Capítulo 3.

Modelos y herramientas utilizadas

Reconocer y entender la voz humana requiere de una cantidad considerable de conocimiento lingüístico: una disposición de las convenciones fonológicas, léxicas, semánticas, gramáticas y pragmáticas que constituyen un lenguaje [Ehsani & Knodt, 99].

3.1 Modelado de la herramienta de Verificación de Pronunciación

Haciendo un análisis sobre las modalidades en el aprendizaje, vistas en el primer capítulo, éste trabajo de tesis se adapta a la modalidad de herramienta: la finalidad es proveer una aplicación que sirva como auxiliar en el aprendizaje del Español hablado en México, en éste caso un herramienta que apoye a los estudiantes en la práctica de su pronunciación.

3.1.1 Formulación de propósitos y objetivos

Los objetivos para este proyecto son:

Investigar cuales son los problemas que tienen los estudiantes al comenzar a adquirir un segundo idioma, en específico, los problemas de pronunciación del Español Mexicano y que parámetros deben seguirse para poder resolver dichos problemas.

Diseñar un método para llevar a cabo el análisis de la pronunciación del usuario mediante la aplicación de un algoritmo basado en tecnología de reconocimiento de voz.

Desarrollar los componentes de software necesarios para la implementación de la herramienta de verificación de pronunciación.

3.1.2 Perfil del usuario

Esta aplicación esta enfocada al apoyo de estudiantes extranjeros (norteamericanos), que desean aprender o practicar el Español hablado en México. Puede servir a estudiantes que comienza con el aprendizaje del Español, debido a que el vocabulario que se tiene no es muy extenso.

3.1.3 Selección del contenido

El vocabulario que se utilizo consta de palabras y frases agrupadas en diferentes tópicos. Los tópicos son:

abecedario

acciones

alimentos

animales

bebidas

colores

dígitos

en la hora de comer

fecha y hora

frases comunes

frutas y verduras

Dicho vocabulario da énfasis a la pronunciación de fonemas del Español Mexicano.

3.1.4 Selección de estrategias de enseñanza y de aprendizaje

Susan Bull menciona cuatro puntos que fueron seleccionados como sugerencias importantes en el aprendizaje de un segundo idioma dentro de un ambiente de aprendizaje [Bull, 94]:

1. El orden de adquisición de las principales reglas del idioma
2. Contar con algunas estrategias para el aprendizaje de idiomas
3. Transferencia del lenguaje
4. Concientización del lenguaje.

Las estrategias de aprendizaje de un idioma han sido identificadas como factores significantes en SLA; una filosofía para el estudiante de cómo se aprende el lenguaje puede determinar su acercamiento con el objetivo, lo cual influye en el nivel de logros alcanzados.

Las estrategias consideradas para este prototipo son: estrategias metacognoscitivas, estrategias cognoscitivas y estrategias socio-afectivas. La aplicación práctica de estas categorías ha sido explorada por Oxford [Oxford, 90], y nos muestra algunos ejemplos de estas tres agrupaciones de estrategias:

1. Estrategias Metacognoscitivas : Viendo las ideas y conceptos principales, planeando partes y secuencias, decidiendo por adelantado a lo que se prestará la atención, cada persona controla su desarrollo y hace evaluaciones sobre lo que ha aprendido.

2. Estrategias Cognoscitivas: *Resourcing*, agrupando y clasificando la información, tomando nota, resumiendo, deducción e inducción, elaboración, transferencia e inferencia, substituciones y traducciones.

3. Estrategias Socio-afectivas: preguntar para aclarar, mostrar donde esta el error para la aclarar, cooperación.

3.1.5 Diseño de la herramienta

Para llegar a este punto, debemos hacer la planeación del proyecto y el análisis de requerimientos como el que ya se ha hecho y hemos descrito en los puntos anteriores. La interfaz de interacción con el estudiante es una parte importante en el desarrollo de nuestro prototipo debido a que es con ésta que los estudiante podrán hacer uso de la herramienta para verificar su pronunciación, los módulos con los cuales interactua se observan en la figura 3.1.

Figura 3.1 Módulos con los que interactua la interfaz para la verificación de pronunciación

La conexión a la base de datos es un módulo mediante el cual se obtiene el vocabulario y otros datos, el módulo de grabación de voz realiza el proceso de grabado de voz, para reconocer lo que se grabó utilizamos el módulo de Reconocimiento de Voz y el proceso de verificación marca los errores que el estudiante tenga al pronunciar una palabra o frase. Estos módulos son explicados a detalle en el siguiente capítulo.

