点序列中的关系密切的节点拼接起来。这里所谓关系密切包括:节点之间有一些密切的语法关系,如数字和量词在一起时拼接成数量词是合适的;也包括:关系密切的节点只有当拼接成一个节点后才有了更为完整的含义,如小数 23.31,三个子节点拼接成一个节点后才有了完整的含义。合并部分在规则中不是必须的,因为如果节点之间的关系不密切,那么合并部分可以为空。

各个部分的定义形式可以有多种,下面为一种比较简单直观的形式,其中 X、Y 为字符类型(H、S、Y 和 F)中的任何一个:

® 条件部分 Q 定义节点形式:

X1(条件 1),……,Xn(条件 n)。

其中:圆括号中为对应节点的限定条件,如可以限定拼音信息是否为空、数字节点的数值范围,符号节点的字符串信息等。

® 处理部分 P 形式:

[Y0(处理操作)][X1(处理操作)][Y1(处理操作)],……,[Xn(处理操作)][Yn(处理操作)]

其中:Xi 为条件部分定义节点,Yi 为新引入的节点。方括号表示可选,圆括号中为对应节点的处理操作,如可以设置对应节点的汉字信息等。

® 合并部分 C 形式:

[Xk(拼接若干节点 + 处理操作)],……,[Xk + 1(拼接若干节点 + 处理操作)]

其中:圆括号含义为按特定顺序拼接若干前面已经定义的节点成一个新节点,同时可以进行设置各种信息的处理操作。

2.3.2 规则的构造

构造规则时必须先清楚规则目的是要解决哪种特殊符号情况,同时由于规则匹配效率的问题,必须考虑这种特殊符号情况是否具有普遍意义,也就是说规则不能定义的过细。

下面以“23.31 %”为例子,来构造匹配相应的规则。人对“23.31 %”理解是首先这是一个百分数,然后是这个百分数是有“23.31”这个数字和符号“ %”组成的。计算机可以模拟人的这一过程,那么第一步计算机必须首先识别“23.31”,然后识别整个百分数,所以要构造 2 条规则。

小数规则 R1 可以按如下步骤构造:

® 根据现有知识确定规则条件部分

对于小数,常识告诉我们由三部分组成:数字 + . + 数字,因此我们可以定义小数规则 R1 条件部分 Q(R1):

Q(R1) :S1(拼音为空) F1(节点为“.”) S2(拼音为空)

® 对条件部分节点做相应处理

根据常识,可以这样定义小数规则的处理部分 P(R1):

P(R1) :S1(修改成按值方式读后的汉字和拼音等信息) F1(修改成“点”的汉字信息和拼音信息) S2(修改成按串方式读后的汉字和拼音等信息)

® 关系密切的节点合并成新节点

显然前面定义的三个节点的关系密切,所以定义合并部分 C(R1)为:

C(R1) :S3(拼接 S1,F1,S2,同时从子节点上合并汉字拼音等信息)

® 给出规则权值

根据规则的层次关系给出一个权值,比如把小数规则的权值设为 1000。

由于小数规则是比较常用的规则,经常会和其他规则发生冲突。如"23.31%"的节点序列中,就有两条匹配规则:小数(23.31)规则和百分数(31%)规则,显然小数规则应该先匹配,所以确定小数规则的权值后,其他规则的权值的确定必须要以小数规则的权值为参考。

对于百分数规则 R2 可以按同样步骤构造:

Q(R2) :S1(L 属性为无符号整数或小数) F1(节点为"%")

P(R2) :S1(不作任何处理) F1(修改汉字信息为"百分之"和拼音信息)

C(R2) :S2(拼接 F1,S1,同时从子节点上合并汉字拼音等信息)

由于小数规则的权值为 1000,而且百分数规则依赖于 L 属性为"小数",所以可以设百分数规则的权值为 900。

整个规则库就是按照上面的方法,根据已有的知识,归纳出的各条规则组成。规则库构造过程中,规则的权值设定是非常重要而又比较困难的,它直接影响了规则的匹配顺序,因此权值需要经过大量文本的测试来调整,最终达到稳定状态。目前在我们的规则库中,定义了 31 种 L 属性和四百多条规则。

三、外部规则的匹配

3.1 层次化匹配

层次化匹配是利用计算机模拟人识别文本过程的一个重要特征。正确的信息是通过输入文本在不同层上的处理后得到的。本文系统中规则匹配的第一层是 L 属性为空的层,以后的每一层均是建立在前面层处理的基础上,规则匹配所处的层次取决于规则的条件和权值。

