Capitulo 5 Síntesis de voz utilizado Selección de Unidades

5.1 Definición de Unit Selection

La técnica de selección de unidades (Unit Selection) se define como la búsqueda a través de un corpus de voz, de aquellas unidades que posean características similares o iguales a las de la frase que se desea sintetizar [Hunt, 1996]. Es una metodología de síntesis de voz mediante la cual podemos concatenar las formas sonoras de diferentes estructuras gramaticales, tales como los fonemas, difonemas, trifonemas, etc., esto con la finalidad de obtener sonidos más naturales. Considera características de prosodia (subjetivamente), de identidad fonética y la preferencia por incluir unidades de diferentes tamaños (de preferencia contiguas), cuando se eligen los elementos que participarán en la construcción de una frase [Flores, 2001].

5.2 Desarrollo de sintetizador

Retomando la tesis de Leonardo Flores, en la que generó un algoritmo y su respectiva programación para la ejecución de la síntesis de voz, nos explica las necesidades y requerimientos para su elaboración. En el desarrollo del corpus realizado en este trabajo, se requiere de este sistema para concatenar las unidades y reproducir la voz que se puede generar con el corpus Gama.

5.2.1 Interfaz del sintetizador

El sintetizador es un ente que responde a la entrada que le da el usuario para lo cual va a tratar de obtener su correspondiente valor en audio.

- El usuario cargará o ejecutará el sistema con las unidades que se trabajarán del corpus, por eso, el usuario describe la ruta donde se encuentra el folder que contiene todo el corpus de voz y sus transcripciones.
- El usuario introducirá la cantidad de archivos en los que desea se realice la búsqueda.
- El usuario procede a introducir la información que desea sintetizar. Si él escribe el texto "Hola", el sintetizador va a reproducir como respuesta una onda sonora que contenga la palabra "Hola".
- Para ejecutar el sintetizador tendrá que dar un click en el botón "Leer".

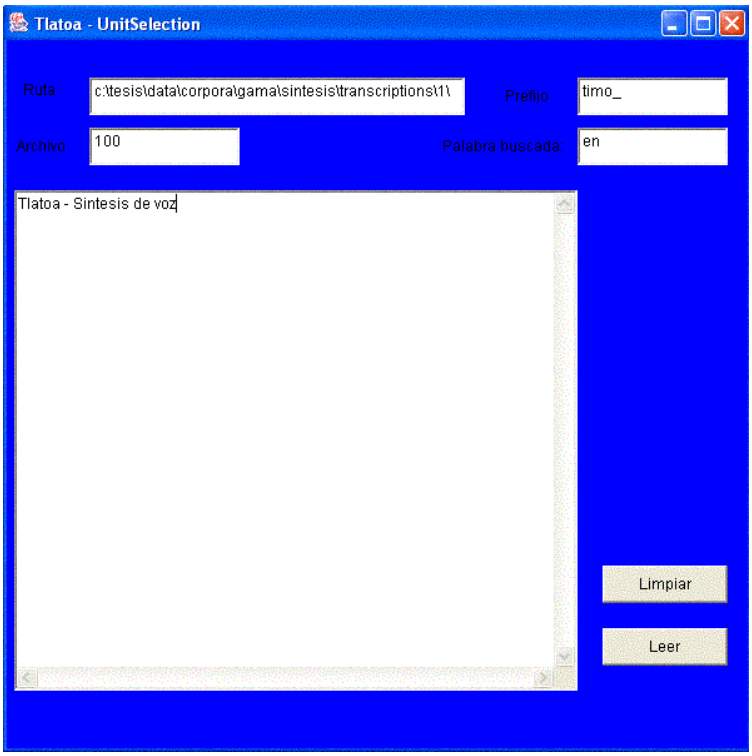

Figura 5.1 Interfaz del sintetizador

La reproducción de sonido es claro, ya que este corpus abarca todos los posibles sonidos y genera dos voces tanto de hombre como de mujer, para lograr un corpus con fines generales y tener una amplia gama de opciones para buscar fines específicos.

