UPUTE

Kako bi se nastavio rad na projektu potrebno je preuzeti nekoliko aplikacija. U nastavku bit će navedene potrebne aplikacije sa hipervezama za njihovo preuzimanje te ukratko opisan postupak snimanje prognoza i automatske transkripcije.

Audacity

Audio editor pomoću kojega su snimljene vremenske prognoze te ujedno izrezane na manje fraze odnosno rečenice. Potrebne konfiguracije su frekvencija od 16 000 Hz, mono kanal te odgovarajuća nomenklatura datoteka. Datoteke se imenuju u formatu spol govornika (sm/sz), datum prognoze (ddmmgg), redni broj prognoze toga dana (12 ili 07) te redni broj fraze unutar prognoze. Može se preuzeti sa https://www.audacityteam.org/ te je instalacija vrlo intuitivna.

Nakon instalacije koristeći Audacity može se snimiti još snimki vremenskih prognoza i tako nastaviti rad na projektu, a ako se vremenske prognoze snime s ispravnim konfiguracijama moguće je proširiti gramatiku na jednostavan način. U već izrađenu datoteku prompts.txt dodaju se nove rečenice u formatu /*nazivdatoteke rečenica te se primjenom odgovarajuće Julia skripte stvara lista riječi, a pomoću HTK naredbe HDMan stvara se novi rječnik. U nastavku biti će navedene sve Julia skripte i HTK naredbe koji su korištene.

HTK Toolkit

Radi se o ključnom alatu za izradu i obradu skrivenih Markovljevih modela (HMM) te se primarno koristi za raspoznavanje govora. Za razliku od Audacity-a potrebno je najprije registrirati se na https://htk.eng.cam.ac.uk/register.shtml kako bi se mogao preuzeti alat, a za ovaj projekt važni su sam alat koji se preuzima sa https://htk.eng.cam.ac.uk/ftp/software/htk-3.3-windows-binary.zip te htk-book koji predstavlja dokumentaciju koja se može preuzeti sa https://htk.eng.cam.ac.uk/ftp/software/htkbook_pdf.zip. Nakon preuzimanje toolkit-a potrebno je koristeći Windows Explorer otvoriti zip datoteku koja sadrži sve potrebne alate. Detaljnije o preuzimanju i instalaciji na http://www.voxforge.org/home/dev/acousticmodels/windows/create/htkjulius/tutorial/download.

Od alata važnih za ovaj projekt ističu se HVite, HERest, HHEd, HDMan te HLED.

HVite – koristi se za izradu aligned.mlf datoteke koja sadrži label datoteke ili ispis tih label datoteke istovremeno. Naredba korištena za generiranje .lab datoteka i aligned.mlf datoteke: HVite -A -D -T 1 -l * -o SW -b SENT-END -C config -H hmm15/macros -H hmm15/hmmdefs -i aligned.mlf -m -y lab -a -I words.mlf -S train.scp dict monophones1 > HVite_log

HERest – koristi se za procjenu parametara HMM. Primjer naredbe: HERest –A –D –T 1 –C config –I phones1.mlf –t 250.0 150.0 3000.0 –S train.scp –H hmm5/macros –H hmm5/hmmdefs –M hmm6 monophones1

HHEd – koristi se za manipulaciju setom HMM. Učitava niz HMM te nad njima provodi određene operacije. Primjer naredbe: HHEd –A –D –T 1 –H hmm9/macros –H hmm9/hmmdefs –M hmm10 mktri.hed monophones1

HDMan – koristi se za izradu fonetskog rječnika iz jednog ili više izvora. Primjer naredbe: HDMan –A –D –T 1 –m –w wlist –n monophones1 –i –l dlog dict ../lexicon/VoxForgeDict.txt

HLEd – editor za upravljanje label datotekama. Primjer naredbe: C:>HLEd –A –D –T 1 –l * -d dict –i phones1.mlf mkphones1.led words.mlf

HTK skripte:

