Nota Del Festival

Esta página web tiene como objetivo crear lecturas accesibles y se puede ver en cualquier navegador web

Origen

Podemos encontrar que aunque el software de síntesis de voz de código abierto en el extranjero como Festival está disponible para investigación, no se ha tratado en chino. Aunque la investigación nacional es avanzada pero no ha sido posible encontrar una versión de código fuente abierto, festival con los logros existentes, no reinventar la rueda, espera crear un software exclusivo de la síntesis de la libertad de expresión Aunque muchos resultados de investigación avanzada de la mala doméstica para una cierta distancia, pero es una parte de la población china, sino por el efecto de este programa inicie la aglomeración rápida Fuerzas de la comunidad OSS, el Festival modificado para que pueda ser la piedra angular de la síntesis del habla mandarín.

Introducción

Festival El sistema de síntesis de voz fue nombrado después del Festival de Edimburgo por la Universidad de Edimburgo en Escocia

1. Última versión: 1.95-beta (14/07/2004)
2. Autor principal: Alan W Negro
3. Lenguaje de desarrollo:
   - Núcleo: C ++
   - Control externo: un esquema llamado SIOD un dialecto de Lisp
4. Idioma de pronunciación: inglés (británico, americano), español, galés
5. El discurso sintético utilizado por el software del Festival tiene un subproyecto llamado FestVox

Instalar

Instalado en rpm (solo use el festival para sonar)

Punto de descarga

- festival 1.4.3 // El archivo rpm listado se usa con mandrake 9.1, si hay otra falla en la distribución o en el punto de descarga, por favor, los lectores pueden encontrar su propia rpmfind
  - ftp://mirrors.kernel.org/mandrake/9.1/contrib/i586/festival-1.4.3-2mdk.i586.rpm
  - ftp://mirrors.kernel.org/mandrake/9.1/contrib/i586/festival-devel-1.4.3-2mdk.i586.rpm
  - ftp://mirrors.kernel.org/mandrake/9.1/contrib/i586/festlex-CMU-1.4.3-1mdk.noarch.rpm
  - ftp://mirrors.kernel.org/mandrake/9.1/contrib/i586/festlex-POSLEX-1.4.3-1mdk.noarch.rpm
  - ftp://mirrors.kernel.org/mandrake/9.1/contrib/i586/festvox-kallpc-common-1.4.3-1mdk.noarch.rpm
  - ftp://mirrors.kernel.org/mandrake/9.1/contrib/i586/festvox-kallpc16k-1.4.3-1mdk.noarch.rpm
  - ftp://mirrors.kernel.org/mandrake/9.1/contrib/i586/festvox-kallpc8k-1.4.3-1mdk.noarch.rpm
  - ftp://mirrors.kernel.org/mandrake/9.1/contrib/i586/festvox-kedlpc-common-1.4.3-1mdk.noarch.rpm
  - ftp://mirrors.kernel.org/mandrake/9.1/contrib/i586/festvox-kedlpc16k-1.4.3-1mdk.noarch.rpm
  - ftp://mirrors.kernel.org/mandrake/9.1/contrib/i586/festvox-kedlpc8k-1.4.3-1mdk.noarch.rpm

Pasos de instalación

#rpm -Uvh fest *

Para la instalación de tarball (si, al mismo tiempo, para crear un archivo de voz sintético)

Punto de descarga

- festival 1.4.3 http://www.festvox.org/packed/festival/1.4.3/
- festvox 2.0 http: // festvox. org / festvox-2.0 / festvox-2.0-release.tar.gz

Lista de archivos

Conjunto de sistema principal		
*	speech_tools-1.2.3-release.tar.gz	Biblioteca de herramientas de voz y archivos de origen
*	festival_1.4.3-release.tar.gz	archivo fuente del festival
*	festvox-2.0-release.tar.gz	festvox el archivo de origen
Kit de diccionario		
*	festlex_POSLEX.tar.gz	Diccionario de componentes de voz
*	festlex_CMU.tar.gz	Diccionario de inglés americano (necesidad de combinar el archivo de sonido estadounidense)
	festlex_OALD.tar.gz	Inglés británico diccionario (necesidad de combinar el archivo de sonido en inglés)
El kit de sonido principal		
*	festvox_kallpc16k.tar.gz	Versión 'kal' 16kHz masculina americana
	festvox_kallpc8k.tar.gz	Versión 'kal' 8kHz masculina americana
	festvox_kedlpc16k.tar.gz	Versión 'ked' 16kHz masculina americana
	festvox_kedlpc8k.tar.gz	Versión estadounidense de 8 kHz 'ked' masculino
	festvox_rablpc16k.tar.gz	Versión 'rab' masculina británica 16kHz
	festvox_rablpc8k.tar.gz	Versión 'rab' 8kHz masculina británica
Otro kit de sonido		
	festvox_us1.tar.gz	Otros archivos de sonido estadounidenses incompletos
	festvox_us2.tar.gz	Otro archivo de sonido estadounidense incompleto 2
	festvox_us3.tar.gz	Otro archivo de sonido estadounidense incompleto 3
	festvox_don.tar.gz	Otros archivos de sonido estadounidenses incompletos
	festvox_en1.tar.gz	Otros archivos de sonido estadounidenses incompletos

festvox_ellpc11k.tar.gz	Archivo de sonido en español

Nota * para el kit de instalación mínimo requerido

Pasos de instalación

Suponiendo que el directorio de inicio del usuario es peter, copie todos los archivos descargados desde arriba en el subdirectorio de proyectos en el directorio de inicio del usuario. Los siguientes pasos también pueden implementarse como un script de shell.

1. Escribir script para descomprimir todos los archivos obtendrá 3 directorios speech_tools, festival, festvox

```
#! / bin / sh
para i en * .gz
do
  tar -zxvf $ i
done
```

2. En el directorio speech_tools, ejecute ./configure, gmake, gmake test

```
cd speech_tools
./configure
make
make test
make install
```

3. Para el directorio del festival, ejecute ./configure, gmake, gmake test

```
cd ../festival
./configure
make
make test
make install
```

4. Agregue 4 variables de entorno (~ / .bashrc)

```
Exportar el PATH = $ PATH: / Home / Peter / Proyecto / Festival / bin
exportación PATH = $ PATH: / Home / Peter / Proyecto / speech_tools / bin
exportación ESTDIR = / home / Peter / proyecto / speech_tools
exportación FESTVOXDIR = / home / Peter / proyecto / festvox
```

5. Para el directorio festvox, ejecuta make

```
cd festvox
./configure
make
```

Uso

Shell

# festival	// llamar al modo chat
# echo "hello world" | festival --tts	// Cadena de eco a la pronunciación del festival (2 dash Oh)
# festival --tts news.txt	El modo de comando leyó el contenido del nombre de archivo especificado (2 guiones Oh)
# festival -b nombre de archivo.scm	// Ejecuta el archivo por lotes especificado en modo de comando
# / usr / share / festival / examples / saytime	// ejemplo: decir que es hora (directorio de instalación de rpm)

festival

festival Debido al enlace a la biblioteca readline, las teclas son las mismas que emacs debajo del shell

festival> (SayText "hola")	Lea el texto especificado
festival> (SayPhone "hh eh ow")	// leer los fonemas especificados (fonemas fonémicos según el conjunto de fonemas utilizado, el sistema

	actual tiene el teléfono de EE. UU. y el teléfono del Reino Unido) dos
festival> (voice_kal_diphone) o (voice_ked_diphone)	// Cambiar los datos de voz a kal o ked versión 1.95 voz femenina: (voice_cmu_us_slt_arctic_hts)
festival> (PhoneSet.list)	// Listar el conjunto de fonemas actualmente usado (radio)
festival> (voice_rab_diphone)	// Cambiar los datos de voz a rab
festival> (PhoneSet.list)	// Lista de conjuntos de fonemas actualmente utilizados (radio mrpa)
festival> (PhoneSet.description '(silencia teléfonos))	// Listar todos los fonemas disponibles
festival> (tts "nombre de archivo")	// puede leer el contenido del nombre de archivo especificado
festival> (Parameter.set 'Duration_Stretch 2.0)	// alargar la pronunciación (leer más despacio)
festival> ayuda	// Ayuda
festival> (dejar de fumar)	// Salir del modo de chat, o presionar Ctrl + D
festival> (lex.lookup 'hola)	// Descubre el fonema de la palabra hola
festival> (introducción)	// Prueba de Pronunciación: 2 oraciones Festival de Sinopsis
festival> (festival_avance)	// Mostrar el alcance de la autorización del Festival
festival> libdir	// directorio de biblioteca de la tienda
festival> (pwd)	// mostrar el directorio actual
festival> (pow 5 3)	// 5 potencia de 3, mostrando 125
festival> (load 'test.scm)	// Cargar archivo de script de esquema

Perfil

En la clave del festival sobre el comando: ~ .festival_history

Festival Speech Architecture

Este artículo se centra en la estructura de datos de síntesis de voz utilizando el gigante del software gratuito Festival (http://www.festvox.org/packed/festival/1.4.3/) de síntesis de voz para diseccionar y documentar la familia Lisp. Esquema como herramienta de investigación.

