ACOUSTIC MODELING

Dr. Ivan Kraljevski, Dr. Frank Duckhorn

Fraunhofer Institute for Ceramic Technologies and Systems IKTS, Dresden

Daniel Sobe

Stiftung für das sorbische Volk, Bautzen

Table of contents

1 Introduction

Description: Improvement of the acoustic model. Optimization of the acoustic modelling performance with respect to the challenges identified in the feasibility study.

For this purpose, the phonetic inventory used in the acoustic model will be redefined and proved by an expert (phonetician/linguist) or a native speaker.
It will correspond to Upper Sorbian phonemes better than the adapted German acoustic model.

In addition, the performance will be further improved by the possibility of speaker-dependent (SD) adaptation of the model.

Finally, we want to upgrade the technology used and create the possibility to replace it in the future phases with the State-of-the-art technology (neural networks for feature vectors).

- Improvement of phoneme mappings, new acoustic model with native phoneme inventory for Upper Sorbian (revision/optimization of the phonetic transcription, especially by including the Sorbian phonemes) retraining of the acoustic model with the data from the feasibility study.
- Speaker-dependent adaptation. Speaker profiles through user enrollment (reading of short texts with adaptation sentences)

Impact: Optimized base model. Executable program for speaker-dependent adaptation and report. The models could be used on the existing demonstrator without reconfiguration.

Prerequisites: Support from a linguist, phonetician, or a native speaker.

2 Acoustic modeling

2.1 Phoneme inventory

The Table 1 represents the new phoneme inventory specified according to various sources[1][2] and to the best knowledge of native speakers involved in the project.

The phoneme labels may differ than the closest matching X-SAMPA[3] symbol.
In the default inventory the labels for the garbage ("#") and the pause (".") are included as well.

[1] "Tanja Anstatt / Christina Clasmeier / Sonja Wölke Obersorbisch Aus der Perspektive der slavischen Interkomprehension

[2] https://en.wikipedia.org/wiki/Upper_Sorbian_phonology

[3] https://en.wikipedia.org/wiki/X-SAMPA

Table 1. Grapheme to phoneme mapping

Grapheme	Ph. label	IPA	AM	Description
A	a	[a]	a	open central unrounded vowel, or low central unrounded vowel
B	b	[b]	b	voiced bilabial plosive or stop
C	ts	[ts]	ts	voiceless alveolar sibilant affricate
Č, Ć	tS	[tʃ]	tS	voiceless palato-alveolar sibilant affricate
D	d	[d]	d	Voiced alveolar plosive
E	E	[ɛ]	E	open-mid front unrounded vowel
Ě	ji	[ɪi]	ji	near-close front unrounded vowel
F	f	[f]	f	voiceless labiodental fricative
G	g	[g]	g	voiced velar plosive or stop
H	h	[h]	h	voiceless glottal fricative
I	i	[ɪ]	i	close front unrounded vowel
J	j	[j]	j	voiced palatal approximant
K	k	[k]	k	voiceless velar plosive or stop
Ł, W	w	[w]	w	voiced labial–velar approximant
L	l	[l]	l	voiced alveolar lateral approximant
M	m	[m]	m	voiced bilabial nasal
N	n	[n]	n	voiced alveolar nasal
Ń	jn	[ɲ][nʲ]	jn	Voiced palatal nasal
O	O	[ɔ]	O	open-mid back rounded vowel
Ó	U	[ʊ]	U	near-close near-back rounded vowel
P	p	[p]	p	voiceless bilabial plosive or stop
R	R	[ʁ]	r	voiced uvular fricative
Ř, Š	S	[ʃ]	S	Voiceless palato-alveolar fricative
S	s	[s]	s	Voiceless alveolar sibilant
T	t	[t]	t	Voiceless alveolar plosive
U	u	[u]	u	close back rounded vowel
Y	1	[ɨ]	1	close central unrounded vowel
Z	z	[z]	z	voiced alveolar sibilant
Ž	Z	[ʒ]	Z	Voiced postalveolar fricative
CH	x, C	[x][ç]	x, C	voiceless velar fricative or voiceless palatal fricative
DŹ	dZ	[dʒ]	dZ	Voiced postalveolar affricate
V	v	[v]	f	voiced labiodental fricative
X	k s	[ks]	k s	
Q	k w	[kw]	k w	
glottal stop	Q	[ʔ]	Q	Glottal stop

2.2 Pronunciation variants

The exceptions rules defined in the feasibility study are redefined according to the new phoneme inventory definitions with new additions.