global.ded

```
AS sp
RS cmu
MP sil sil sp
```

mkphones0.led

```
EX
IS sil sil
DE sp
```

mkphones1.led

```
EX
IS sil sil
```

mktri.led

```
WB sp
WB sil
TC
```

maketriphones.ded

```
AS sp
MP sil sil sp
TC
```

tree1.hed

```
RO 100 "stats"

TR 0

QS  "R_NonBoundary"           { *+* }
QS  "R_Silence"                       { *+sil }
QS  "R_Stop"               { *+p,*+pd,*+b,*+t,*+td,*+d,*+dd,*+k,*+kd,*+g }
QS  "R_Nasal"                         { *+m,*+n,*+en,*+ng }
QS  "R_Fricative"      { *+s,*+sh,*+z,*+f,*+v,*+ch,*+jh,*+th,*+dh }
QS  "R_Liquid"                        { *+l,*+el,*+r,*+w,*+y,*+hh }
QS  "R_Vowel"                        {
*+eh,*+ih,*+ao,*+aa,*+uw,*+ah,*+ax,*+er,*+ar,*+ir,*+ur,*+ay,*+oy,*+ey,*+iy,*+ow
}
QS  "R_C-Front"                     { *+p,*+pd,*+b,*+m,*+f,*+v,*+w }
QS  "R_C-Central"     {
*+t,*+td,*+d,*+dd,*+en,*+n,*+s,*+z,*+sh,*+th,*+dh,*+l,*+el,*+r }
QS  "R_C-Back"                      { *+sh,*+ch,*+jh,*+y,*+k,*+kd,*+g,*+ng,*+hh }
QS  "R_V-Front"                     { *+iy,*+ih,*+eh }
QS  "R_V-Central"     { *+eh,*+aa,*+er,*+ar,*+ir,*+ur,*+ao }
QS  "R_V-Back"                      { *+uw,*+aa,*+ax,*+uh }
QS  "R_Front"                         {
*+p,*+pd,*+b,*+m,*+f,*+v,*+w,*+iy,*+ih,*+eh }
QS  "R_Central"                      {
*+t,*+td,*+d,*+dd,*+en,*+n,*+s,*+z,*+sh,*+th,*+dh,*+l,*+el,*+r,*+eh,*+aa,*+er,*+
ar,*+ir,*+ur,*+ao }
QS  "R_Back"              {
*+sh,*+ch,*+jh,*+y,*+k,*+kd,*+g,*+ng,*+hh,*+aa,*+uw,*+ax,*+uh }
```

```
QS  "R_Fortis"                        {
*+p,*+pd,*+t,*+td,*+k,*+kd,*+f,*+th,*+s,*+sh,*+ch }
QS  "R_Lenis"              { *+b,*+d,*+dd,*+g,*+v,*+dh,*+z,*+sh,*+jh }
QS  "R_UnFortLenis"             { *+m,*+n,*+en,*+ng,*+hh,*+l,*+el,*+r,*+y,*+w }
QS  "R_Coronal"                     {
*+t,*+td,*+d,*+dd,*+n,*+en,*+th,*+dh,*+s,*+z,*+sh,*+ch,*+jh,*+l,*+el,*+r }
QS  "R_NonCoronal" { *+p,*+pd,*+b,*+m,*+k,*+kd,*+g,*+ng,*+f,*+v,*+hh,*+y,*+w }
QS  "R_Anterior"        {
*+p,*+pd,*+b,*+m,*+t,*+td,*+d,*+dd,*+n,*+en,*+f,*+v,*+th,*+dh,*+s,*+z,*+l,*+el,*
+w }
QS  "R_NonAnterior" { *+k,*+kd,*+g,*+ng,*+sh,*+hh,*+ch,*+jh,*+r,*+y }
QS  "R_Continuent"   {
*+m,*+n,*+en,*+ng,*+f,*+v,*+th,*+dh,*+s,*+z,*+sh,*+hh,*+l,*+el,*+r,*+y,*+w }
QS  "R_NonContinuent"         {
*+p,*+pd,*+b,*+t,*+td,*+d,*+dd,*+k,*+kd,*+g,*+ch,*+jh }
QS  "R_Strident"        { *+s,*+z,*+sh,*+ch,*+jh }
QS  "R_NonStrident"  { *+f,*+v,*+th,*+dh,*+hh }
QS  "R_UnStrident"   {
*+p,*+pd,*+b,*+m,*+t,*+td,*+d,*+dd,*+n,*+en,*+k,*+kd,*+g,*+ng,*+l,*+el,*+r,*+y,*
+w }
QS  "R_Glide"                         { *+hh,*+l,*+el,*+r,*+y,*+w }
QS  "R_Syllabic"          { *+en,*+m,*+l,*+el,*+er,*+ar,*+ir,*+ur }
QS  "R_Unvoiced-Cons"         {
*+p,*+pd,*+t,*+td,*+k,*+kd,*+s,*+sh,*+f,*+th,*+hh,*+ch }
QS  "R_Voiced-Cons" {
*+jh,*+b,*+d,*+dd,*+dh,*+g,*+y,*+l,*+el,*+m,*+n,*+en,*+ng,*+r,*+v,*+w,*+z }
QS  "R_Unvoiced-All"            {
*+p,*+pd,*+t,*+td,*+k,*+kd,*+s,*+sh,*+f,*+th,*+hh,*+ch,*+sil }
QS  "R_Long"              { *+iy,*+aa,*+ow,*+ao,*+uw,*+en,*+m,*+l,*+el }
QS  "R_Short"                         {
*+eh,*+ey,*+aa,*+ih,*+ay,*+oy,*+ah,*+ax,*+uh }
QS  "R_Dipthong"      {