5.2.2 Diseño del sistema

Fue necesario primero contar con las transcripciones fonéticas a nivel texto (.txt), palabra (.wrd) y fonema (.phn) para poder construir las frases solicitadas. Para esto Leonardo Flores utilizó la técnica de Pellom [Pellom, 2001] utilizando listas ligadas, debido a que esta estructura es la más adecuada y fácil de implementar. Sin embargo, esta técnica hace una búsqueda lenta de las unidades, pero de cierta manera es eficiente en la concatenación de unidades.

Para poder cargar el sistema con el conjunto de elementos que constituyen nuestro corpus, el usuario debe de seleccionar la voz que utilizará para realizar el proceso de síntesis y posteriormente disponerse a teclear el texto que desea sintetizar. Internamente este proceso consiste en identificar la ruta del folder donde están contenidas todas las transcripciones a nivel de palabra (.wrd) y fonemas (.phn); asimismo, se considera el prefijo con el cual empiezan todos los archivos de este corpus ("timo_" o "lizy_") y se define el número de elementos que participan en el experimento, esto es, el número de frases del corpus de voz para la cual se tengan los wavs y las transcripciones fonéticas.

La información de la ubicación del fólder, sirve para poder hacer la lectura de todos los archivos que contienen las transcripciones fonéticas, esta función será realizada por el módulo de Lectura de archivos. Cada archivo de texto será leído línea por línea e introducido a un vector de cadenas que será regresado al Administrador de Información.

El vector de cadenas que sea regresado por el módulo de Lectura de Archivos, el Administrador dividirá cada enunciado de manera que puedan obtenerse un total de 4 atributos o parámetros:

1. La palabra, que representa la fracción de texto que se está extrayendo (en el caso de los archivos .phn es un fonema).
2. El archivo wav que lo contiene, que servirá para buscar el archivo de audio que posea la representación acústica de la oración.
3. Las fronteras superior e inferior, que serán los límites que identificarán la posición que debe de cortarse al wav para producir un nuevo elemento.

Estos parámetros serán pasados por el Administrador a un objeto de clase *AlmacenamientodeDatos*, que representa una estructura creada para almacenar específicamente estos valores. Cada objeto de clase *AlmacenamientodeDatos* será pasado al módulo de *CreacióndeListas*, donde será agregado como nodo a lista ligada. Esto quiere decir que a la larga existirán tantos nodos como palabras o fonemas haya en una oración.

El vector de cadenas es el que se encargará de ejecutar correctamente el sistema de síntesis de voz, ya que una vez que haya procesado el texto de entrada, será pasado junto con la información que se desea sintetizar, a los módulos encargados de la construcción de oraciones y palabras.

5.2.3 Procesamiento de Texto

El módulo de Procesamiento de Texto es el encargado de esta tarea y tiene como objetivos:

1. Separar al texto en tokens.
2. Procesar los signos de puntuación.
3. Expandir las abreviaturas a enunciados normales.
4. Procesar las excepciones que pudieran existir en el idioma.

La primera de las tareas consiste en separar a toda la cadena que se desee procesar en unidades lexicográficas más pequeñas denominadas tokens. Dichas unidades son divididas unas de otras mediante delimitadores que por lo regular son espacios en blanco, tabulaciones, saltos de línea y retornos de carro. [Deitel, 1998]

La segunda tarea consiste en procesar los signos de puntuación de manera que podamos eliminarlos del token y transcribirlos como una pausa dentro del texto.

El punto que corresponde a la expansión de abreviaturas tiene que ver con la traducción de notaciones especiales como: números, abreviaturas y precios a las palabras que representan. A nivel de código esta función es realizada por el método *expansiónDeAbreviaturas* de la clase *ProcesamientoDeTexto* [Barbosa, 1997].

La última de las funciones del módulo de Procesamiento de texto está encaminada a resolver aquellas excepciones que encontramos en el idioma y que son producto de los alófonos (variaciones en los sonidos de un fonema) de la lengua con la que se está trabajando. Lo que se pretende encontrar con esta evaluación es una forma de pronunciar correctamente palabras como “México” que debe de escucharse Méjico pero que bajo las normas de trascripción fonética normales se interpreta de otra forma.

