音节切分歧义方法研究

刘政怡,吴建国,刘慧婷
(安徽大学 计算机科学与技术学院,安徽 合肥 230039;
安徽大学 计算智能与信号处理教育部重点实验室,安徽 合肥 230039)

摘 要:音节切分是整句拼音转换的基础,由于拼音的特殊性,存在歧义切分的可能。如果采用最少分词算法只能得到一种切分结果,不能保证整句拼音转换的正确性。提出一种音节切分算法,通过插入音素节点不断构造合法音节节点,进而生成状态空间,遍历算法遍历状态空间可获得所有的切分可能,而当用户进行删除操作时,只需删除部分相关节点。整个状态空间随用户的操作进行局部调整,分布均匀。该算法有利于存在歧义切分问题的整句拼音转换,可从保留下来的所有切分可能中选出一个全局最优的语句候选,保证整句转换的正确性。

关键词:音节切分; 切分算法; 切分歧义; 整句输入; 状态空间
中图分类号:TP391.1 **文献标识码**:A **文章编号**:1673-629X(2008)08-0035-04

Research on Syllable Segmentation Method

LIU Zheng-yi,WU Jian-guo,LIU Hui-ting
(School of Computer Science and Technology,Anhui University, Hefei 230039,China;
Ministry of Education Key Laboratory of Intelligent Computing & Signal Processing,
Anhui University,Hefei 230039,China)

Abstract:Syllable segmentation lays the foundation for sentence pinyin conversion. As a result of the particularity of pinyin syllable segmentation has different ways. One partition result can be got by least participle algorithm so that correctness of sentence pinyin conversion can not be guaranteed. Presents a syllable segmentation algorithm which is composed of two kinds of nodes. One is phoneme node, the other is syllable node. Phoneme nodes are integrated into syllable node. They link with chain to form state space. Can get all the possible partition results by traversal algorithm. When encountering user's deletion operation it only need to delete some nodes related. State space is adjusted locally and equably along with user's operation. This algorithm is in favor of sentence conversion with different partition ways. It can get an optimal answer from all the possible partition.

Key words:syllable segmentation;partition algorithm;various segmentation;sentence input method;state space

1 音节切分歧义问题

拼音输入是一种最简单的汉字输入方法,它的发展非常快,从第一代的以字输入为主,发展到第二代以词为主并具有智能调频功能,现在拼音输入已经发展到了第三代,即语句输入。但是在语句输入实现过程中,如果拼音串不含分隔符号,可能存在歧义[1],如果仅采用最少分词算法[2,3],可能导致错误切分,使得整个音字转换过程失去意义,无法得到正确的候选语句。有人说,音节切分的歧义问题可以通过要求用户提供隔音符来解决[4,5],例如将"西安"的歧义拼音码"xian"改为"xi'an"。但是这样做明显加重了用户的负担,与整句输入法为用户提供方便的宗旨相违背。用户需要判断什么时候有可能出现歧义,需要加隔音符,什么时候不需要。对于常见的歧义,用户可能比较容易判断,但对于不常见的,用户可能由于词汇量的限制没办法很快做出反应,例如:在准备通过拼音码"xiangai"输入"相爱"时,可能不会想到会有"宪改"这个词与之发生混淆。这只是局限于词的输入,如果是在句子输入中,例如: wohenfanganzhezhongshiqing(我很反感这种事情),如果采用最少分词算法,该句子被切分为 wo'hen'fang'an'zhe'zhong'shi'qing,之后无论再使用什么样的方法都无法得到匹配的语句。那是因为过早的切分错误,将正确的语句排除在选择之外,文中提出一种音节切分算法可保留所有的切分可能,再使用统计和语法相结合的方法在所有可能的候选中进行选择,保

收稿日期:2007-11-29
基金项目:安徽大学人才队伍建设经费资助项目(02203105)
作者简介:刘政怡(1978-),女,博士,讲师,主要研究领域为自然语言处理;吴建国,博士,教授,主要研究领域为中文信息处理。

证获得全局最优的语句候选。

2 音节切分算法

为保留所有的切分可能,定义两类节点:拼音音素节点(PYCharNode)和合法拼音节点(LegalPYNode),如图 1。音素指的是从音节中分析出来的最小语音单位,例如:"绿"可以分析出 l、i 和 ü 三个音素,"红"可以分析出 h、o、n、g 四个音素。

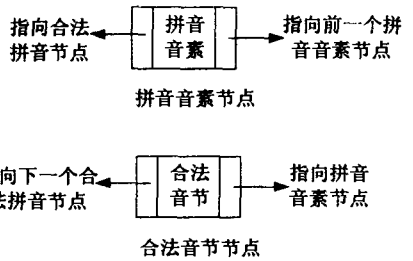

图 1 节点定义

2.1 插入算法

音节切分结果随着用户的输入随时变化。当用户输入一个拼音音素时,即在状态空间中插入一个拼音音素节点,它与前面已经输入的拼音音素可能组成的合法拼音节点都需要插入到状态空间模型中。

