ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.5, May 2006, pp.1026−1033
DOI: 10.1360/jos171026

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种特征匹配方法:稀疏特征树*

周雅倩+, 黄萱菁, 吴立德

(复旦大学 计算机科学与工程系,上海 200433)

A Feature Matching Method: Sparse Feature Tree

ZHOU Ya-Qian, HUANG Xuan-Jing, WU Li-De

(Department of Computer Science and Engineering, Fudan University, Shanghai 200433, China)

+ Corresponding author: Phn: +86-21-55664844, E-mail: ZhouYaqian@fudan.edu.cn, http://www.cs.fudan.edu.cn/mcwil/irnlp

Zhou YQ, Huang XJ, Wu LD. A feature matching method: Sparse feature tree. *Journal of Software*, 2006, 17(5):1026−1033. http://www.jos.org.cn/1000-9825/17/1026.htm

Abstract: Computational efficiency is an important concern for machine learning algorithms, especially for applications on large test sets or in real-time scenarios. In this paper, a novel data structure and the corresponding algorithms for the execution system of the maximum entropy model are described. This data structure, called sparse feature tree, is used to represent the feature set to speed up the process of feature search (or feature matching), so that speed up the process of probability calculation and execution system. Experiments on chunking recognition and Part-of-Speech tagging are conducted to show that the new data structure greatly speeds up the feature matching process while keeping the same space complexity.

Key words: sparse feature tree; maximum entropy model; feature matching

摘 要: 在大规模或实时环境要求下,机器学习算法的计算效率非常重要.描述了用于最大熵模型执行系统的一种高效的数据结构及其相关的生成和查找算法.这种数据结构称为稀疏特征树,用于表示特征集合,以提高特征查找(或特征匹配)的速度,从而提高概率计算和执行系统的速度.基本短语识别和词性标注的实验显示,这种新的数据结构的确能够极大地加快最大熵方法执行系统的速度,同时保持空间复杂度不变.

关键词: 稀疏特征树;最大熵模型;特征匹配
中图法分类号: TP181 文献标识码: A

最大熵方法在自然语言处理方面的应用非常广泛.1992 年,Della Pietra 首先把最大熵方法应用于自然语言处理[1].十几年来,最大熵方法在自然语言处理领域的成果越来越令人瞩目.1996 年,Berger 等人提出了解决条件最大熵方法两个基本任务(特征选择和模型选择)的基本算法,并把它用于机器翻译、词序重排和断句等任务[2].Rosenfeld 把最大熵模型用于语言模型[3,4].Adwait Ratnaparkhi[5]在最大熵方法用于自然语言处理方面做了一系列工作,包括把它用于介词短语修饰问题[6]、词性标注[7]、断句[8]和句法分析[9].这些基于最大熵模型的词性标注器、句法分析器的性能都达到或接近目前报导的最好水平.最大熵方法在自然语言处理方面的应用还有:短语识别[10]、指代消解[11]、文本分类[12]、问题回答[13]等.虽然近年来最大熵方法得到了发展,但仍有待于进一步

* Supported by the National Natural Science Foundation of China under Grant No.60435020 (国家自然科学基金)
Received 2005-02-14; Accepted 2005-08-25

从速度和性能上加以提高.

当特征选择和模型选择(参数计算)完成时,整个最大熵模型就建立了起来.在执行过程中要考虑如何高效地使用这个模型,也就是计算概率 $p(y|x)$.关于最大熵模型方面的大部分工作都围绕在模型训练方面,在特征匹配方面的工作几乎没有,原因有两个:第一,对于特征集合或测试集合的规模不大的任务,高效的特征匹配不是必需的;第二,虽然在参数估计和特征选择过程中也需要特征匹配的过程,但这个过程只需要一次.然而在实际任务中,特征集合的规模往往很大[2,5,10],例如文献[5]提到,在其词性标注的任务中使用了 119 910 个特征.在执行过程中,测试数据往往非常大,或者要求实时处理,所以提高特征匹配过程的速度非常有必要.我们将提出一种新的、高效的表示最大熵模型特征集的数据结构及其相应的特征匹配算法.

