# JRTk and JANUS

The Ibis-Gang

(IBIS V5.1.1 P000)

# Contents

# Chapter 1

# Introduction

This manual describes *JRTk*, the Janus Recognition Toolkit, in version *V5.1.1 P000*, which includes the *Ibis* decoder. This manual also contains pointers, where to look for further information. One important page is the online JRTK documentation available at `http://isl.ira.uka.de/~jrtk/janus-doku.html`.

In the following chapter 2 you'll find the information you'll need to get Tcl and Janus up and running. We focus on the UNIX variants, although much of the information also applies for Windows installations. You might want to have a look at a sample `.janusrc` first, which is the main configuration file for Janus. The basic concepts of the JANUS user interface are discussed in 3. Chapter 5 covers all you'll need to know in order to train a system using JRTk, while chapter 6 covers the Ibis decoder. If you're experiencing difficulties and need help in either installing Janus, configuring it properly, or running scripts, the trouble-shooting section 7 contains (hopefully) useful information.

The JANUS interface having an object-oriented style, you'll find descriptions of all modules in chapter 8; this will be of interest to both the user and the would-be C programmer. The Tcl-library, which should save you a lot of effort when building systems at script level, is described in chapter 9. The "Janus Scripts Collection", which comprises a number of standard scripts to build and test systems, also relies on the Tcl-library. It is discussed in section 3. Chapter 10 describes some of JANUS' files and their formats.

The people who have worked on JANUS over the time can be found in 11. At the end of this document, you'll also find a bibliography and a glossary. Chapter 2 also contains some information on how to use this manual, available in Postscript, HTML and PDF format. If you have questions or problems with JRTk, please send e-mail to `jrtk@ira.uka.de`.

Janus was successfully used in a number of evaluations, see [1, 13, 8].

# Chapter 2

# Basics

## 2.1 What is it?

The goal of the ISL's JANUS project is to build a general–purpose speech recognition toolkit useful for both research and applications. Currently, the software consists of JRTk, the Janus Recognition Toolkit for the development of speech recognition systems, including the Ibis decoder. This document attempts to serve two purposes: the first one is to jump–start users in getting the basic jobs done with JANUS, be it for research projects, or be it to build another system using JANUS, while the second purpose is to also give an overview of the current research done within the JANUS project. This document is for incoming researchers and students as well as external partners in order to familiarize themselves with the options and procedures to make the most of the existing code-base. At the end of this document, you find a list of references to JANUS and an index, covering the most important concepts, files and commands used in JANUS.

### Terminology

Over time, a number of terms have evolved, refering to different parts of the system, although JANUS' nomenclature is not always strictly adhered to:

**JRTk** refers to the ASR Toolkit developed at the *ISL* in Karlsruhe (*UKA*) and Pittsburgh (*CMU*). It is implemented in C code, with an interface in Tcl/Tk, having an object-oriented look-and-feel.

**janus** means the `janus` executable.

**JANUS or Janus** can often be replaced by JRTk or janus.

**Ibis** denotes the one-pass decoder available in Janus V5.0 and later.

### Why the names?[1]

**Janus:** Roman god of doorways and archways, after whom the month of January is named.

[1] From http://concise.britannica.com.

Often depicted as a double-faced head, he was a deity of beginnings. The worship of Janus dated back to the earliest years of Rome, and the city had many freestanding ceremonial gateways called jani, used for symbolically auspicious entrances or exits. The festival of Janus, the Agonium, took place on January 9.

**Ibis:** Egyptian Djhuty, also spelled Djhowtey.

In Egyptian religion, a god of the moon, of reckoning, of learning, and of writing. He was held to be the inventor of writing, the creator of languages, the scribe, interpreter, and adviser of the gods, and the representative of the sun god, Re.

Thoth in turn was frequently represented in human form with an ibis' head.

## 2.2 About the documentation

This documentation is intended to cover most aspects that you'll need to know to use JRTk at the Tcl/Tk level. You should also find a lot of useful information if you need to change the C-Source. If you find errors or omissions, feel free to contact one of the maintainers (section 11) or send e-mail to `jrtk@ira.uka.de`. Don't forget to look at the "trouble-shooting" section 7, too.

This documentation contains four main parts:

1. A cookbook of training procedures in chapter 5. Basic system training can most easily be done by using the Janus Scripts Collection documented in 3.4.