*+ey,*+ay,*+oy,*+aa,*+er,*+ar,*+ir,*+ur,*+en,*+m,*+l,*+el }
QS  "R_Front-Start"   { *+ey,*+aa,*+er,*+ar,*+ir,*+ur }
QS  "R_Fronting"       { *+ay,*+ey,*+oy }
QS  "R_High"              { *+ih,*+uw,*+aa,*+ax,*+iy }
QS  "R_Medium"                     {
*+ey,*+er,*+ar,*+ir,*+ur,*+aa,*+ax,*+eh,*+en,*+m,*+l,*+el }
QS  "R_Low"               { *+eh,*+ay,*+aa,*+aw,*+ao,*+oy }
QS  "R_Rounded"                    { *+ao,*+uw,*+aa,*+ax,*+oy,*+w }
QS  "R_Unrounded"   {
*+eh,*+ih,*+aa,*+er,*+ar,*+ir,*+ur,*+ay,*+ey,*+iy,*+aw,*+ah,*+ax,*+en,*+m,*+hh,*
+l,*+el,*+r,*+y }
QS  "R_NonAffricate"            { *+s,*+sh,*+z,*+f,*+v,*+th,*+dh }
QS  "R_Affricate"       { *+ch,*+jh }
QS  "R_IVowel"                      { *+ih,*+iy }
QS  "R_EVowel"                     { *+eh,*+ey }
QS  "R_AVowel"                     { *+eh,*+aa,*+er,*+ar,*+ir,*+ur,*+ay,*+aw }
QS  "R_OVowel"                     { *+ao,*+oy,*+aa }
QS  "R_UVowel"                     { *+aa,*+ax,*+en,*+m,*+l,*+el,*+uw }
QS  "R_Voiced-Stop"  { *+b,*+d,*+dd,*+g }
QS  "R_Unvoiced-Stop"         { *+p,*+pd,*+t,*+td,*+k,*+kd }
QS  "R_Front-Stop"   { *+p,*+pd,*+b }
QS  "R_Central-Stop"            { *+t,*+td,*+d,*+dd }
QS  "R_Back-Stop"     { *+k,*+kd,*+g }
QS  "R_Voiced-Fric"  { *+z,*+sh,*+dh,*+ch,*+v }
QS  "R_Unvoiced-Fric"           { *+s,*+sh,*+th,*+f,*+ch }
QS  "R_Front-Fric"    { *+f,*+v }
QS  "R_Central-Fric" { *+s,*+z,*+th,*+dh }
QS  "R_Back-Fric"     { *+sh,*+ch,*+jh }
QS  "R_aa"                 { *+aa }
QS  "R_ae"                 { *+ae }
QS  "R_ah"                 { *+ah }
QS  "R_ao"                 { *+ao }
QS  "R_aw"                { *+aw }
QS  "R_ax"                 { *+ax }
QS  "R_ay"                 { *+ay }
QS  "R_b"                   { *+b }
QS  "R_ch"                 { *+ch }
QS  "R_d"                   { *+d }
QS  "R_dd"                 { *+dd }
QS  "R_dh"                 { *+dh }
QS  "R_dx"                 { *+dx }
QS  "R_eh"                 { *+eh }
QS  "R_el"                  { *+el }
QS  "R_en"                 { *+en }
QS  "R_er"                  { *+er }
QS  "R_ar"                  { *+ar }
QS  "R_ir"                  { *+ir }
QS  "R_ur"                  { *+ur }
QS  "R_ey"                  { *+ey }
QS  "R_f"                    { *+f }
QS  "R_g"                   { *+g }
QS  "R_hh"                 { *+hh }
QS  "R_ih"                  { *+ih }
QS  "R_iy"                  { *+iy }
QS  "R_jh"                  { *+jh }
QS  "R_k"                   { *+k }
QS  "R_kd"                 { *+kd }
QS  "R_l"                    { *+l }
QS  "R_m"                  { *+m }
QS  "R_n"                   { *+n }
QS  "R_ng"                 { *+ng }
QS  "R_ow"                { *+ow }
QS  "R_oy"                 { *+oy }
QS  "R_p"                   { *+p }
QS  "R_pd"                 { *+pd }
QS  "R_r"                    { *+r }
QS  "R_s"                    { *+s }
QS  "R_sh"                  { *+sh }
QS  "R_t"                    { *+t }
QS  "R_td"                  { *+td }
QS  "R_th"                  { *+th }
QS  "R_ts"                  { *+ts }
QS  "R_uh"                 { *+uh }
QS  "R_uw"                { *+uw }