1 条件最大熵执行系统

条件最大熵模型最初的任务或者说最大熵训练系统的任务是估计条件概率 $p(y|x)$.但事实上,由于条件最大熵模型使用了一个近似或假设 $p(x)=\tilde{p}(x)$,在训练的时候,只需估计 $\tilde{p}(x)\neq 0$ 的实例 x 的条件概率,也就是在训练样例中出现的实例的条件概率.训练样例中未出现过的实例通过特征参数对$\{(f,\lambda)\}$,直接用下面的公式计算出来:

$$p(y\mid x)=\frac{\exp\sum_{i=1}^{n}\lambda_i f_i(x,y)}{\sum_{y=1}^{Y}\exp\sum_{i=1}^{n}\lambda_i f_i(x,y)} \tag{1}$$

其中:x 表示实例或事件的条件;y 表示目标概念类,取值范围在$\{1,2,\ldots,Y\}$;f_i 就是所谓的指示函数或特征值,当 x 和 y 满足特定的条件时,它为真;λ_i 是特征 f_i 的权重或参数.

通常,训练样例的实例集合只能覆盖实例空间很小的一部分.然而在执行时,实例空间中的任何实例执行系统都可能会遇到.现有的计算机存储量设备是无法存储实例空间中所有实例的条件概率的,但可以存储数量相比小得多的特征及其相应参数的集合.所以,最大熵训练系统的一般输出一组特征参数对$\{(f,\lambda)\}$,而执行系统必须准备随时计算 $p(y|x)$.

1.1 特征的表示

由于本文将描述特征匹配,会涉及到数据表示的一些细节,所以这里首先介绍特征表示的问题.假设事件有 M 个名义型(或符号型)属性,$a_1,a_2,\ldots,a_M$,属性 a_i 取值范围为$\{1,2,\ldots,n_i\}$.其中:n_i 称为第 i 个属性的值数;M 称为属性的维数.那么,实例 x 可以以向量的形式表示成$(x_1,x_2,\ldots,x_M)$,其中 x_i 是属性 a_i 的值.

回顾一般特征的定义:

$$f(x,y)=\begin{cases}1, & \text{如果}x\text{和}y\text{满足一定的条件}\\ 0, & \text{否则}\end{cases} \tag{2}$$

为了简化特征的表示,研究人员[2,5,10]一般只考虑属性合取的特征,那么,特征可以表示为 $f=(v_1,v_2,\ldots,v_M,t)$.其中:$v_i\in\{0\}\cup\{1,2,\ldots,n_i\}$,$v_i=0$ 意味着第 i 个属性可以“不用考虑”;t 是特征的目标概念类,$t\in\{1,2,\ldots,Y\}$.我们分别称 $v_1,v_2,\ldots,v_M$ 和 t 为这个特征的条件和目标概念类.特别地,我们把 $v_i\in\{0\}\cup\{1,2,\ldots,n_i\}$称为第 i 个属性的特征属性值.当在特征中提到属性值时,一般是指特征属性值.假设训练过程选择了 n 个特征,并且估计了相应的参数.那么,特征及其相应的参数可以以向量形式表示为$(v_1,v_2,\ldots,v_M,t,\lambda)$.请看一个简单的示例:假设任务的属性维数是3,目标概念类的个数是 2,特征集合有 5 个特征,那么特征集合的表示如图 1 所示.

如果在特征的向量表示中,把“不用考虑”(属性值为 0)的属性省略,改写条件部分,特征 f 可以重写为 $f=(j_1{:}v_{j1},j_2{:}v_{j2},\ldots,j_k{:}v_{jk},t)$.其中:$k$ 称为特征的维数,表示特征中非 0 属性的个数;$j_i\in\{1,2,\ldots,M\}$是非 0 属性的编号,$v_{ji}\in\{1,2,\ldots,n_i\}$是其值.我们把这种特征表示方式称为特征的稀疏向量表示.那么,特征及其相应的参数可以以稀疏向量形式表示为$(j_1{:}v_{j1},j_2{:}v_{j2},\ldots,j_k{:}v_{jk},t,\lambda)$.那么,对于图 1 的特征集合,稀疏向量形式表示如图 2 所示.很显然,对于平均特征维数和属性维数比小于 0.5 的特征集合,稀疏向量表示比普通向量表示更加节省空间.本文中将