Festival El sistema de síntesis de voz fue nombrado después del Festival de Edimburgo por la Universidad de Edimburgo en Escocia. Su núcleo está escrito en C ++, pero puede ser controlado y utilizado con el uso de la biblioteca SIOD. La versión estable actual versión 1.43, la última versión es 2004/7/14 versión 1.95.

Carnegie Mellon junto con el Festival desarrolló un subproyecto: Festvox (http://festvox.org/), diseñado para proporcionar una voz más diversa y una mejor calidad de voz y documentación manual.

Produce objetos de declaración

Una vez completada la instalación del Festival, podemos simplemente intentar decir cómo decirlo, escribir el mensaje (SayText "Esto es un ejemplo")

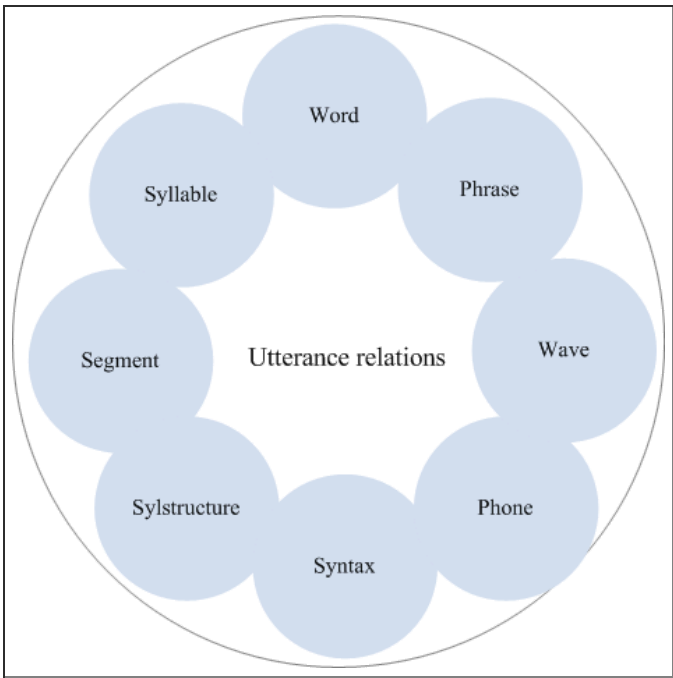
```
festival> (SayText "Este es un ejemplo")
# <Expresión 0x404054e8>
```

Respondió que el festival va a convertir texto a un usuario escribió el nombre de la Expresión de estructuras de datos y se almacena en la dirección 0x404054e8 memoria, debido a que la estructura de datos de seguimiento a ser desmontado para este discurso, y por lo tanto la definición de un "sen" Señale para que pueda analizar esta estructura de presentación

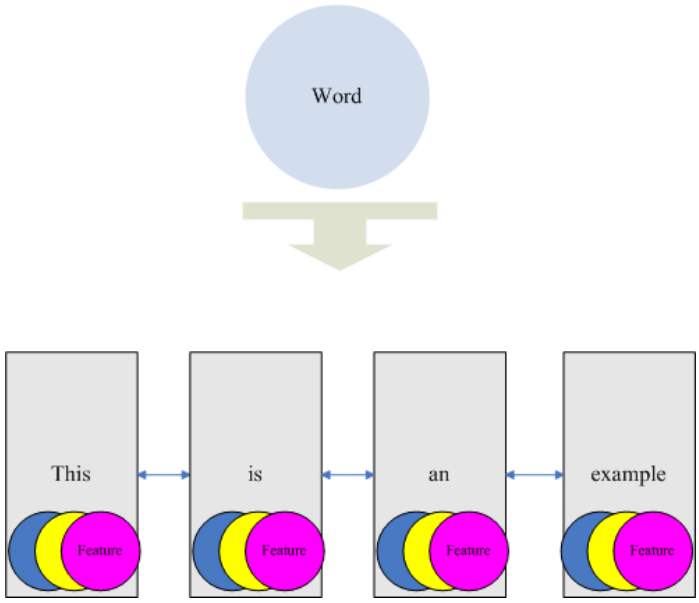
```
festival> (set! sen (SayText "Este es un ejemplo"))
# <Utterance 0x40554b98>
```

Estructura de la declaración

En el nivel de fonética, al menos una palabra (Expresión) se puede analizar desde diferentes perspectivas, como frase, palabra, sílaba, sílaba, etc. Estas diferentes Aquí podemos llamarnos Relaciones. Por ejemplo: Palabra de la Relación de vista, es por este <-> es <-> un <-> ejemplo de la serie juntos, y esto "Este", "es", "una", "ejemplo" en este Llámalo un elemento, y cada elemento tiene un conjunto de propiedades que describen sus propias características, aquí llamadas Características.

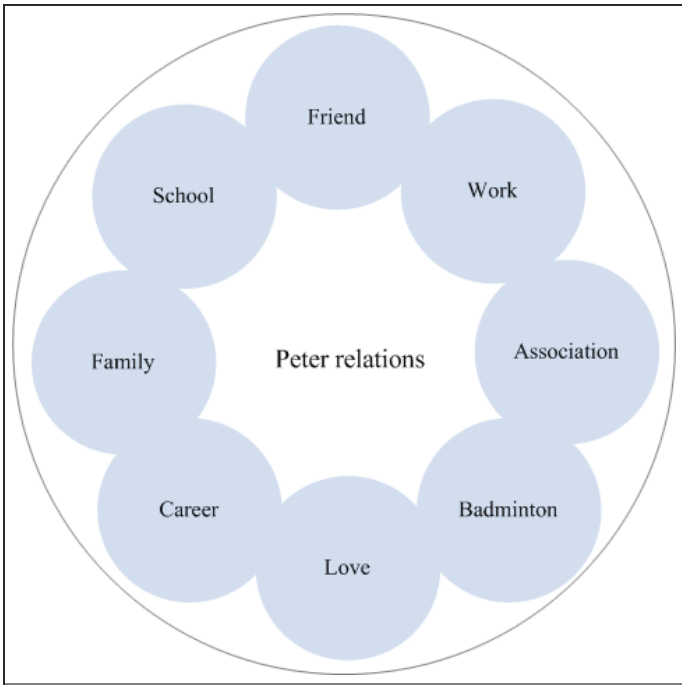


Composición de las relaciones de presentación

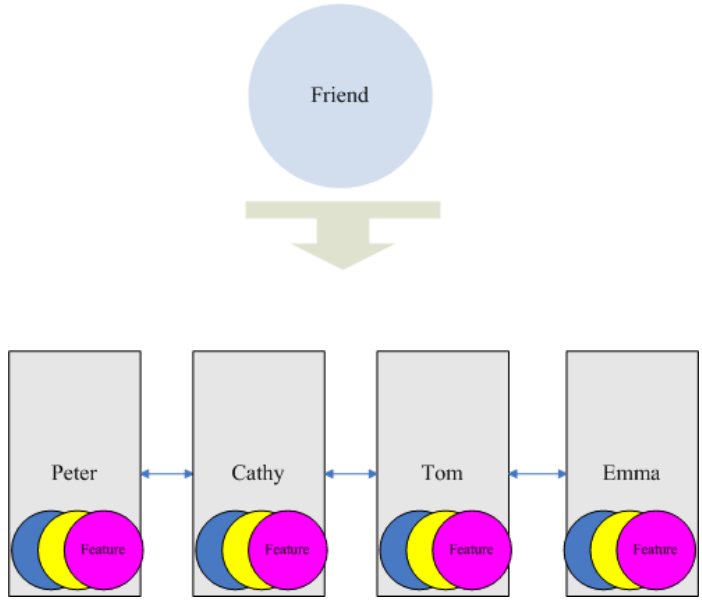


Composición de la relación de palabras

Si no lo entendemos bien, tomemos otro ejemplo. Hay un alumno Peter (como el enunciado), sus relaciones sociales pueden ser como una relación de palabras, como una relación de sílaba, una relación familiar. relación segmento), y las relaciones con los amigos, por ejemplo, Peter comprensión de Cathy, Cathy saben Tom, Tom conoce a Emma, que (como elemento) sí tienen sus propias características (como características), como el género, la edad, la altura, el tipo de sangre y así sucesivamente.