El código fuente que realiza esta función está especificado en el método *excepcionesEspañolMexicano* de la clase *ProcesamientoDeTexto.*

Una vez que la información ha sido trabajada y ésta está separada por palabras, es agregada a un vector que llamaremos información a sintetizar. Este vector es enviado al módulo de síntesis, junto con el vector de listas ligadas para palabras y el vector de listas ligadas para fonemas que contienen las transcripciones fonéticas del corpus. El encargado de recibir estos tres vectores es el administrador de la información que únicamente los pasa al módulo de selección de unidades donde son captados por el Constructor de Oraciones quien se encargará de manejar esta información.

5.2.4 Construcción de los archivos de audio

El responsable de la búsqueda y obtención de datos de las palabras candidatas a concatenarse es el Constructor de Oraciones. Este es un módulo que se encarga de verificar si existe en el corpus, cada palabra que está almacenada en el vector de información a sintetizar. Si este módulo la encuentra en alguno de los nodos de las listas ligadas que almacenan la transcripción fonética, la considera como propia para la concatenación, aunque está sujeta a un proceso de decisión que se describirá a continuación. Si el Constructor de Oraciones no es capaz de encontrar una palabra la manda al Constructor de Palabras para que aquí pueda ser elaborada.

5.2.5 Proceso de depuración del vector de contextos

Las palabras que constituyen nuestro abanico de posibilidades para formar la oración que especificamos poseen mucha redundancia, así que tiene que depurarse. Para eso se

estableció una serie de reglas en el método *depurandoCandidatos* de la clase *ConstructorDeOraciones*.

a) **Para dos unidades encontrados en un mismo wav**

b) **Para dos unidades encontradas en diferentes archivos de audio.**

5.2.6 Proceso de construcción de una palabra

Para construir una palabra a partir de sílabas, primero se tiene que transcribir la palabra a un nivel de transcripción fonético, para eso es necesario aplicar todas las reglas que se tienen en una lista de etiquetas (ver Apéndice A), proporcionadas por el grupo TLATOA, utilizado para representar los fonemas del Español Mexicano. Ya que se tiene la transcripción fonética a nivel de fonemas de la palabra que se quiere construir, se procede a silabificar usando las reglas especificadas en el conjunto que a sido modificado de su versión inicial, para adaptarlo al proceso de etiquetación que utiliza el laboratorio de TLATOA en la actualidad.

Estas reglas sirven para averiguar si un fonema pertenece a la sílaba o no. Para determinarlo se extraen uno a uno y de derecha a izquierda, los caracteres de la transcripción fonética que se esté evaluando. Se compara cada valor con los fonemas encontrados en la parte izquierda de la tabla. Si coincide el fonema actual con alguno de éstos, determinamos cual es el fonema que lo antecede en la transcripción y lo comparamos con la lista que se encuentra a la derecha de este valor. Esta lista representa el conjunto de fonemas válidos que pueden anteceder al fonema actual, así que si el fonema anterior al actual coincide con estos caracteres, quiere decir que el fonema anterior pertenece a la sílaba actual, en caso contrario el fonema anterior representa el

inicio de otra sílaba. Este proceso resuelve correctamente las palabras de tipo CV (Consonantes-Vocales), sin embargo, tiene problemas para hacerlo con las formas VC o CVC así que se propone que se asignen las consonantes que queden sobrando (aisladas) a la sílaba que se encuentre a su izquierda.

Cuando ya se tiene silabificada la palabra se agregará a cada unidad al vector de sílabas, posteriormente enviaremos dicho vector al método *localizaCandidatos* de la clase *ConstructorDeOraciones*, que representa al elemento encargado de la selección de las unidades que participarán en la concatenación. Cuando hayamos decidido que unidades son las que consideraremos para construir la oración, se ejecutará un script de TCL para cortar los fragmentos de audio que se requieren para reproducir, posteriormente se ejecutará otro script para construir el archivo final para su reproducción.

5.2.7 Composición del sistema

En esta sección se da una breve explicación del funcionamiento de la aplicación, de esta manera se menciona cada una de las clases que conforman el sistema antes mencionado y su diagrama de clases.

- **unitselection.java**

  Es la clase principal que agrupa todas las clases que se mencionan más adelante, también realiza la interfaz para la interacción entre el usuario y el sistema.

