## Apéndice B

```
# ------------------------------------------------------------------
# convierte.tcl -- crea archivos .fone a partir de archivos .wrd & .phn
#
# command-line arguments:
#   "-files fname" -- file listing the .wrd files to act upon
#   "-noskip"      -- flag, if present, overwrites already-present files
#
# example:
#   convierte.tcl -files digits.train.files
#   (creates a phone file  for each file especified)
# ------------------------------------------------------------------

# load CSLU packages
  package require Label
  package require TrainLibrary 1.0

# load my own packages
  package require CLI
  package require UTIL
  package require FIO
  package require RECOG

#-------------------------------------------------------------------
---------------------------
  proc create_unique_vocab_map {vocab_list pronun_list} {

    assert [expr "[llength $vocab_list] == [llength $pronun_list]"]
\
         "conversion error: mismatch between num words and num
pronunciations"
    set i 0
    foreach word $vocab_list {
   set pronun [lindex $pronun_list $i]
   regsub -all {\( *([^\|]+).*\|*\)} $pronun {\1} first_pronun
   regsub -all {  +} $first_pronun { } first_pronun
   set vocab($word) [string trim $first_pronun]
   incr i
    }
    return [array get vocab]
  }

#-------------------------------------------------------------------
# parse the command-line arguments into key/value pairs

  array set args [CLI_parse $argv "-files -phones" "-noskip"]

  set files_fname $args(-files)
  set fon_fname   $args(-phones)

  if {[string compare $fon_fname ""] != 0} {
```

```
     readVocab $fon_fname vocab_list pronun_list n_used_wd_mod
fon_list grammar
     array set fon [create_unique_vocab_map $vocab_list
$pronun_list]
  }

  set fons [array names fon]
  set files_fid [open $files_fname "r"]

  while {[gets $files_fid fileset] > 0} {
    set phn_fname [lindex $fileset 1]
    set wrd_fname [lindex $fileset 3]

    regsub ".wrd" $wrd_fname ".fone" out_fname
    puts "salida:  $out_fname"
    puts "Processing file $wrd_fname..."

    # read the .wrd file
    readPhn $wrd_fname msec wrd_lola

    # read the .phn file
    readPhn $phn_fname msec phn_lola
    set silencio { \.PAU \.BN \.BR \.LN }

    set fonemas {}
    set frase {}
    foreach item $wrd_lola {
   set wrd_start_time [lindex $item 0]
   set wrd_end_time   [lindex $item 1]
   set word           [lindex $item 2]
   set fonemas {}
        set fin 0

        if {[lsearch $silencio $word] != -1 } {
           continue
        } else {
       foreach item $phn_lola {

         set phn_start_time [lindex $item 0]
         set phn_end_time   [lindex $item 1]
         set phn            [lindex $item 2]

         if {$fin == 1} {
             lappend fonemas $phn
         }
         if {[string compare $wrd_start_time $phn_start_time]== 0} {
             lappend fonemas $phn
             set fin 1
         }
         if {[string compare $wrd_end_time $phn_end_time] == 0} {
             break
         }
       }

       lappend frase $fonemas
        }
```

```
    }
    # write out the txt file
       puts "frase: $frase"
  set cadena [join $frase]
       set final {}
       foreach elemento $cadena {
     set pron $fon($elemento)
     if {[string compare $pron ""] != 0} {
         lappend final $pron
     }
       }
  puts [join $final]
  set frase_final [join $final]
  set out_fid [open $out_fname "w"]
  puts $out_fid $frase_final

    }

    close $files_fid



# ------------------------------------------------------------------
# texto2fonemas.tcl – crea archivos .fone a partir de archivos .txt
#
# command-line arguments:
#   "-files fname" -- file listing the .txt files to act upon
#   "-rules fname" -- (optional) file listing TTS rules to use
#   "-vocab fname" -- (optional) pronunciations that override the TTS rules
#   "-noskip" -- flag, if present, overwrites already-present files
#
#
# tcl txt2fonemas.tcl -files -rules tts_rules.txt -vocab exceptions.vocab -noskip
#
# ------------------------------------------------------------------

# load my own packages

  package require Label
  package require Wave
  package require Rtcl
  package require TrainLibrary 1.0

  package require UTIL
  package require CLI
  package require FIO
  package require RECOG
  package require CORPUS

#-------------------------------------------------------------------
