Apéndice D.

Código Fuente

Script de verificación

```
#============================================================================


  # Nancy Aguas Garcia =
  # Script: verifica.tcl =
  # Dado un archivo wav, realiza forced aligment sobre este y da la salida  a =
  # nivel fonetico. =
  # =
  # Forma de usar: =
  # Debes de tener una red, tu archivo vocab, tu archivo olddesc, =
  # el texto con que quieres forzar tu wav, el wav y el nombre =
  # de tu archivo de salida =
  # =
  # Ejemplo de corrida: =
  # tcl verifica.tcl "u n o" uno.wav salida.out =
  # =
  # Fecha de ultima modificacion: =
  # 11/10/99 =
  #============================================================================
  #!/bin/sh
  # -*- TCL -*- \
  exec cslush-tcl "$0" ${1+"$@"}
  #Llamamos a las librerias o paquetes necesarios
  foreach package {Context Encode Feature Garbage Mx Nnrun Prep
  Rtcl Search TrainLibrary Tree Wave Word} {
  package require $package
  }
  # si no pones los argumentos entonces te muestra un mensaje como debe de
  ser
  if {$argc != 3} {
  puts "Argumentos: verifica.tcl \"texto\" <archivo.wav>
  <archivo.out>"
  puts "Escribiste: $argv "
  exit -1
  }
  # asignamos el nombre de cada argumento de acuerdo al index de lista
  set a_texto [lindex $argv 0]
  set a_wav [lindex $argv 1]
  set a_salida [lindex $argv 2]
  #Definimos la red, vocab y desc que vamos a usar
  set a_red "nnet.30"
  set a_vocab "vocabulario.vocab"
  set a_desc "archivo.olddesc"
  # la salida a nivel fonetico
  set nivel "p"
  # lee el codigo donde estan las definiciones de nuestro reconocedor
  set fid [open $a_desc]
  set desc ""
  while {[gets $fid line] >= 0} {
  if {[string first "/*" $line] < 0} {
  lappend desc $line
  }
  }
  # Concatena la lista de elementos de desc
  set desc [join $desc]
  close $fid
  # solo reconocera lo que esta en el desc
  set recognizer [recognizer describe $desc]
  # vuelve a leer el a_desc para verificar que las especificaciones
  # del codigo del usuario se encuentren en las dadas
  read_desc $a_desc map contexts define_commands sampling_rate
  if {![catch {set fob [obfile open $a_red]} msg]} {
  if {[catch {set nnet [obfile read $fob __nnet__]}]} {
  set nnet [nnet optload $a_red]
  }
  obfile close $fob
  } else {
  set nnet [nnet optload $a_red]
  }
  # definimos nuestras palabras y gramatica leyendo el archivo vocab
  readVocab $a_vocab vocab pronun wordmodel phones grammar_list
```

```
set special "{forced_alignment_separator {.pau \[.garbage\] .pau}}"
set wordlist [concat [word create $wordmodel] [word create $special]]
set expanded [word context $wordlist $recognizer]
# definimos la gramatica
set gram_string "\[forced_alignment_separator\]"
foreach word $a_texto {
# Si el texto no esta en el vocabulario, muestra un mensaje
if {[lsearch $vocab $word] < 0} {
puts "Error en Forced-Alignment: la palabra $word no esta en el vocabulario"
exit -1
}
# si no creo la gramatica de acuerdo al texto
lappend gram_string $word
lappend gram_string "\[forced_alignment_separator\]"
}
set gram_string [join $gram_string " "]
set gram_string [format "(%s);" $gram_string]
set G "\{ grammar \{ \$grammar = $gram_string\}\}"
set grammar [grammar build $G]
# construimos el estado de la gramatica que direccione la busqueda viterbi
set search [search create $grammar $expanded $recognizer \
-pruning 1.0 -phonemebt]
# hace un backtrace stuff
set btheap [backtrace create 500000 500000]
backtrace reset $btheap
# nuke sobre algunos stuff que no se necesitan
foreach word $wordlist {
nuke $word
}
# lee el archivo wave y obtiene las salidas de la red
set wave [wave read $a_wav]
# crea el vector de caracteristicas en base al archivo .wav leído
get_features $wave feat $sampling_rate
compose_feature_vector $wave $feat $sampling_rate myNetIn
#crea la matriz de probabilidades
set myNetOut [nnet x $nnet $myNetIn]
set myNetOutG [garbage median -N 5 $myNetOut]
nuke $feat $wave
nuke $myNetIn $myNetOut
# hace la actualizacion de la puntuacion de los estados
search update $search $myNetOutG $btheap
# obtenemos nuestros resultados
set answer [search getbest $search $btheap]
set answer [lindex $answer 2]
set answer [mapCdPhn $answer]
# graba el archivo de salida
set salida [open $a_salida w]
foreach seg $answer {
set label [lindex $seg 2]
# mapea <.garbage> a <.pau>
if {[string compare $label "<.garbage>"] == 0} {
set label ".pau"
}
if {[string compare $label ".garbage"] == 0} {
set label ".pau"
}
puts $salida "$label"
}
close $salida
nuke $myNetOutG
nuke $btheap
```

Clase para ventana de grabación.