- **ConstructorDeOraciones.java**

  Esta clase servirá para crear los wavs para síntesis sólo para palabras y se

encarga de encontrar dentro de todo el vector de listas ligadas, a aquellos elementos que sean iguales a la palabra que se extrajo de la información a sintetizar y sus contextos derecho e izquierdo, también se encarga de depurar el conjunto de candidatos que se obtuvieron en el vector de contextos del método que *localizaCandidatos*. una vez depurado nuestro vector, los valores que quedan pueden ser ocupados para realizar la síntesis a nivel de palabras.

- **Datos.java**

  Esta es una clase que contendrá los datos principales de un archivo .wrd (palabra) inicializa los atributos que se señalaron como necesarios para manipular la información que tenemos en los archivos .wrd (el valor es asignado internamente), también del exterior. Recibe como parámetros dos *long* que se refieren a las fronteras de la palabra ya sea inferior o superior, También recibe dos cadenas una para identificar la palabra y otra para el wav correspondiente.

- **Lista.java**

  La clase *Lista* se encarga de implementar una lista enlazada, agregar objetos de clase *Datos*. Tiene un método que imprime la información que se encuentra almacenada, también regresa un objeto de clase Datos y su propósito es verificar si el String que se recibe como parámetro coincide con el atributo palabra de uno de los nodos de la lista ligada y regresa una lista ligada que tendrá valor null (vació) si no tiene almacenado ningún objeto y en el caso contrario tendrá Objetos de clase Datos.

- **execute.java**

  Esta clase nos sirve para realizar la transcripción fonética de las palabras y nos servirá para seleccionar a los elementos que habrán de participar en la concatenación.

- **CandidatosConcatenacion.java**

  Regresa el valor del wav candidato, los valores del contexto izquierdo del wav candidato. Regresa los valores del contexto central del wav candidato, los valores del contexto derecho del wav candidato, el peso correspondiente al número de contextos que se encontraron en el wav candidato a síntesis, el número correspondiente a la palabra que es parte del wav candidato a síntesis y el código del wav candidato. Se encarga de asignar el nombre del wav que es candidato a sintetizarse, el contexto izquierdo del wav que es candidato a sintetizarse, el contexto derecho del wav que es candidato a sintetizarse, el contexto central del wav que es candidato a sintetizarse, el peso correspondiente a los contextos que se manejan en el wav que es candidato a sintetizarse, puede manejar sólo 3 valores, 1 si tiene un sólo contexto en este caso el central, 2 si maneja los contextos central-izquierdo o central-derecho y 3 si todos los contextos se encuentran presentes, también asigna el numero de palabra que corresponde al contexto central que se esta evaluando, este número proviene de la posición que ocupa la palabra que representa el contexto central dentro de la frase a sintetizar.

  Por ejemplo en la transcripción fonética de “hola amiguito”, pau -> hola -> amiguito -> pau. El numero de palabra correspondiente a amiguito es el 3.El código es una simbología que sirve para identificar los contextos que se tienen presentes para una palabra dada. Por ejemplo, "111" si tenemos los tres contextos, "010" si sólo se tiene al contexto central, "110" si el contexto es central-izquierdo o "011" central-derecho

- **LecturaArchivos.java**

  Esta clase permitirá hacer la lectura de un archivo secuencial, recibe como parámetro una cadena que indica la ruta donde encontramos el archivo que se desea leer y podemos tener acceso a la información que existe en el archivo, así cada línea será accesada de manera individual e introducida a un Vector de Strings (cadenas).

- **Fronteras.java**

  Con el fin de dar las fronteras correctas regresa un long que corresponde al valor de la frontera inferior, otro que corresponde al valor de la frontera superior y un String que corresponde al valor de la palabra. Se encarga de asignar un valor (long) a la frontera inferior, otro a la frontera superior y asigna un valor (String) al atributo palabra.

- **ProcesamientoDeTexto.java**

  El procesamiento de texto nos sirve para realizar la transcripción fonética de un texto a nivel de textos, la información que de aquí se obtenga es almacenada en un vector y regresada al módulo.