Peter relaciones sociales de los estudiantes



La composición de una relación de amigo

Y hay tres formas principales de estructura para el habla o lingüística en el Festival

1. Elementos: el elemento se refiere a una unidad de idioma único como teléfono, palabra, sílaba, nodo sintáctico, frase de entonación y más. Cada elemento tiene sus propias características (valores propios) , para describir sus propios atributos locales, como el nombre, la longitud de la pronunciación. El contenido de las características puede ser alfanumérico o funcionar.
2. Relaciones (relación): relación El elemento anterior junto, por ejemplo, puede haber una palabra (palabra única), sintaxis (sintaxis), sílaba (sílaba), etc. relación. Las relaciones generalmente se expresan en estructuras de datos gráficos, con listas, árboles, y la más común es una cadena de columnas bidireccionales, por ejemplo: la relación de palabras es una cadena de columnas bidireccional, que a su vez vincula todo el enunciado palabra Las relaciones también pueden estar representadas por la estructura de datos del árbol. Por ejemplo, la relación de una sílaba contiene las estructuras de inicio, coda, núcleo y rima. Es importante tener en cuenta que los elementos pueden tener más de una relación, por ejemplo: la relación de una gramática es una estructura de árbol cuyas hojas son palabras, y las palabras mismas están contenidas en la relación de palabras.
3. Expresiones (entonación): emisiones compuestas de elementos, cada elemento tiene muchas características, los elementos pueden tener 1 o más relaciones entre ellos.

Relaciones en serie

Luego llegamos a comprender la composición de la relación de Word, aquí tenemos que usar la variable de sen definida anteriormente, eche un vistazo a la relación de Word hay algunos elementos

```
Festival> (SEN utt.relation.leafs 'Word)
(# <Elemento 0x87a1680>
  # <Elemento 0x87a2170>
  # <Elemento 0x87a2418>
  # <Elemento 0x87a26c8>)
```

Aquí vemos el índice de 4 elementos, también podemos especificar directamente el primer elemento para obtener su relación de Word

```
festival> (utt.relation.first sen 'Word)
# <item 0x87a1680>
```

La respuesta es una estructura de datos de elementos, almacenada en la memoria 0x87a1680, tenga en cuenta que el primer elemento de utt.relation.leafsd y utt.relation.first item para la misma dirección, el mismo para mayor comodidad, Definimos una primera palabra variable para señalarla

```
festival> (set! firstword (utt.relation.first sen 'Word))
```

Dado que tenemos un artículo llamado firstword, por supuesto, tenemos que ver qué características tiene

```
Festival> (item.features FirstWord)
((ID "_5")
  (nombre "Este")
  (pos_index 2)
  (pos_index_score 0)
  (POS "DT")
  (phr_pos "DT")
  (pbreak_index. 1)
  (pbreak_index_score 0)
  ( pbreak "NB"))
```

También podemos especificar ver solo una característica, por ejemplo: para ver la primera palabra este elemento llamado "nombre"

```
festival> (item.feat firstword "nombre")
"Esto"
```

Además de especificar directamente el orden de los artículos, también podemos usar la ubicación actual del artículo, obtener el siguiente elemento hacia atrás o hacia adelante para obtener el elemento anterior

```
Festival> (SET! Secondword (item.next FirstWord))
# <Elemento 0x87a2170>
Festival> (item.feat secondword "nombre")
"ES"
Festival> (item.feat secondword "p.name")
"Este"
```

También podemos utilizar este método para observar la expresión de otras relaciones, como la sílaba, por segmentos, etc., y por qué estas relaciones son parte de una lista enlazada escribirla? Puede ser la misma que la del primer artículo y el artículo que utt.relation.first número de utt.relation.leafs

Relaciones en forma de árbol

A continuación llegamos a entender la relación SylStructure composición, y en este tenemos que utilizar variables sen definidos anteriormente, echar un vistazo a una relación SylStructure pocas elemento

```
Festival> (utt.relation.leafs SylStructure SEN')
(# <Elemento 0x87a30b8>
 # <Elemento 0x87c3060>
 # <0x87a57b8 Elemento>
 # <Elemento 0x87a59e8>
 # <Elemento 0x87c2cb8>
 # <Elemento 0x87c2d48>
 # <Elemento 0x87a42d8>
 # < 0x87a4758 Elemento>
 # <Elemento 0x87a48f8>
 # <Elemento 0x87a4c78>
 # <Elemento 0x87a4e18>
 # <Elemento 0x87a4fb8>
 # <Elemento 0x87a5158>
 # <Elemento 0x87a54d8>
 # <Elemento 0x87be800>)
```

Lo mismo vemos en el primer elemento SylStructure relación ¿por qué?

```
festival> (utt.relation.first sen 'SylStructure)
# <item 0x87a1680>
```

Curso experimental

Inicialización De Estructura De Voz	
(set! sen (texto de pronunciación "Este es un ejemplo"))	Defina la variable sen para apuntar al objeto Utterance
(utt.play sen)	Feature Wave Error no definido aparece
(utt.relation.first sen 'Word)	Aparece el error de la palabra característica no definida
(utt.synth sen)	Esta síntesis de forma de onda debe realizarse antes de que la acción no sea incorrecta
(set! sen (SayText "Este es un ejemplo"))	Defina la variable de sen para que apunte a la estructura de objetos de Otorgamiento completa (puede omitir los pasos sintéticos)
Observación De La Relación De Palabras	
(set! firstword (utt.relation.first sen 'Word))	Defina la variable firstword para señalar el primer elemento de la relación de Word
(item.features firstword)	Observe todas las características de firstword este artículo
(item.feat firstword "nombre") (item.name firstword)	Observe la característica llamada "nombre" para este artículo firstword
(item.feat firstword "n.name")	Observe la característica de "nombre" del siguiente elemento en el elemento firstword
(set! secondword (item.next firstword))	Defina la variable de la segunda palabra para señalar el siguiente elemento en la primera palabra
(item.feat secondword "nombre")	Observe la segunda palabra de este elemento llamada característica "nombre"
(item.feat secondword "p.name")	Observe la característica "nombre" del último elemento del artículo de la segunda palabra
(utt.features sen 'Word' (id nombre pos))	Sen sentidos características de relación de palabras (función personalizada de mapleaf similar)
Observación De Relación Sílaba	
(set! firstsyllable (utt.relation.first sen 'Sílaba))	Defina la variable de la primera sílaba para señalar el primer elemento de la relación