----------------------------------
  proc create_unique_vocab_map {vocab_list pronun_list} {
```

```
    assert [expr "[llength $vocab_list] == [llength $pronun_list]"]
\
           "txt2wrd error: mismatch between num words and num
pronunciations"

    set i 0
    foreach word $vocab_list {
   set pronun [lindex $pronun_list $i]
   regsub -all {\( *([^\|]+).*\|*\)} $pronun {\1} first_pronun
   regsub -all {  +} $first_pronun { } first_pronun
   set vocab([string toupper $word]) [string trim $first_pronun]
   incr i
    }

    return [array get vocab]
  }

#-----------------------------------------------------------------------
# MAIN
# parse the command-line arguments into key/value pairs

  array set args [CLI_parse $argv "-files" "-rules -vocab" "-
noskip"]

  set rules_fname   $args(-rules)
  set vocab_fname   $args(-vocab)
  set files_fname   $args(-files)


  if {[string compare $vocab_fname ""] != 0} {
      readVocab $vocab_fname vocab_list pronun_list
not_used_word_model phone_list grammar
      array set vocab [create_unique_vocab_map $vocab_list
$pronun_list]
  }

  if {[string compare $rules_fname ""] != 0} {
      set rules [FIO_read_rules $rules_fname]
  }

  assert [expr "[info exists rules] || [info exists vocab]"] \
      "Usage error: must specify either a vocab file or a rules
file, or both."

  set files_fid [open $files_fname "r"]
    while {[gets $files_fid fileset] > 0} {

      set txt_fname [lindex $fileset 3]

      # do nothing if word file already exists and noskip wasn't
specified
      if {!$args(-noskip) && [file exists $txt_fname]} {
   puts "Skipping file $txt_fname..."
   continue
      }
```

```
      puts "Processing file $txt_fname..."

      # read the .txt file
      assert [file exists $txt_fname] "txt2wrd error: unable to open
file $txt_fname"
      set words [string toupper [string trim [FIO_read_txt_file
$txt_fname]]]
      set cadena ""
      foreach word $words {


           regsub -all {\[[A-z]*\]} $word "" word1
           regsub -all {\*} $word1 "" word1
           regsub -all {\[[A-z]*\]} $word "" word2

     if {[info exists vocab($word1)]} {
          set pronunciation $vocab($word1)
        } elseif {[info exists rules]} {
          set pronunciation [RECOG_apply_rules rules $word1]
              } else {
          display_message "txt2wrd.tcl Error: no TTS rules and
$word not found in vocab"
          exit
        }

          set cadena "$cadena $pronunciation"
    set nuevo [format "%s2" $txt_fname]
          set salida [open $nuevo "w"]
          puts $salida $cadena
    close $salida
   }

  puts "$cadena"
}

   close $files_fid




# ------------------------------------------------------------------
# validawrd.tcl – clasifica a un archivo word en habla, silencio y basura
#
# command-line arguments:
#   "-files fname" -- file listing the .wrd files to act upon
#   "-noskip"      -- flag, if present, overwrites already-present files
#
# example:
#   validawrd.tcl -files phonemas.files -noskip
# ------------------------------------------------------------------

# load CSLU packages
  package require Label
  package require TrainLibrary 1.0
```

```
# load my own packages
  package require CLI
  package require UTIL
  package require FIO

proc WriteLola {lola filename resolution} {
  puts "entro"
  set fp [open $filename w]
  puts $fp "MillisecondsPerFrame: $resolution\nEND OF HEADER"
  foreach s $lola {
    puts -nonewline $fp "[expr int([lindex $s 0])] "
    puts -nonewline $fp "[expr int([lindex $s 1])]"
    foreach w [lindex $s 2] {
      puts -nonewline $fp " $w"
    }
    puts $fp ""
  }
  close $fp
}

proc classify {cadena dur} {

set lista_silencio { \.PAU \.BN \.BR \.LN \.UNK }

if { [ lsearch $lista_silencio $cadena ] == -1 } {
   set label "HABLA"
   return $label }
if { $dur >= 200 } {
       set label "SILENCIO"
       return $label
   }
set label "HABLA"
return $label
}

#-----------------------------------------------------------------
# parse the command-line arguments into key/value pairs

  array set args [CLI_parse $argv "-files" "" "-noskip"]

  set files_fname $args(-files)
  set files_fid   [open $files_fname "r"]

  while {[gets $files_fid fileset] > 0} {

     set wrd_fname [lindex $fileset 3]

     # read the .wrd file

     readPhn $wrd_fname msec wrd_lola
     puts "Processing file $wrd_lola..."