El modelo general, Figura 3.2, nos plantea el diseño de los procesos que deben llevarse a cabo en la verificación de pronunciación. Un locutor produce habla, una palabra o frase. La señal de la voz pasa por un reconocedor de voz para identificar lo que dijo el locutor. El resultado pasa por un módulo de análisis, el cual está basado en la gramática, vocabulario y modelo fonético de una base de datos de voz y finalmente se aplica un algoritmo para obtener los errores en la pronunciación que el locutor haya tenido.

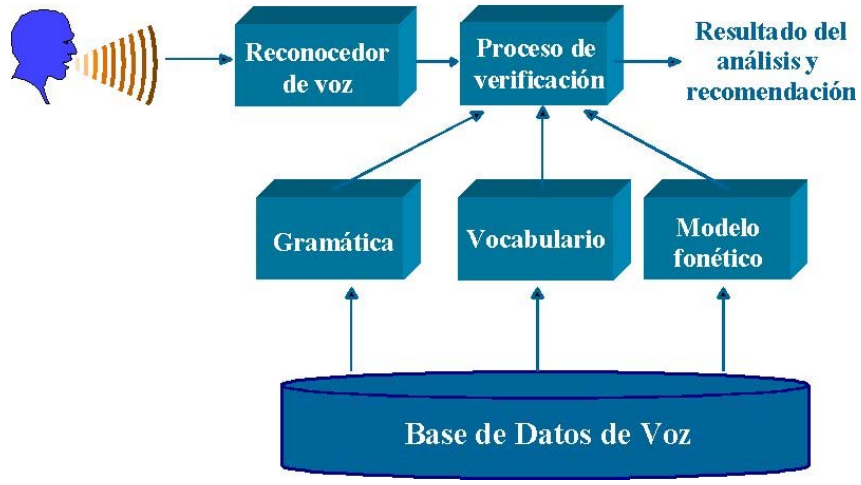

Figura 3.2 Modelo general de la herramienta de verificación de pronunciación

En base a este modelo general, se llevó a cabo el diseño e implementación de cada componente, mismo que se explica con mayor detalle en el siguiente capítulo.

En cuanto al diseño de la Interfaz Persona-Maquina se toman en cuenta los siguientes factores humanos [Gómez, 99]:

Percepción visual. El humano recibe información, la almacena y la procesa. En el uso de interfaces intervienen principalmente los sentidos visual, táctil y auditivo.

Psicología de lectura. La comunicación visual es un punto clave en un sistema amigable. La mayor parte de la interacción se lleva a cabo por medio de texto que el usuario lee, por lo que la lectura es muy importante.

Memoria humana. El ser humano cuenta con memoria a largo plazo y a corto plazo.

Razonamiento inductivo y deductivo. La mayoria de las personas no aplican razonamiento inductivo o deductivo para tomar decisiones, si no que utilizan "heurísticas" para resolver el problema.

Comportamiento del usuario. Es importante considerar las habilidades humanas con las que cuenta el usuario al diseñar sistemas.

Además de los factores humanos debemos considerar también el tiempo de respuesta, el tipo de ayuda y el manejo de errores para el usuario.

Para éste trabajo se considera que el tiempo de respuesta debe ser rápido, la ayuda esta siempre disponible y los mensajes de error se manejan textualmente, indicando al estudiante el error o errores que tuvo.

El diseño general de la interfaz esta basado en el manejo de menús, listas de opciones, cuadros de textos e iconos. En la Figura 3.3 se muestran los elementos principales para la interacción y sus

componentes:

Figura 3.3 Elementos principales en la interacción con el usuario y sus componentes

En la Figura 3.4, diseño general del proceso de verificación, se muestra como se lleva a cabo éste proceso. Tomando la voz, se restringe la gramática de la palabra o frase a evaluar, pasa por el proceso de reconocimiento y se obtiene la secuencia exacta de fonemas. Esta secuencia es comparada con la pronunciación correcta de la palabra o frase y finalmente se muestra le resultado del análisis y la recomendación para el locutor.