	Sílaba
(item.features firstsyllable)	Observe todos los ítems de primer plano de esta característica
(item.feat firstsyble "nombre") (item.name firstsyllable)	Observe la característica llamada "nombre" del artículo de la primera sílaba
(item.feat firstsyllable "n.name")	Observe la función "nombre" del siguiente elemento del primer elemento
(set! secondsyllable (item.next firstsilable))	Define la variable secondsyllable que apunta al siguiente elemento en firstword
(item.feat secondsyllable "id")	Observe los segundos, este elemento llamado característica "id"
(item.feat secondsyllable "p.id")	Observe la función "id" del último elemento de este artículo
(utt.features sen 'Syllable' (id nombre de estrés))	Observe las características de sen en la relación de Sílaba
	Observación De Relación De Segmento
(set! firstsegment (utt.relation.first sen 'Segment))	Defina la variable de primer término para señalar el primer elemento en la relación Segmento
(item.features primer plano)	Observe todas las características del elemento del primer elemento
(item.feat firstsegment "nombre") (item.name firstsegment)	Observe el elemento de primer segmento llamado función "nombre"
(item.feat firstsegment "n.name")	Ver la función "nombre" del siguiente elemento en el elemento del primer elemento
(set! secondsegment (item.next primerosexto))	Defina la variable del segundo segmento para apuntar al primer elemento del primer registro
(item.feat secondsegment "id")	Observe el segundo segmento de este elemento llamado función "id"
(item.feat secondsegment "p.id")	Observe la función "id" del último elemento de este artículo
(utt.features sen 'Segment' (id nombre dur_factor final))	Observe las características de sen en la relación de Sílaba
	Observación De La Relación De Frase
(set! firstphrase (utt.relation.first sen 'Frase))	Defina la variable firstphrase para señalar el primer elemento de la relación Phrase
(item.features firstphrase)	Observe todas las características del artículo firstphrase
(set! secondphrase (item.next firstphrase))	Defina la variable de la segunda frase para apuntar al siguiente elemento en la primera frase (return nil)
(item.daughters firstphrase)	Vea la primera frase que este artículo tiene varias hijas (con 4 hijas)
(set! fpl (item.daughter1 firstphrase))	Defina la variable fpl para señalar a la hija del elemento de la primera palabra
(item.features fpl)	Observe la característica de fpl, que es relación de Word
(item.feat fp1 "R: SylStructure.daughter1.stress")	Cambia la vista SylStructure incluso, mira la característica de estrés de la hija
	Observación De La Relación SylStructure
(set! firstword (utt.relation.first sen 'SylStructure))	Defina la variable firstword para señalar el primer elemento de la relación SylStructure
(item.features firstword)	Observe todas las características de firstword este artículo
(item.feat firstword "nombre") (item.name firstword)	Observe la característica llamada "nombre" para este artículo firstword
(item.feat firstword "n.name")	Observe la característica de "nombre" del siguiente elemento en el elemento firstword
(item.daughters firstword)	Observe la primera palabra que este artículo tiene varias hijas (solo 1 hija)
(set! firstsyl (item.daughter1 firstword))	Defina la variable firstsyl para señalar a la primera hija de este elemento
(item.features firstsyl)	Observación de Firstsyl de la característica, que es relación de Sílaba
(item.name (item.parent firstsyl))	Observe el primer padre de este artículo, que es la relación de Word: "Esto"

(item.daughters firstsyl) Mira primero, este objeto tiene varias hijas (3 hijas)

(set! secondseg (item.daughter2 firstsyl)) Defina la variable secondseg para apuntar a la primera hija del ítem ítem primer

(item.features secondseg) Observe la función de Secondseg, que es la relación de Segmento

(utt.features sen 'SylStructure' (nombre de id) Observe las características de sen en la relación SylStructure

Acabado de observación

Relación	Estructura de datos	Características		
Palabra	**lista**	1. identificación 2. nombre 3. pos_index 4. pos_index_score 5. pos 6. phr_pos 7. pbreak_index 8. pbreak_index_score 9. pbreak		
Sílaba	**lista**	1. identificación 2. nombre 3. estrés		
Segmento	**lista**	1. identificación 2. nombre 3. dur_factor 4. fin 5. source_end		
Frase	**árboles** Las raíces son Frase Las hojas son palabras	1. identificación 2. nombre		1. identificación 2. nombre 3. pos_index 4. pos_index_score 5. pos 6. phr_pos 7. pbreak_index 8. pbreak_index_score 9. pbreak
SylStructure	**árboles** las raíces son palabras mediolas son sílabas las hojas son segmentos	1. identificación 2. nombre 3. pos_index 4. pos_index_score 5. pos 6. phr_pos 7. pbreak_index 8. pbreak_index_score 9. pbreak	1. identificación 2. nombre 3. estrés	1. identificación 2. nombre 3. dur_factor 4. fin 5. source_end

Función personalizada

```
(define (valot utt rel)
  to_end (utt.relation.first utt rel)
)
```

```
(define (to_end node)
  (if node
    (cons (sub_tree node) (to_end (item.next node)))
    nil
  )
)

(define (sub_tree node)
  (if (item.daughters node)
    (mapcar sub_tree (item.daughters node))
    node
  )
)

(El DEFINE (primera mapleaf Diversión)
  (cond
   ((nulo? 1º) nil)
   ((primera Atom) (1er diversión))
   (t (cons (mapleaf Diversión (CAR primero)) (mapleaf Diversión (CDR primero))))
  )
)
```

Observación De La Relación De Palabras	
(valot sen 'Word)	La función de auto-función para observar los cuatro elementos es una cadena de estructura en tándem
(mapleaf item.features (valot sen 'Word))	Para personalizar la función para observar cada una de las características de los cuatro elementos (9)
Observación De Relación Sílaba	
(valot sen 'Sílaba)	Personalizar la función para observar los seis elementos también está vinculada a la estructura en tándem
(mapleaf item.features (valot sen 'Sílaba))	Para personalizar la función para observar cada una de las seis características del elemento (3: id, nombre, estrés)
Observación De Relación De Segmento	
(valot sen 'Segmento)	Con una función personalizada para observar su elemento 17 también está vinculado a la estructura en tándem
(mapleaf item.features (valot sen 'Segmento))	Con una función personalizada para observar los 17 elementos, cada uno tiene qué funciones (5: id, nombre, dur_factor, end, source_end)
Observación De La Relación SylStructure	
(valot sen 'SylStructure)	Observado con una función personalizada
(mapleaf item.features (valot sen 'SylStructure))	Observado con una función personalizada

Utiliza el festival para enviar un experimento de habla en chino

El primer pensamiento es usar el alfabeto romano para decir SayText, emitió "Mi nombre es Zhu Xiaoguo"

```
festival> (SayText "wo di ming zi giao ju hsiao koa")
```

Me siento muy bien, pero para ser tan probado todos los sonidos demasiado cansado, demasiado Wufajiejue Por otra parte, la cuestión de tono, por lo tanto sustituido por hasta observar desde el festival de la arquitectura, saca primero "hola" mirada palabra a nivel de palabra Practica

```
Festival> (SET! SEN (Expresión, Palabras (Hello)))
Festival> (utt.synth SEN)
Festival> (utt.play SEN)
# <Expresión, 0x408cb4d8>
```

Busque "SayText" con grep Encuentre una definición en lib \ synthesis.scm que además defina "SynthText" y "SayPhones"

```
(define (Texto de Texto de Texto) (utt.play (utt.synth (eval (texto de texto de 'Expresión')))))

(define (texto SynthText) (utt.synth (eval (lista texto de texto 'Utterance'))))

(definir (teléfonos SayPhones) (utt.play (utt.synth (eval (enumerar teléfonos 'Utterance' Phones)))))
```

Puede explicar, una expresión de texto debe primero utt.synth utt.play puede reproducir una voz para

El script de esquema en el festival se ubica principalmente en synthesis.scm y festival.scm en el directorio lib

- utt.synth
- utt.play
  - wave.play
    - utt.wave

Obviamente desde utt.synth y utt.play estas dos funciones para comenzar a rastrear, primero siguiendo utt.play

```
(define (utt.play utt) (wave.play (utt.wave utt)) utt)

(define (utt.wave utt) (item.feat (utt.relation.first utt "Wave") "wave"))
```

Ven y sigue utt.wave ----------->

Las dos anteriores no funcionan para nada, y luego siguen utt.synth ---------- ---------->

En la penúltima línea 4 imprimir el cuerpo, archivar y luego volver a cargar el festival, restablecer las variables, re-utt.synth sen
aparece como se muestra a continuación, parece hacer 10 movimientos

```
(La define (utt.synth UTT)
(apply_hooks before_synth_hooks UTT)
(let ((del tipo (utt.type UTT)))
 (let ((Definición (Assoc de los UttTypes tipo)))
  (SI (nulo? Definición)
   (error "desconocido Expresión, tipo tipo")
   (LET ((cuerpo (la eval (cons 'lambda
                 (cons' (UTT) (CDR definición))))))  (cuerpo de impresión)
    (cuerpo UTT)))))
(apply_hooks after_synth_hooks UTT)
UTT)
```

```
(CIERRE (UTT) (el inicio (la inicialización UTT)
(POS UTT)
(Phrasify UTT) (Word UTT) (pausas UTT)
(entonación UTT) (PostLex UTT) (UTT la duración)
(Int_Targets UTT) (Wave_Synth UTT)))
```

