一个 TTS 系统的实现方案

袁　嵩
（武汉科技大学计算机科学与技术学院，武汉 430081）
E-mail：yuansong_2002@163.com

摘　要　针对 TTS 技术在一些小型应用上存在着大量占用系统资源，执行速度慢等问题，该文介绍了一个高效而精简的汉语文语转换系统的实现方案。首先介绍了整个系统的设计思想，然后分步介绍了其具体实现。

关键词　文语转换　GBK　WAV 文件　语音合成

文章编号 1002-8331-(2004)21-0121-02　**文献标识码** A　**中图分类号** TP311

An Implementation Scheme of TTS System

Yuan Song
(College of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan 430081)

Abstract: In the light of some problems, such as a lot of system resources occupation, low execution speed and so on occurring in some simple applications of TTS technology, this paper gives an implementation scheme for simple and effective Chinese Text-to-speech system. After introducing the design ideas of the whole system, the paper discusses the process of this implementation scheme in detail.

Keywords: Text To Speech(TTS), GBK, WAV file, speech synthesis

1　引言

文语转换(Text To Speech，简称 TTS)，是将文字形式的信息转换成自然语音的一项技术，在人机语音交流、文字信息处理领域有着广泛的应用，例如人机对话、电话咨询、自动播音、助讲助读、语音教学、电话翻译等。汉语 TTS 指的是利用计算机将给定的汉字文本信息转换成汉语语音。随着 TTS 应用的普及，人们对 TTS 系统的要求也越来越高，尤其在一些小型应用上，大量占用系统资源，执行速度慢等弊端就成为 TTS 应用的瓶颈。结合以上问题，该文介绍了一个高效而精简的汉语文语转换系统的实现方案。

2　设计思想

实现这个汉语 TTS 系统主要完成两项工作：第一，建立一个语音库，记录在普通话中所有汉字的读音；第二，建立汉字机内码到汉字读音的访问方法，实现由输入的汉字机内码得到该汉字的读音。具体而言，首先根据汉字编码规则获取汉字字符集的文本文件；然后利用报读软件生成汉字字符集的语音文件；接着对语音文件进行处理，使每个汉字都能对应它在语音文件中的位置和数据；最后根据用户输入，在处理过的语音文件中截取每个汉字的语音数据，合成语音输出，实现汉语文语转换。

3　实现步骤

3.1　根据汉字编码规则获取汉字字符集的文本文件

所谓编码，是以固定的顺序排列字符，并以此作为记录、存贮、传递、交换的统一内部特征。一个汉字有 ASCⅡ码、区位码等与之对应。ASCⅡ码中对应于码值 161 到 254 的字符用于表示汉字，每个汉字用两个 ASCⅡ码值对应的字符表示。区位码用 4 位数字表示，前两位从 01 到 94 称区码，后两位从 01 到 94 称位码。一个汉字的前一半是 ASCⅡ码为“160+区码”的字符，后一半是 ASCⅡ码为“160+位码”的字符。例如：“刘”的区位码是 3385，其意为区码 33 和位码 85，它是由 ASCⅡ码为 160+33=193 和 160+85=245 的两个字符组成。

该文所说的汉字字符集一般是指 ISO 10646.1 即 GB 13000.1。在 Windows 95/98/2000 中，微软提供了“汉字扩展内码规范(GBK)”以解决汉字的收字不足、简繁共存、简化代码体系间转换等汉字信息交换的瓶颈问题，利用 GBK 可以方便解决“镕”、“碁”等大量汉字的交换问题而不必自行造字了。

GBK 字库共分为 5 部分，其中 GBK/1 和 GBK/5 为符号部分，GBK/2 为国标汉字部分，GBK/3 和 GBK/4 为扩展汉字部分。其中，第 16 区至 55 区为一级汉字，以拼音排序，共计 3755 字，56-87 区为二级汉字，按偏旁部首排序，共计 3008 字[1]。为简化起见，以一级汉字为例讨论系统的实现。