还是以"占有 23.31%"为例子,首先小数规则和百分数规则的条件同时得到满足,结果是权值较高的小数规则得到匹配,生成新的数字节点 S3,并把 S3 的 L 属性设为"小数";接下来百分数规则得到了匹配,生成了节点 S4,并且把 L 属性为"百分数"。图 3 清晰表明了 2 次规则匹配的过程,其中节点中的 C 代表汉字信息,S 代表拼音信息。

图 4 的左边为我们系统中特殊符号处理前的结果,是整个处理的输入;右边为处理后的结果,可以看到"占有 23.3%"被正确识别了。

3.2 歧义的消除

特殊符号处理中的歧义包括两种,一种是局部与全局的歧义,另一种是全局与全局的歧义。前者如"23.31%",存在着"31%"和"23.31%"之间的歧义,象这样的歧义可以通过用规则权值控制匹配顺序来消除;后者如"18:25",不可能断定它是表示时间还是表示比分,对它的处理在全局上有了歧义,消除这种歧义的方法就是有效利用上下文信息。L 属性在消除这种歧义中起了非常重要的作用。

如"2 局比分 18:25、21:25"和"2 场开始时间 18:25、21:25"这两个句子,对于"18:25"的歧义,通过直接上下文环境的不同很容易消除;而"21:25"直接的上下文信息显然不够,但是由于前面的"18:25"已经拼接成了一个节点,这个节点有着确定的 L 属性,可以通过这个间接的上下文信息构造规则,从而消除"18:25"的歧义。