     set foo_list {}
     set temp [lindex [lindex $wrd_lola 0 ] 0 ]
     set end [ llength $wrd_lola ]
```

```
     for {set x 0 } {$x < $end } {incr x } {

   set item      [lindex $wrd_lola $x]
   set linferior [lindex $item 0]
       set lsuperior [lindex $item 1]
   set word      [string toupper [lindex $item 2]]
   set duracion  [expr ($lsuperior - $linferior) ]
   set label [classify $word $duracion]
   if {$x < [ expr ($end -1)] } {
    set y              [ expr ( $x + 1) ]
    set next_item  [lindex $wrd_lola $y]
    set lsup       [lindex $next_item 1]
    set linf       [lindex $next_item 0]
    set next_dur   [ expr ( $lsup - $linf ) ]
    set next_label [ classify [lindex $next_item 2] $next_dur]
   }

         # si son diferentes tipos de etiquetas y no es la última

       if { [string compare $label $next_label] != 0  &&  [string
       compare $next_label ""] != 0 }  {
      set new_item "$temp $lsuperior $label"
      puts $new_item
            lappend foo_list $new_item
      set temp [lindex $next_item 0]
    }

         # ultima etiqueta
    if { [string compare $next_label ""] == 0} {
          set new_item  "$temp $lsuperior $label"
          lappend foo_list $new_item
    }

    # limpia
         set next_label ""
     }

     set fnamewrd [file tail $wrd_fname]
     set path [file dirname $wrd_fname ]
     puts $fnamewrd
     set nuevo  [format "%s/N%s" $path $fnamewrd]
     WriteLola $foo_list $nuevo 1.0

}

close $files_fid
```

A continuación se incluye un tipo de  archivo de vocabulario usado por el script convierte.tcl, para generar los archivos .fone

**phones.vocab**

```
.pau          {};
.unk          {};
.br           {};
.bn           {};
```

```
.ln             {};
f               {F_F};
tS              {CH_F};
o_h             {O_F};
a_h             {A_F};
e_h             {E_F};
i_h             {I_F};
i_bn            {I_F};
u_h             {U_F};
u_bn            {U_F};
e_x             {};
e_bn            {E_F};
e_ln            {E_F};
o_x             {};
o_ln            {O_F};
o_bn            {O_F};
e_fp            {E_F};
i_fp            {I_F};
a_fp            {A_F};
a_bn            {A_F};
a_ln            {A_F};
o_fp            {O_F};
u_fp            {U_F};
n_fp            {N_F};
x_fp            {X_F};
dc              {};
bc              {};
gc              {};
kc              {};
pc              {};
tc              {};
tSc             {};
dZc             {};
ia              {I_F A_F};
ie              {I_F E_F};
io              {I_F O_F};
iu              {I_F U_F};
ua              {W_F A_F};
ue              {W_F E_F};
ui              {W_F I_F};
uo              {W_F O_F};
a               {A_F};
b               {B_F};
D               {D_F};
V               {B_F};
e               {E_F};
G               {G_F};
i               {I_F};
x               {X_F};
l               {L_F};
j               {LL_F};
dZ              {LL_F};
m               {M_F};
m_bn            {M_F};
nj              {Ñ_F};
N               {N_F};
```

```
n        {N_F};
o        {O_F};
p        {P_F};
r        {R_F};
rZ       {R_F};
rr       {R_F};
s        {S_F};
u        {U_F};
w        {W_F};
d        {D_F};
k        {K_F};
t        {T_F};
t_bn     {T_F};
t_ln     {T_F};
p        {P_F};
g        {G_F};
s_v      {S_F};
tc_bn    {};
tc_ln    {};
tSc_bn   {};
kc_ln    {};
kc_bn    {};