2. A How-To on decoding strategies and using existing systems with the Ibis one-pass decoder in chapter 6.

3. The alphabetical list of modules available at Tcl-level with their description in chapter 8; a list of functions provided by the Tcl-library can be found in chapter 9.

4. A description of files and formats needed or used in JANUS in chapter 10.

The source and documentation are kept under SVN, it is assumed that you are familiar with version control. To build documentation from the sources, you should be able to use "make pdf" with the provided "Makefile" as follows:

```
~/janus/doc > ../src/Linux/janus lib2tex.tcl
~/janus/doc > ../src/Linux/janus tcl2tex.tcl
~/janus/doc > make pdf
```

Generating documentation was last tested on "Snow Leopard" using Tcl 8.4.

## 2.3 Installation

All Janus software is contained in a `janus` directory, which you can either find on your distribution media, copy from somewhere, or check out from CVS/SVN. Installing JRTk consists of the following three steps:

1. Copy the janus distribution directory somewhere on your file system.

   We suggest to create a janus directory in your home directory. This directory will be referred to as <JANUSHOME> in the future, it should contain the library, tcl-lib, gui-tcl and optionally the bin, src and doc subdirectories.

2. Set environment variables appropriately.

   Set your search path, so that the correct janus executable for your system and architecture can be found. On a Linux system, you can for example add <JANUSHOME>/src/Linux.gcc to your PATH environment variable. Alternatively, you can copy the executable(s) to a location already on your search path (e.g. ~/bin).

   On Unixes, janus needs three environment variables:

   JANUS_LIBRARY needs to be set to <JANUSHOME>/library

   HOST should be set to the name of your node. On Linux machines using a tcsh, you can say setenv HOST ‘uname -n‘

   HOME should contain the path to your home directory. In principle, this can be any directory.

   Note that on some Unix machines, it might also be necessary to set TCL_LIBRARY and TK_LIBRARY to appropriate values (often /usr/lib/tcl8.4 and /usr/lib/tk8.4), and you might need to set LD_LIBRARY_PATH, if you want to use dynamic linking. An example .tcshrc excerpt:

   ```
   # JRTk
   setenv JANUS_LIBRARY   "${HOME}/janus/library"
   setenv TCL_LIBRARY     "/usr/lib/tcl8.4"
   setenv TK_LIBRARY      "/usr/lib/tk8.4"
   setenv LD_LIBRARY_PATH "${HOME}/tools/nist/lib:${HOME}/tools/portaudio/lib/.libs:${HOME}/t
   ```

3. Adapt the startup configuration file .janusrc to your needs.

   Copy the <JANUSHOME>/scripts/janusrc file to ${HOME}/.janusrc, i.e. the directory declared with the HOME environment variable. Open the .janusrc file with a text editor and change the lines setting the JANUSHOME variable to the value of <JANUSHOME>. If you experience difficulties when creating logfiles on a Windows platform, try uncommenting the set LOGFILE "janus.log" line.

If you are used to a Unix-style environment and work on a Windows platform, you might consider looking at the Cygwin tools (http://www.cygwin.com/); although Janus will run just as well without them.

The default .janusrc automatically optimizes your Janus setup depending on the architecture, operation system and location you use. If janus doesn't start or these automatically detected settings are incorrect, there are several things you can to check:

1. Are all the dynamically linked libraries there? This is not the case if Janus complains about missing libraries, it can be fixed by setting the environment variable `LD_LIBRARY_PATH` accordingly. Tcl/Tk has to be available in the correct version.

2. Does Janus not start because of wrong X settings? You either have to set the `DISPLAY` environment variable or run a Janus binary compiled without X support.

3. Are the paths set correctly (cf. `~/.tcshrc`)?

4. Can Janus initialize properly? The environment variable `JANUS_LIBRARY` should be set to `~/janus/library` (or whatever is appropriate) and `~/.janusrc` should contain the lines

   ```
   set JANUSLIB  "$env(HOME)/janus/gui-tcl"
   set auto_path "$env(HOME)/janus/tcl-lib \
                  $JANUSLIB $auto_path"
   ```

   or equivalent. These lines tell Janus where to find the Tcl scripts needed to initialize Tcl and Janus itself properly. You can override the settings for `gui-tcl` and `tcl-lib` in your own scripts, but you have to know what you're doing ;-)

5. If Janus runs all right, but blocks (i.e. stops) when it tries to use `fgets` (which is used in most procedures provided in `tcl-lib`), you're most likely experiencing an `fgets`-Problem and you want to read the 7 section.