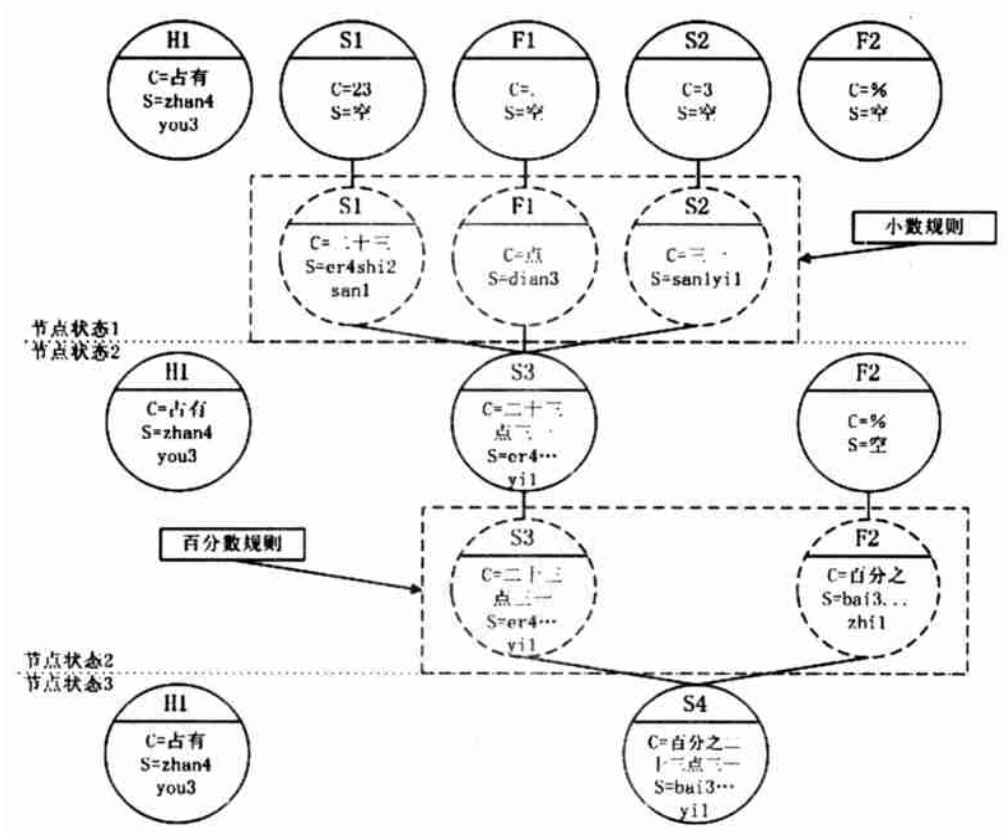

图 3 “占有 23.31 %”的 2 次规则匹配

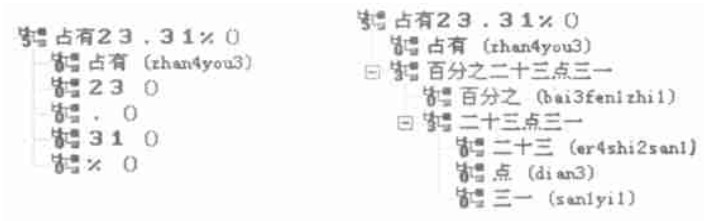

图 4 “占有 23.31 %”的输入(左)和输出结果

因此,在消除特殊符号歧义时应该灵活利用 L 属性,善于总结规律。

四、测试结果

我们用 1900 多句包括各种特殊符号情况的句子作为构建规则库的语料,这些句子主要来自于平时的收集和一些实际应用中错误处理的反馈,针对其中每一种有普遍意义的情况构造相应的规则加入到规则库中。最后这些句子的集内测试结果达到 98.7 %。

我们还进行了集外测试:从网络上下载的各种文本中,挑选出含有不同类别特殊符号情况的句子 1705 句,正确识别的有 1642 句,测试正确率达到 96.3 %。经统计发现,文本中常见的特殊符号均为比较容易处理的情况,因此又进行了一次大规模的开放测试,在 24000 句的开放测试中特殊符号识别正确率达到 99.76 %。

表 1 是一些句子在本文系统的处理结果,其中大写英文字母表示按照字母方式读,小写表

示按拼音或单词方式读。最后 3 句未能正确处理,原因是上下文的信息不够,不能给出有普遍意义的处理规则。

表 1　一些句子的处理结果

处理前的汉字信息	处理后的汉字信息	对错
2 局比分 21 :25、18 :25	两局比分二十一比二十五,十八比二十五	√
2 局开始时间 18 :25、21 :25	二局开始时间十八点二十五分、二十一点二十五分	√
9.11 事件	九么么事件	√
9.11 个百分点	九点一一个百分点	√
- 34.32 ℃	零下三十四点三二摄氏度	√
32.3 - 34.32 ℃	三十二点三至三十四点三二摄氏度	√
0.25 元 - 0.28 元 KWH	零点二五元至零点二八元每千瓦时	√
90、94、98 年 3 届世界杯	九零、九四、九八年三届世界杯	√
du-feilong @ccw.com.cn	du 下划线 feilong at CCW 点 com 点 CN	√
C:\ KBCE2.0	C 盘 KBCE 二点零	√
生产日期 2001 - 11 - 2	生产日期二零零一年十一月二日	√
17 :05PM	下午十七点零五分	√
1000 元	人民币一千元	√
0551 - 3601363 - 812	零五五么三六零么三六三转八么二	√
乐坛群星闪耀 5.1 工体	乐坛群星闪耀五点一工体	×
总体费用为 2 - 3	总体费用为二减三	×
东方 110	东方一百一十	×

总体来说,以上方法能有效对含有特殊符号的句子进行标准化。

五、结束语

文本标准化是中文语音合成系统中一个非常重要的部分,对合成效果有着很大的影响。本文提出了一种层次化匹配规则的方法,能有效模拟人识别文本的过程,从而在规则尽可能精简的前提下提高了正确率。利用该方法构造的规则库,可以识别绝大部分的特殊符号句子,开放测试正确率达到 99.76 %。

从识别错误的句子中,我们发现错误原因主要集中表现在:这些句子中特殊符号的正确识别必须理解上下文的含义,也就是说只用我们定义的节点属性难以提供足够信息。

用本文介绍的方法构造的外部规则库,具有很好的易维护性和可扩展性。由于中文的不断发展,表示新含义的特殊符号情况会不断出现,因此应该不断的对外部规则库进行更新,给出正确的处理规则。

参 考 文 献:

[1] Richard Sproat. Multilingual text analysis for text - to - speech synthesis [C], ICSLP'96.

[2] Richard Sproat, Alan Black, Stanley Chen, Shankar Kumar, Mari Ostendorf, Christopher Richards. Normalization of Non - Standard Words [C]: WS '99 Final Report (1999).

[3] Wu Xiaoru. Special Text Processing Based External Descriptor Rule [C], ICSLP'2000.

[4] Andrew Breen, Barry Eggleton. Refocussing on the text normalization process in Text-to-speech Systems [C]. ICSLP'2002.

[5] Mehryar Mohri, Richard Sproat. A Efficient Compiler for Weighted Rewrite Rules [C]. Meeting of the Association for Computational Linguistics, 1996.

[6] 陈意云. 编译原理和技术 [M]. 合肥:中国科技大学出版社.