插入算法的思想是,首先向中心链结尾插入新的拼音音素节点,使之成为尾节点,然后从尾节点开始遍历中心链,当遍历到一个拼音音素节点时,若从当前节点到尾节点的拼音音素序列所组成的拼音音节是一个合法的音节,则将该音节插入以当前拼音音素节点的前一个节点为头的合法拼音链中,并将该合法拼音节点的 pPYCharNode 域指向末尾节点,循环这一过程直至当前节点的拼音音素符号为 root 或者拼音音素个数已大于最长合法拼音音节长度。

算法 1 插入一个拼音字母节点。

```
PYCharNode_ Insert (PYCharNode * finalPYCharNodePointer,
char pinyin){
    pyCharNode = new PYCharNode ();
    pyCharNode. PinYinChar = pinyin;
    pyCharNode. pPrePYCharNode = finalCharPYNodePointer;
    pyCharNode. pLegalPY = NULL;
    finalCharPYNodePointer = pyCharNode;
    p = finalPYCharNodePointer;
    while (! Empty(p)){
      IniStack(S);
      curPoint = finalPYCharNodePointer;
      p = p->pPrePYCharNode;
      while (curPoint ! = p){
       Push(S,curPoint);
       curPoint = curPoint->pPrePYCharNode;
      }
      //从 S 栈底到栈顶所有节点中的字符组成 curPY-
CharList;
      GetPYList(S,curPYCharList);
      //找出拼音音素组成的合法音节
      if (curPYCharList 是合法音节){
       legalPyNode = new LegalPYNode (curPYCharList);
       LegalPYNode -Insert (p, legalPyNode);
         legalPyNode -> pPYCharNode = finalPYCharN-
odePointer;
      }
    }
}
```

算法 2 插入一个合法音节节点。

```
LegalPYNode_ Insert (PYCharNode * headLegalPYNodePointer,
LegalPYNode legalPYNode){
    p1 = headLegalPYNodePointer ->pLegalPY;
    if(p1 = =NULL){
    headLegalPYNodePointer -> pLegalPY = legalPYNode;
    legalPYNode ->pNextLegalPYNode=NULL;
    }else{
    while(p1 -> pNextLegalPYNode! =NULL){
    p1 = p1 -> pNextLegalPYNode;
    }
    p1 ->pNextLegalPYNode= legalPYNode;
    }
}
```

该算法的时间主要花费在遍历中心链、判断 curPYCharList 是否是合法音节以及插入合法音节节点上。假设拼音音素序列长度为 n,合法音节个数可看作常数 T,在合法音节集合中顺序查找需花费常数时间,插入合法音节节点,需花费常数时间,则总共花费时间为 $O(n * T)$。

2.2 遍历算法

构造好状态空间模型中的各节点后,可以通过遍历算法得到所有可能的拼音组合。

遍历状态空间的基本思想:初始化栈 Stack,从首节点 Root->pLegalPY 开始遍历。当遍历到某一节点 p 时,p 入栈,在 p 不是叶节点的情况下,如果 p->pPYCharNode->pLegalPY 中有节点未遍历则遍历该节点,否则遍历 p->pNextLegalPYNode;在 p 是叶节点的情况下,则把当前合法音节放入拼音音节集合 SetOfPYList 中,加入一个拼音分隔符号,否则算法从栈中上一个节点继续遍历,循环这一过程直至栈空。

算法 3 遍历状态空间算法。

```
Traversal_Graph(PYCharNode * Root, SetOfPYList Set){
    InitStack(S);
    if (Root = = NULL) return;
```

```
    curPoint = Root->pLegalPY;
    while (curPoint){
    //从S的栈底到栈顶所有节点的PinYinChar域中的字符组成curPYStr,且以'分隔;
    curPYStr=GetPYStr(S);
    if (curPoint->pPYCharNode->pLegalPY){
    Push(S,curPoint);
    curPoint = curPoint->pPYCharNode-> pLegalPY;
    continue;
    }else if (curPoint是叶节点){
    curPYStr=curPYStr+"\'"+curPoint->sLegalPY;
    Add(SetOfPYList, curPYStr);
    Pop(S,curPoint);
    }
    do{
    if (curPoint->pNextLegalPYNode == NULL){
    Pop(S,curPoint);
    }
    }while (! Empty(S)&& (curPoint->pNextLegalPYNode ==NULL));
    if (curPoint ≠ NULL) curPoint = curPoint->pNextLegalPYNode;
    }
}
```

