文章编号:1001 - 9081(2006)03 - 0651 - 04

英文文语转换系统中基于决策树的词性标注的非监督学习

王永生[1],柴佩琪[2]
(1. 同济大学 留德预备部, 上海 200092; 2. 同济大学 计算机科学与工程系, 上海 200092)
(yshwang@online.sh.cn)

摘　要:英文文语转换系统中的韵律生成模块和多音词消歧模块均必须用到单词的词性信息,因而词性标注是英文 TTS 系统中一个非常重要的部分。讨论了在只有一个词库的有限条件下,如何通过决策树中的 C4.5 算法进行词性标注的非监督学习,同时讨论了未登录词的词性猜测问题。

关键词:文语转换;词性标注;决策树;自然语言处理
中图分类号: TP391.1　　**文献标识码**:A

Unsupervised learning of part-of-speech tagging using decision trees in English TTS

WANG Yong-sheng[1], CHAI Pei-qi[2]
(1. *German College, Tongji University, Shanghai* 200092, *China*;
2. *Department of Computer Science and Engineering, Tongji University, Shanghai* 200092, *China*)

Abstract: Part-of-Speech is essential for prosody generation module and homograph disambiguation module in English TTS(text to speech) system. An unsupervised learning of part-of-speech tagging using decision trees was described, under the condition of a lexicon. In addition, the problem of unknown word guessing was also discussed.

Key words: text-to-speech; part-of-speech tagging; decision tree; natural language processing

0　引言

所谓词性标注(Part-of-Speech Tagging),是指给文本中的每个单词及符号标注正确的词性标记。词性标注是自然语言处理中一个非常重要的课题,它可以应用于多个方面,如文语转换、机器翻译、信息提取、拼写检查等。以文语转换为例,对于一段要合成的文本,取得每个单词的词性信息是非常重要的,因为文语转换系统中的一些模块会用到这些词性信息,如在韵律生成模块中,在进行句法分析时,显然必须先知道单词的词性;再比如说,有些单词因其词性的不同而读音不同,如 advocate,名词时读['ædvəkit],而作动词时读['ædvəkeit],对于这类词,没有词性信息显然难以判别其正确的读音。加之,许多英文单词存在多个词性,即使通过查找词典,也仅仅能知道某个单词可能的词性有哪些。要准确地知道合成文本中的每个词的确切词性,必须通过词性标注算法,充分利用单词的形态结构及上下文信息来解决。

自从 1971 年 TAGGIT 词性标注系统[1](用于标注 Brown 语料库)问世以来,词性标注问题就一直是自然语言处理中一个比较热门的课题,并出现了多种标注算法,主要有三类,即基于语言学、基于统计学和基于机器学习方法。基于语言学的标注算法主要是依据语言学、形态学等知识编制约束规则用于词性标注,如 TOSCA 系统[2]和约束语法算法[3]等;基于统计学模型的算法,主要采用一元语法、二元语法、三元语法或 HHM 等统计模型来进行词性标注学习,如 CLAWS 词性标注系统[4],以及能量函数优化法[5]和最大熵法[6]等;而基于机器学习方法的标注算法主要有基于转换的学习算法[7]、约束语法规则法[8]及基于决策树的标注算法[9]等。

上述三类方法中,第一类方法要求有较深的语言学功底,后两类方法大多以大规模标注语料库为基础。对一般的研究人员而言,他们既没有深厚的语言学功底,也很难取得大规模的标注语料库,而如果自己动手创建,显然不是短期所能完成的。本课题目前可资使用的仅仅是一个有 27 000 个词的词库,其中包含每个词所有可能的词性,本文所讨论的问题就是如何在这样有限的条件下,完成词性标注任务。

当然仅有一个词库是远远不够的,对于一段要标注的文本,最多只能通过查找词库,取得每个单词所有可能的词性。假设我们现在还有一个未标注的语料库,通过查找词库,将该语料库中的单词均标注上所有可能的词性。

例句 1:

In(IN) an(DT) interview(NN,VB) he(PRP) told(VBD, VBN) a(DT,SYM) lie(VB,NN) .(.)