New phoneme mappings and rules are included in the delivery of the work package 4 (WP4).

Table 2. List of pronunciation rules, with notable examples.

#C_Ł_$	*	zmysł, přinjesł, wotrostł, móhł ("l" na koncu po konsonance so smorny)
$_CH_C	*	chcedźa, chce, chcyše, chcemy, chcu, chcych, chceće (twjerdy "ch" na zapocatku pred "c" so smorny) - ale: nic preco?
$_CH_#V	k	chłódźa, Chróśćicach, chórje, chowanka, chětro, chlěbom (twjerdy "ch" za zapocatku, jeli njesleduje "c", so zmeni na "k")
$_CH_L	k	chłódźa, Chróśćicach, chórje, chowanka, chětro, chlěbom (twjerdy "ch" za zapocatku, jeli njesleduje "c", so zmeni na "k")
$_Ł_#C	*	łžicy ("l" na zapocatku pred konsonantom so smorny)
A_Š_Ł	j S	zašłosće, zašłe, našła ("j" so mjez "a" a "sl" zasuhny)
U_Š_Ł	j S	wušło, dalse priklady??? ("j" so mjez "u" a "sl" zasuhny)
Ó_Ž_D	j Z	kóždy, kóždolětnu, kóždeho, kóždu, kóžde, kóždemu ("j" so mjez "o" a "zd" zasuhny)
#V_Ž_K	S	nałožki, chěžki, třiróžku, podłožki, nóžki, wotnožkow, knižku, hdźežkuli, hněžko, ćežko (po wokalu a pred "k" zmeni so "z" na "S")
#V_Z_K	s	zwjazk, wotrězk, rjećazk, wurězki, rozkładźe, tyzku, pomazkami (po wokalu a pred "k" zmeni so "z" na "s")
_Ž_$	S	kaž, tež, kotrež, dołhož, hdyž, jenož, dokelž, štož, ručež ("z" na koncu slowa so zmeni na "S")
Ě_H_#C	*	přěhrać, něhdźe, wuzběhny, něhdy, lěhwo, přěhrać ("h" po "e" pred konsonantom so smorny)
E_CH_	C	kładźechu, nańdźechu, přednjesechmy, přińdźechmy, nańdźech, potrjechi ("ch" po "e" so zmeni na "C")
I_CH_	C	serbskich, młódšich, nichtó, jich, přichodnje ("ch" po "i" so zmeni na "C") - ale: přichadźachu, přichwata
I_J_$	*	studij, Handrij, Maćij, gymnazij, julij, kij ("j" na koncu slowa po "i" so smorny)
K_Ń_$	*	tykń, wotamkń, dalse priklady??? ("n" na koncu slowa po "k" so smorny)
S_Ń_$	*	njećisń, dalse priklady??? ("n" na koncu slowa po "s" so smorny)
#V_Ń_$	n	kamjeń, zakoń, tydźeń, woheń, swjedźeń ("n" na koncu slowa po wokalu zmeni so na "n")
I_Ń_	n	přińdź, začińtaj ("n" po "i" so zmeni na "n")
T_Ř_I	ts	kotřiž, třiróžku, tři, njeroztřihaš, jutřišeje, wótřiši, někotři (kombinacija "tri" so wurjekuje "tsi") - ale: problem dwojny t
T_Ř_Ě	ts	třěšneho, třěšny, třěćinje, třělić (kombinacija "tre" so wurjekuje "tse") - ale: problem dwojny t
_B_$	p	wob, chlěb, Serb, hrib, rub ("b" na koncu slowa zmeny so na "p")