用 C++实现的获取 GBK 中一级汉字字符的程序清单如下：

```
// getgbk.cpp -获取 GBK 汉字码文件
#include <strstream.h> // 字符串 I/O 操作
#include <fstream.h> // 文件 I/O 操作
……
unsigned char oneline[4];
ofstream ofs("gbhz.txt",ios::binary);
oneline[2]=163;
oneline[3]=172;//逗号
for(int qm=176;qm<=215;qm++)//区码 0XB0-0XD7
for(int wm=161;wm<=254;wm++)//位码 0XA1-0XFE
    if(!((qm==215)&&(wm>=250)))// 剔除 GBK 中没有编码
```

作者简介：袁嵩(1976-)，男，硕士，主要研究方向：计算机应用。

的字位

```
{
    oneline[0]=qm ;//汉字区码
    oneline[1]=wm ;//汉字位码
    ofs.write((char *)&oneline,4) ;//写一行至 gbhz.txt
} //if end
```

运行程序生成 gbhz.txt 文本文件，其中就包括 3755 个汉字。

3.2 获取汉字字符集的语音文件

利用报读软件将汉字字符集报读出来，生成汉字字符集的语音文件。笔者选用 Miscrosoft Speech SDK 5.1 语音引擎，导入文本文件 gbhz.txt，生成语音文件 hzyy.wav。

3.3 对语音文件进行处理

3.3.1 理解 WAV 文件格式

WAVE 文件作为多媒体中使用的声波文件格式之一，它是以 RIFF 格式为标准的。每个 WAVE 文件的头 4 个字节便是“RIFF”。WAVE 文件由文件头和数据体两大部分组成。其中文件头又分为 RIFF/WAV 文件标识段和声音数据格式说明段两部分[2]。WAVE 文件格式说明见表 1。

表 1 WAVE 文件格式说明表

	偏移地址	字节数	数据类型	内 容
文件头	00H	4	char	"RIFF"标志
	04H	4	long int	文件长度
	08H	4	char	"WAVE"标志
	0CH	4	char	"fmt"标志
	10H	4		过渡字节(不定)
	14H	2	int	格式类别(10H 为 PCM 形式的声音数据)
	16H	2	int	通道数,单声道为 1,双声道为 2
	18H	2	int	采样率(每秒样本数),表示每个通道的播放速度,
	1CH	4	long int	波形音频数据传送速率,其值为通道数×每秒数据位数×每样本的数据位数/8。播放软件利用此值可以估计缓冲区的大小。
	20H	2	int	数据块的调整数(按字节算的),其值为通道数×每样本的数据位值/8。播放软件需要一次处理多个该值大小的字节数据,以便将其值用于缓冲区的调整。
	22H	2		每样本的数据位数,表示每个声道中各个样本的数据位数。如果有多个声道,对每个声道而言,样本大小都一样。
	24H	4	char	数据标记符"data"
	28H	4	long int	语音数据的长度

可以以时域-幅度的方式显示出原始声音的波形，这是最简单同时也是最直接的信息处理方式。在时域范围内，可以观察该信号波形是否连续，中间是否有跳变等。据发现，经语音引擎处理后，每个汉字所对应的语音数据长度不尽相同，这给以后截取每个汉字的语音数据造成了困难。因此，为了区分每个汉字的语音数据，在生成汉字字符集时，在每个汉字后添加了一个逗号作为间隔符。这样生成的语音文件的波形图(部分)如图 1 所示。

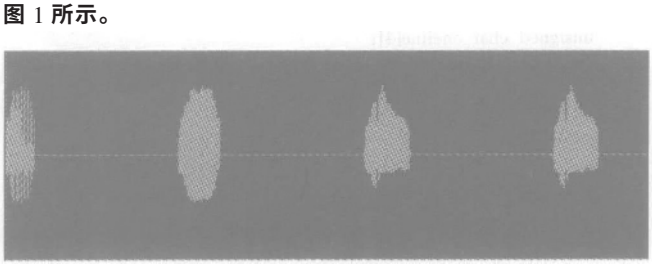

图 1 部分 hzyy.wav 的波形图

3.3.2 生成新的汉字字符集的语音文件的算法

(1)打开 hzyy.wav，从 hzyy.wav 中读入文件头 CHAR HEAD[44]。44 为文件头长度。

(2)从 hzyy.wav 中读入第一个汉字的语音波形数据放入 CHAR BUFFER[3200]。3200 为一个汉字的语音波形数据长度，这取决于形成语音的速率、音质等因素。