If the above installation did not work for you, there are a few additional things to check. You'll find some information in section 7.

On `Windows`, JRTk can be installed as follows:

- install gunzip (GnuWin32) for windows, preferable in version 1.3.5 (gzip-1.3.5-bin.exe)
- copy .janusrc to your home directory (Documents and Settings/... user name ...)
- edit .janusrc as appropriate
- install Tcl8.4 (ActiveTcl8.4.19.1.286921-win32-ix86-threaded.exe)
- add to system variables (Settings, Control Panel, System, Advanced, Environment Variables):
  - new JANUS_LIBRARY "...path.../janus/library" (Unix notation!)
  - add to PATH: "...path...\GnuWin32\bin" and "...path...janus\tcl-lib" (you may need to do this as an administrator, Windows notation!)
- reboot windows

Compilation on Windows is usually done with Visual Studio. Also, be aware that Windows uses CR+LF line endings in text files, which under some circumstances cannot be read properly on Unixes. Also, if you get weird errors when loading acoustic models, try if the gzip and gunzip commands can be executed successfully from the command line.

## 2.4 Language Models

The generation of language models is not part of JRTk. The standard LM in the Ibis decoder, created and loaded with

```
[LingKS lm$SID NGramLM] load $lmDesc
```

can read a standard ARPA-format file. These can be created by a number of toolkits:

- The CLAUSI tools available at ISL.
- The CMU-SLMT toolkit, available at `http://svr-www.eng.cam.ac.uk/~prc14/toolkit.html`.
- The SRI Language Modeling Toolkit, available at `http://www.speech.sri.com/projects/srilm/`.

Which one to use depends on availability and experience. Note that language models can become very big. Even a compressed (.gz) file will take up more space on disc and take longer to load than a so-called language model dump file.

## 2.5 Scoring

A comprehensive scoring package is not part of Janus. Instead, the Ibis decoder can write hypotheses in CTM format, which can directly be processed with NIST's SCLite scoring package. Using Tcl, it is straightforward to write hypos in almost any format you might need.

Additionally, the Tcl-library implements an "align" function, which you can use to compute string edit distances for simple alignment problems. See `align.tcl` for details. This file also defines a set of lower-level functions, which you might find useful.

## 2.6 Compilation

Using the Makefile provided in the `src` directory, it is possible to compile Janus on Linux, Mac OS X, and Solaris. You can also set switches like `-DMTHREAD` for a multi-threaded version of Janus. Currently, two main targets are supported by this Makefile:

**janus_opt** the default version, contains everything to train and test

**janusNX_opt** janus_opt without X-Windows and readline support

Simply type `make` or `make janus_opt` to compile Janus. On SUNs, you might have to use `gmake` instead of `make`. To build debugging or profiling versions, replace `opt` by `dbg` or `prf`. For cleaning the object directory, simply type e.g. `make clean_opt` for the optimized code. This is especially neccessary, when using another main target, because the object files for both targets are taking place in the same directory. Different directories will be created for `opt`, `dbg` and `prf` version.

Depending on the exact system configuration you're using, Janus (on Unix, particularly Linux) depends on the following libraries:

**System Librarys** : `ld, m, X11` (until compiled without X11 support), `readline` (GNU), `termcap, ncurses, pthread`

**Tcl/Tk 8.0** or greater; if you use Tcl/Tk 8.4 or greater you should add `-DUSE_COMPAT_CONST` to your `CFlags`

**SPHERE** : `sp, util` from NIST's SPHERE library[2], if you want to read files in SPHERE format and you're using `-DSPHERE`

The exact libraries you'll need depend on your system and if you want to use static or dynamic linking. A complete description of compile-time options and optimizations is beyond the scope of this documentation, on Linux systems we're using the following switches for `gcc`: `-O3 -ffast-math -fomit-frame-pointer -march=pentium4`, you might also try `-mfpmath=sse -msse` on PIII systems.

The following configurations are well tested: On SOLARIS machines, we compile Janus with the SUN compiler WS5.0 and the GNU compiler gcc 2.95; on Linux we mainly use the Intel C++ compiler icc6.0 (7.0) and gcc 2.95 or greater (gcc 3.0, 3.2, 3.3). On Windows platforms, the Microsoft compiler VisualC++ is used. A work space file can be found in the `src/Windows` directory.