- **TranscripcionFonetica.java**

  Es una clase que almacene la transcripción fonética para los fonemas como "b" su transcripción fonética es "bc b". Se asigna una subcadena (una o más letras) al atributo fonemas y la transcripción fonética correspondiente a la cadena que se especificó en el atributo fonemas. Se encarga de regresar el valor que tenga el atributo fonemas y el valor del atributo transcripción, también se encarga de deducir la transcripción fonética para una cadena determinada

- **ConstructorDePalabaras.java**

  Hay que recibir en el constructor al vector de listas ligadas y a la palabra que se

requiere construir, partimos la subcadena en dos porciones partiendo de derecha a izquierda, se determina como estaban establecidos los fonemas que quedaban dentro del vector de secuencias de fonemas, se concatena la parte derecha del fonema actual con el resto de las secuencias, se crea una nueva secuencia de fonemas con la nueva cadena y actualizamos el vector de secuencia de fonemas.

- **Duerme.java**
- **Audio.java**
- **Transcripcion.java**
- **DatosFonema.java**
- **Silabifica.java**

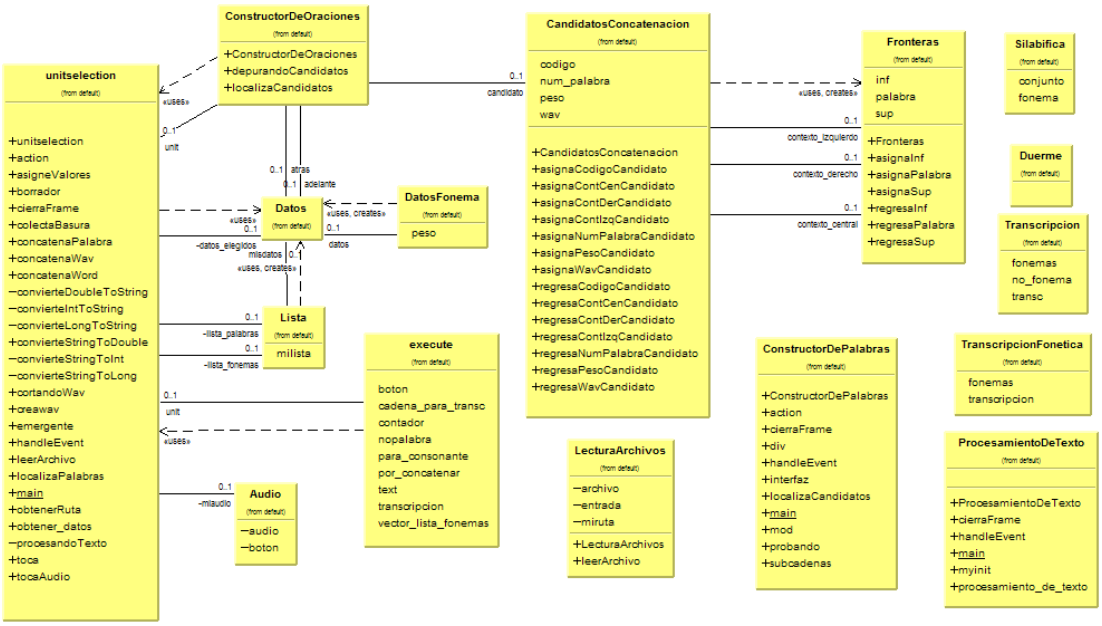

Figura 5.2 Diagrama de clases del sintetizador realizado por Leonardo Flores, 2001

5.3 Pruebas de comparación entre el corpus Gama y el corpus Fraga.

- La mejor de las ventajas es que el corpus Gama esta diseñado y balanceado con fines generales y considero que agrupan todos los posibles sonidos del lenguaje español mexicano.

- La calidad de la voz tanto en tonalidad como en su volumen es mejor por las características de ambos.
- El corpus Gama esta grabado con voz de hombre (timo) y de mujer (lizy) y de esta manera se puede utilizar para la realización de cualquier proyecto relacionado con síntesis de voz.
- Puede ser utilizado para cualquier fin lucrativo o no lucrativo.
- No se había realizado un corpus para el español de México.
- Punto de partida para trabajos posteriores.