(3)读后面的数据，如果是 0x80 则不做处理，继续往后读。0x80 是逗号语音数据。

(4)直到读到第一个不是逗号的语音数据，开始记录第二个汉字的语音波形数据到 CHAR BUFFER[3200]。

(5)重复 3,4 步直到读完全部的语音波形数据。

(6)修改 CHAR HEAD[44]中的 5~8 字节(文件长度)和最后 4 个字节(语音数据的长度)的值生成新的文件头，合成新的汉字字符集的语音文件 yyk.wav，其波形图(部分)如图 2 所示。

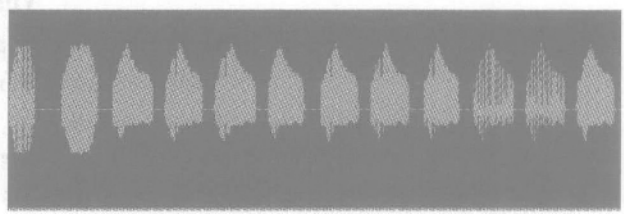

图 2 部分 yyk.wav 的波形图

实际上，在处理完所有汉字的语音波形数据后，可以不必加载文件头合成新的 WAV 文件，因为所关心的仅仅是语音波形数据。但是之所以这样做是为了便于对处理后的语音文件进行测试。

3.4 根据用户输入，合成语音输出

设计用户界面，根据用户的输入，在语音文件 yyk.wav 中截取每个汉字的语音数据，合成语音输出，实现中文文语转换。

如何根据汉字截取其语音数据呢？归根结底就是解决汉字在字符集中的定位问题。方法是根据汉字的两个字节中的值，从高字节算出汉字的位 wm，从低字节算出汉字的区 qm，(qm-176)*94+wm-160 就是该汉字在汉字集里的位置 position，而该汉字所对应的语音数据的偏移量就是 (position -1)*3200 + 44。部分代码如下：

```
……
String ls ;// 用户输入
length=ls.Length() ;
……
for(int m<=length ;m+=2)
{
    int qm=ls[m]& 0x00FF ;//计算区码
    int wm=ls[m+1] & 0x00FF ;//计算位码
    int position=(qm-176)*94+wm-160 ;//找到汉字在 3755 个汉字中的相对位置 */
    position=(result-1)*3200 + 44 ;
    fseek(stream,position,SEEK_SET) ;//找到汉字在语音文件 yyk.wav 中的偏移量
    int readresult=fread(buffer,sizeof(char),3200,stream) ;//读该汉字相应的语音数据到 buffer 中去
    ……
}
```

(下转 229 页)

4.5 将 FP-tree 转换为决策树

设将上述 FP-tree 的所有实例分成含有项目 10 和不含有项目 10 2 P 类和 N 类。由于实例总数为 9，其中 P 类为 6

$$H(X)=-\frac{6}{9}\lg\frac{6}{9}-\frac{3}{9}\lg\frac{3}{9}=0.2764$$；

若选取 20 属性作为测试属性，由于包含 20 的实例为 8，包含 20 且为 P 类的实例数为 6，有

$$H(\frac{X}{b})=\frac{8}{9}(-\frac{6}{8}\lg\frac{6}{8}-\frac{(8-6)}{8}\lg\frac{2}{8})+(-\frac{(9-8)}{9}\lg 1)=0.2171$$

若选取 30 属性作为测试属性，由于包含 20 的实例为 7，包含 30 且为 P 类的实例数为 5，有

$$H(\frac{X}{c})=\frac{7}{9}(-\frac{5}{7}\lg\frac{5}{7}-\frac{(7-5)}{7}\lg\frac{2}{7})+\frac{(9-7)}{9}(-\frac{(6-5)}{2}\lg\frac{1}{2}-\frac{(2-1)}{2}\lg\frac{1}{2})=0.2680$$

若选取 40 属性作为测试属性，由于包含 40 的实例为 5，包含 40 且为 P 类的实例数为 3，有

$$H(\frac{X}{d})=\frac{5}{9}(-\frac{3}{5}\lg\frac{3}{5}-\frac{(5-3)}{5}\lg\frac{2}{5})+\frac{(9-5)}{9}(-\frac{(6-3)}{4}\lg\frac{3}{4}-\frac{(4-3)}{4}\lg\frac{1}{4})=0.2709$$