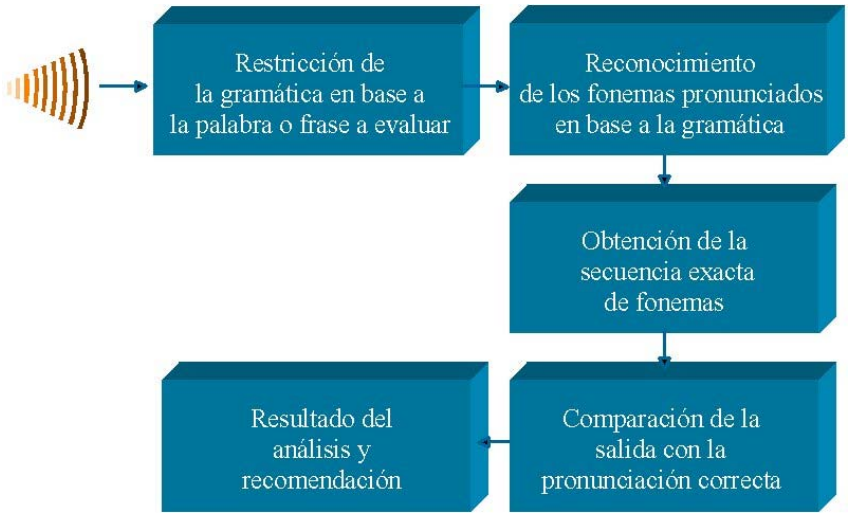

Figura 3.4 Diseño general del proceso de verificación de pronunciación

Para este diseño se tomó en cuenta que el estudiante, en el momento de interactuar con la interfaz, debe elegir un tópico, luego una palabra o frasa (misma con la que desea practicar o verificar su pronunciación), debe grabar su voz y con esto podrá llevar a cabo el proceso de verificación de pronunciación. Si en el proceso de la verificación el estudiante tuvo errores de pronunciación estos son marcados, de otra forma se le da una felicitación. El proceso puede llevarse a cabo cuantas veces sea requierido. En la figura 3.5 se muestra el diagrama de flujo que se sigue.

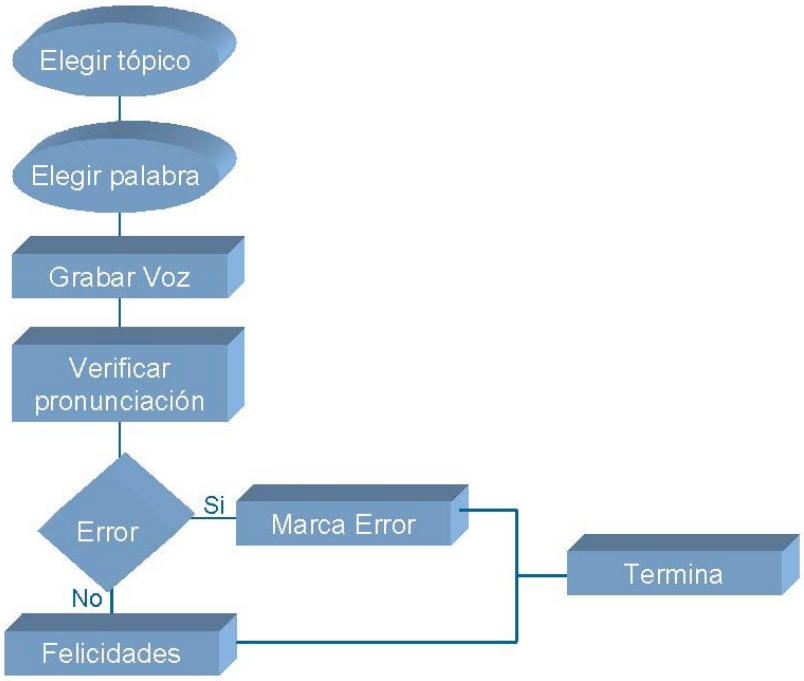

Figura 3.5 Diagrama de flujo

3.1.6 Selección y uso de los medios de aprendizaje

Para el medio en que se desenvuelve la herramienta, se necesita de una computadora. El desarrollo se lleva a cabo utilizando los lenguajes de programación Java, Tcl y la aplicación Jacl. Para el manejo de la Base de Datos se utiliza el manejador de Bases de Datos Access.

3.2 Modelo del Reconocedor

El objetivo de este trabajo es desarrollar un reconocedor fonético de dominio restringido para el habla continua aprovechando las herramientas que proporciona el *CSLU Toolkit* para el entrenamiento de un reconocedor basado en redes neuronales. La figura 3.6, muestra como se lleva a cabo el proceso de reconocimiento de voz siguiendo la metodología del *CSLU Toolkit*.

La primera fase del proceso de reconocimiento de voz consiste en

convertir la señal análoga a una señal digital de manera que pueda ser procesada por la computadora. La segunda fase, el proceso de extracción de características, consiste en calcular un conjunto de parámetros que midan las propiedades esenciales de la señal de voz para obtener una mejor representación de esta. Generalmente, este proceso involucra realizar un análisis espectral de la señal. Al conjunto de parámetros resultante de esta extracción, se le denomina vector de características. En la tercera fase, el objetivo es clasificar cada unidad del habla de acuerdo con el conjunto de unidades utilizadas por el sistema.