_B_K	p	wobkrućа, wobknježa, wuhibki, hubkowaše, žabka, babka ("b" pred k so zmeny na "p")
_B_S	p	serbskim, wobstejachu, wobswětowych, herbstwa ("b" pred s so zmeny na "p")
_B_Š	p	wobšěrnym, serbšćiny, najlubšej, hłubši ("b" pred s so zmeny na "p")
_B_CH	p	wobchody, wobchad, wobchować ("b" pred ch so zmeny na "p")
_D_$	t	lód, zawod, sad, před, zakład, zarjad ("d" na koncu slowa zmeny so na "t")
_D_K	t	zrědka, doprědka, srědki, přehladka, wuslědkow ("d" pred k zmeny so na "t")
_D_S	t	předsydki, předstajeja, wobsydstwo, susodstwje ("d" pred s zmeny so na "t")
_D_Š	t	najradšo, najmłódšej, předšulskich, podšmórnyło, rědšo ("d" pred s zmeny so na "t")
_D_CH	t	předchadźacym ("d" pred ch zmeny so na "t")
_D_Č	*	přeswědčeny, předčitać, wobswědčeja, wuswědčenju ("d" pred c so smorny)
_DŹ_$	tS	srjedź, pojědź, wosrjedź, přińdź, jědź, snadź ("dz" na koncu slowa zmeny so na "tS")
_Z_$	s	mjez, bjez, naraz, wukaz, raz, z - ale: z druhdy jako "z"?
E_J_I	*	stejimy, meji, čejim (w kombinacije "eji" so "j" smorny)
_E_DŹ	e	chcedźa, srjedź, njeskedźbliwa, swjedźenskim, dowjedźe, posedźenja, rjedźe (zacinjeny "e" pred "dz")
_E_J	e	prawopisneje, poskitkej, stej, měšćanskeje, ćežkej, wjesnej, čornej (zacinjeny "e" pred "j")
_E_Ń	e	mjeńšin, swjedźeń, dźeń, woteńdźeš, škleńcu, woheń, njeńdźe, prózdnjeńcowe - ale: wustajeńcu, kamjeń, Leńka (???)
_E_Ž	e	tež, knježi, leži, hdźež - ale: kotrež, šćežce(???), Drježdźanach, ćežišća
_H_$	*	sněh, žiwjenjoběh, wobsah ("h" na koncu slowa so smorny)
_H_#C	*	njewobhladujemy, zahroda, njehraje, wothłós, hrodźišću, hłós, hdźež, hłownu, hdyž, hladajće ("h" pred konsonantom so smorny)
Ě_CH_	C	spěchować, běchu, dźěchmy, třěchi, wuspěcha, wšěch, wujěcha, něchtó ("ch" po "e" so zmeni na "C")
_O_Ł	o	kołowokoło, dospołnosće, rozmołwy, zamołwitosći, powołaja, tołsta, hołbicu ("o" pred "l" je zacinjeny) - ale: delnjołužiskim (hranica slowow)
_O_W	o	powšitkowne, pruwowanski, hłownaj, jadrowej, towarstwow, institucijow, słowjanskej ("o" pred "w" je zacinjeny)
O_W_#C	u	sadowcami, bratrowska, powšitkownych, kralownu, sportowni ("w" po "o" pred konsonantom so zmeni na "u") - ale: druhdy so wuwosta
O_Ł_#C	u	zamołwiteho, žołtu, rozmołwjał, namołwja, tołstu ("l" po "o" pred konsonantom so zmeni na "u") - ale: druhdy so wuwosta
#C_W_J	*	zwjetša, wobsydstwje, wuměłstwje ("w" po konsonance pred "j" so smorne)
#V_W_#C	u	přestawce, Žitawčanow, najnowšim, stawje, nawróća, zjawnje ("w" pred konsonantom zmeni so na "u") - ale: předewšěm
#C_W_#C	u	dito

$_W_J	*	wjacorych ("wj" na zapocatku slowa so "w" smorny) - ale: wjele, wječorach
$_W_#C	*	wšak, wsy ("w" na zapocatku slowa pred konsonantom so smorny) - ale: "we wsy"
O_W_$	*	Błótow, etapow, wosadow, rjadow ("w" na koncu slowa po "o" so smorny)
A_W_$	u	staw ("w" na koncu slowa po "a" so zmeni na "u")
#C_W_$	*	čerw ("w" na koncu slowa po konsonance so smorny)
O_Ł_$	*	poł, sokoł, šoł ("l" na koncu slowa po "o" so smorny)
A_Ł_$	u	widźał, jednał, Michał ("l" na koncu slowa po "a" so zmeni na "u")
I_Ł_$	u	rozsudźił, stworił, přełožił ("l" na koncu slowa po "i" so zmeni na "u")
$_P_T	*	ptačim (p na zapocatku slowa pred t so smorny)

2.3 Lexicon creation

The lexicon and other metadata are created using the BASGenerator with the corresponding phoneme inventory and pronunciation rules. Words with initial vowel are also preceded with a glottal stop.