```
/**
  * Class Name : Grabaciones.java
  * Graba
  *
  * Descripción:
  *
  * Grabaciones.java
  *
  * Construye un panel con botones e invoca a la clase graba
  *
  * Graba
  *
  * Esta clase lleva a cabo la grabación de la voz, y la
  * salva como un archivo wav (actual.wav). Tambien puede
  * tocar la voz grabada al usuario.
```

```
*
* @Author: Nancy Aguas García
*
* @version 2.1 - 06/11/99
*
*/
import java.applet.AudioClip;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.URL;
import javax.media.sound.sampled.*;
import javax.swing.JButton;
import javax.swing.JPanel;
import javax.swing.JFrame;
import javax.swing.ImageIcon;



public class Grabaciones extends JPanel implements ActionListener
{
//componentes
AudioClip wave;
JButton Record,Stop,Play;
URL path;
Graba grabar = new Graba();
public Grabaciones()
{
Botones();
}



//botones para la ventana de grabado
public void Botones()
{
ImageIcon BotonRecord = new ImageIcon("record.gif");
Record = new JButton("Record", BotonRecord);
Record.setMnemonic('R');
ImageIcon BotonStop = new ImageIcon("stop.gif");
Stop = new JButton("Stop", BotonStop);
Stop.setMnemonic('S');
ImageIcon BotonPlay = new ImageIcon("play.gif");
Play = new JButton("Play", BotonPlay);
Play.setMnemonic('P');
Record.addActionListener(this);
Stop.addActionListener(this);
Play.addActionListener(this);
Record.setToolTipText("Record");
Stop.setToolTipText("Stop");
Play.setToolTipText("Play");
//agregamos los componentes
add(Record);
add(Stop);
add(Play);
}
//Tocamos el archivo actual.wav, utilizamos un audioClip
public void Toca()
{
try
{
path = new URL("file:/c:/Programs/temporal/actual.wav");
wave = java.applet.Applet.newAudioClip(path);
wave.play();
}
catch (Exception e)
{
System.out.println("Malformed URL");
}
}
//manejo de los eventos
public void actionPerformed(ActionEvent e)
{
Object object = e.getSource();
JButton button = (JButton) e.getSource();
if (object instanceof JButton)
{
if (button.getText().equals("Record"))
{
grabar.start();
System.out.println("Grabando...");
```

```
}
else if (button.getText().equals("Stop"))
{
grabar.stop();
System.out.println("Termino y guardo");
}
else if (button.getText().equals("Play"))
{
Toca();
System.out.println("Tocando el audio");
}
}
}
public static void main(String[] args)
{
JFrame frame = new JFrame("Recording");
frame.addWindowListener(new WindowAdapter() {
public void windowClosing(WindowEvent e) {
System.exit(0);
}
});
frame.getContentPane().add(new Grabaciones(), BorderLayout.CENTER);
frame.pack();
frame.setVisible(true);
}
}//fin de la clase Grabaciones



/* Clase Graba
*
* En esta clase especificamos el formato y atributos para la grabación
*
*/
class Graba implements Runnable
{
AudioFormat audioFormat;
ByteArrayOutputStream byteArrayOutputStream;
int frameSizeInBytes;
TargetDataLine targetDataLine;
Thread thread;
double duration;
//utilizamos un thread para grabar la voz el tiempo que se desee
public void start()
{
thread = new Thread(this);
thread.setName("Capturing");
thread.start();
}
//paramos el thread, lo que indica que se acabo el tiempo de grabación
public void stop()
{
thread = null;
}
public void run()
{
duration = 0;
System.out.println("Cargando el formato");
Type encoding = AudioFormat.PCM_SIGNED;
float fsampleRate = 8000F;
int nsampleSizeInBits = 16;
int nchannels = 1; //mono
int frameSizeInBytes = nchannels * nsampleSizeInBits / 8;
float fframeRate = fsampleRate;
boolean obigEndian = false; //falso para archivos .wav
//asignamos el formato para el audio
AudioFormat audioFormat = new AudioFormat(encoding,
fsampleRate,nsampleSizeInBits,nchannels,frameSizeInBytes,fframeRate,obigEndian);
//System.out.println("Formato del Audio = " + audioFormat.toString());
//inicializamos la linea para grabación
TargetDataLine targetDataLine = null;
DataLine.Info targetDataLineInfo = new DataLine.Info(TargetDataLine.class,
null, null, new Class[0], audioFormat, 1000); //AudioSystem.NOT_SPECIFIED);
//1024
if (!AudioSystem.isSupportedLine(targetDataLineInfo))
{
System.out.println("Target line " + targetDataLineInfo + " no puede
soportarse.");
System.exit(1);
}
// obtiene y abre el TargetDataLine para captura.