QS  "R_v"                   { *+v }
QS  "R_w"                  { *+w }
QS  "R_y"                   { *+y }
QS  "R_z"                    { *+z }
QS  "L_NonBoundary"          { *-* }
QS  "L_Silence"                    { sil-* }
QS  "L_Stop"              { p-*,pd-*,b-*,t-*,td-*,d-*,dd-*,k-*,kd-*,g-* }
QS  "L_Nasal"                      { m-*,n-*,en-*,ng-* }
QS  "L_Fricative"      { s-*,sh-*,z-*,f-*,v-*,ch-*,jh-*,th-*,dh-* }
QS  "L_Liquid"                     { l-*,el-*,r-*,w-*,y-*,hh-* }
QS  "L_Vowel"                     { eh-*,ih-*,ao-*,aa-*,uw-*,ah-*,ax-*,er-
*,ar-*,ir-*,ur-*,ay-*,oy-*,ey-*,iy-*,ow-* }
```

```
QS  "L_C-Front"                    { p-*,pd-*,b-*,m-*,f-*,v-*,w-* }
QS  "L_C-Central"     { t-*,td-*,d-*,dd-*,en-*,n-*,s-*,z-*,sh-*,th-*,dh-*,l-
*,el-*,r-* }
QS  "L_C-Back"                     { sh-*,ch-*,jh-*,y-*,k-*,kd-*,g-*,ng-*,hh-* }
QS  "L_V-Front"                    { iy-*,ih-*,eh-* }
QS  "L_V-Central"     { eh-*,aa-*,er-*,ar-*,ir-*,ur-*,ao-* }
QS  "L_V-Back"                     { uw-*,aa-*,ax-*,uh-* }
QS  "L_Front"                        { p-*,pd-*,b-*,m-*,f-*,v-*,w-*,iy-*,ih-
*,eh-* }
QS  "L_Central"                     { t-*,td-*,d-*,dd-*,en-*,n-*,s-*,z-*,sh-
*,th-*,dh-*,l-*,el-*,r-*,eh-*,aa-*,er-*,ar-*,ir-*,ur-*,ao-* }
QS  "L_Back"             { sh-*,ch-*,jh-*,y-*,k-*,kd-*,g-*,ng-*,hh-*,aa-*,uw-
*,ax-*,uh-* }
QS  "L_Fortis"                       { p-*,pd-*,t-*,td-*,k-*,kd-*,f-*,th-*,s-
*,sh-*,ch-* }
QS  "L_Lenis"             { b-*,d-*,dd-*,g-*,v-*,dh-*,z-*,sh-*,jh-* }
QS  "L_UnFortLenis"            { m-*,n-*,en-*,ng-*,hh-*,l-*,el-*,r-*,y-*,w-* }
QS  "L_Coronal"                    { t-*,td-*,d-*,dd-*,n-*,en-*,th-*,dh-*,s-*,z-
*,sh-*,ch-*,jh-*,l-*,el-*,r-* }
QS  "L_NonCoronal" { p-*,pd-*,b-*,m-*,k-*,kd-*,g-*,ng-*,f-*,v-*,hh-*,y-*,w-* }
QS  "L_Anterior"         { p-*,pd-*,b-*,m-*,t-*,td-*,d-*,dd-*,n-*,en-*,f-*,v-
*,th-*,dh-*,s-*,z-*,l-*,el-*,w-* }
QS  "L_NonAnterior" { k-*,kd-*,g-*,ng-*,sh-*,hh-*,ch-*,jh-*,r-*,y-* }
QS  "L_Continuent"   { m-*,n-*,en-*,ng-*,f-*,v-*,th-*,dh-*,s-*,z-*,sh-*,hh-*,l-
*,el-*,r-*,y-*,w-* }
QS  "L_NonContinuent"        { p-*,pd-*,b-*,t-*,td-*,d-*,dd-*,k-*,kd-*,g-*,ch-
*,jh-* }
QS  "L_Strident"         { s-*,z-*,sh-*,ch-*,jh-* }
QS  "L_NonStrident"  { f-*,v-*,th-*,dh-*,hh-* }
QS  "L_UnStrident"   { p-*,pd-*,b-*,m-*,t-*,td-*,d-*,dd-*,n-*,en-*,k-*,kd-*,g-
*,ng-*,l-*,el-*,r-*,y-*,w-* }
QS  "L_Glide"                        { hh-*,l-*,el-*,r-*,y-*,w-* }
QS  "L_Syllabic"         { en-*,m-*,l-*,el-*,er-*,ar-*,ir-*,ur-* }
QS  "L_Unvoiced-Cons"        { p-*,pd-*,t-*,td-*,k-*,kd-*,s-*,sh-*,f-*,th-*,hh-
*,ch-* }
QS  "L_Voiced-Cons" { jh-*,b-*,d-*,dd-*,dh-*,g-*,y-*,l-*,el-*,m-*,n-*,en-*,ng-
*,r-*,v-*,w-*,z-* }
QS  "L_Unvoiced-All"            { p-*,pd-*,t-*,td-*,k-*,kd-*,s-*,sh-*,f-*,th-
*,hh-*,ch-*,sil-* }
QS  "L_Long"             { iy-*,aa-*,ow-*,ao-*,uw-*,en-*,m-*,l-*,el-* }
QS  "L_Short"                        { eh-*,ey-*,aa-*,ih-*,ay-*,oy-*,ah-*,ax-
*,uh-* }
QS  "L_Dipthong"      { ey-*,ay-*,oy-*,aa-*,er-*,ar-*,ir-*,ur-*,en-*,m-*,l-*,el-
* }
QS  "L_Front-Start"   { ey-*,aa-*,er-*,ar-*,ir-*,ur-* }