要提到的其他数据结构和算法都将以特征的稀疏向量表示为基础.

f_1:(1,0,0;1,λ_1)
f_2:(2,0,1;1,λ_2)
f_3:(0,2,0;2,λ_3)
f_4:(0,2,2;2,λ_4)
f_5:(0,0,0;1,λ_5)

Fig.1 An example of feature set: represented as Vector

图 1 特征集例子:向量表示

f_1:(1:1;1,λ_1)
f_2:(1:2,3:1;1,λ_2)
f_3:(2:2;2,λ_3)
f_4:(2:2,3:2;2,λ_4)
f_5:(;1,λ_5)

Fig.2 An example of feature set: represented as Sparse vector

图 2 特征集例子:稀疏向量表示

1.2 特征匹配

如果把式(1)中的特征值是 0 的因子省略,那么,$p(y|x)$可以重写为

$$p(y\mid x)=\frac{\exp\sum_{i=1}^{n}\lambda_i f_i(x,y)}{\sum_{y=1}^{Y}\exp\sum_{i=1}^{n}\lambda_i f_i(x,y)}=\frac{\exp\sum_{i=1,f_i(x,y)\neq 0}^{n}\lambda_i f_i(x,y)}{\sum_{y=1}^{Y}\exp\sum_{i=1,f_i(x,y)\neq 0}^{n}\lambda_i f_i(x,y)}=\frac{\exp\sum_{i\in mfs(x,y)}\lambda_i f_i(x,y)}{\sum_{y=1}^{Y}\exp\sum_{i\in mfs(x,y)}\lambda_i f_i(x,y)} \tag{3}$$

其中, $mfs(x,y)=\{i\mid f_i(x,y)\neq 0,1\le i\le n\}$.所以,为了计算 $p(y|x)$,必须找到匹配实例 x 的所有特征,我们用 $mfs(x)=\bigcup_y mfs(x,y)$ 来表示匹配实例 x 的特征集合,把搜索集合 $mfs(x)$的过程称为特征匹配.很明显,集合 $mfs(x)$只与 x 有关.

下面介绍特征值的计算过程.假设给定向量表示的实例 $x=(x_1,x_2,\ldots,x_M)$和稀疏向量表示的特征 $f=(j_1:v_{j1},j_2:v_{j2},\ldots,j_k:v_{jk},t)$,那么,$f(x,y)$可按下式计算:

$$f(x,y)=\begin{cases}1, & \text{如果}t=y\text{并且}\forall i,1\le i\le k:\ v_{ji}=x_{ji}\\ 0, & \text{否则}\end{cases} \tag{4}$$

计算概率 $p(y|x)$的主要任务就是特征匹配.由于特征匹配是一个集合查找的过程,而传统的查找方法只能查找单个元素,所以无法直接用于特征匹配.一种最直接的解决方法是依次比较特征集合中的每个特征和实例 x,找到所有匹配的特征.

比较高效的解决方案是:用特征模板(表示单个属性或属性组合)把特征集合切割成子集,属于同一个特征模板的特征组成一个子集合.在每一个子集合中,与某个实例 x 匹配的特征可能有多个,但这些特征有相同的特征条件.所以,再把子集合中的特征按特征条件归类.这样,找到了匹配的特征条件就等于找到了所有匹配的特征.而在子集合中查找匹配的特征条件就可以化为一个多维属性的查找问题.对于图 1 和图 2 中的例子,由于 5 个特征属于5个不同的特征模板,所以分成5个子集合:$\{f_1\}$,$\{f_2\}$,$\{f_3\}$,$\{f_4\}$和$\{f_5\}$.我们称这种方法为基于特征子集的二分查找方法.它将作为我们提出的更高效的方法的比较对象.

2 稀疏特征树

在本文中,我们基于特征的稀疏向量表示,提出一种树状的数据结构——稀疏特征树(简称特征树)来表示特征集合(或对特征集合建索引),以提高特征匹配的效率.下面将具体描述这种数据结构.