```

```
try
{
targetDataLine = (TargetDataLine) AudioSystem.getLine(targetDataLineInfo);
targetDataLine.open(audioFormat, targetDataLine.getBufferSize());
}
catch (LineUnavailableException ex)
{
System.out.println("Imposible abrir la linea: " + ex);
ex.printStackTrace();
}
ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
int bufferLengthInFrames = targetDataLine.getBufferSize(); // / frameSizeInBytes;
int bufferLengthInBytes = bufferLengthInFrames * frameSizeInBytes;
byte[] data = new byte[bufferLengthInBytes];
int numFramesRead = 0;
//comenzamos a grabar
targetDataLine.start();
//hasta que se de click sobre stop.
while (thread != null)
{
if ((numFramesRead = targetDataLine.read(data, 0, bufferLengthInFrames))
== -1)
{
break;
}
byteArrayOutputStream.write(data, 0, (numFramesRead * frameSizeInBytes));
}
// Se para y cierra la linea.
targetDataLine.stop();
targetDataLine.close();
targetDataLine = null;
// para y cierra el output stream
try
{
byteArrayOutputStream.flush();
byteArrayOutputStream.close();
}
catch (IOException ex)
{
ex.printStackTrace();
}
byte audioBytes[] = byteArrayOutputStream.toByteArray();
ByteArrayInputStream bais = new ByteArrayInputStream(audioBytes);
AudioInputStream audioInputStream = new AudioInputStream(bais, audioFormat,
audioBytes.length / frameSizeInBytes);
long milliseconds = (long)((audioInputStream.getLength() * 1000) /
audioFormat.getFrameRate());
duration = milliseconds / 1000.0;
//guardamos el archivo como actual.wav
try
{
File outfile = new File("actual.wav");
AudioFileFormat[] fileFormats =
AudioSystem.getAudioFileFormats(AudioFileFormat.WAVE, audioInputStream);
if (fileFormats.length == 0)
{
System.out.println("No puede salvar el stream como " + AudioFileFormat.WAVE);
System.exit(1);
}
AudioSystem.write(audioInputStream, fileFormats[0], outfile);
}
catch (Exception ex)
{
ex.printStackTrace();
}
try
{
audioInputStream.reset();
}
catch (Exception ex)
{
ex.printStackTrace();
}
}
} //fin de la clase graba
```

Script de control de GUI's

```
#=====================================================================
```