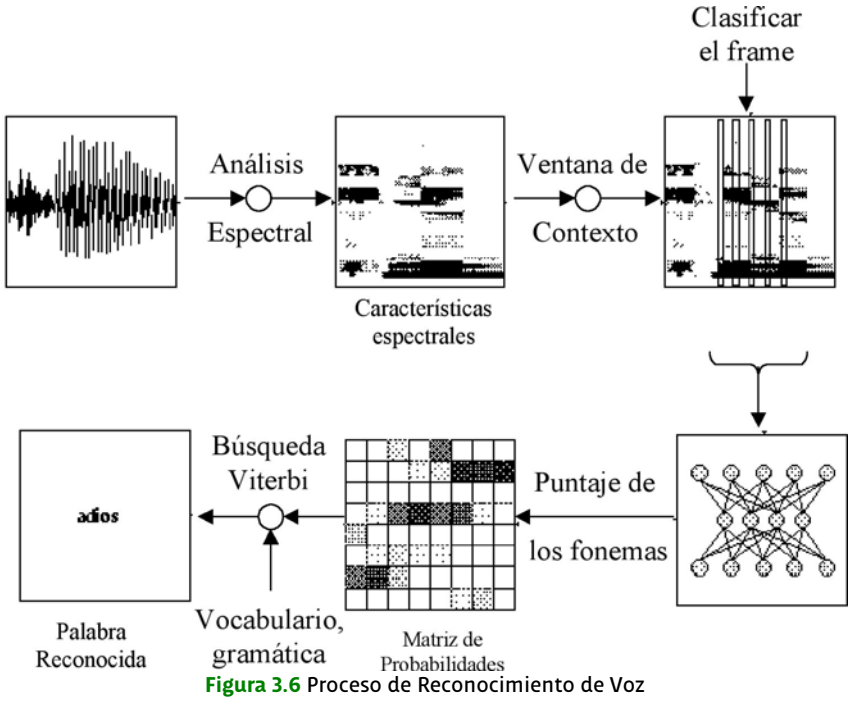

Figura 3.6 Proceso de Reconocimiento de Voz

La entrada del clasificador es un vector de contexto el cual corresponde al de la ventana de contexto. La salida del clasificador proporciona por cada fonema la probabilidad de que dicho fonema es lo que genero el vector de contexto. Con la matriz de probabilidades obtenida procedemos a realizar una búsqueda para encontrar la secuencia de fonemas con mayor probabilidad de reconocimiento. La mayoría de los reconocedores usan el algoritmo de búsqueda *Viterbi*.

El algoritmo *Viterbi* determina la secuencia de palabras mas probable usando información estadística de las duraciones, máximas y mínimas de cada fonema, con el objetivo de restringir las opciones. A medida que los limites de duraciones para cada fonema se afinan, el nivel de reconocimiento se incrementa [Cole et al., 99].

3.2.1. Etapas previas al desarrollo de un reconocedor de voz.

Los sistemas modernos de reconocimiento de voz están basados en datos en vez de los conocimientos de investigadores. Los parámetros del sistema se entrenan automáticamente a partir de muchas muestras

de cada fonema. Por eso, la primera etapa del diseño de un sistema consiste en la colección de una base de datos (un corpus) que contiene grabaciones de personas hablando [Serridge, 98].

Las grabaciones deben estar hechas bajo condiciones similares en el ambiente, el tipo de micrófono, y las características de los locutores. Se necesitan muchos datos para entrenar un sistema de buena calidad.

El siguiente paso, después de la recolección del corpus, es realizar las transcripciones, es decir escribir lo que la persona dijo en cada grabación. Esto es necesario ya que las personas pueden haber dicho diferentes palabras y porque existen varias maneras de decir algunas letras. En la tabla 3.1 podemos ver los ejemplos de transcripciones de las palabras vaca y ayer, deletreadas por diferentes personas.