假设训练过程选择了 n 个特征,并且估计了相应的参数.那么,第 i 个特征及其相应的参数可以表示成 $(j_{1i}:v_{j1i},j_{2i}:v_{j2i},\ldots,j_{ki}:v_{jki},t_i,\lambda_i)$.特征集合可以用一棵树来表示,如图 3 所示.在图 3 中,a_1,a_{j1},a_{j2},a_M 等分别表示特征的属性;$0\le j_1,j_2\le M$ 表示属性编号,$0\le i_1,i_2\le n_{j1}$ 表示属性值;$1\le k_1,k_2\le Y$ 表示是目标概念类;“NULL”表示指针为空.特征树的层数和特征集的最大特征维数 k 相关,有$(3+2k)$层.它有 5 类节点:根节点、属性节点、属性值节点、指示节点、目标概念及参数节点.由于它没有一定的层次分布特性,下面将按节点类别来说明整棵树的结构.

位于第 0 层的根节点指向 0 到 M 个属性节点,以及 0 或 1 个指示节点.

属性节点为图中椭圆形的节点,每个节点表示一个属性 a_i,它要么指向空,要么指向一个表示 a_i 值的属性值

节点.

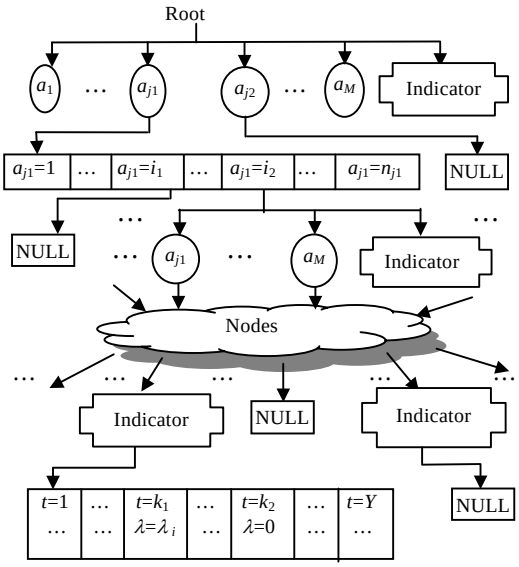

Fig.3 The structure of the sparse feature tree

图 3 稀疏特征树的结构

属性值节点为图中长条格节点,每个节点包括 n_i 个子节点,每个子节点对应于{1,2,…,n_i}中的单个属性值,每个节点表示以特定的 i–1 个属性值开始的所有特征集合,这些节点同时也表明了一个依赖于第 i 个属性值的 n_i-路的分支.每个子节点指向 0~M 个属性节点,以及 0 或 1 个指示节点.在实际中,我们并不生成空的子节点.

指示节点为图中十字形的节点,每个指示节点要么指向空,要么指向一个目标概念及参数节点.为表示方便,图中把空的指示节点画了出来,而在实际中,只保留非空的指示节点.而引入指示节点的目的是分割属性值层和目标概念及参数层,使具有相同特征条件的特征在同一个节点(指示节点)下,以方便查找.

目标概念及参数节点(叶子节点)为图中长条格节点,每个节点包括 Y 个子节点,其中每一维对应于{1,2,…,Y}中的单个目标概念类,每个节点表示具有相同条件的一个或多个特征,这些节点同时也表明了一个依赖于目标概念类的 Y-路的分支.每个子节点要么表示特征集中的一个特征及其参数,要么指向空.在实际中,我们并不生成空的子节点.所以,如果特征集合有 n 个特征,那么目标概念及参数层就有 n 个非空子节点.

对于属性维数与特征维数的比值较大的模型,这是一种比较高效的数据结构.在实际应用中,模型的特征维数不会取得过大,而属性维数可以非常大.所以,它是一种实用的数据结构.若把树中包含 k 个子节点的节点都当作 k 节点来计数,从理论上分析,在最坏的情况下,当所有的非根节点都只有一个儿子时,特征树所需空间的复杂度是(n×(AFD×2+2)+1).这里,n 表示特征的个数,AFD 表示平均特征维数.