The lexicon is exported in two formats, with space as delimiter (*.lex) for the phonemes and without (*.ulex). In the latter case the lexicon parser considers first the longest sequence, therefore is it recommended in case of creating acoustic models for a new language that the phoneme labels are unique and not longer than ASCII characters.

Some examples of the generated lexicon:

- With the added “glottal stop” and exception rule.

```
AWTOBIOGRAFISKE      a u t O b i O g r a f i s k E
AWTOBIOGRAFISKE      a w t O b i O g r a f i s k E
AWTOBIOGRAFISKE      Q a u t O b i O g r a f i s k E
AWTOBIOGRAFISKE      Q a w t O b i O g r a f i s k E
```

- Random selection:

```
KRUTY         k r u t 1
NJERJANA      n j E r j a n a
PŁUCO         p w u ts O
FACHOWA       f a x O w a
DELEGOWANA    d E l E g O w a n a
WOTEBĚRA      w O t E b ji r a
MANDŹELSKU    m a n dZ E l s k u
PĚSTOWAĆ      p ji s t O w a tS
STUPIŁA       s t u p i w a
ROZMOŁWU      r O z m o w u
DŹĚŁAŁA       dZ ji w a w a
NAWUKNYCHU    n a w u k n 1 x u
KAJKU         k a j k u
ROZHŁOSU      r O z w O s u
HALI          h a l i
```

2.4 Usage of pretrained acoustic models

It is possible to use a pretrained model of a different language (for example German or English) and rename or combine existing phoneme models into new. Such model can be used either for recognition or forced-alignment phoneme segmentation (labeling).

The script `mapAM.py` takes a YAML configuration file with the specified mappings and parameters. The script requires compiled python (3.7) wrapper of the dLabPro software. The created acoustic model must be used with the source `feainfo.object` file containing statistics necessary for correct feature extraction.

Command syntax:

```
mapAM.py hsb.yaml
```

Example of the YAML configuration file (hsb.yaml).

```
# Configuration file

source: '3_20.hmm' #source AM model
target: 'hsb.hmm'  #target AM model

gauss_tie: false   #Gaussian tying (default: false)

mapping:           #Phoneme mappings (same order as in the classes.txt file)
    '.': ['.']
    '#': ['#']
    '1': ['Y']
    'C': ['C']
   …
    'U': ['U']
    'Z': ['z','S']
    'a': ['a:']
    'b': ['b']
    'd': ['d']
    'dZ': ['d','z','S']
    'e': ['e:']
   …
    'i': ['i:']
    'j': ['j']
    'ji': ['j','i:']
    'jn': ['j','n']
   …
    'o': ['o:']
   …
    'tS': ['t','S']
    'ts': ['t','s']
    'u' : ['u:']
    'w' : ['U','v']
    'x' : ['x']
    'z' : ['z']

#Classes file for validation (generated from the BASGenerator script)
classfile: "classes.txt"
```

2.5 Forced-alignment phoneme segmentation

To train and evaluate new acoustic model from scratch, it is necessary to provide phoneme segmentations for each recording in the datasets.
This can be done either manually using suitable tools (e.g., Praat or Wavesurfer) or automatically by forced alignment procedure using a pretrained model (see section 2.4).

The phoneme segmentation procedure requires a word loop lexicon (generated by the BASGenerator script) located in the `“uasr_configuration/grammar”` folder.

The model must have the same phoneme inventory, otherwise the resulting phoneme segmentations must be mapped to the target phoneme labels.

The labels must conform to the following format:

```
Header
#
   “Starting position (sec)” “121” “phoneme label”
```

Example:

```
Created by dLabPro.
File format: ESPS/waves+
#
  0.00000000 121 #
  0.66000000 121 .
  1.05000000 121 #
  1.34000000 121 p
…
```

The used format is modified `ESPS/waves+,` with the difference that starting time is used instead of the ending time for the phoneme labels.