```
# Nancy Aguas Garcia =
# Script: formats.tcl =
# script que controla los guis de java =
# =
# Forma de usar: =
# se puede llamar desde una clase de con Jacl.
# =
# Fecha de ultima modificacion: =
# 8/10/99 =
#============================================================================
foreach c {RELATIVE REMAINDER NONE BOTH HORIZONTAL VERTICAL CENTER
NORTH NORTHEAST EAST SOUTHEAST SOUTH SOUTHWEST WEST NORTHWEST} {
set gridConst($c) [java::field java.awt.GridBagConstraints $c]
}



proc formato {master slave args} {
global gridConst
if {[expr [llength $args] %2] != 0} {
error "wrong # args"
}
set constr [java::new java.awt.GridBagConstraints]
set insets [java::field $constr insets]
# para el control de ubicación
foreach {opt val} $args {
set x [string toupper $val]
if [info exists gridConst($x)] {
set val $gridConst($x)
}
case $opt {
-x {
java::field $constr gridx $val
}
-y {
java::field $constr gridy $val
}
-width {
java::field $constr gridwidth $val
}
-height {
java::field $constr gridheight $val
}
-fill {
java::field $constr fill $val
}
-ipadx {
java::field $constr ipadx $val
}
-ipadx {
java::field $constr ipady $val
}
-weightx {
java::field $constr weightx $val
}
-weighty {
java::field $constr weighty $val
}
-anchor {
java::field $constr anchor $val
}
-padx {
java::field $insets left $val
java::field $insets right $val
}
-pady {
java::field $insets top $val
java::field $insets bottom $val
}
-insettop {
java::field $insets top $val
}
-insetbottom {
java::field $insets bottom $val
}
-insetleft {
java::field $insets left $val
}
-insetright {
java::field $insets right $val
}
```

```
default {
error "unknown optopn $opt"
}
}
}
set layout [$master getLayout]
if ![java::instanceof $layout java.awt.GridBagLayout] {
set layout [java::new java.awt.GridBagLayout]
$master setLayout $layout
} else {
# If it is a GridBagLayout cast it up from LayoutManager
set layout [java::cast java.awt.GridBagLayout $layout]
}
$layout setConstraints $slave $constr
$master {add java.awt.Component} $slave
}
```

Métodos para el manejo de Información con la base de datos

```
/*
* Conexion a la base de datos cuando se pide un topico.
*/
public void Conexion_topico(String P)
{
String url = "jdbc:odbc:BDSistema";
String qs = "SELECT PALABRA_ESPAÑOL "
+ "FROM INFORMACION WHERE INFORMACION.TOPICO = '"
+ P + "'";
try
{
// Baja el protocolo jdbc-odbc
Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" );
// Intenta la conexión con el driver
Connection con = DriverManager.getConnection( url, "", "" );
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery( qs );
//limpiamos nuestra lista
Palabras_español.clear();
while( rs.next() )
{
String e=rs.getString(1);
//hacemos la lista de palabras
Palabras_español.add(e);
}
rs.close();
}
catch( java.lang.Exception ex )
{
// Descripcion de la excepción.
System.out.println( "** Error en la conexion con la BD. ** " );
ex.printStackTrace();
}
}
```

```
/*
* Conexion a la base de datos si seleccionas una palabra de la lista
*/
public void Conexion_palabra(String w)
{
String url = "jdbc:odbc:BDSistema";
String qs = "SELECT PALABRA_INGLES, PRONUNCIACION "
+ "FROM INFORMACION WHERE INFORMACION.PALABRA_ESPAÑOL = '"
+ w + "'";
try
{
// Baja el protocolo jdbc-odbc
Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" );
// Intenta la conexión con el driver
Connection con = DriverManager.getConnection( url, "", "" );
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery( qs );
while( rs.next() )
{
t = rs.getString(1);
p = rs.getString(2);
}
Campo_Traduccion.setText(t);
```

```
Pantalla_Salida.append(p + "\n");
rs.close();
}
catch( java.lang.Exception ex )
{
// Descripcion de la excepción.
System.out.println( "** Error en la conexion de la BD con algun elemento
gráfico. ** " );
ex.printStackTrace();
}
}



/*
* Conexion a la base de datos para obtener informacion de la palabra
* seleccionada
*/
public void Conexion_texto(String w)
{
String url = "jdbc:odbc:BDSistema";
String qs = "SELECT TEXTO, FONEMAS "
+ "FROM INFORMACION WHERE INFORMACION.PALABRA_ESPAÑOL = '"
+ w + "'";
try
{
// Baja el protocolo jdbc-odbc
Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" );
// Intenta la conexión con el driver
Connection con = DriverManager.getConnection( url, "", "" );
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery( qs );
while( rs.next() )
{
text = rs.getString(1);
fonemas = rs.getString(2);
}
rs.close();
}
catch( java.lang.Exception ex )
{
// Descripcion de la excepción.
System.out.println( "** Error en la conexion de la BD con algun elemento
gráfico. ** " );
ex.printStackTrace();
}
}
```


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.