2.1 稀疏特征树的生成和查找

从图 3 中可以看到,特征树的属性值节点是多维的.而在自然语言处理的实际应用中,有些属性(例如单词)的取值可能上万.所以,使用高效的数据结构来对子树的导航进行决策是相当必要的.关于子树的导航决策结构有多种选择:数组、链表和二分查找树.为了简单明了起见,图 3 中所采用的是数组结构,然而在实际应用中,可以采用能结合数组在时间和链表在空间上的优势的二分查找树来管理这些节点.也就是说,只保留非空的节点以节省空间,同时保持子节点的按属性值的排列次序,这样可以使用二分查找以节省时间.

特征树的生成算法如算法 1 所示.这个算法首先生成根节点;然后生成根节点和指示节点之间包括指示节点的属性节点和属性值节点;最后生成指示节点下所有的目标概念及参数节点.这个算法使用一个队列作为广度优先的建树策略的载体.图 2 的特征集合按照算法 1 实际生成的特征树如图 4 所示.

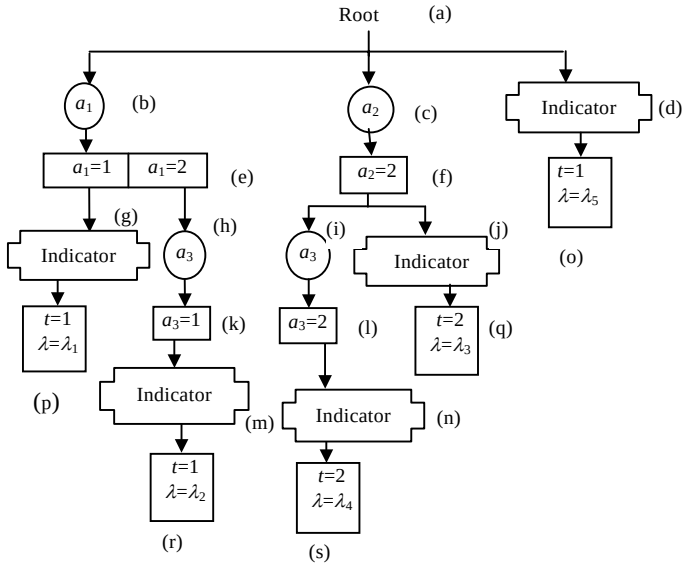

Fig.4 An example of the sparse feature tree

图 4 稀疏特征树例子

算法 **1**. 稀疏特征树生成算法.

1. 生成根节点:把对应到整个特征集合的根节点从后面推入队列.
2. 生成属性层和属性值层:
   - 2.1. 若队列的首节点是指示节点,则转到第 3 步;否则,移出该节点;
   - 2.2. 若该节点是属性值节点(假设对应于第 i 个属性)或根节点(i=0),那么,把该节点对应的所有特征按最小的非 0 属性分类:假设共 $j(0\le j\le M-i)$类,生成 j 个节点作为该节点的儿子,分别对应于这 j 类特征.把这些儿子节点推入队列;如果特征集合中存在没有非 0 属性的特征,那么生成指示节点作为根节点的儿子来对应于这些特征;
   - 2.3. 若该节点是属性节点(假设层次为 i),那么把该节点对应的所有特征按第 i 个属性值排序分类:假设共 $n(1\le n\le n_i+1)$类,生成 n 个节点作为该节点的儿子,分别对应于这 n 类特征.把这些儿子节点推入队列;
   - 2.4. 重复第 2 步.
3. 生成目标概念及参数层:
   - 3.1. 若队列为空,则结束;
   - 3.2. 从队列中移出首节点,假设该节点对应 $n(1\le n\le Y)$个特征,生成 n 个参数节点作为该节点的儿子;
   - 3.3. 重复第 3 步.

算法 2 描述了在特征树中查找匹配实例 $x=(x_1,x_2,\ldots,x_M)$的所有特征的过程.第 1 步是初始化.第 2 步是特征查找,这其实是一个从特征树的第 1 层到叶子层,从上到下查找匹配实例 x 的所有特征的过程.如果没有特征匹配实例 x,那么这个过程可能在特征树的某一层结束.第 3 步是累计λ,也就是把第 2 步找到的所有特征按目标概念的类别 $y(1\le y\le Y)$分别累计权重,得到未归一化的 $p(y|x)$.