括号内的符号为词性标记(本文采用的是 Penn Treebank 定义的词性标记集,其中一些主要的标记有:IN - 介词;DT - 限定词;NN - 名词单数;VBD - 动词过去式;VBN - 动词过去分词;VBZ - 动词第三人称单数;PRP - 人称代词;PRP $ - 物主代词;SYM - 符号;JJ - 形容词;RB - 副词;CC - 连词)。在这个句子中有 4 个词的词性不确定,如何对它们进行消歧呢?以 interview 为例,由于该词的前一个词为不定冠词 an,通过遍历整个语料库,查看在 an 后面的、且无词性歧义的词,发现他们可以是名词、形容词等,就是没有一个是动词,依据这一点,可以断定上述句子中的 interview 的词性为 NN。也就是说,我们可以利用初始标注的语料库中无词性歧义的词来对

收稿日期:2005 - 10 - 08
作者简介:王永生(1972 -),男,江苏东台人,博士研究生,主要研究方向:语言合成、自然语言理解; 柴佩琪(1935 -),女,上海人,教授,博士生导师,主要研究方向:语音处理、人工智能、自然语言处理.

有词性歧义的词消歧。这实际上将整个词性标注问题转化成一个分类问题,对于可能是 NN 和 VB 的所有词形成一类(下文中统一将此类称为 NN_VB 类),消歧的实质就是将所有属于此类的实例最终分成两类,一类的词性为 NN,而另一类的词性为 VB。本文将通过决策树中的 C4.5 算法[10]来构造分类器,解决词性的消歧问题。

此外,在一段要标注的文本中,总有可能存在某些词在词库中找不到,即所谓的未收录词。对于这些单词,必须先猜测出它们可能的词性,然后再将整个文本通过词性标注算法进行标注。

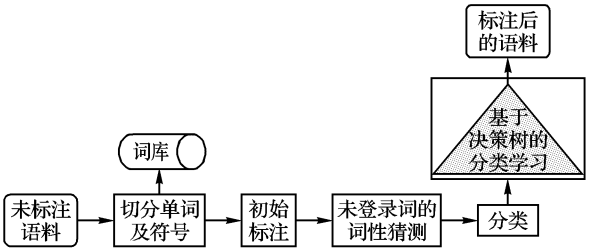


图 1 词性标注学习的过程

除了已创建好的词库外,我们还需要一个未标注的语料库。未标注的文本还是比较容易取得的,我们从 Internet 上取得一些英文网页,经过编辑整理,形成一个有 160 000 个词的语料库,留出其中的 20 000 个词的文本作测试用,其余 140 000 个词的文本用作学习。

1 基于决策树的词性标注学习

首先通过查找词库,对学习语料进行初始标注,即给每个单词及符号标注上所有可能的词性(本文采用的是 Penn Treebank 定义的词性标记集[11],共有 45 个标记,其中有 9 个标记是标点符号标记,36 个是具体的英语单词标记),并且假设其中的未收录词的词性已经过处理。经过初始标注的学习语料,有 35.6% 的单词是有歧义的,平均每个单词有 2.39 个词性标记,整个语料库有 238 个不同的歧义类(如所有可能是 NN 和 VB 的单词组成 NN_VB 类),其中有 42 个频率最高的类占 85.71%。

决策树是数据挖掘中应用最广的归纳推理算法之一,是解决分类问题的一个强有力的工具,对噪声数据有很好的健壮性。决策树通过构建一个二叉树或多叉树,把实例从根结点排列到叶子结点来分类实例,叶子结点即为实例所属的分类。树上的每一个结点指定了对实例的某个属性的测试,并且该结点的每一个分支对应于该属性的一个可能值。分类实例的方法是从根结点开始,测试这个结点指定的属性,然后按照给定实例的该属性值向下生长,然后这个过程在以新结点为根的子树上重复[12]。

1.1 属性的选取

实例是用"属性 - 值"对来表示的,我们针对每个实例选择 5 个属性:P_ - 1:前一个词的词性标记;W_ - 1:前一个词;W_0:当前要进行词性消歧的词;P_1:下一个词的词性标记;W_1:下一个词。

以例句 1 中的 interview 为例,其对应的 5 个属性为(DT, an, interview, PRP, he)。表 1 选取了 NN_VB 类中的 10 个实例。