QS  "L_Fronting"       { ay-*,ey-*,oy-* }
QS  "L_High"             { ih-*,uw-*,aa-*,ax-*,iy-* }
QS  "L_Medium"                    { ey-*,er-*,ar-*,ir-*,ur-*,aa-*,ax-*,eh-*,en-
*,m-*,l-*,el-* }
QS  "L_Low"              { eh-*,ay-*,aa-*,aw-*,ao-*,oy-* }
QS  "L_Rounded"                   { ao-*,uw-*,aa-*,ax-*,oy-*,w-* }
QS  "L_Unrounded"   { eh-*,ih-*,aa-*,er-*,ar-*,ir-*,ur-*,ay-*,ey-*,iy-*,aw-*,ah-
*,ax-*,en-*,m-*,hh-*,l-*,el-*,r-*,y-* }
QS  "L_NonAffricate"            { s-*,sh-*,z-*,f-*,v-*,th-*,dh-* }
QS  "L_Affricate"       { ch-*,jh-* }
QS  "L_IVowel"                      { ih-*,iy-* }
QS  "L_EVowel"                     { eh-*,ey-* }
QS  "L_AVowel"                     { eh-*,aa-*,er-*,ar-*,ir-*,ur-*,ay-*,aw-* }
QS  "L_OVowel"                    { ao-*,oy-*,aa-* }
QS  "L_UVowel"                    { aa-*,ax-*,en-*,m-*,l-*,el-*,uw-* }
QS  "L_Voiced-Stop"  { b-*,d-*,dd-*,g-* }
QS  "L_Unvoiced-Stop"         { p-*,pd-*,t-*,td-*,k-*,kd-* }
QS  "L_Front-Stop"   { p-*,pd-*,b-* }
QS  "L_Central-Stop"           { t-*,td-*,d-*,dd-* }
QS  "L_Back-Stop"     { k-*,kd-*,g-* }
QS  "L_Voiced-Fric"  { z-*,sh-*,dh-*,ch-*,v-* }
QS  "L_Unvoiced-Fric"         { s-*,sh-*,th-*,f-*,ch-* }
QS  "L_Front-Fric"    { f-*,v-* }
QS  "L_Central-Fric" { s-*,z-*,th-*,dh-* }
QS  "L_Back-Fric"     { sh-*,ch-*,jh-* }
QS  "L_aa"                { aa-* }
QS  "L_ae"                { ae-* }
QS  "L_ah"                { ah-* }
QS  "L_ao"                { ao-* }
QS  "L_aw"               { aw-* }
QS  "L_ax"                { ax-* }
QS  "L_ay"                { ay-* }
QS  "L_b"                  { b-* }
QS  "L_ch"                { ch-* }
QS  "L_d"                  { d-* }
QS  "L_dd"                { dd-* }
QS  "L_dh"                { dh-* }
QS  "L_dx"                { dx-* }
QS  "L_eh"                { eh-* }
QS  "L_el"                 { el-* }
QS  "L_en"                { en-* }
QS  "L_er"                 { er-* }
QS  "L_ar"                 { ar-* }
QS  "L_ir"                 { ir-* }
QS  "L_ur"                 { ur-* }
QS  "L_ey"                 { ey-* }
QS  "L_f"                   { f-* }
QS  "L_g"                  { g-* }
QS  "L_hh"                { hh-* }
QS  "L_ih"                 { ih-* }
QS  "L_iy"                 { iy-* }
QS  "L_jh"                 { jh-* }
QS  "L_k"                  { k-* }
QS  "L_kd"                { kd-* }
QS  "L_l"                   { l-* }
QS  "L_m"                 { m-* }
QS  "L_n"                  { n-* }
QS  "L_ng"                { ng-* }
QS  "L_ow"               { ow-* }
QS  "L_oy"                { oy-* }
QS  "L_p"                  { p-* }
QS  "L_pd"                { pd-* }
QS  "L_r"                   { r-* }
QS  "L_s"                   { s-* }
QS  "L_sh"                 { sh-* }
QS  "L_t"                   { t-* }
QS  "L_td"                 { td-* }
QS  "L_th"                 { th-* }
QS  "L_ts"                 { ts-* }
QS  "L_uh"                { uh-* }
QS  "L_uw"               { uw-* }
QS  "L_v"                  { v-* }
QS  "L_w"                 { w-* }
QS  "L_y"                     { y-* }
QS  "L_z"                      { z-* }

TR 2
```