If you want to generate a version which does not include the three-pass decoder and the neural net code, do not include `$(SEARCHOBS)` during compilation and don't include `Search_Init()` in `src/main/janusInit.c` (use `-DIBIS -DNO_NET`). This is accomplished by using the `ibis_opt` target. For a version which does not rely on Tcl/ Tk, use the `ibisNTCL_opt` target. More information can be found in the `Makefile` available in the `src` directory.

## Defines

Some properties of the Ibis decoder can only be changed by altering `#define`s and `typedef`s in `src/ibis/slimits.h`. Changing these values will require you to re-compile Janus and will also make it impossible to re-use object dump-files, because the internal representation of data structures has changed.

The most frequently used settings are:

**LVX, LVX_MAX** sets the maximum language model vocabulary size to 65535 ($2^{16} - 1$) or $2^{32} - 1$.

**SVX, SVX_MAX** sets the maximum search vocabulary size to 65535 ($2^{16} - 1$) or $2^{32} - 1$. This setting has to be at least as big as the setting of `LVX, LVX_MAX`. It has some influence on runtime memory consumption.

[2] http://www.nist.gov/speech/tools/sphere_26atarZ.htm, be warned: the original version needs modifications to compile under modern Linuxes and contains bugs

## 2.7 Version history

The version of Janus you are running can be determined from the start-up message the `janus` binary displays:

```
# =======================================================================
#  ____  ____  _____ _
# |__  ||  _ \|_   _| | _     V5.0 P012 [Nov 11 2002 11:11:11]
#    | || |_| | | | | |/ |  ---------------------------------------------
#    | ||  _ <  | | |   <     University of Karlsruhe, Germany
#    | ||_| |_| |_| |_|\_|    Carnegie Mellon University, USA
#   _| | JANUS Recognition
#  \__/        Toolkit        (c) 1993-2002 Interactive Systems Labs
#
# =======================================================================
started janus: on i13pc234.10934, Fri Oct 25 10:03:53 CEST 2002
```

This means that this executable was compiled on November 11, 2002. The version is "Janus V5.0, patch-level 12". Some versions of Janus were "stamped" with an extra tag (e.g. "fame", "glory", ...), which will then also appear printed in this line. Also, CVS or SVN version information might be embedded in the start-up message. The last line of output, `started janus: ...` is generated in the file `.janusrc` and logs the start-up time of this process. The differences between different versions and patch-levels of janus are listed below:

**V5.2, P000** released on 2008-??-??

- added various feature enhancement techniques
- incorporates all code written for the RT07S Meeting eval
- added description of pre-processing methods to documentation
- added ICA

**V5.0, P014** released on 2004-09-24

- bugfixes over P013
- incorporates all code written for the RT04S Meeting eval
- added the description of adaptation methods to documentation
- some code for array (pre-)processing integrated
- changes in CFG implementation: support of additional grammar file formats (FSM, PFSG), support of weight definitions through JSGF, support for generating random terminal sequences
- support for reading LMs with unsorted n-gram sections, e.g. produced by the SRILM-Toolkit
- x86-64 code cleaning

**V5.0, P013** released on 2003-08-13

- support for training on Windows (bugfixes in IslSystem)
- cleaner interface to NGets
- incorporates all code written for the RT03 CTS eval
- major changes in grammar implementation
- discriminative training (MMIE)

- bugfixes (splitting of trees, interpolated LMs)
- changed glatComputeGamma and Confidence

**V5.0, P012** released on 2002-11-27

- redesigned filler words
- code-cleaning for windows

**V5.0, P011** released on 2002-10-10

**LingKS:** redesign of language model interface (Tcl-scripts have to be adapted, see ibisInit for a comparison of the two interfaces). Basically, a language model now is an object of type LingKS, while before the language model could be of different types (LModelNJD, MetaLM, PhraseLM, CFGSet). Now, a language model has a type-specific sub-object. The methods and configuration options change accordingly.