表 1 NN_VB 类的实例

Instance	P_ - 1	W_ - 1	W_0	P_1	W_1
I1	TO	to	lift	DT	the
I2	DT	the	buckle	IN	on
I3	CC	and	curb	PRP$	your
I4	CC	and	interest	VBZ	is
I5	DT	the	field	IN	of
I6	TO	to	launch	DT	a
I7	DT	the	board	IN	as
I8	DT	a	launch	IN	of
I9	DT	the	plant	IN	for
I10	CC	and	finance	PRP$	their

1.2 定义分类函数

假设现在要对表 1 的 10 个实例构建决策树,怎么知道当前哪一个属性是最佳的分类属性呢?为此,我们必须定义一个分类函数。

1.2.1 歧义类中不同标记的词分布的相对比例

首先来考虑最简单的情况,即两类问题(如 NN_VB 类),定义:

$$P_C(X) = \frac{f_C(X) * f(Y)}{f_C(X) * f(Y) + f_C(Y) * f(X)}$$

$$P_C(Y) = \frac{f_C(Y) * f(X)}{f_C(X) * f(Y) + f_C(Y) * f(X)}$$

其中:X,Y 表示词性标记;$f(X)$ 表示语料中标注为 X 的词出现的次数;$f(Y)$ 表示语料中标注为 Y 的词出现的次数;$f_C(X)$ 表示在上下文环境 C 下标注为 X 的词出现的次数;$f_C(Y)$ 表示在上下文环境 C 下标注为 Y 的词出现的次数。显然这里定义的 $P_C(X)$ 和 $P_C(Y)$ 分别表示在整个语料中,在上下文环境 C 下标注为 X 和 Y 的词的相对比例。

对于多类问题,用 $X_i(i = 1,2,\cdots,N)$ 表示该歧义类中的 N 个词性标记,则在上下文环境 C 下标注为 X_i 的词的相对比例定义如下:

$$P_C(X_i) = \frac{f_C(X_i)\prod_{j=1}^{N} f(X_j)}{\sum_{h=1}^{N}\left[f_C(X_h)\prod_{k=1}^{N} f(X_k)\right]}, i \neq j, h \neq k$$

1.2.2 实例分布的熵

对于两类问题,再定义:

$$E(S_C) = -P_C(X)\log_2 P_C(X) - P_C(Y)\log_2 P_C(Y)$$

其中 S_C 表示某个目标概念在上下文环境 C 下的实例集,因而 $E(S_C)$ 表示在上下文环境 C 下的实例分布的熵。

更一般地,对于多类问题,在上下文环境 C 下的实例分布的熵定义为:

$$E(S_C) = \sum_{i=1}^{N} -P_C(X_i)\log_2 P_C(X_i)$$

1.2.3 实例分类函数

最后定义实例的分类函数如下:

$$T(S,A) = \sum_{C \in Values(A)} \frac{|S_C|}{|S|} E(S_C)$$

其中 S 表示某个目标概念的整个实例集,A 表示实例的某个属性,$Values(A)$ 表示属性 A 所有值的集合,则 $T(S,A)$ 表示用属性 A 分类 S 后熵的期望值。

$T(S,A)$ 即为定义的分类函数,它是决策树增长过程中每

一步选取最佳属性的度量标准,当某个属性 A 的 $T(S,A)$ 值最小,说明经过它分类的结点中的实例集最纯。

以表 1 的 NN_VB 类为例,假设整个实例集 S 仅由这 10 个实例组成,即 $|S| = 10$,令 $X = \mathrm{NN}$,$Y = \mathrm{VB}$,C 为"P_ - 1 = CC",则 $|S_{\mathrm{P\_-1=CC}}| = 3$,经过对学习语料库统计,得 $f(\mathrm{NN}) = 8\,462$,$f(\mathrm{VB}) = 3\,064$,$f_{\mathrm{P\_-1=CC}}(\mathrm{NN}) = 226$,$f_{\mathrm{P\_-1=CC}}(\mathrm{VB}) = 56$,则:

$$P_{\mathrm{P\_-1=CC}}(\mathrm{NN}) = 226 \times 3\,064/(226 \times 3\,064 + 56 \times 8\,462) = 0.594$$

$$P_{\mathrm{P\_-1=CC}}(\mathrm{VB}) = 56 \times 8\,462/(226 \times 3\,064 + 56 \times 8\,462) = 0.406$$

$$E(\mathrm{S}_{\mathrm{P\_-1=CC}}) = -0.594 \times \log_2 0.594 - 0.406 \times \log_2 0.406 = 0.975$$