試著以自訂函數 step10 來取代上述的無名函數

```
(define (step10 utt) (begin (Initialize utt) (POS utt)
(Phrasify utt) (Word utt) (Pauses utt) (Intonation utt)
(PostLex utt) (Duration utt) (Int_Targets utt)
(Wave_Synth utt) ))
```

再依照上述定義 step10方式，根據10個步驟再定義 step9 ~ step1 總共有10個函數，然後針對 sen 作用後再觀察其資料結構，整理如下：

- step1 增加了 Word 結構
- step3 增加了 Phrase 結構
- step4 增加了Syllable , Segment ， SylStructure 結構
- step6 增加了 IntEvent , Intonation 結構
- step9 增加了 Target 結構
- step10 增加了 Unit , SourceCoef , f0 ,TargetCoef , US_map , Wave 結構

```
festival>(step1 sen)
festival>(utt.relationnames sen)
(Word)
festival>(step2 sen)
festival>(utt.relationnames sen)
(Word)
festival>(step3 sen)
festival>(utt.relationnames sen)
(Word Phrase)
festival>(step4 sen)
festival>(utt.relationnames sen)
(Word  Phrase  Syllable  Segment  SylStructure)
festival>(step5 sen)
festival>(utt.relationnames sen)
(Word  Phrase  Syllable  Segment  SylStructure)
festival>(step6 sen)
festival>(utt.relationnames sen)
(Word  Phrase  Syllable  Segment  SylStructure IntEvent Intonation)
festival>(step7 sen)
festival>(utt.relationnames sen)
(Word  Phrase  Syllable  Segment  SylStructure IntEvent Intonation)
festival>(step8 sen)
```

亦可利用下列指令觀察指定資料結構：

```
(utt.relation.item sen 'Word)
```

亦可利用下列指令觀察指定資料結構的詳細樹狀內容：

(utt.relation_tree sen 'Word)

在執行step9之後來利用 utt.relationnames 觀看已產生的結構 ，發現有8種結構，再以(utt.play sen)測試是否可以發聲，發現無法發出聲音，證明少掉最後一個步驟就無法發聲了

在執行step10之後來利用 utt.relationnames 觀看已產生的結構 ，發現有14種結構(增加了6個)，再以(utt.play sen)測試是否可以發聲，發現果然可以發聲

```
festival>(utt.relationnames sen)
(Word Phrase Syllable Segment SylStructure IntEvent Intonation)
festival>(step9 sen)
festival>(utt.relationnames sen)
(Word Phrase Syllable Segment SylStructure IntEvent Intonation
Target)
festival>(utt.play sen)
-=-=-=-=-=- EST Error -=-=-=-=-=-
{FND} Feature Wave not Defined
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
festival>(step10 sen)
festival>(utt.relationnames sen)
(Word Phrase Syllable Segment SylStructure IntEvent
Intronation Target Unit SourceCoef f0 TargetCoef US_map
Wave)
festival>(utt.play sen)
```

1. 刪掉除了Wave以外的13個資料結構
2. 以utt.play仍然可以發聲，表示 utt.play參考 Wave_Synth 產生的6種結構中的最後一種(Wave)
3. 可見只要執行過 Wave_Synth後 ，除了Wave這個資料結構外的改變都不會影響語音的輸出，或可說只要Wave_Synth可以正確執行，就能產生 Wave 的資料結構。
4. 因此要探討 Wave_Synth可正確執行的條件為何？是參考哪些資料結構？

```
festival>(utt.relation.delete sen 'Word)
festival>(utt.relation.delete sen 'Phrase)
festival>(utt.relation.delete sen 'Syllable)
festival>(utt.relation.delete sen 'Segment)
festival>(utt.relation.delete sen 'SylStructure)
festival>(utt.relation.delete sen 'IntEvent)
festival>(utt.relation.delete sen 'Intronation)
festival>(utt.relation.delete sen 'Target)
festival>(utt.relation.delete sen 'Unit)
festival>(utt.relation.delete sen 'SourceCoef)
festival>(utt.relation.delete sen 'f0)
festival>(utt.relation.delete sen 'TargetCoef)
festival>(utt.relation.delete sen 'US_map)
festival>(utt.relationnames sen)
(Wave)
festival>(utt.play sen)
```

改在做 Wave_Synth步驟前， 嘗試以下的方法

1. 刪除Word資料結構，可執行 Wave_Synth
2. 刪除Phrase資料結構，可執行 Wave_Synth
3. 刪除Syllable資料結構，可執行 Wave_Synth
4. 刪除Segment資料結構，則無法執行 Wave_Synth

因此可說明 Wave_Synth 必須參考的資料結構是 **Segment**

```
festival>(set ! wd (Utterance Words (hello)))
festival>(step9 wd)
festival>(utt.relation.delete wd 'Word)
festival>(utt.relation.delete wd 'Phrase)
festival>(utt.relation.delete wd 'Syllable)
festival>(utt.relation.delete wd 'Segment)
festival>(Wave_Synth wd)
-=-=-=-=-=- EST Error -=-=-=-=-=-
{FND} Feature Segment not Defined
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
festival>(utt.relationnames wd)
(SylStructure IntEvent Intronation Target)
```

重新定義變數 wd ，進行9個步驟的函數處理

除了Segment之外，嘗試是否可以刪除其後的資料結構

1. 刪除Word資料結構，可執行 Wave_Synth
2. 刪除Phrase資料結構，可執行 Wave_Synth
3. 刪除Syllable資料結構，可執行 Wave_Synth
4. 刪除SylStructure資料結構，可執行 Wave_Synth
5. 刪除IntEvent資料結構，可執行 Wave_Synth
6. 刪除Intronation資料結構，可執行 Wave_Synth
7. 刪除Target資料結構，則無法執行 Wave_Synth

```
festival>(set ! wd (Utterance Words (hello)))
festival>(step9 wd)
festival>(utt.relation.delete wd 'Word)
festival>(utt.relation.delete wd 'Phrase)
festival>(utt.relation.delete wd 'Syllable)
festival>(utt.relation.delete wd 'SylStructure)
festival>(utt.relation.delete wd 'IntEvent)
festival>(utt.relation.delete wd 'Intonation)
festival>(utt.relation.delete wd 'Target)
festival>(Wave_Synth wd)
-=-=-=-=-=- EST Error -=-=-=-=-=-
```

因此可說明 Wave_Synth 還必須參考的資料結構是 **Target**

{FND} Feature Target not Defined

festival>(utt.relationnames wd)
(Segment)

觀察以上的結果，覺得可能可以不要參考到Segment結構，因此想針對Segment結構做刪減，看看是否能再簡化下去，希望能達到「只要自訂 Target 資料結構，就可以交給 Wave_Synth 來合成」

刪除所有的 7 個 item 後，雖然能正確執行第10個步驟「Wave_Synth」，卻無法正確發聲，若改成只刪除4個item後，再執行 Wave_Synth，會發覺發出的聲音被切一半，由此證明合成聲音時的確會參考到 Segment

原因是 Target 的結構與 Segment 的結構交叉參照了，若刪除所有Segment結構，則Target結構也不見了

真的要刪，就只能刪除Segment的 first 與 last 的item (pau)，而看起來甚至連m與b都可以刪，但實際還是會影響

(item.delete (item.prev (utt.relation.last sen 'Segment)))

festival>(utt.relation_tree wd 'Segment)
((("pau" ((id "_10") (name "pau") (dur_factor 0) (end 0.22))))
(("s" ((id "_4") (name "s") (dur_factor 0.457137) (end 0.350805))))
(("aa" ((id "_5") (name "aa") (dur_factor 0.153892) (end 0.460469))))
(("m" ((id "_6") (name "m") (dur_factor -0.470784) (end 0.521869))))
(("b" ((id "_7") (name "b") (dur_factor -0.59933) (end 0.581946))))
(("ax" ((id "_9") (name "ax") (dur_factor 1.69917) (end 0.677405))))
(("pau" ((id "_11") (name "pau") (dur_factor 0) (end 0.897405)))))
festival>(item.delete (utt.relation.first wd 'Segment))
festival>(item.delete (utt.relation.last wd 'Segment))
festival>#.....再重複5次, 刪除所有的item.....
festival>(utt.features wd 'Segment '(id name dur_factor end))
nil
festival>(utt.relationnames wd)
(Segment Target)
festival>(Wave_Synth wd)
festival>(utt.play wd)