pc_ln    {};
bc_ln    {};
dc_ln    {};
tSc_ln   {};
gc_ln    {};
s_ln     {S_F};
s_bn     {S_F};
V_bn     {B_F};
bc_bn    {};
gc_bn    {};
pc_bn    {};
dc_bn    {};
m_fp     {M_F};
n_bn     {N_F};
#        {};

# ------------------------------------------------------------------
# wavecut.tcl – corta la señal de voz, extrayendo solo partes de habla
#
# command-line arguments:
#   "-files fname" -- file listing the .wrd files to act upon
#   "-noskip"      -- flag, if present, overwrites already-present files
#
# example:
#   wavecut.tcl -files phonemas.files -noskip
# ------------------------------------------------------------------


# load packages
package require TrainLibrary
package require Hmm
package require Wave
```

```
package require Rtcl
package require Label
package require TrainLibrary 1.0

# load my own packages
package require CLI
package require UTIL
package require FIO

proc WriteLola {lola filename resolution} {
  puts "entro"
  set fp [open $filename w]
  puts $fp "MillisecondsPerFrame: $resolution\nEND OF HEADER"
  foreach s $lola {
    puts -nonewline $fp "[expr int([lindex $s 0])]"
    puts -nonewline $fp "[expr int([lindex $s 1])]"
    foreach w [lindex $s 2] {
      puts -nonewline $fp " $w"
    }
    puts $fp ""
  }
  close $fp
}

# parse the command-line arguments into key/value pairs
array set args [CLI_parse $argv "-files"]

set files_fname $args(-files)
set num_modified 0

set files_fid [open $files_fname "r"]
while {[gets $files_fid fileset] > 0} {
    set wav_fname [lindex $fileset 0]
    set wrd_fname [lindex $fileset 4]

    if {[string compare $wrd_fname ""] == 0 || \
      [string compare $wrd_fname "NULL"] == 0} {
   puts "Skipping file $wav_fname -- no wrd fname given"
   continue
    }

    assert [file exists $wav_fname] \
      "wavecut.tcl error: file $wav_fname does not exist"
    set w [wave read $wav_fname]

    readPhn $wrd_fname msec wrd_labels
    set count 1
    set num_wrds [llength $wrd_labels]
    set modified 0

  puts $wav_fname
    foreach wrd_label $wrd_labels {

        set wrd       [string toupper [lindex $wrd_label 2]]
   set empieza   [lindex $wrd_label 0]
   set termina   [lindex $wrd_label 1]
```

```
   set dur [expr "$termina - $empieza"]
   # puts $wrd
   # puts "-- $num_wrds $count"
   if {$count == 1 && [string compare $wrd "HABLA"] == 0 } {
       set modified 1
       set n_termina [expr $termina + 100]
       set temp [wave chop $w -end $n_termina]

   } elseif {$count == $num_wrds && [string compare $wrd "HABLA"] ==
0 } {

       set modified 1
       set n_empieza [expr $empieza - 100]
       puts "$n_empieza"
       set temp [wave chop $w -begin $n_empieza]

   } elseif {[string compare $wrd "HABLA"] == 0 } {
       # otra etiqueta de habla en cualquier parte
       set modified 1
       set n_termina [expr $termina + 100]
       set n_empieza [expr $empieza - 100]
       set temp [wave chop $w -begin $n_empieza -end $n_termina]
   }

   incr count
   if {$modified} {
       incr num_modified
       file copy -force $wav_fname $wav_fname.old
       set new_wav_fname [ format "story_u.s%s.wav" $num_modified ]
       wave write $temp
"d:/data/corpora/stories_cortos/speechfiles/$new_wav_fname"
       nuke $temp
       set n_termina 0
       set n_empieza 0
       set modified 0
   }
     }
     nuke $w
}
close $files_fid
```