又由于 $E(\mathrm{S}_{\mathrm{P\_-1=DT}})$ 和 $E(\mathrm{S}_{\mathrm{P\_-1=TO}})$ 均为 0,则:

$$T(\mathrm{S,P\_-1}) = 0 \times 5/10 + 0 \times 2/10 + 0.975 \times 3/10 = 0.293$$

1.3　决策树的构建

一开始,所有的实例组成根结点,根据每个属性的分类函数的值,来决定哪一个属性最先用于分类。5 个属性对应的分类函数的值分别为:

$T(S,\ \mathrm{P\_-1}) = 0.293$

$T(S,\ \mathrm{W\_-1}) = 0.357$

$T(S,\ \mathrm{W\_0}) = 0.683$

$T(S,\ \mathrm{P\_1}) = 0.376$

$T(S,\ \mathrm{W\_1}) = 0.485$

因而选取属性 P_-1 为第一个分类属性,该属性有三个值,即 DT,TO 和 CC,因而生成的树有三个分叉。当 P_ - 1 为 DT 和 TO 时,由于 $E(S_{P\_-1=DT})$ 和 $E(S_{P\_-1=TO})$ 均为 0,说明其中的实例分布最纯,所以这两个结点就成为叶子结点,无需继续生长。而对于 P_ - 1 = CC 的这个结点,其 $E(S_{\mathrm{P\_-1=CC}})$ 不为 0,所以继续向下生长。由于 $T(\mathrm{S}_{\mathrm{P\_-1=CC}},\mathrm{P\_1}) = 0$,在剩余的 4 个属性的分类函数值中最小,所以再依据属性 P_1 来分类,并最终生成两个叶子结点,从而将 10 个实例成功分成两类。

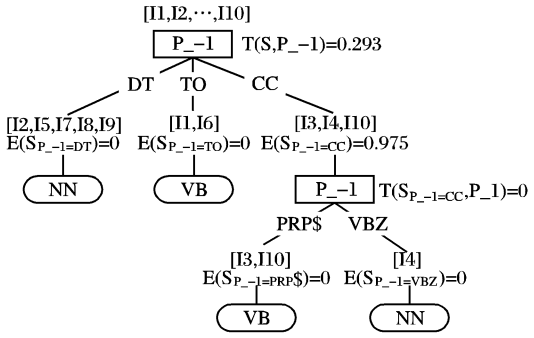


图 2　生成的决策树

对于这样一棵决策树,对应着以下 4 条 NN_VB 类的词性消歧规则:

if P_ - 1 = DT then POS = NN

if P_ - 1 = TO then POS = VB

if P_ - 1 = CC and P_1 = PRP$ then POS = VB

if P_ - 1 = CC and P_1 = VBZ then POS = NN

通过对 238 个歧义类构建决策树进行学习,总生成 3 872 条词性标注规则。

2　未登录词的词性猜测

许多词性标注系统均讨论了未登录词的词性猜测问题,但大多仍以大规模的标注语料库为基础,通过算法猜测未登录词最可能的词性,然后再将之与其他词一起进行词性标注学习[13,14]。由于我们没有标注语料库,可资使用的只是一个词库和一个未标注语料库,因而不能使用那些基于标注语料库的算法。其实,细细分析 Penn Treebank 标记集中的 45 个标记,其中标点符号标记有 9 个,另外一些标记,如情态动词(MD)、连词(CC)、人称代词(PRP)、物主代词(PRP$)、介词(IN)等涉及的英语单词是有限的、可罗列的,我们将这些词均添加到词库中,然后再将不规则的名词复数、不规则的动词过去式及过去分词、不规则的形容词和副词的比较级及最高级添加到词库中。这样一来,出现的未登录词只可能是以下几类:

名词　包括名词单数(NN)、名词复数(NNS)、专有名词(NNP)和专有名词复数(NNPS)。

动词　包括动词的基本形式(VB)、动词第三人称单数(VBZ)、动名词(VBG)、规则的过去式(VBD)和规则的过去分词(VBN)。

形容词　包括形容词原形(JJ)、规则的形容词比较级(JJR)和规则的形容词最高级(JJS)。

副词　包括副词原形(RB)、规则的副词比较级(RBR)和规则的副词最高级(RBS)。

我们知道,规则的名词复数形式、动词第三人称单数、动名词、规则的动词过去式和过去分词、规则的形容词和副词的比较级及最高级均是通过添加特定词尾来构成的,而许多形容词和副词也可以通过特定的前缀或后缀来判断其词性。比如说,对于单词 shouting,由于其后缀为 ing,且去除 ing 后的部分是一个动词,则显然该词是一个动名词,而动名词在句子中可以作 VBG,JJ 及 NN。因而,针对这类词可定义规则:

if --3 = ing and POS = VB then POS = (VBZ, JJ, NN)

针对未登录词,定义以下几类规则:

1) 某个词含有特定的前缀或后缀。如:

if ~~4 = able then POS = (JJ)

表示如果后缀为 able,则该词的词性标记为 JJ。

if 2 ~~ = un and ~~2 = ed then POS = (JJ)

表示如果前缀为 un,后缀为 ed,则该词的词性标记为 JJ。

2) 如果某个词加上一个特定前缀或后缀后形成的词在词库中存在。如:

if ++2 = ly and POS = RB then POS = (JJ)

表示如果加上后缀 ly 后是一个副词,则该词的词性为 JJ。

3) 如果某个词减去某个特定前缀或后缀后形成的词在词库中存在。如:

if --2 = ed and POS = VB
then POS = (VBD, VBN, JJ)

表示如果删除词尾的 ed 后是一个动词,则该词的词性可能是 VBD,VBN 及 JJ。

(4) 除此以外,定义最通用的规则:

if 首字母大写
　then POS = (NNP, NN, VB, JJ, RB)
　else POS = (NN, VB, JJ, RB)

即如果前面所有的规则均不适用,则应用此规则。

我们一共定义了 87 条未登录词词性猜测规则。

3 实验结果

为了测试上述基于决策树的词性标注算法的效果,我们通过半自动的方式对测试语料进行了词性标注,以便与经过算法标注的结果进行对比与分析。

首先对原始的未标注的测试语料通过查找词库进行初始标注,发现其中有 2752 个未登录词。然后再使用未登录词词性猜测规则,猜测出未登录词可能的词性,发现在所有未登录词中,词性猜测正确的有 2546 个(所谓猜测正确,是指猜测出的词性组合中包含其正确的词性),其正确率为 92.51%。其中猜测错误的主要是那些词性比较复杂的词,如 nominal,由于 nominally 在词库中存在,因而依据规则,nominal 加上 ly 后,其作为一个副词在词库中存在,因而判定 nominal 的词性为 JJ,而事实上 nominal 还可作 NN。

未登录词经过处理后,测试语料中的所有词均被标注上其可能的词性,现在再将测试语料通过经学习语料学习生成的决策树进行词性消歧,并最终生成每个词和符号只有一个词性标记的语料,发现共有 978 个标注不正确,标注正确率为 95.11%。

4 结语

显然 95.11% 的标注正确率与其他一些标注系统大多能达到 97%(最高近 99%)以上的高正确率相比,还有不小的差距,然而由于我们既缺少一个大规模的标注语料库用于标注学习及决策树的修剪,且由于词库规模有限,有相当一部分未登录词只能通过词性猜测规则来猜测词性,能达到 95.11% 的正确率应该说还算是一个不错的结果。

参考文献:

[1] GREENE BB, RUBIN GM. Automatic Grammatical Tagging of English[R]. Department of Linguistics, Brown University, 1971.

[2] OOSTDIJK N. Corpus Linguistic and the automatic analysis of English[Z]. Rodopi, Amsterdam, 1991.

[3] KARLSSON F, VOUTILAINEN A, HEIKKILA J, *et al*. Constraint Grammar. A Language-Independent System for Parsing Unrestricted Text[M]. Berlin, New Youk: Mouton de Gruyter, 1995.

[4] GARSIDE R, LEECH G, SAMPSON G. The Computational Analysis of English[M]. London and New York: Longman, 1987.

[5] SCHMID H. Part-of-speech tagging with neural networks[A]. Proceedings of 15th International Conference on Computational Linguistics[C], 1994.