算法 **2**. 稀疏特征树查找算法.

1. 初始化:把根节点的所有儿子节点从后面推入队列.
2. 特征查找:
   - 2.1. 若队列为空,则转到第 3 步;否则,移出首节点;

2.2. 若该节点为指示节点,则保存该节点;

2.3. 若该节点为属性节点(第 i 个属性),那么,如果该节点所指的属性值节点中存在值为 x_i 的子节点,则把这个子节点推入队列;

2.4. 若该节点为属性值节点,则把该节点的所有儿子节点推入队列;

2.5. 重复第 2 步.

3. 累计λ:把所有保存的指示节点下的目标概念及参数节点按目标概念的类别 $y(1\le y\le Y)$分别累计权重.

例如,我们根据图 4 的特征树来计算实例 $x(1,2,1)$的两个条件概率 $p(1|x)$,$p(2|x)$.特征匹配过程中队列中的节点和保存的指示节点(在括号中分号之前表示队列中的节点,分号之后表示保存的指示节点)是这样的:(*bcd*;),(*cde*;),(*def*;),(*ef*;*d*),(*fg*;*d*),(*gij*;*d*),(*ij*;*dg*),(*jl*;*dg*),(*l*;*dgj*),(;*dgj*).所以,最后保留下 3 个指示节点 *d*,*g* 和 *j*,对应特征 f_1,f_3 和 f_5,那么,$p(1|x)=\exp(\lambda_1+\lambda_5)$,$p(2|x)=\exp(\lambda_3)$.

显然,对于所有特征的维数为 1 的集合,基于特征树的查找方法就退化成了基于特征子集的二分查找方法.

3 实验分析

我们设计了许多实验来证实稀疏特征树这种数据结构及其查找算法的合理性.这些实验基于英文短语识别[14]和词性标注[7].所有实验都使用了一个 1.13Ghz CPU 和 768MB 内存的笔记本计算机.在短语识别任务中采用 SGC 算法[15],在词性标注任务中采用频数阈值方法[7],分别选取了 2000,4000,8000,16000,32000,64000 个特征来分别比较相应的时间.

图 5 和图 6 中的“FM”表示特征匹配时间,包括特征查找和λ累计的时间.而“FM-ftree”表示的是基于特征树的特征匹配时间;“FM-bisearch”表示的是基于特征子集二分查找的特征匹配时间.“probability”表示条件概率计算时间,包括特征匹配时间和条件概率初始化和归一化的时间.“executing”表示应用系统执行时间,包括知识库装入、特征装入、实例向量的数据转换、条件概率计算以及结果输出等所有的时间.这里的“条件概率计算”和“执行时间”都基于特征子集二分查找.