Tabla 3.1 Ejemplo de transcripciones para las palabras deletreadas:
vaca las palabras a y ayer

Transcripción original	La persona dijo	Nueva Transcripción
VACA	/b/ /e/ /a/ /s/ /e/ /a/	V A C A
VACA	/u/ /b/ /e/ /a/ /s/ /e/ /a/	UVE A C A
AYER	/a/ /jh/ /e/ /e/ /e/ /r/ /e/	A Y E R
AYER	/a/ /i/ /g/ /r/ /i/ /e/ /g/ /a/ /e/ /e/ /r/ /e/	A IGRIEGA E R
AYER	/a/ /i/ /e/ /e/ /r/ /e/	A I E R

Las transcripciones contienen las palabras que dijo la persona, pero sin información que sirva para identificar que parte de la señal de voz corresponde a que letra. Después de verificar que las transcripciones sean correctas, el siguiente paso es etiquetar los datos que se grabaron. Etiquetar los datos significa marcar los límites donde ocurre cada fonema y cada palabra, información que es necesaria para entrenar una red neuronal [Kirschning, 99]

Actualmente existe la herramienta de *Forced Aligment* [Olivier, 99] para crear las etiquetas automáticamente. Esta herramienta es sumamente útil ya que ahorra mucho tiempo en la tarea de etiquetado. Sin embargo, llega a tener algunos errores en determinar las fronteras de algunos fonemas, por lo que se hace necesario una revisión manual posterior para ajustar y revisar las etiquetas hechas. Las etiquetas son a nivel de palabras y de fonemas.

Teniendo etiquetado el corpus, se deben preparar los archivos .vocab y .parts que se necesitan para entrenar el reconocedor. Después se puede entrenar la red neuronal automáticamente usando los programas que forman parte del *CSLU Toolkit*.

El archivo .vocab define dos cosas: la pronunciación de cada palabra en

el vocabulario y la gramática. Consiste de dos partes, en la primera parte se definen las posibles pronunciaciones para cada palabra. El formato es el siguiente:

```
DOS { (dc d | D) o s };
CUATRO { kc k w a tc t r [o]};
```

Del lado izquierdo se escribe la palabra del vocabulario a reconocer, y a su derecha se especifica su pronunciación a nivel fonético.

La segunda parte del archivo describe la gramática que vamos a permitir durante el proceso de reconocimiento. La definición de la gramática tiene dos partes, primero se definen los grupos de palabras que componen el vocabulario, por ejemplo:

```
$digito = CERO | UNO | DOS | TRES | CUATRO | CINCO | SEIS | SIETE | O
```

Además se define la gramática, con la cual el reconocedor buscará la frase más cercana a lo que dijo el locutor. Por ejemplo:

```
$grammar = [ .sil%% ] < $digito [ .pau%% ] > [ .sil%%] ;
```

La sintaxis de estos datos es la que utiliza el *CSLU Toolkit*. Los < > significan que lo que esta adentro puede repetirse una o más veces. Los [ ] indican que lo que esta adentro es opcional. Así que esta gramática se interpreta como: " Un silencio, opcional, y después uno o más ejemplos de dígitos, cada uno con una pausa opcional y finalmente un silencio".

El archivo .parts define el número de partes en las que se va a dividir un fonema y las clases de fonemas que constituyen los contextos. El *CSLU Toolkit* divide fonemas en partes y usa las clases para limitar el número posible de contextos.

Las opciones para dividir un fonema en partes son:

Una (1) parte: el fonema es independiente del contexto

Dos (2) partes: la primera mitad del fonema depende del contexto izquierdo y la segunda mitad del contexto derecho

Tres (3) partes: la primera parte del fonema depende del contexto izquierdo, el centro es independiente del contexto y la ultima parte depende del contexto derecho.

Right-dependent(r): el fonema no se divide en partes, pero si depende del contexto derecho.

3.2.2 Entrenamiento de una red neuronal.

El algoritmo que se utiliza para el entrenamiento de una red neuronal dentro del Toolkit es denominado *back propagation learning* [Rumelhart & McClellar, 86]. El objetivo de este algoritmo es modificar los pesos de la red neuronal por medio de un proceso iterativo, con el

propósito de obtener cada vez valores de salida más similares a los que se desea obtener.

Para esta etapa se requieren los siguientes *scripts* desarrollados por el *CSLU Toolkit*:

Gen_catfiles.tcl. Crea las etiquetas alineadas con el tiempo de las categorías a entrenar. Este proceso se realiza tomando las transcripciones textuales o de las transcripciones fonéticas alineadas con el tiempo. Estas categorías son escritas en archivos con la extensión .cat y se colocan en subdirectorios que reflejan la estructura del directorio del corpus a ser usada.

Genvec.tcl. Se utiliza para crear el vector de características para cada frame que se entrenará. El proceso se lleva a cabo seleccionando los *frames* para entrenamiento, de los archivos creados con gen_catfiles.tcl, y procesa las características de todos los *frames*.

Make_recognizer.tcl. Encuentra los archivos para entrenamiento y determina las categorías que serán clasificadas por el reconocedor.