[6] ROSEFELD R. Adaptive Statistical Language Modeling: A Maximum Entropy Approach[D]. School of Computer Science, Carnegie Mellon University, 1994.

[7] SAMUELSSON C, TAPANAINEN P, VOUTILAINEN A. Inducing Constraint Grammars[A]. Proceedings of the 3rd International Colloquium on Grammatical Inference[C], 1996.

[8] DAELEMANS W, ZAVREL J, BERCK P, *et al*. MTB: A Memory-Based Part-of-speech Tagger Generator[A]. Proceedings of 4th Wordshop on Very Large Corpora[C], 1996.

[9] MARQUEZ L, RODRíGUEZ H. Part-of-Speech Tagging Using Decision Trees[A]. Proceedings of the 10th European Conference on Machine Learning, ECML[C]. Chemnitz, Germany, 1998.25 – 36.

[10] QUINLAN JR. C4.5: Programs for Machine Learing[M]. San Mateo, CA: Morgan Kaufmann, 1993.

[11] MARCUS M, SANTORINI B, MARCINKIEWICZ MA. Building a large annotated corpus of English: The Penn Treebank[J]. Computational Linguistics, 1993, 19(2): 313 – 330.

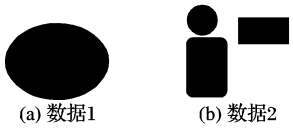
[12] MITCHELL T. Machine Learning[M]. McGraw Hill, 1997.

[13] ERIC B. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part of Speech Tagging[J]. Computational Linguistics, 1995, 21(4): 543 – 565.

[14] VASILAKOPOULOS A. Improved Unknown Word Guessing by Decision Tree Induction for POS Tagging with TBL[A]. Proceedings of CLUK 2003[C], 2003.

(上接第 646 页)

单一,数据量较少,如图 3(a)。数据 2 区域划分较多,数据量比较大,如图 3(b)。



(a) 数据1 (b) 数据2

图 3 测试数据

在 5 节点(1 个主节点 4 个从节点)和 9 节点(1 个主节点 8 个从节点)情况下分别对两组数据并行处理。试验结果如表 1 ~ 表 2。

表 1 5 节点加速比

数据区域	步骤 Step1	Step2	Step3	Step4
1	2.6	4.0	1.8	2.6
2	2.25	4.0	1.6	2.25

表 2 9 节点加速比

数据区域	步骤 Step1	Step2	Step3	Step4
1	5.2	8.0	2.7	5.2
2	4.5	8.0	2.3	4.5

通过试验结果我们可以看出,在数据量较大、计算比较复杂的情况下,使用我们的基于 MPI 的并行小波聚类算法获得了较好的加速比,表明我们的算法具有较好的性能。

参考文献:

[1] LIANG H, ZHAO G-S, LI W-S. A New MPI-Based Parallel Wave-Cluster Algorithm [A]. 第 22 届全国数据库学术会议(NDBC2005)[C], 2005.

[2] ZHANG T, RAMAKRISHNAN R, LIVNY M. BIRCH: An Efficient Data Clustering Method for Very Large DataBases[A]. ACM SIGMOD International Conference on Management of Data[C], 1996.

[3] GANTI V, GEHRKE J, RAMAKRISHNAN R. CACTUS – clustering Categorical Data Using Summaries[A]. ACM SIGKDD International Conference on Knowledge Discovery and Data Dining[C], 1999.

[4] WANG W, YANG J, MUNTZ R. STING: A Statistical Information Grid Approach to Spatial Data Mining[A]. 23rd VLDB Conference [C], 1997.

[5] CHENG CH, FU AW, ZHANG Y. Entropy-based Subspace Clustering for Mining Numerical Data[A]. ACM SIGKDD International Conference on Knowledge Discovery and Data Dining[C], 1999.

[6] ZHOU B, SHEN JY, PENG QK. Parallel Clustering Algorithm for PCs Cluster[J]. Computer Engineering, 2004, 30(4).

[7] WANG K-B, CHIA T-L, CHEN Z, *et al*. Parallel Execution of a Connected component labeling Operation on a Linear Array Architecture[J]. Journal of Information Science and Engineering, 2003, 19(2): 353 – 370.