Za detaljnije upute i pregled ostalih alata i njihovih uporaba pogledati na http://www.seas.ucla.edu/spapl/weichu/htkbook/

Julia

Julia je skriptni jezik za računalnu komputaciju. Čitav toolkit akustičnog modela koji je izrađen u sklopu projekta pisan je u Julia skriptnom jeziku. U nastavku bit će navedene sve korištene skripte i dane hiperveze gdje se one mogu preuzeti. Sama Julia preuzima se sa https://julialang.org/downloads/ te se otvara zip datoteka u kojoj se nalaze sve potrebne komponente.

Detaljnije o preuzimanju i instalaciji na

http://www.voxforge.org/home/dev/acousticmodels/windows/create/htkjulius/tutorial/download

Skripte:

Mkdfa.jl – izrada gramatike

```
####################################################################################
#
#    Copyright (C) 2015  VoxForge
#
#    This program is free software: you can redistribute it and/or modify
#    it under the terms of the GNU General Public License as published by
#    the Free Software Foundation, either version 3 of the License, or
#    (at your option) any later version.
#
#    This program is distributed in the hope that it will be useful,
#    but WITHOUT ANY WARRANTY; without even the implied warranty of
#    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#    GNU General Public License for more details.
#
#    You should have received a copy of the GNU General Public License
#    along with this program.  If not, see <http://www.gnu.org/licenses/>.
#
####################################################################################
#
# port of Julius perl script: mkdfa.pl
#
####################################################################################
```

```
if VERSION < v"1.0"
   @warn("the VoxForge scripts require version 1.0 and above")
end

function reverse_grammar(rgramfile,gramfile)
  rgramfile_fh=open(rgramfile,"w")

  gramfile_arr=open(readlines, gramfile) # automatically closes file handle
  n=0
  for lineln=gramfile_arr
    if ! occursin(r"^[\n|\r]", lineln)
      line=replace(chomp(lineln), r"#.*" => "") # remove line endings & comments
      (left, right)=split(line,r"\:")
      category_arr=split(right,r"\s")
      reverse_category_arr=reverse(category_arr)

      write(rgramfile_fh, left * ":")
      write(rgramfile_fh, join(reverse_category_arr," ")  )
      if occursin(r"\r$", lineln) # windows line ending
        write(rgramfile_fh, "\n\r")
      else
        write(rgramfile_fh, "\n")

      end
      n=n+1
    end
  end

  close(rgramfile_fh)
  println("$gramfile has $n rules")
  println("---")
end


function make_category_voca(vocafile,termfile,tmpvocafile)
  tmpvocafile_fh=open(tmpvocafile,"w")
  termfile_fh=open(termfile,"w")

  vocafile_arr=open(readlines, vocafile) # automatically closes file handle
  n1=0
  n2=0
  termid=0
  for lineln=vocafile_arr
    if occursin(r"\r$", lineln)
      lineend="\r\n" # windows line ending
    else
      lineend="\n" # unix/linux line ending
    end
    line=replace(chomp(lineln), r"#.*" => "") # remove line endings & comments

    m=match(r"^%[ \t]*([A-Za-z0-9_]*)", line)
    if m == nothing
      n2=n2+1
    else
      found=m.captures[1]

      write(tmpvocafile_fh, "#$found$lineend")
      write(termfile_fh, "$termid\t$found$lineend")
      termid=termid+1
      n1=n1+1
    end
  end

  close(tmpvocafile_fh)
  close(termfile_fh)
  println("$vocafile has $n1 categories and $n2 words")
  println("generated: $termfile")
  println("---")
end


function voca2dict(vocafile, dictfile)
  dictfile_fh=open(dictfile,"w")

  vocafile_arr=open(readlines, vocafile) # automatically closes file handle
  newid=-1
  for lineln=vocafile_arr
    if occursin(r"\r$", lineln)
      lineend="\r\n" # windows line ending
    else
      lineend="\n" # unix/linux line ending
    end

    line=replace(chomp(lineln), r"#.*" => "") # remove line endings & comments
    if occursin(r"^[\s\t]*$", line) # skip blank lines
      continue
    end

    if occursin(r"^%", line)
      newid=newid+1
    else
      line_arr=split(line,r"[\s\t]+")
      name=popfirst!(line_arr)
      write(dictfile_fh, "$(newid)\t[$(name)]\t$(join(line_arr," "))$(lineend)")
    end
  end

  close(dictfile_fh)

  println("generated: $dictfile")
end


function main()
  grammar_prefix=ARGS[1] # can include path
  if ! isfile(grammar_prefix * ".grammar")
    error("can't find gramfile file: $(grammar_prefix).grammar")
  end
  if ! isfile(grammar_prefix * ".voca")
    error("can't find voca file: $(grammar_prefix).voca")
  end
  if length(ARGS) > 1
    error("mkdfa: too many arguments for call from command line")
  end


  mkfa= Sys.iswindows() ? "mkfa.exe" : "mkfa"
  dfa_minimize= Sys.iswindows() ? "dfa_minimize.exe" : "dfa_minimize"
  workingfolder=mktempdir()

  rgramfile= "$(workingfolder)/g$(getpid()).grammar"
  gramfile="$(grammar_prefix).grammar"
  vocafile=grammar_prefix * ".voca"
  termfile=grammar_prefix * ".term"
  tmpvocafile="$(workingfolder)/g$(getpid()).voca"
  dfafile=grammar_prefix * ".dfa"
  dictfile="$(grammar_prefix).dict"
  headerfile="$(workingfolder)/g$(getpid()).h"

  reverse_grammar(rgramfile,gramfile)
  make_category_voca(vocafile,termfile,tmpvocafile)
  run(`$mkfa -e1 -fg $rgramfile -fv $tmpvocafile -fo $(dfafile).tmp -fh
$headerfile`)
  run(`$dfa_minimize $(dfafile).tmp -o $dfafile`)
  voca2dict(vocafile, dictfile)

  rm("$(dfafile).tmp")
  rm(rgramfile)
  rm(tmpvocafile)
  rm(headerfile)
end

# called from command line
if length(ARGS) > 0
  main()
end
```