该算法是用栈模拟非递归来实现的,其时间复杂度和空间复杂度较利用递归方法的小。利用音素构造网状搜索空间,再通过遍历这个空间,就可以得到所有可能的切分结果集合。

2.3 删除算法

当用户删除某个音素时,对应地进行删除操作。删除算法的思想是,删除中心链中的尾节点,更新该链的尾节点,然后在状态空间中心链上各个节点对应的合法音节链中查找,如果链中的节点的右指针域指向原来的尾节点,那么删除该节点。

算法 4　删除拼音音素节点。

```
void DeletePYNode (PYNode *&pHeadList){
    PYNode *p;
    p = pHeadList->pPrePYNode;
    while(p! =NULL){
    curCandidateNode=p->pCandidateList;
    while(curCandidateNode! =NULL){
    if(curCandidateNode->pPYNode== pHeadList){
    DeleteCandidateNode(curCandidateNode);
    }
    curCandidateNode =curCandidateNode->pNextCandidate;
    }
    p=p->pPrePYNode;
    }
    pHeadList = pHeadList->pPrePYNode;
}
```

删除算法的优点在于只需更新用户删除的那个音素节点对应的合法音节节点,对于已经输入的其他音素,不用重新构造。算法的时间复杂度也是 $O(n*T)$。

3 音节切分结果

利用提出的音节切分算法对存在歧义的音节进行切分,实验证明该算法是切实可行的。下面以两个例子验证该算法。第一个例子 fangan,见图 2,通过该算法可得到 fang' an(方案)和 fan'gan(反感)两种切分可能。当用户输入 f,由于 f 不是合法音节,不做任何操作;接着用户输入 a,a 是合法音节,将其插入到 f 的合法音节链中,且其 pPYCharNode 指针指向 a,接着沿着中心链往前,f 和 a 组合成合法音节 fa,将其插入到 root 的合法音节链中,且其 pPYCharNode 指针指向 a;用户接着输入 n,首先判断 n,因其不是合法音节,忽略,接着沿着中心链指针往前与 a 组合成 an,an 是合法音节,插入到 f 的合法音节链中,再接着往前与 f 组合成 fan,fan 也是合法的,将其插入 root 的合法音节链中,以此类推,整个状态空间构造完成。可以看出整个构造过程完全对应于用户的输入,随着用户输入不断变化。最后通过遍历算法获得两种切分可能 fang'an 和 fan'gan。

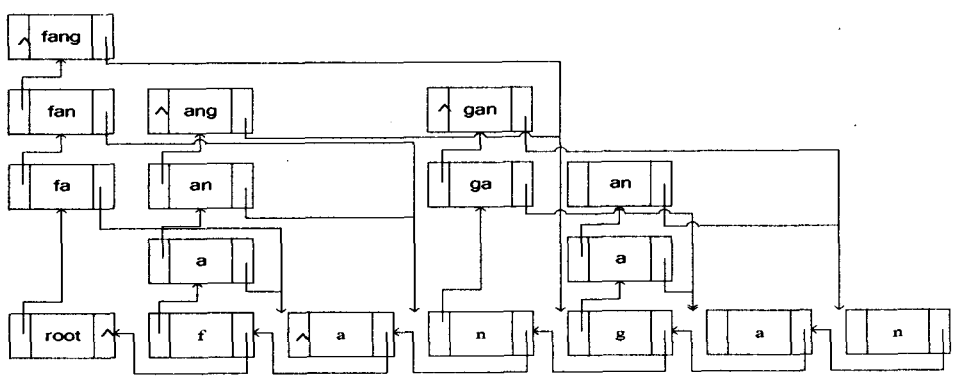

图 2　fangan 的切分算法状态图

第二个例子取自句子的一部分 xiangang(西安刚...),见图 3,通过该算法得到四种切分可能:xi'an'gang,xian'gang,xiang'ang,xi'ang'ang,其中包含了正确的切分结果。如果采用最少分词算法,只能得到一种错误的 xiang'ang,在错误的基础上进行音字转换,