For example, when loading the label file into Wavesurfer for visual analysis, the labels should be shifted one segment later.

For each entry in the file list there must be a signal (recording) and a corresponding transliteration file (textual file) where each word is in a new line.
All the words from the transliterations must be part of the lexicon in the word-loop grammar file.

Example for the recording 0001HSB_1_000_2:

```
WÓN
JE
PŘEDSYDU
DOMOWINY
W
NALEŽNOSĆI
WO
PODPĚRU
PROSYŁ
```

The labeling process itself is called with the dLabPro/UASR command:

```
dlabpro $UASR_HOME/scripts/dlabpro/HMM.xtp lab HSB-01/info/label.cfg
```

The configuration file "`label.cfg`" contains some parameters that should be adjusted. For the more information about the configuration keys, refer to the UASR documentation[1].

This example uses pretrained model "`hsb.hmm`" created with the mapAM.py script. The configuration key `uasr.filelist.test = "all.flst"` contains all the recordings that should be labeled regardless whether they are used for training or testing and performance evaluation.

```
## UASR configuration file
## - Verbmobil database

uasr.db                 = "db-hsb-asr";
uasr.exp                = "HSB-01";
uasr.customize          = "$UASR_HOME-data/db-hsb-asr/HSB-01/info/de-
fault.itp";

uasr.sig.srate          = 16000;
uasr.pfa                = "UPFA";

uasr.flist.test         ="all.flst"
uasr.from               ="T"
uasr.out                ="lab"
uasr.type               ="phn"

uasr.dir.sig            ="$UASR_HOME-data/db-hsb-asr/common/sig"
uasr.lab.ext            ="lab";

uasr.am.classes ="$UASR_HOME-data/db-hsb-asr/HSB-01/info/classes.txt"

uasr.am.model ="hsb"
uasr.am.sil   =0;
uasr.am.gbg   =1;

uasr.lm       ="fsg"
uasr.lm.fsg   ="$UASR_HOME-data/db-hsb-asr/HSB-01/grammar/hsb.grm";

uasr.dir.out  ="$UASR_HOME-data/db-hsb-asr/common/lab"
uasr.dir.trl  ="$UASR_HOME-data/db-hsb-asr/common/trl"
uasr.trl.ext  ="lab";

## EOF
```

IMPORTANT: The destination folder `uasr.dir.out` for the labeling must be empty! The script will take existing label files (if any) for re-alignment.

2.6 Acoustic model training with dLabPro/UASR

The training of a new acoustic model from scratch requires the phoneme segmentation created in the previous step (`uasr.dir.lab`) and train and test file lists.

The best performing model on the test data is split for further training of the higher order models. To avoid the bias in the training it is very important to make sure that speakers in the training set do not occur in the testing set.

The training process itself is called with the dLabPro/UASR command:

```
dlabpro $UASR_HOME/scripts/dlabpro/HMM.xtp trn HSB-01/info/train.cfg
```

[1] https://rawgit.com/matthias-wolff/UASR/master/manual/index.html

An example of the configuration file for training, later can be used for evaluation as well:

```
## UASR configuration file
## - Verbmobil database

uasr.db           = "db-hsb-asr";
uasr.exp          = "HSB-01";
uasr.customize    = "$UASR_HOME-data/db-hsb-asr/HSB-01/info/default.itp";

uasr.sig.srate    = 16000;
uasr.pfa          = "UPFA";

uasr.flist.train  = "train.flst"
uasr.flist.test   = "test.flst"

uasr.dir.sig      ="$UASR_HOME-data/db-hsb-asr/common/sig"
uasr.dir.lab      ="$UASR_HOME-data/db-hsb-asr/common/lab"
uasr.lab.ext      ="lab";

uasr.am.sil       = 0;
uasr.am.gbg       = 1;

uasr.am.classes   ="$UASR_HOME-data/db-hsb-asr/HSB-01/info/classes.txt"

## EOF
```

Training could usually last for many hours, depending on the used hardware.

The output can be redirected into a log file for better monitoring, for each split and iteration the evaluation is performed on the test set.