Prompts2wlist.jl – izrada liste riječi

```
####################################################################################
#
#    Copyright (C) 2015  VoxForge
#
#    This program is free software: you can redistribute it and/or modify
#    it under the terms of the GNU General Public License as published by
#    the Free Software Foundation, either version 3 of the License, or
#    (at your option) any later version.
#
#    This program is distributed in the hope that it will be useful,
#    but WITHOUT ANY WARRANTY; without even the implied warranty of
#    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#    GNU General Public License for more details.
#
#    You should have received a copy of the GNU General Public License
#    along with this program.  If not, see <http://www.gnu.org/licenses/>.
#
####################################################################################
using Printf

if VERSION < v"1.0"
   @warn("the VoxForge scripts require version 1.0 and above")
end

function prompts2wlist(prompts, wlist)
  if ! isfile(prompts)
    error("can't find prompts file: $prompts")
  end
  wordhash = Dict{String, Int32}()
  prompts_fh=open(readlines, prompts)
  for lineln=prompts_fh
    line=chomp(lineln)
    line_array=split(line,r"\s+");
    popfirst!(line_array)
    for word=line_array
      wordhash[word]=1
    end
  end
  wordhash["SENT-END"]=1
  wordhash["SENT-START"]=1

  wordlist = keys(wordhash) # returns an iterator
  wlist_arr=Array{String}(undef,length(wordhash))
  i=1
  for word=wordlist
    wlist_arr[i] = word * "\n"
    i=i+1
  end
  sortedwlist_arr=sort(wlist_arr)

  wlist_fh=open(wlist,"w");
  #write(wlist_fh, serialize(sortedwlist_arr) );
  for line=sortedwlist_arr
    write(wlist_fh,line)
  end
  close(wlist_fh)
end

# if called from command line
if length(ARGS) > 0
  if ! isfile(ARGS[1])
    error("can't find prompts file: $ARGS[1]")
  end
  if length(ARGS) <= 2
    prompts2wlist(ARGS[1],ARGS[2] )
  else
    error("prompts2list: too many arguments for call from command line")
  end

end
```

Prompts2mlf.jl – izrada Master Label File datoteke koja sadržava oznake (label)

```
################################################################################
#
#    Copyright (C) 2015  VoxForge
#
#    This program is free software: you can redistribute it and/or modify
#    it under the terms of the GNU General Public License as published by
#    the Free Software Foundation, either version 3 of the License, or
#    (at your option) any later version.
#
#    This program is distributed in the hope that it will be useful,
#    but WITHOUT ANY WARRANTY; without even the implied warranty of
#    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#    GNU General Public License for more details.
#
#    You should have received a copy of the GNU General Public License
#    along with this program.  If not, see <http://www.gnu.org/licenses/>.
#
####################################################################################
if VERSION < v"1.0"
   @warn("the VoxForge scripts require version 1.0 and above")
end

function prompts2mlf(prompts, mlf)
  mlf=open(mlf,"w");

  write(mlf,"#!MLF!#\n")
  prompts_arr=open(readlines, prompts)
  for lineln=prompts_arr
    line=chomp(lineln)
    line_array=split(line,r"\s+");
    fname=popfirst!(line_array)
    write(mlf,"\"$fname.lab\"\n")
    for word=line_array
        write(mlf,"$word\n")
    end
    write(mlf,".\n")
  end

  close(mlf)
end

# if called from command line
if length(ARGS) > 0
  if ! isfile(ARGS[1])
    error("can't find prompts file: $ARGS[1]")
  end
  if length(ARGS) <= 2
    prompts2mlf(ARGS[1],ARGS[2] )
  else
    error("prompts2list: too many arguments for call from command line")
  end

end
```

Mktrihed.jl – izrada datoteke koja sadrži CL naredbu i koristi se za povezivanje stanja

```
####################################################################################
#
#    Copyright (C) 2015  VoxForge
#
#    This program is free software: you can redistribute it and/or modify
#    it under the terms of the GNU General Public License as published by
#    the Free Software Foundation, either version 3 of the License, or
#    (at your option) any later version.
#
#    This program is distributed in the hope that it will be useful,
#    but WITHOUT ANY WARRANTY; without even the implied warranty of
#    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#    GNU General Public License for more details.
#
#    You should have received a copy of the GNU General Public License
#    along with this program.  If not, see <http://www.gnu.org/licenses/>.
#
################################################################################
if VERSION < v"1.0"
   @warn("the VoxForge scripts require version 1.0 and above")
end

function mktrihed(monophones, triphones, mktri)
  monophones_arr=open(readlines, monophones)  # automatically closes file handle

  hed=open(mktri, "w")
  write(hed, "CL $triphones\n")
  for phoneln=monophones_arr
    phone=chomp(phoneln)
    if length(phone)>0
      write(hed,"TI T_$phone {(*-$phone+*,$phone+*,*-$phone).transP}\n")
    end
  end
  close(hed)
end

# if called from command line
if length(ARGS) > 0
  if ! isfile(ARGS[1])
    error("can't find monophones file: $ARGS[1]")
  end
  if ! isfile(ARGS[2])
    error("can't find triphones file: $ARGS[2]")
  end
  if length(ARGS) > 3
    error("prompts2list: too many arguments for call from command line")
  end

  mktrihed(ARGS[1], ARGS[2], ARGS[3])
end
```