希望能手動產生Segment與Target兩個資料結構，可以有2個作法：
1.直接定義一個list的變數

(set ! sen '((id "_10") (name "pau") (dur_factor 0) (end 0.22)))

2.修改舊有的結構

(item.insert (item.next utt.relation.first sen 'Segment) 'before)

3.新增Segment與Target二個結構，如右

festival>(set! w1 (Utterance Words ()))
festival>(utt.relation.create w1 'Segment)
festival>(set! tt1 (utt.relation.append w1 'Segment))
festival>(item.set_feat tt1 "id" "_10")
festival>(item.set_feat tt1 "name" "pau")
festival>(item.set_feat tt1 "dur_factor" 0)
festival>(item.set_feat tt1 "end" 0.22)
festival>(set! tt2 (utt.relation.append w1 'Segment))
festival>(item.set_feat tt2 "id" "_4")
.....
(utt.relation.create w1 'Target)
(set! tt1 (utt.relation.append w1 'Target))
(item.set_feat tt1 "id" "_4")
(item.set_feat tt1 "name" "s")
(item.set_feat tt1 "dur_factor" 0.457137)
(item.set_feat tt1 "end" 0.350805)
(set! tt1_d1 (item.append_daughter tt1))
(item.set_feat tt1_d1 "id" "_14")
(item.set_feat tt1_d1 "f0" 99.8505)
(item.set_feat tt1_d1 "pos" 0.22)
(set! tt2 (utt.relation.append w1 'Target))
(item.set_feat tt2 "id" "_5")
(item.set_feat tt2 "name" "aa")
.....
(utt.relation_tree w1 'Segment)
(utt.relation_tree w1 'Target)

尚未完成.....

嘗試以 segment 的層次來切入，首先看看在 lexicon 中

festival>(lex.lookup 'hello)

的 diphone 為何

定義好Utterance，在合成時就發生錯誤了
不認識Segment？看來要走另一條路了

```
("hello" nil (((hh ax l) 0) ((ow) 1)))
festival>(set! sen (Utterance Segment ((hh 0.058) (ax 0.039) (l 0.069) (ow 0.219))))
festival>(utt.synth sen)
SIOD ERROR: Unknown utterance type : Segment
```

lexicon中的對應是否可以修改呢？

果然發出reagan的聲音喔

```
festival>(lex.lookup 'hello)
("hello" nil (((hh ax l) 0) ((ow) 1)))
festival>(lex.add.entry '("hello" nil ((( r ey g) 0) ((ow) 1))))
festival>(SayText "hello")
```

Para aprovechar las ventajas de la fiesta emitido por china bastante limitada, ya que hay que aprovechar el lenguaje existente y el festival léxico se puede combinar con un sonido similar, pero léxico proporcionado actualmente son Estados Unidos, Reino Unido, español, tres phoneset, después de festival de carga, Echemos un vistazo al léxico predeterminado y escuchemos a los Estados Unidos o el Reino Unido, y cambiemos el idioma, y luego observemos los cambios

```
Festival> (PhoneSet.list)
(Radio)
Festival> (Intro)
Festival> (voice_rab_diphone)
Festival> (PhoneSet.list)
(Radio MRPA)
Festival> (Intro)
Festival> (voice_don_diphone)
Festival> (PhoneSet.list)
(Radio mrpa holmes)
festival> (introducción)
```

El ejemplo anterior, para los países que no hablan Inglés es un poco difícil para nosotros eran acento americano Inglés e Inglés británico, pero hicieron ver un tel cambio, veamos más detalles de estos tel

Inglés británico: voice_rab_diphone

Español: voice_el_diphone

¿Extraño es que la elección original que corresponde al idioma, su PhoneSet también se debe seguir, pero los resultados medidos no son necesariamente verdaderos?

```
Festival> (voice_rab_diphone)
(rab_diphone)
Festival> (PhoneSet.list)
(Radio MRPA)
Festival> (PhoneSet.description)
((nombre MRPA) .....
Festival> (voice_el_diphone)
el_diphone
Festival> (PhoneSet.list)
(Radio MRPA español)
Festival> (PhoneSet.description)
((nombre español) .....
```

Tour de función esencial de estructura de voz

(utt.relationnames UTT)
Devolver UTT esta lista contiene todo el nombre de las relaciones

```
Festival> (SET! SEN (Expresión, Palabras (Hello)))
Festival> (utt.synth SEN)
(sílaba palabra Frase Segmento SylStructure IntEvent Intronation Objetivo Unidad SourceCoef F0 TargetCoef US_map Wave)
```

(utt.relation.items UTT RELATIONNAME)
Devuelve UTT esta lista todos los indicadores RELATIONNAME items

```
Festival> (utt.relation.items Segmento SEN')
<# <Elemento 0xb28a828>
  # <Elemento 0xb289fc8>
  # <Elemento 0xb28a168>
  # <Elemento 0xb28a308>
  # <Elemento 0xb28a688>
  # <>> Artículo 0xb28a8e0
```

(utt.relation_tree UTT RELATIONNAME)

```
Festival> (utt.relation_tree Segmento SEN')
```

Devuelve la lista UTT de todos los elementos en RELATIONNAME detalles, y el árbol dijo

```
((( "Pau"
  ((ID "_9")
  (nombre de "Pau")
  (dur_factor 0)
  (0,22 End)
  (0,081826 source_end))))
(( "HH"
  (el id anteriormente mencionado "_4")
  (nombre de "HH")
  (dur_factor -0.296956)
  (0,277954 End)
  (source_end 0,0188655))))
...
...
```

Descripción del Festival Abstracts

Inicio rápido

festival> (set! **utt1** (texto de pronunciación "hello world"))	// Genera una pronunciación del objeto almacenado en esta variable utt1
festival> (utt.synth **utt1**)	// Sintetiza el objeto utt1
festival> (utt.play **utt1**)	// utt1 objeto para tomar el sonido
festival> (SayText "hola mundo")	// Simplifica los tres pasos anteriores
festival> (utt.save.wave **utt1** 'myutterance.wav)	/ / El objeto utt1 se generará después del archivo waveform wav especificado
festival> (Palabras de presentación (hola mundo))	// Producir una palabra del objeto
festival> (Utterance Phones (pau hh ah l ow pau))	// Crear un objeto de teléfonos
festival> (Frase de Utterance ((Frase ((nombre B)) Vi)))	// Crea un objeto Phrase

Esquema

El festival está controlado por la sintaxis Scheme , que es un conjunto de declaraciones encerradas entre paréntesis

Emacs y Sable

Disponible por Emacs como una interfaz para seleccionar los bloques deseados en la traducción de ival Fest
Sable este lenguaje de marcado de XML formato de datos basado en proporcionar una buena voz, la primera versión se basa SSML (SpeechSynthesis Markup Language)

festival> (tts "filename.sable" 'sable)	// Se puede leer el archivo de formato de sable especificado
# festival --tts nombre de archivo.sable	// modo de comando lee directamente el archivo de formato de sable especificado

Expresiones

festival El objetivo básico de la síntesis del habla es la pronunciación. Aquí se explica cómo convertir las palabras en pronunciación

1. tokenización
2. identificación del token
3. token a palabra
4. parte del discurso
5. fraseo prosódico
6. búsqueda léxica
7. Acentos entonacionales

8. asignar duración
9. generar contorno F0 (sintonizar)
10. renderizar forma de onda

Use software gratuito para implementar el discurso sintetizado que se aplica al campo exclusivo

Resumen

Este artículo describe una forma confiable y eficiente de construir un discurso sintético que se aplica a un dominio limitado. Un discurso sintetizado completo es capaz de todas las palabras e incluso símbolos. Sin embargo, en algunas aplicaciones, piezas y plazo fijo sin la necesidad de que cada palabra puede sonar, como un 117 estaciones de hora normal o sistema de predicción del tiempo sólo se requiere el dominio exclusivo de la voz sintetizada, vamos a implementar el siguiente conjunto para grabar su propia voz Del archivo de voz de tiempo con las herramientas de síntesis de voz Festival y las herramientas de creación de sonido Festvox.