Revise_desc.tcl. Se utiliza para asegurar que todas las categorías tienen suficientes ejemplos para entrenar. Si alguna categoría no tiene muchos ejemplos, entonces se necesita los archivos .vocab y .parts, o esas categorías necesitarán ser agregadas a categorías con mas ejemplos. También se utiliza para establecer los limites de duración mínimos y máximos para cada categoría. Se agregan los limites de duraciones a los archivos .desc y .olddesc.

Train_and_test_nets.tcl. Entrena la red neuronal en el archivo vector name.train.vec. Este programa crea un archivo de pesos en cada iteración.

Como se mencionó, el proceso de entrenamiento es iterativo, es decir la red será entrenada cierto número de veces el cual puede variar. Cada iteración es guardada en el archivo *nnet*. Se ha observado que después de la treintava iteración la curva de reconocimiento tiende a bajar.

Terminando el proceso de entrenamiento, se debe determinar cuál fue la red que obtuvo el desempeño más alto. Esto se hace evaluando el nivel de error alcanzado en cada iteración utilizando los vectores de características generados para esta etapa. En el *CSLU Toolkit* se calcula el nivel de reconocimiento en términos del grado de error generado [Cole et al., 99]. La formula es:

$$E = \frac{S + I + D}{N} * 100$$

En donde N es el número total de palabras en el conjunto de prueba, S es el número de substituciones, I es el número de inserciones y D el número de eliminaciones. Este método es aplicado a nivel unidades y frases.

Al terminar esta fase se obtiene la red con mejor desempeño.

El desarrollo del reconocedor usando redes neuronales y el *CSLU Toolkit* se resume en los siguientes pasos:

Especificar las categorías fonéticas que la red reconocerá.

Encontrar muchos ejemplos de cada una de esas categorías en la base de datos de voz.

Entrenar una red para reconocer esas categorías.

Evaluar el desempeño de la red utilizando un conjunto de prueba.

3.3 Herramientas utilizadas

Para el desarrollo de éste componente se utilizaron algunas de las herramientas del *CSLU Toolkit*, con las que cuenta el grupo Tlatoa , se usó además el manejador de bases de datos Access, la aplicación Jacl y dos lenguajes de programación: Java y Tcl.

3.3.1 El CSLU Toolkit

Desde 1993, el *Center for Spoken Language Undestanding* (CSLU) se ha enfocado en incorporar tecnología del lenguaje hablado dentro de un ambiente de software portable, flexible y fácil de usar. El resultado de esos esfuerzos es el *CSLU Toolkit*.

El *CSLU Toolkit* es una caja de herramientas para el desarrollo e investigación de sistemas del lenguaje hablado, el cual cuenta con diversas tecnologías, provee una arquitectura modular abierta que soporta el trabajo distribuido o en red cliente/servidor.

Los principales componentes del Toolkit son:

Reconocimiento de voz Soporta varios módulos en los que utiliza redes neuronales artificiales (ANN), modelos ocultos de Markov (HMM''s) y sistemas de segmentos. Incluye vocabulario independiente y específico para los reconocedores. El trabajo en la Universidad de las Américas, en Puebla México, ha producido reconocedores de Español Mexicano para diversos contextos (dígitos, ciudades, palabras deletreadas, etc.) [Serridge et. al., 98], así como un reconocedor de propósito general.

Síntesis de voz Integra el sistema de síntesis texto a voz *Festival,* desarrollado en la Universidad de Edinburgh. El CSLU ha desarrollado un componente "plug-in" de síntesis-forma de onda (waveform syntesis) y seis voces, en las que se incluyen las voces de hombre y mujer en Español Mexicano [Barbosa, 97] e Inglés Americano. *Festival* provee de un ambiente completo para aprendizaje, investigación y desarrollo de habla sintética, incluyendo módulos de normalización transformación de texto en una secuencia o segmentos fonéticos con duraciones apropiadas, asignación de niveles prosódicos y generación de voz

usando difonemas o concatenación.

Animación facial *Baldi*. Es una cara parlante animada en 3D, desarrollada por la Universidad de California, Santa Cruz. Se maneja mediante los componentes de reconocimiento y síntesis de voz, es capaz de sincronizar automáticamente la voz natural, grabada o sintética con los labios, lengua, boca y movimientos faciales. La cara puede ponerse transparente para poder observar los movimientos de los dientes y lengua mientras se produce la voz. La orientación de la cara se puede cambiar para observar desde diferentes perspectivas sus movimientos mientras habla. Además, puede expresar emociones de sorpresa, felicidad, enojo, tristeza, disgusto y miedo mediante expresiones faciales.