Within a split the best performing model (iteration) will be used to start the next Gaussians split.

The training stops either by terminating the process or giving the explicit number of iterations and splits (see dLabPro/UASR documentation).

2.7 Performance evaluation and analysis

For the performance analysis the same configuration file is used as for training with the addition of configuration key for the desired acoustic model (`uasr.am.model="3_8";`).

The evaluation process itself is called with the dLabPro/UASR command:

```
dlabpro $UASR_HOME/scripts/dlabpro/HMM.xtp evl HSB-01/info/train.cfg -Puasr.am.model="3_8" -v2
```

With the verbose level 2, the output will contain the confusion matrices for the frame and phoneme labels.

The option `uasr.am.eval.lab=TRUE;` provides the reference and the recognized phoneme label files that can be used for more thorough analysis.

Selection of best performing models with different splits are created during the training is provided as the part of the delivery.

Table 3. Results on the test set evaluation

Model ID	Parameters (k)	Frame label Correctness	Label sequences Correctness	Label sequences Accuracy	Lattice Density
0_5	**36.2**	**60.9 % +-0.9 %**	**62.1 % +-0.6 %**	**54.1 % +-0.6 %**	**0.969 +-0.004**
1_10	**72.4**	**62.5 % +-0.9 %**	**65.3 % +-0.5 %**	**56.9 % +-0.6 %**	**0.988 +-0.004**
2_11	**144.8**	**64.3 % +-0.9 %**	**68.9 % +-0.5 %**	**60.2 % +-0.6 %**	**1.000 +-0.004**
3_8	**289.5**	**67.1 % +-0.8 %**	**71.4 % +-0.5 %**	**61.7 % +-0.6 %**	**1.017 +-0.004**
4_4	**579**	**68.1 % +-0.8 %**	**73.0 % +-0.5 %**	**61.8 % +-0.6 %**	**1.039 +-0.005**

The table presents the model ID that shows the split and the iteration, number of parameters in thousands (k), frame label correctness, label (phoneme) sequences correctness and accuracy and the label density.

The metrics are calculated by the following formulas:

• Correct frames/phonemes:	C
• Substitutions:	S
• Insertions:	I
• Deletions:	D
• Length of the reference sequence:	LT = C+S+D
• Length of the recognized sequence:	LR = C+S+I
• Correctness:	COR = C/LT
• Accuracy:	ACC = 1-(S+I+D)/LT = (C-I)/LT
• Lattice density:	LD = LR/LT

3 Speaker-Dependent Adaptation

3.1 Adaptation text selection

The content of the adaptation texts should be as general as possible and suitable for all types of the speakers (children, females, and males).
The texts should be phonetically balanced with the mono-phones as scoring criteria.
The number of sentences should be as small as possible that the speaker could read them in max 10-15 minutes.
The BASGenerator will generate the Speechrecorder projects, a third-party recording application can be used (Audacity, Wavesurfer) or a custom application tailored for speaker enrollment.
It is important that the recordings are performed on the audio devices (microphones, mic-arrays) that will be used by the speaker, providing in the same moment speaker dependent and equipment dependent adaptation of the acoustic models.
Each recording in the signals folder (sig) must have the matching transliteration file in the corresponding folder (trl), according to the conventions used in dLabPro/UASR.

3.2 Acoustic model adaptation with dLabPro/UASR

The adaptation process itself is called with the dLabPro/UASR command:

```
        dlabpro $UASR_HOME/scripts/dlabpro/HMM.xtp adp HSB-01/info/adapt.cfg
```

Important: Note that the adaptation set should be assigned as a development file list, if not it will fall back to the test file list for adaptation)

An example of the configuration file for adaptation:

```
## UASR configuration file
## - Verbmobil database

uasr.db              ="db-hsb-asr";
uasr.exp             ="HSB-01";
#uasr.customize       ="$UASR_HOME-data/db-hsb-asr/HSB-01/info/default.itp";

uasr.sig.srate       =16000;
uasr.pfa             ="UPFA";

uasr.flist.dev       ="adptrain.flst"
uasr.flist.test      ="adptest.flst"

uasr.am.eval         ="eval";
uasr.am.eval.mode    ="T";

uasr.lx              ="$UASR_HOME-data/db-hsb-asr/HSB-01/lexicon/hsb_sampa.ulex"

uasr.dir.trl         ="$UASR_HOME-data/db-hsb-asr/common/trl"
uasr.dir.sig         ="$UASR_HOME-data/db-hsb-asr/common/sig"
uasr.trl.ext         ="trl";

uasr.am.classes      = "$UASR_HOME-data/db-hsb-asr/HSB-05/info/classes.txt"

uasr.am.model         ="3_8";   #The model to be adapted
```