Introducción

También conocido como el habla de síntesis de voz de texto (Text-to-Speech), la gama de aplicaciones que puede ser bastante extensa, por ejemplo, como el tiempo o el sistema de distribución de mensajes de hora normal, DQ del sistema, sistema de activación con voz Mangbao, libros de audio, con el aumento se produjo en aplicaciones de voz se puede encontrar, la mayoría de la voz es todavía el método de tomar pre-grabado sin el uso de sistemas de síntesis de voz para producir el habla, la razón principal es que, si es aceptable a la entrada de un texto o un artículo , el propio texto no contiene ningún características acústicas (característica acustic, como los tonos altos y bajos, el modo de pausa, pronunciación y longitud prosódicos), sólo las características lingüísticas, debe ser posible generar carácter acústico de estos mecanismos a través de pronóstico automático enfoque cuando las características predichas del llamado mecanismo automático, por lo general basada en reglas con dos tipos de datos impulsadas, pero la calidad de la síntesis de dos técnicas de sonar aburrido y poco atractivo y desea mantener experimentando expresión continua o rendimiento de sonido del altavoz no es bueno , Así que la concatenación de sílaba popular reciente (concatenación de sílaba basada en corpus), es a una Buena corpus de voz récord como el tema de la comparación, la unidad de composición de la captura correspondiente al corpus, en comparación con el enfoque basado en reglas tienen que ver con los datos de la rima impulsado ajustar los detalles de lo que reduce mucho, por lo No solo tiene la misma naturalidad que el método pregrabado, sino que también reduce el tiempo y el costo de la grabación masiva.

Tal como se usa en este documento, el centro de investigación británico en software de síntesis de voz el lenguaje de la Universidad de Edimburgo desarrollado por el Festival, que utiliza dos tipos de tecnología para mejorar la calidad de síntesis de voz, unidad de síntesis de voz método de selección (síntesis de selección de la unidad) y la síntesis diphone UTV (diphone generales síntesis), y con características de voz únicas de campos fijos y una pequeña cantidad de cambio en Benpian utilizan para seleccionar la unidad de síntesis de voz se simplifica el proceso es para hacer avanzar la voz deseada grabado, de forma automática cuando la síntesis de voz Seleccione unidades fonéticas similares para el sonido, por lo tanto, en comparación con el TTS general basado en la selección de unidades, el sonido es más suave y natural.

Planificación para la producción de entorno de voz

Antes de hacer un archivo de voz en off, tenemos que hacer algunos preparativos

1. Encontrar el hablante correcto:
   dado que el timbre del hablante afecta la calidad del habla sintetizada, también le da libertad al hablante para interpretar la base de datos de voz sintética
2. Encontrar micrófonos y tarjetas de sonido de buena calidad:
   porque queremos construir una biblioteca que permita utilizar las técnicas de selección de células
3. Instale el siguiente kit relacionado: la
   instalación y el uso básico pueden hacer referencia a http://www2.cyut.edu.tw/~s9154610/festival.html#install
   1. Instalar herramienta de síntesis de voz: festival
   2. Instalar herramientas de producción de voz: festvox
   3. Instalar herramientas de análisis de voz: emulabel
4. Lugar de instalación:
   / home / peter / projects / (de acuerdo con el lingüista instalado en el subdirectorio de proyectos del directorio de inicio, el caso del nombre del hablante peter)
5. Establecer variables de entorno relacionadas:

   La ruta de la PATH = $ exportación: / Home / Peter / Proyectos / Festival / bin
   exportación FESTVOXDIR = / home / pedro / Proyectos / FestVox para
   exportación ESTDIR = / home / pedro / Proyectos / speech_tools

6. Probar el efecto de grabación y la calidad:
   Elegimos a la computadora personal con sistema operativo Linux para la grabación, la desventaja es que la calidad puede verse

afectada por la señal electrónica, la ventaja del control más conveniente.

- Use el programa aumix para ajustar el volumen de entrada del micrófono y el volumen de salida del auricular.
- Si ve este mensaje de error "No se puede abrir el archivo de salida '/ dev / dsp': dispositivo o recurso ocupado" puede tener 2 soluciones
  - Para una tarjeta de sonido estándar: como la tarjeta Sound Blaster
  - Utilice los ajustes del menú Xwindow / Sistema de KDE / audio / sonido en el que la voz me pestaña de E / S, el modo de entrada y salida de sonido de "entrada Moyoushengyin / salida"
- Grabe un tono de 5 segundos, escuche el estado del ruido del micrófono y continúe ajustando el ajuste al estado OK.

```
$ ESTDIR / bin / na_record -f 16000 -time 5 -o test.wav -otype riff
$ ESTDIR / bin / na_play test.wav
```

- O la grabación de Sox, pero también el uso de la reproducción de Sox, escuchar para ver el estado del ruido del micrófono y continuar ajustando el ajuste al estado OK.

```
ESTDIR $ / bin / Medias -c2 -sw ossdsp -r -t 16000 / dev / DSP C1 -sw -r 16000 test.wav
$ ESTDIR / bin / Medias test.wav -c2 ossdsp -t / dev / DSP
```

- También puede mostrar formas de onda y espectro (necesita instalar emulabel con tcl y emu)

```
$ FESTVOXDIR / src / general / display_sg test.wav
```

Hay varias maneras de reducir la interferencia de ruido
1 para evitar el cable del micrófono y otro cable, especialmente el cable de alimentación.
2. cambiar la perspectiva y ubicación de la computadora.
3. Cambia la posición de la ranura de la tarjeta de sonido.

7. La paciencia y la perseverancia:
   Para grabar una voz satisfactoria a veces toma un tiempo considerable, trata de ser un registro completo, porque la voz del estado en diferentes momentos diferentes para todo el mundo, en busca de los mismos entornos de grabación y condiciones o estado de ánimo, todo el equipo Ajuste la prueba al mejor estado posible para lograr la mejor calidad de voz.

Construye tu voz

Crear una base de datos de cronometraje

首先我們在使用者家目錄中建立存放語音的目錄 data/time，然後透過festvox提供的命令稿(scheme file)讓我們可根據應用的「機構」，「語者」，「領域」的參數建立在語音目錄下所需的資料庫

- 所屬機構名稱：cyut(朝陽科技大學，若僅為個人使用建議為net)
- 應用領域名稱：time (本例應用在報時上)
- 錄音的語者名稱：peter

```
mkdir -p ~/data/time
cd ~/data/time
$FESTVOXDIR/src/ldom/setup_ldom cyut time peter
```

執行 setup_ldom 這個命令稿後我們可以發現它在 data/time 下產生了許多為了存放合成語音的目錄與檔案，特別的是在 data/time/festvox 子目錄下有4個用來控制 festival 的 scheme 命令稿，我們稍後可能需要編輯它。(檔案變動列表)

文後所提及之目錄名稱，除非特別指定，否則皆以 ~/data/time 為相對目錄起始點。

檔案變動列表利用 ls -1R 取得目錄列表。

設計提示語(Designing the prompts)

根據應用的領域我們必須設計一個提示語的參考檔，通常是以 **DOMAIN**.data 命名，如本例應用在語音領域就應該取名為 time.data 這個提示語的參考檔必須放在語音目錄的etc下，但因為我們使用了 festvox 提供的自製報時語音命令稿(下面例子的 build_ldom.scm)，因此 etc/time.data 已被自動建立，值得注意的是若我們要自行產生 DOMAIN.data ，其中不可有空行，且左括號之後要空一格。

```
(time0001 "El momento es ahora, ...")
(time0002 "El tiempo es ahora, ...")
(time0003 "El tiempo es ahora, ...")
.....
.... ....
(time0024 "El momento es ahora, ...")
```

De acuerdo con la grabación rápida de voz (Grabar las indicaciones)

Festival ofrece se emite una orden según la build_ldom.scm sugerido idioma etc / time.data utilizado para generar una grabación de voz, que producirá time0001 ~ time0024 en cada línea de laboratorio, pedirá-UTT, pedirá-wav otros tres directorios 24 archivos. (Lista de cambios de archivos)

```
festival -b festvox / build_ldom.scm '(build_prompts "etc / time.data")'
```

Siguiente necesidad de teclear la instrucción prompt_them, festival se lee de acuerdo con el pronto establecimiento de una oración ha llegado, cuando vemos "a partir de la grabación", seguido después de leerlo, altavoz del festival será emitido por la voz generada time0001 almacenados en el directorio wav ~ time0024 y otros 24 archivos wav, debe tenerse en cuenta que más de 10 segundos de voz gastarán mucho espacio de intercambio. (Lista de cambios de archivos)

```
bin / prompt_them etc / time.data
```