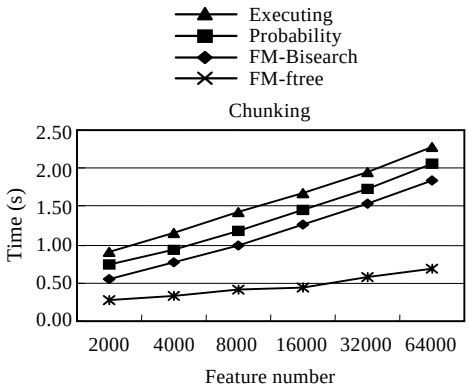

Fig.5 Comparing the time for chunking

图 5 时间比较:基本短语识别

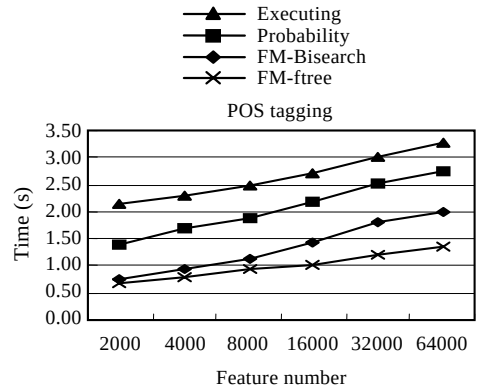

Fig.6 Comparing the time of POS tagging

图 6 时间比较:词性标注

从图 5 和图 6 中我们可以看到 3 个事实.第一,即使已经使用了比较高效的特征匹配方法,特征匹配时间在应用系统的执行时间中占有很大的比例,当特征数量增加时,这个比例也随之增加.所以,提高特征匹配速度在实际应用中非常有必要.第二,特征树在这两个任务中确实比二分查找方法更为快速.当特征数量增加时,特征树在速度上的优势更加明显.第三,特征树与二分查找方法相比的优势在短语识别中更为明显,其原因是短语识别任务的平均特征维数比较大的缘故.

在实际应用中,短语识别和词性标注分别在 6 000 和 60 000 个左右特征的时候性能达到最佳.在性能最佳的时候,特征树比二分查找,在这两个任务中分别节约 39%和 20%的时间.

4 比较与讨论

本文描述的最大熵执行系统的特征匹配,实际上是在有通配符的多维名义型数据集合中查找某一无通配符的多维名义型数据的过程.多维数据的存储和查找并不是一个新的主题,在数据库方面,它至今还是一个重要的研究方向.过去的研究者已经提出了各种 tree 和 trie 来存储和查找多维数据,例如 trie 结构[16],k-d-trie[16],k-d-tree[17],AD-tree[18]等.但这些数据结构都不能解决最大熵执行系统的特征匹配的问题.文献[16]把 trie 定义为一种 *m*-路的树.trie 表示一本词典,它提供的是查找单个单词的功能,trie 中每个单词是由特定的字符组成的.而特征树表示的是一个特征集合,它提供的是查找特征集合的功能,并且特征中包含了通配符,所以 trie 结构不是特征树.k-d-trie 和 k-d-tree 都是用于多维数据的部分匹配问题,其任务是在样例集合中查找匹配某个特征的所有样例,而特征匹配是在特征集合中查找匹配某个样例的所有的特征,所以,k-d-trie 或 k-d-tree 不是特征树.AD-tree 是 k-d-tree 的一个变种,在机器学习算法中用于快速计数训练样例,所以,AD-tree 也不是特征树.

本文中所描述的特征树方法也适用于联合最大熵模型.这是因为联合最大熵模型的特征和条件最大熵模型的特征只相差一个目标概念类 *y*,而特征匹配是只考察特征的条件部分与实例 *x* 的关系.

References:

[1] Pietra SD, Pietra VD, Mercer RL, Roukos S. Adaptive language modeling using minimum discriminant estimation. In: Proc. of the DARPA Workshop on Speech and Natural Language. San Mateo: Morgan Kaufmann, 1992. 103−106. http://www.cs.mu.oz.au/acl/H/H92/H92-1020.pdf

[2] Berger AL, Pietra SAD, Pietra VJD. A maximum entropy approach to natural language processing. Computational Linguistic, 1996, 22(1):39−71.

[3] Rosenfeld R. Adaptive statistical language modeling: A maximum entropy approach [Ph.D. Thesis]. Pittsburgh: Carnegie Mellon University, 1994.

[4] Rosenfeld R. A maximum entropy approach to adaptive statistical language modeling. Computer, Speech and Language, 1996,10: 187−228.

[5] Ratnaparkhi A. Maximum entropy models for natural language ambiguity resolution [Ph.D. Thesis]. Philadelphia: University of Pennsylvania, 1998.

[6] Ratnaparkhi A, Reynar J, Roukos S. A maximum entropy model for prepositional phrase attachment. In: Proc. of the ARPA Workshop on Human Language Technology. San Mateo: Morgan Kaufmann, 1994. 250−255. http://acl.ldc.upenn.edu//H/H94/H94-1048.pdf

[7] Ratnaparkhi A. A maximum entropy model for part-of-speech tagging. In: Brill E, Church K, eds. Proc. of the Conf. on Empirical Methods in Natural Language Processing. Somerset: Association for Computational Linguistics, 1996. 133−142.