The resulting model is saved along with the original with the suffix "_A",
(e.g., 3_8_A.hmm)

The evaluation process of the adapted model is called with the dLabPro/UASR command with the `uasr.am.eval.mode="0"`:

```
        dlabpro $UASR_HOME/scripts/dlabpro/HMM.xtp evl \\
        HSB-01/info/adapt.cfg -Puasr.am.model="3_8_A" -Puasr.am.eval.mode="0" -v2
```

In this case the performance will be evaluated by using the transliterations and the lexicon as a word-loop recognition grammar.
The proper way to do the evaluation is the free phoneme recognition (without a grammar) and comparing with the already existing phoneme segmentation:

```
        dlabpro $UASR_HOME/scripts/dlabpro/HMM.xtp evl HSB-01/info/eval.cfg -
        Puasr.am.model="3_8_A" -v2
```

With the corresponding configuration file:

```
## UASR configuration file
## - Verbmobil database

uasr.db              = "db-hsb-asr";
uasr.exp             = "HSB-01";
uasr.customize       = "$UASR_HOME-data/db-hsb-asr/HSB-01/info/default.itp";
uasr.sig.srate        = 16000;
uasr.pfa              = "UPFA";

uasr.flist.test       = "adptest.flst"
uasr.am.eval          ="assess";
uasr.am.eval.asses    ="cmx";


uasr.am.classes       = "$UASR_HOME-data/db-hsb-asr/HSB-01/info/classes.txt"
uasr.dir.sig          ="$UASR_HOME-data/db-hsb-asr/common/sig"
uasr.dir.lab          ="$UASR_HOME-data/db-hsb-asr/common/lab"
uasr.lab.ext          ="lab";

uasr.am.sil           =0;
uasr.am.gbg           =1;

uasr.am.model         ="3_8";

## EOF
```

Using higher level of verbosity (>1) also provides the confusion matrices of the frame and label sequences, that could be used for more detailed evaluation and pinpointing the systematic issues with some phonemes.
The differences in the results of both approaches are illustrated in the next section.

3.3 Phoneme recognition comparison SI vs. SD models

The adaptation experiment is carried out with the recordings provided by one speaker (D. Sobe). To get the phoneme recognition performance, for the provided adaptation and test sentences the phoneme segmentation is necessary as already described in the Section 2.5, but in this case the existing model "3_8" was used for labeling.

Table 4. The performance evaluation using the lexicon as a word-loop recognition grammar.

Model ID	Parameters (k)	Frame label Correctness	Label sequences Correctness	Label sequences Accuracy	Lattice Density
3_8	**289.5**	**67.1 % +-4.1 %**	**58.2 % +-5.8 %**	**45.0 % +-7.1 %**	**1.044 +-0.034**
3_8_A	**289.5**	**65.4 % +-3.3 %**	**69.6 % +-4.1 %**	**61.2 % +-5.4 %**	**0.996 +-0.036**

It is obvious that the phoneme recognition results are too optimistic and biased due to the using of the matching word-loop (lexicon) as the recognition grammar.

Table 5. The performance evaluation using free phoneme recognition.

Model ID	Parameters (k)	Frame label Correctness	Label sequences Correctness	Label sequences Accuracy	Lattice Density
3_8	**289.5**	**61.4 % +-2.6 %**	**48.3 % +-4.3 %**	**28.0 % +-5.4 %**	**1.146 +-0.051**
3_8_A	**289.5**	**62.4 % +-3.2 %**	**61.5 % +-3.5 %**	**49.0 % +-3.9 %**	**1.035 +-0.047**

The speaker dependent adaptation significantly improves the performance of the acoustic modelling of the phonemes and consequently the overall speech recognition in the voice applications.