Etiquetar automáticamente las indicaciones

Escriba comando make_labs para grabar un buen indicador de comparación de voz, el resultado será en cep, lab, prompt-cep y otros tres directorios tienen time0001 ~ time0024 los 24 archivos. (Lista de cambios de archivos)

```
bin / make_labs prompt-wav / *. wav
```

A continuación, debe escribir el comando build_ldom en el directorio festival / utts time0001.utt ~ time0024.utt 24 archivos. (Lista de cambios de archivos)

```
festival -b festvox / build_ldom.scm '(build_utts "etc / time.data")'
```

Extracción de símbolos de tono y construcción de coeficientes de LPC

Tomar buenas marcas de tono de wavs es de gran ayuda para mejorar la calidad de los composites, generando 24 archivos, como time0001.pm ~ time0024.pm, en el directorio pm. (Lista de cambios de archivos)

```
bin / make_pm_wave wav / *. wav
```

Ajuste de la marca de afinación para mejorar la calidad, pero no nuevos catálogos o archivos (lista de cambios de archivos)

```
bin / make_pm_fix pm / *. pm
```

Regularización, resultando en el directorio wavn y generando time0001.wav ~ time0024.wav 24 archivos. (Lista de cambios de archivos)

```
bin / simple_powernormalize wav / *. wav
```

Produzca parametrización de sincronización de tono, en el directorio mcep time0001.mcep ~ time0024.mcep 24 archivos. (Lista de cambios de archivos)

```
bin / make_mcep wav / *. wav
```

Eliminar una unidad de síntesis de voz de la pronunciación (Crear un sintetizador basado en clunit a partir de las emisiones)

La unidad de síntesis extraída contiene propiedades fonéticas y prosódicas para construir un árbol de decisión.

Si no registra el tiempo, solo aproximadamente 3 minutos para completar la configuración.

También puede usar el idioma que no sea inglés para establecer el archivo de voz que se usa en el control de tiempo Oh.

- Se agregó un archivo para cyut_time_peter.catalogue en el directorio festival / clunits
- 132 archivos agregados en el directorio festival / proezas (* .feats)
- En el directorio festivales / árboles, se agregaron 133 archivos (* .tree)

(Lista de cambios de archivos)

```
festival -b festvox / build_ldom.scm '(build_clunits "etc / time.data")'
```

Prueba y ajuste

Comienza el festival para cargar el nuevo sonido

```
festival festvox / cyut_time_peter_ldom.scm '(voice_cyut_time_peter_ldom)'
```

La lectura apareció a tiempo

```
(saytime)
```

Leer el tiempo especificado

```
(saythistime "11:23")
```

Referencias

1. Black, A. y Lenzo, K. Creación de voces en el sistema de síntesis de discursos del Festival. Http://festvox.org, 2000
2. Black, A., and Lenzo, K. Limited Domain Synthesis.

說明檔翻譯(未完成)

- 第20章 UniSyn synthesizer(統一的合成器)
  自1.3版起加入了新的通用型合成模組。這個設計用來取代舊的雙音素合成器將在下個章節來說明，為了連續音而非雙音素的的語音進行一般化的波形合成器與訊號處理模組而需要 重新設計之，同時在完整的雙音素資料庫的預處理函數會被加至語音工具函式庫
- 20.1 Unisyn database format(統一的合成資料庫格式)
  統一的合成模組以兩種基本格式來使用資料庫,分離式與群組式,分離格式指的是當所有的檔案(信號,音高符號與係數檔)在合成時分別地被存取,這是在資料庫發展時的標準使用方 式,群組格式指的是蒐集了所有聲波合成所需的資訊到一個獨特的檔案的資料庫,這種格式是設計用來散佈與一般使用的資料庫.
  資料庫應該由一組波形(或許應該稱為訊號處理方法所需的一組係數),一組音高符號與索引所組成.我們現在的信號處理是需要音高符號來作音高的同步
- 20.1.1 Generating pitchmarks(產生音高符號)
  音高符號可以從語音工具函式庫中驗證過的程式pitchmark所產生的喉音圖檔案中得出,針對這個程式要使用正確的參數還是有點藝術形式.第一個主要議題是lar檔 案格式的未來走向.我們已經看過2者雖然它看起來像是CSTR機構提供的亂七八糟的東西,相較於其他機構的正確方向上,在pitchmark上的-inv這個參數正是為了 此檔案格式所提供,其它的議題是獲取音高符號的校正,基本產生音高符號的命令如下
- ```
  pitchmark -inv lar/file001.lar -o pm/file001.pm -otype est -min 0.005 -max 0.012 -fill -def 0.01 -wave_end
  ```

  -min,-max,-def(無聲的範圍)這些參數都需根據與者的音高範圍來調整.以上適合男性的語者.-fill這個參數代表無聲的段落需要被填上等量的空白音高記號
- 20.1.2 Generating LPC coefficients(產生LPC的係數)
  LPC的係數可用sig2fv這個命令來產生,可分為產生係數與餘數2個步驟,示範命令如下
-

coefficients	sig2fv wav/file001.wav -o lpc/file001.lpc -otype est -lpc_order 16 -coefs "lpc" -pm pm/file001.pm -preemph 0.95 -factor 3 -window_type hamming
residual	sig2fv wav/file001.wav -o lpc/file001.res -otype nist -lpcfilter lpc/file001.lpc -inv_filter

  Es posible que necesite regularizar su intensidad de volumen en algunas bases de datos. Puede ser un poco más difícil normalizar la intensidad del volumen, pero proporcionamos una función de ejemplo que hará que este trabajo sea aceptable, y necesita para ello (la acción tomada y el resto se determinó que después de la normalización en lugar del perfil original de forma de onda)
  ch_wave -scaleN 0,5 WAV / file001.wav -ofile001.Nwav
  volumen normalizado para maximizar la señal se multiplica entonces por la administración Coeficiente. Si la forma de onda de la base de datos está limpia, al hacerlo obtendrá resultados razonables.
- 20.2 Generando un índice de difonos
- 20.3 Declaración de base de datos (declaración de la base de datos)
- 20.4 Hacer grupos de archivos (hacer archivos de grupo)
- 20.5 Selección del módulo UniSyn (Selección del Módulo de Síntesis Unificado)
- 20.6 Selección de diphone (selección de dos teléfonos)

Términos relacionados

Festival de Edinburgo

Un festival de arte internacional (especialmente música y teatro) que comenzó en 1947 se lleva a cabo en Edimburgo, Escocia, cada tres semanas en agosto. Organizado por R. Bing antes de 1950. El director de arte actual es F. Dunlop. Además de las actividades oficiales del festival, el "aumento temporal de programas" está aumentando y el festival es animado y vívido.

CSTR (El Centro de Investigación de Tecnología del Habla)

CSTR es un centro de investigación para todo tipo de aprendizaje, y actualmente está trabajando en la aplicación de la investigación fonética

Recursos de internet

- Festiva
  ```

- Festvox
- 11-752: Horario: primavera de 2004
- wikipedia
- Construyendo Voces Sintéticas
- Creación de voces de dominio limitado para su uso con el Festival
- SÍNTESIS DE DOMINIO LIMITADO
- Construyendo un sintetizador de Dominio Limitado con Festival
- Ejemplo de síntesis en línea
- El Festival Speech Synthesis System - Documentación del sistema
- Aplicación de procesamiento de sonido
- Ejemplos de discurso sintetizado
- Duke Speaks: síntesis de voz libre
- Definición de diphone para la síntesis del habla del mandarín
- Síntesis del habla - Referencias
- Sistema de Síntesis de Voz basado en HMM (HTS)
- Duke Speaks: síntesis de voz libre
- FestVox a FreeTTS
- HTS Voces para el Festival
- Descripción general de la tecnología de síntesis del habla
- Proyecto DSP: Speech Synthesis

Volver a la página principal

Página web principal: http://peterju.notlong.com (actualmente reenviado a http://irw.ncut.edu.tw/peterju/)

WAI-AAA WCAG 1.0 | ANY browser 0.0 | XHTML 1.0 | | | Este trabajo está licenciado bajo una licencia de Creative Commons CSS

VALID RSS SOME RIGHTS RESERVED creative commons