[8] Reynar JC, Ratnaparkhi A. A maximum entropy approach to identifying sentence boundaries. In: Proc. of the 5th Conf. on Applied Natural Language Processing. San Mateo: Morgan Kaufmann, 1997. 16−19. http://acl.ldc.upenn.edu/A/A97/A97-1004.pdf

[9] Ratnaparkhi A. Learning to parse natural language with maximum entropy models. Machine Learning, 1999,34(1-3):151−176.

[10] Koeling R. Chunking with maximum entropy models. In: Proc. of the CoNLL-2000 and LLL-2000. 2000. 139−141. http://acl.ldc.upenn.edu/W/W00/W00-0729.pdf

[11] Luo XQ, Ittycheriah A, Jing HY, Kambhatla N, Roukos S. A mention-synchronous coreference resolution algorithm based on the. bell tree. In: Proc. of the 42nd Annual Meeting of the Association for Computational Linguistics. 2004. 135−142. http://acl.ldc.upenn.edu/acl2004/main/pdf/243_pdf_2-col.pdf

[12] Nigam K, Lafferty L, McCallum A. Using maximum entropy for text classification. In: Proc. of the IJCAI'99 Workshop on Machine Learning for Information Filtering. 1999. 61−67. http://www.kamalnigam.com/papers/maxent-ijcaiws99.pdf

[13] Ittycheriah A, Roukos S. IBM's statistical question answering system for trec-11. In: Voorhees EM, Buckland LP, eds. Proc. of the TREC-11 Conf. Gaithersburg: NIST, 2002. 394−401. http://trec.nist.gov/pubs/trec11/papers/ibm.ittycheriah.pdf

[14] Zhou YQ, Guo YK, Huang XJ. Chinese and English BaseNP recognition based on a maximum entropy model. Journal of Computer Research and Development, 2003,40(3):440−446 (in Chinese with English abstract).

[15] Zhou YQ, Weng FL, Wu LD, Schmidt H. A fast algorithm for feature selection in conditional maximum entropy modeling. In: Proc. of the 2003 Conf. on Empirical Methods in Natural Language Processing. 2003. 153−159. http://acl.ldc.upenn.edu/W/W03/W03-1020.pdf

[16] Knuth DE. The Art of Computer Programming, Volume 3, Sorting and Searching. Addison-Wesley, 1973. Algorithm based on the Bell tree. ACL 2004. 135−142.

[17] Bentley JL. Multidimensional binary search trees used for associative searching. Communications of the ACM, 1975,18(9): 509−517.

[18] Moore A, Lee MS. Cached sufficient statistics for efficient machine learning with large datasets. Journal of Artificial Intelligence Research, 1998,8:67−91.

附中文参考文献:

[14] 周雅倩,郭以昆,黄萱菁,吴立德.基于最大熵方法的中英文基本名词短语识别.计算机研究与发展,2003,40(3):440−446.

周雅倩(1976－),女,江苏太仓人,博士,讲师,主要研究领域为自然语言处理.

吴立德(1937－),男,教授,博士生导师,主要研究领域为自然语言处理,视频处理.

黄萱菁(1972－),女,博士,副教授,主要研究领域为自然语言处理.

关于征集中国计算机事业 50 周年大事记的通知

为纪念中国计算机事业创建 50 周年，中国计算机学会决定编辑出版“中国计算机事业五十周年大事记”。在“大事记”的基础上，由学会选评中国计算机事业发展历程中的 50 件大事。编辑出版完成后，在今年 10 月下旬举行的“纪念中国计算机事业五十周年”活动上发布。

各相关单位或个人均可书面提供“大事记”的内容。“大事记”将反映对中国计算机事业发展有重要影响的事件、项目、发明或成果，“大事记”记录中还包括与该事件相关的主要单位和/或主要人士。

“大事记”提供者须完整填写“大事记”推荐表(推荐表请从学会网站上下载)。

电子邮件发至：ccf@ict.ac.cn；原件同时寄至：北京 2704 信箱中国计算机学会，100080，注明“大事记”字样，也可传真至：010-62527485。

事件征集时间范围：1956 年~2005 年

征集截止日期：2006 年 6 月 30 日