RAD Rapid Application Developer es un módulo mediante el cual es posible diseñar una aplicación de voz rápidamente usando una interfaz del tipo "drag & drop". Como parte primordial integra las tecnologías con otros factores útiles como detección de palabras, interrupción del dialogo con la computadora cuando el locutor habla comenzando el proceso de reconocimiento de voz, restauración de diálogo, interfaces de micrófono y teléfono. Este software hace posible que las personas con poco o ningún conocimiento sobre tecnología de voz puedan desarrollar interfaces de voz y aplicaciones en muy poco tiempo.

Herramientas de análisis de señales Provee de un conjunto completo de herramientas para grabar, representar , desplegar y manipular la voz. Representa la señal de voz como un espectograma, y los formantes se pueden desplegar y manipular en ventanas separadas. Las herramientas desplegadas permiten conocer los resultados.

Ambiente de Programación Posee un ambiente de programación muy completo para C y TCL, los cuales incorporan una colección de librerías de software y un conjunto de API''s. Esas librerías sirven como base en la construcción de bloques de programación para el Toolkit. Son portables de una a otra plataforma y proveen la capacidad de manejo de voz, lenguaje, trabajo en red, entradas, salidas y transferencia de datos del Toolkit. Los módulos de procesamiento de lenguaje natural se desarrollaron en Prolog.

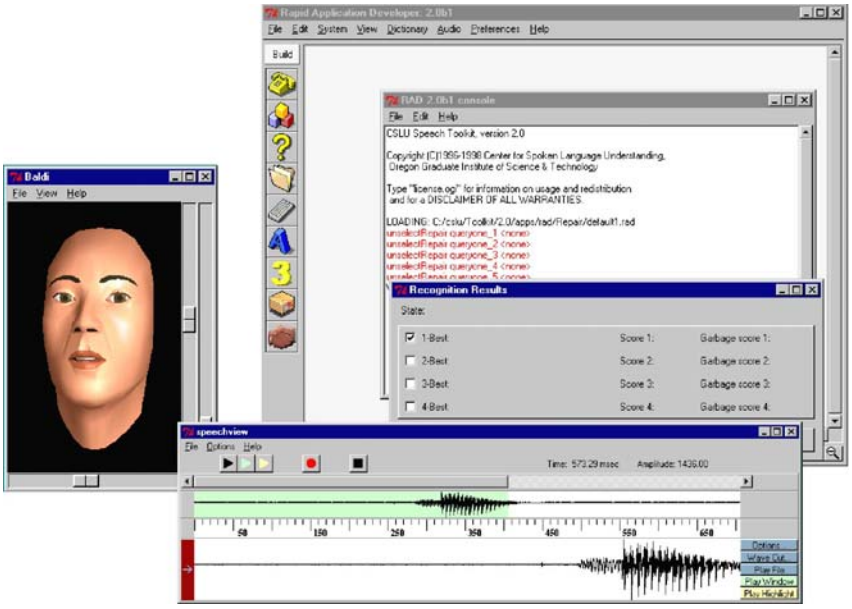

Figura 3.7 Algunas de las principales herramientas con las que cuenta el *CSLU Toolkit*

3.3.2. Base de Datos

El manejador de bases de datos que se utilizo para el presente trabajo fue Microsoft Access 97.

Access, es una base de datos fácil de manejar, ya que permite crear tablas de manera rápida y fácil, de la misma forma se puede agregar, cambiar o borrar información.

Este manejador se utilizo para crear la tabla que almacena la información que usa la herramienta de verificación de pronunciación, en la Figura 3.8 se puede observar la tabla creada. Las columnas son los campos utilizados y los renglones son los registros.

Microsoft Access - [INFORMACION : Tabla]

Archivo Edición Ver Insertar Formato Registros Herramientas Ventana ?

TOPICO	PALABRA_ESP	PALABRA_ING	PRONUNCIACION	TEXTO	FONEMAS
ABECEDARIO	a	a	As the 'a' in Father.	a	a
ABECEDARIO	b	b	As 'be' in Ben.	b e	be
ABECEDARIO	c	c	As 'ce' in cent.	s e	se
ABECEDARIO	ch	ch	As 'cha' in chay.	ch e	che
ABECEDARIO	d	d	As 'de' in Demi.	d e	de
ABECEDARIO	e	e	As 'e' in ten or dress.	e	e
ABECEDARIO	f	f	As ef-fay.	e f e	efe
ABECEDARIO	g	g	As 'he' in hello	g e	ge
ABECEDARIO	h	h	A like the 'a' in Father and 'cha' like in chay.	a ch e	ache
ABECEDARIO	i	i	Always pronounced ee, as in feet, wheel or bee.	i	i
ABECEDARIO	j	j	Like english 'hot' and A 'a' in Father.	x o t a	jota
ABECEDARIO	k	k	As 'ca' in can	k a	ka
ABECEDARIO	l	l	As El-ay.	e l e	ele
ABECEDARIO	ll	ll	E as in elephant, LLE as ye in yes.	e ll e	elle
ABECEDARIO	m	m	E as in elephant, M like an english m and another E	e m e	eme
ABECEDARIO	n	n	E as in elephant, N like an english n and another E	e n e	ene
ABECEDARIO	ñ	ny	E as in elephant, Ñ like the n in onion and another E	e nj e	eñe
ABECEDARIO	o	o	Always pronounces as a short o, as in doctor, but v	o	o
ABECEDARIO	p	p	Like 'pe' in person	p e	pe