可以看出 H(X/b)最小，即对于分类有最大的帮助，所以应选择 20 作为测试属性，这时 20 作为测试属性之后将实例集分为两个子集，生成两个叶结点，对每个叶结点依次利用上面过程就可生成如图 3 所示的决策树。

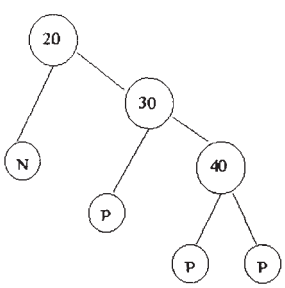

图 3 决策树

注：左分枝不包含该结点项目，右分枝包含该结点项目

5 结论及展望

通过以上分析，可以看出该挖掘频繁集方法主要利用模式仓库来存储病例记录信息，结合 Apriori 算法和 FP-growth 算法对用户定制 FP-tree 进行频繁模式挖掘。

采用模式仓库存储的信息比 FP-tree 多，但比较简洁，不存在冗余，很容易进行动态更新，只需对改变的病例进行处理，不需重新构造整个模式仓库；该模式仓库信息易于裁剪，可根据用户需要定制用户 FP-tree；该模式仓库容易存储和划分，因此可处理大量病例事务。该模式仓库的建立只需扫描一次初始病例记录库。

在 FP-growth 算法中利用 Apriori 算法产生候选频繁集增加了频繁模式挖掘的目的性，减少了条件模式库的递归次数，而候选频繁集产生只需很少额外项目集操作即为模式长度，因此缩短了挖掘时间，提高了挖掘效率。

该文对电子病历的挖掘及分析提出了一些想法，主要是对现有流行挖掘算法进行修改以适应电子病历病例分析的特点，以及如何得到分类决策树。今后电子病历挖掘的研究方向可能是提出一种针对某种标准（如 HL7）电子病历进行挖掘的结构模型，在此基础上，再开发一种好的有效算法来实现挖掘任务。（收稿日期：2003 年 10 月）

参考文献

1.J Han，J Pei，Y Yin.Mining Frequent Patterns without Candidate Generation[C].In：SIGMOD Conference，2000：1~12

2.Richard Relue，Xindong Wu，Hao Huang.Efficient Runtime Generation of Association Rules[C].In：Proc of the Tenth International Conference on Information and Knowledge Management，Atlanta，Georgia，USA，2001：466~477

3.Wenge Guo.Mining Frequent Patterns with FP-tree and Candidate Generation.URL，http：//www.cs.ndsu.nodak.edu/~wguo/cs765-paper.doc，2001

4.毛国君等.基于项目序列集操作的关联规则挖掘算法[J].计算机学报，2002，25(4)：1~3

（上接 122 页）

读取每个汉字的语音数据，最后加载文件头就能合成语音输出了。

4 结束语

TTS 系统的实现方案有很多种，传统的做法是：在普通话中实际存在 1333 种发音，先将这 1333 个语音分别录制成 1333 个独立的 WAV 件，每一个读音都有它自己的序号。然后建立一个语音数据库或是按照读音的序号合成一个大文件，但必须建立两张索引表分别记录汉字对应的读音序号和该读音在文件中的偏移[3,4]。

该文所介绍的汉语 TTS 系统的实现方案，其最大特点是等长截取汉字语音数据、自动计算语音数据的偏移量，尤其在一些小型的 TTS 应用上，例如语音查询系统、报分系统等，和传统方法相比，不仅开发时的工作量小，而且系统资源占用率低，代码执行效率高。

在该系统的基础上，笔者还实现了从键盘和电话按键获取汉字的区位码，通过汉字语音库的转换，经过语音文件的拼接，形成一段汉语语音信息，然后通过计算机上的媒体播放器或电话听筒将这些信息播放给用户。该系统已在某声讯台投入使用，系统运转良好。（收稿日期：2003 年 11 月）

参考文献

1.彭寿全，黄可.汉字信息处理[M].成都：电子科技大学出版社，1994

2.徐济仁.基于多媒体 WAV 文件的语音特征识别[J].计算机工程，2000；26(11)：123~125

3.谌卫军.汉语文语转换系统(TTS)[J].计算机工程与应用，2000；36(6)：1~3

4.吕强.一个汉字语音库的实现[J].苏州大学学报(自然科学)，2001；17(1)：40~46