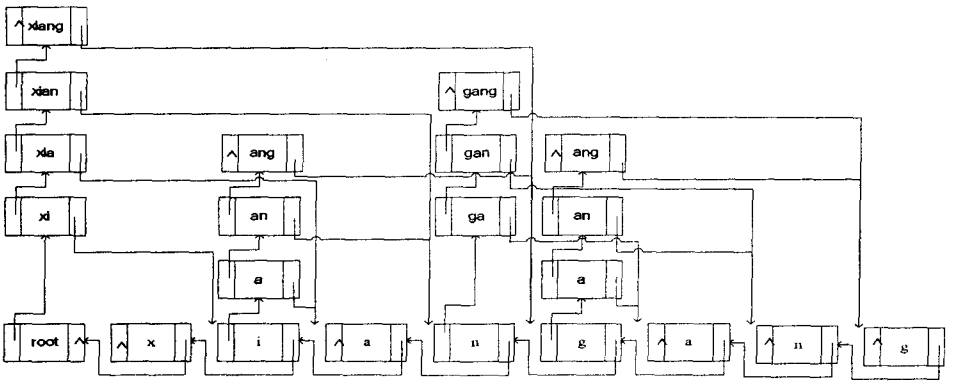

图 3 xiangang 的切分算法状态图

显然无法得到匹配的候选，而且也是一种浪费。

当用户删除最后一个音素节点，状态空间发生变化，如图 4。从音素节点 g 往前搜索，只要某合法音节节点的 pPYCharNode 指针域指向该节点，即由其产生的合法音节，在本例中有 ang 和 gang，则将其删除。最后删除该音素节点，更新中心链的尾指针。由遍历算法可知删除 g 后存在 xi'an'gan，xi'ang'an，xian'gan，xiang'an 四种切分可能。

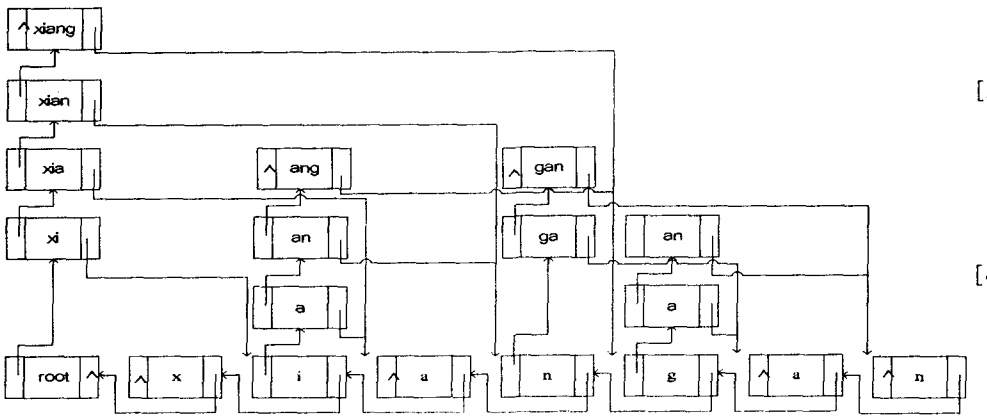

图 4 删除最后一个音素的状态空间

4 结束语

音节切分是整句拼音转换的前提和基础。为避免由于切分错误较早地将正确的语句排除在外，提出一种音节切分算法，通过插入、遍历操作可得到所有的切分可能，再由统计和语法方法可选出全局最优的候选语句。该算法虽然应用于音节切分解决歧义问题，但其方法适用于所有需要全局考虑的问题，将所有的可能性都包含其中。例如手写笔画输入法[6]中，通过方向码构造汉字时，该算法可获得所有可能的笔画序列，再从字库里查找对应的汉字。

参考文献：

[1] 吴根清.统计语言模型研究及其应用[D].北京：清华大学，2004.

[2] 王晓龙，王开铸，李仲蓉，等.最少分词问题及其解法[J].科学通报，1988(11)：1030－1032.

[3] 徐志明，王晓龙，姜守旭.一种语句级汉字输入技术的研究[J].高技术通讯，2000，10(1)：51－55.

[4] 张小衡.不完全拼音码的模版处理——兼谈微软拼音输入法的进一步完善[J].计算机工程与应用，2005(20)：74－101.

[5] 李 炜，贾庆成，刘政怡.汉语拼音输入法中拼音流的切分[J].现代计算机(专业版)，2007(8)：11－13.

[6] 樊庆林.基于笔画的联机手写汉字识别系统的研究与实现[D].合肥：安徽大学，2007.

(上接第 34 页)

法具有更好的调度结果，是一种有效的网格任务调度算法。

参考文献：

[1] Di Martino V. Scheduling in a grid computing environment using genetic algorithms[C]// In: Mililotti M. the 16th and Distributed Processing Symp(IPEPS 2002). Florida USA:[s. n.],2002.

[2] Xu Zhihong, Hou Xiangdan, Sun Jizhou. Ant algorithm - based task scheduling in grid computing[C]// In Proc of 2003 Canadian Conf on Electrical and Computer Engineering. Montreal, Canada: IEEE Computer Society Press, 2003: 1107 - 1110.

[3] Di Martino V, Mililotti M. Suboptimal scheduling in a grid using genetic algorithms[J]. Parallel Computing, 2004, 30: 553 - 565.

[4] Czajkowski K, Fitzgerald S, Foster I, et al. Grid Information Services for Distributed Resource Sharing[C]// In Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC - 10). [s.l.]: IEEE Press, 2001: 181 - 195.

[5] 钟求喜，谢 涛，陈火旺.基于遗传算法的任务分配与调度[J].计算机研究与发展，2000，37(10)：1197 - 1203.