Registro: 1 de 344

Vista Hoja de datos NUM

Figura 3.8 Tabla de datos en Access.

Para poder accesar a los datos de la base de datos se utilizó JDBC de Java. JDBC fue diseñado para poder manejar la información de manera

fácil. Esto significa que el API JDBC hace muy sencillas las tareas diarias de una base de datos, ya que se puede hacer uso de una instrucción tan simple como SELECT. Todos los querys, se generan en SQL, un ejemplo:

```
SELECT * FROM INFORMACION WHERE TOPICO="ABECEDARIO";
```

Aquí se selecciona de la tabla "INFORMACION" todos los registros, donde el tópico sea igual a "ABECEDARIO".

3.3.3. Lenguajes de Programación

Se utilizaron dos lenguajes de programación como base, Tcl y Java, y una aplicación de Java llamada Jacl.

a. **Tcl**
   Tcl se utilizo para desarrollar los scripts que interactúan con el reconocedor debido a que los componentes del *CSLU Toolkit* fueron hechos en Tcl y C. La versión de Tcl con que se trabajo fue la 8.0.

   El Tcl es un "*scripting language*" que se corre con un interprete en lugar de un compilador [Welch, 98].

b. **Java**
   Java se utilizó para desarrollar la interfaz con el usuario, debido a que éste lenguaje de programación es sencillo y permite desarrollar objetos gráficos fácilmente. Se usó también para poder interactuar con la base de datos. La versión utilizada fue la 1.3 Beta ya que es en esta versión donde puede llevarse a cabo grabación de voz .

   El lenguaje de programación Java, fue diseñado por la compañía *Sun Microsystems Inc*, con el propósito de crear un lenguaje que pudiera funcionar en redes de computadoras formadas por más de un tipo de computadora, ya sean PC, MAC's, estaciones de trabajo, etc., y que fuera independiente del sistema operativo que posea la computadora. Esto significa que un programa de Java puede ejecutarse en cualquier máquina o plataforma. [Ayala, 98]

c. **Jacl**
   Para poder interactuar con los dos lenguajes de programación se utilizó una aplicación llamada Jacl, mediante la cual se pueden invocar scripts de Tcl y clases hechas en Java. La versión utilizada fue la 1.1.1.

   Jacl es una aplicación de Tcl escrita 100 % en Java. No es una extensión de Tcl porque no agrega nuevos comandos al intérprete del Tcl. Es simplemente una reescritura del intérprete del Tcl usando código de fuente de Java en vez de código de fuente de C. Usando Jacl, los programadores de Tcl pueden definir variables y procedimientos así como interactuar con los objetos de Java. Esta interacción se hace posible por medio del Java package, que es un

conjunto de los comandos de Tcl compartidos por Jacl y de la mezcla del Tcl que proporcionan acceso al Java reflection API. Otra manera de ver Jacl es en términos de plataformas. El Tcl " clásico " es altamente portable; las versiones estándares están disponibles para Unix, Windows, y el OS de la Macintosh, junto con otros sistemas operativos, incluyendo OpenVMS, OS/400, y VxWorks. Jacl es simplemente una aplicación que extiende esta lista a las máquinas virtuales de Java (JVMs).

Conclusiones

En éste capítulo se hizo un análisis sobre los modelos en los que se basa la herramienta para la verificación de pronunciación. Así también se hizo una descripción breve de las herramientas que se utilizaron para desarrollar el prototipo. En el siguiente capítulo se expone la forma de resolver el problema basado en los modelos citados en éste capítulo.

Aguas García, N. 1999. Verificación de Pronunciación Basada en Tecnología de Reconocimiento de Voz para un Ambiente de Aprendizaje. Tesis Licenciatura. Ingeniería en Sistemas Computacionales. Departamento de Ingeniería en Sistemas Computacionales, Escuela de Ingeniería, Universidad de las Américas-Puebla. Diciembre.