Mkclscript.jl – dodavanje stanja datoteci tree.hed

```
################################################################################
#
#    Copyright (C) 2015  VoxForge
#
#    This program is free software: you can redistribute it and/or modify
#    it under the terms of the GNU General Public License as published by
#    the Free Software Foundation, either version 3 of the License, or
#    (at your option) any later version.
#
#    This program is distributed in the hope that it will be useful,
#    but WITHOUT ANY WARRANTY; without even the implied warranty of
#    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#    GNU General Public License for more details.
#
#    You should have received a copy of the GNU General Public License
#    along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

```
#
##############################################################################

if VERSION < v"1.0"
   @warn("the VoxForge scripts require version 1.0 and above")
end

function mkclscript(monophones0, tree_hed, folder)
  hmmlist=open(tree_hed,"a");

  monophones0_arr=open(readlines, monophones0)
  for i=2:4
    for phoneln=monophones0_arr
      phone=chomp(phoneln)
      write(hmmlist,"TB 350 \"ST_$(phone)_$(i)_\" {(\"$phone\",\"*-
$(phone)+*\",\"$(phone)+*\",\"*-$phone\").state[$i]}\n")
    end
  end

  write(hmmlist,"\n")
  write(hmmlist,"TR 1\n")
  write(hmmlist,"\n")
  write(hmmlist,"AU \"$(folder)/fulllist\" \n")
  write(hmmlist,"CO \"$(folder)/tiedlist\" \n")
  write(hmmlist,"\n")
  write(hmmlist,"ST \"$(folder)/trees\" \n")

  close(hmmlist)
end

# if called from command line
if length(ARGS) > 0
  if ! isfile(ARGS[1])
    error("can't find monophones0 file: $(ARGS[1])")
  end
  if ! isfile(ARGS[2])
    error("can't find tree.hed file: $(ARGS[2])")
  end
  if length(ARGS) == 2
    mkclscript(ARGS[1], ARGS[2], "." )
  elseif length(ARGS) == 3
    if ! isdir(ARGS[3])
      error("can't find directory: $(ARGS[3])")
    end

    mkclscript(ARGS[1], ARGS[2], ARGS[3] )
  end

  if length(ARGS) > 3
    error("mkclscript: too many arguments for call from command line")
  end


end
```

Fixfulllist.jl – ispravljanje liste

```
################################################################################
#
#    Copyright (C) 2015  VoxForge
#
#    This program is free software: you can redistribute it and/or modify
#    it under the terms of the GNU General Public License as published by
#    the Free Software Foundation, either version 3 of the License, or
#    (at your option) any later version.
#
#    This program is distributed in the hope that it will be useful,
#    but WITHOUT ANY WARRANTY; without even the implied warranty of
#    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#    GNU General Public License for more details.
#
#    You should have received a copy of the GNU General Public License
#    along with this program.  If not, see <http://www.gnu.org/licenses/>.
#
################################################################################

if VERSION < v"1.0"
   @warn("the VoxForge scripts require version 1.0 and above")
end

function fixfulllist(in_fulllist, in_monophones0, out_fulllist)
  seen=Dict{String, Int32}()
  in_fulllist_arr=open(readlines, in_fulllist) # automatically closes file
handle
  in_monophones0_arr=open(readlines, in_monophones0) # automatically closes file
handle
  new_fulllist_arr=cat(in_fulllist_arr, in_monophones0_arr, dims=1)

  out_fulllist_fh=open(out_fulllist,"w")

  for phoneln=new_fulllist_arr
    phone=chomp(phoneln)
    if ! haskey(seen,phone) # remove duplicate monophone/triphone names
      seen[phone]=1
      write(out_fulllist_fh,phone * "\n")
    end
  end

  close(out_fulllist_fh)
end

# if called from command line
if length(ARGS) > 0
  if ! isfile(ARGS[1])
    error("can't find fulllist file: $ARGS[1]")
  end
  if ! isfile(ARGS[2])
    error("can't find monophones0 file: $ARGS[2]")
  end
  if length(ARGS) > 3
    error("fixfulllist: too many arguments for call from command line\nusage:
in_fulllist, in_monophones0, out_fulllist")
  end

  fixfulllist(ARGS[1], ARGS[2], ARGS[3])
end
```

Za detaljnije upute o izradi akustičnog modela te generiranju lab datoteka informacije dostupne na http://www.voxforge.org/home/dev/acousticmodels/windows/create/htkjulius/tutorial u sklopu tutoriala praćenog za izradu ovog projekta.