**V5.0, P010** released on 2002-02-27

**XCM:** option for left context dependency only

**STree:** convert search tree representation to general network structure and compress the network with the 'coarset partition' algorithm

**GLat:** changed lattice generation, support to write lattices in HTK format

**LTree:** redesigned ltreeFillCtx changed acoustic rescoring of lattices

**MetaLM:** more efficient interpolation

**PhraseLM:** opimized lct handling, added reading of map files

**CFG:** basic grammar support

**HMM:** training of full context dependent single phone words

**Codebooks/Distribs/Senones/Streams:** a couple of things

**V5.0, P009** released on 2002-01-07

**GLat:**
- changed handling of filler words in forward/backward pass
- added ignoreFTag option in glatAlign
- improved handling of dis-connected nodes in glatConnect
- added nodeN option in glatPrune

**PhraseLM:** fixed lct handling in ScoreArray function

**SMem/STree/SPass:** removed position mapper

**V5.0, P008** released on 2001-12-05

- Changed search space memory management
- Fixed trace function in stree
- Changes according to the intel compiler

**V5.0, P007** released on 2001-11-15
fixed final (?) problem with deletion of dictionary entries.

**V5.0, P006** released on 2001-11-14
fixed remaining problem with deletion of dictionary entries.

**V5.0, P005** released on 2001-11-07

- Increased data size for PHMMX in slimits.h

- Added configuration options for svmap and phraseLM
- Made praseLM relocatable

**V5.0, P004** released on 2001-11-06

- Support for arbitrary HMM-topologies
- Added one-state fast match module
- Support for streams in scoreA and mladaptS
- Deactivated LCT-checker in strace

**V5.0, P003** released on 2001-10-30

Bugfixed and some new features: removed a memory allocation bug in the semitied covariance code, which showed up under Linux. Also made the query of codebooks with distribution names working. Made some conversion problems if distribution and codebooks have different names. Made featureADC a bit more portable. Made deletion of words from the dictionary work. Added saving to disc of a single distribution or codebook into a distributionSet or CodebookSet.

**V5.0, P002** released on 2001-10-19

Update of the windows environment to the IBIS code.

**V5.0, P001** released on 2001-10-15

Established Ibis branch from former Janus main branch jtk-01-01-15-fms.

# Chapter 3

# The Janus User Interface

## 3.1 Tcl basics in 5 minutes

Tcl stands for 'tool command language' and is pronounced 'tickle.'

### Starting

You start tcl by typing `tcl` or `tclsh` in your Unix shell. Thus you enter an interactive mode within Tcl. You can leave with the tcl command `exit`. If you want to use the tcl tool kit (TclTk) you use `wish` instead of `tcl`.

```
> tcl
tcl> # this is a comment because the line starts with '#'
tcl> # now we define the variable text
tcl> set text "hello world"
tcl> puts $text
hello world
tcl> exit
>
```

### Variables

Variables in tcl can be defined with the command `set` and the value can be used with `$variable_name`. Arrays can be indexed with arbitrary names in (). Curly braces are used to separate variable names from following characters.

```
tcl> set name1 Hans
tcl> puts $name1
Hans
tcl> set name2 $name1
tcl> puts ${name2}_im_Glueck
Hans_im_Glueck
tcl> set data(name) Hans
tcl> set data(age)  35
tcl> set data(1,2)  something
tcl> set index name
tcl> puts $data($index)
Hans
```

## Commands, grouping and procedures

Commands and procedures are called with their name followed by arguments. Arguments are separated by spaces. They can be grouped together with ”” or . The difference is that variables within ”” will be replaced. ’;’ separates commands in one line.

```
tcl> set a 1
tcl> puts "$a + 1"
1 + 1
tcl> puts {$a + 1}
$a + 1
tcl> puts "{$a} + 1"
{1} + 1
tcl> set b 1; puts $b;     # bla bla
```

A command and arguments within [] will be executed and [command arg1 arg2 ..] will be replaced with the return value.

```
tcl> expr 1 + 2
3
tcl> puts "1 + 2 = [expr 1 + 2]"
1 + 2 = 3
```

The interpretation of $variable and [] can be switched off with
.

```
tcl> set a 999
tcl> puts "\[$a \$\]"
[999 $]
tcl> puts {[$a $]}
[$a $]
```

New commands or better procedures can be defined with the command `proc`.

```
tcl> proc add {a b} {return [expr $a + $b]}
tcl> add 1 2
3
```

Note that the procedure name ’add’, the variable list ’a b’ and the body of the function ’return [expr $a + $b]’ are the arguments of the command ’proc’. You can also use optional arguments with their default value.

```
tcl> proc printText {times {text "hello word"}} {
=>     for {set i 0} {$i<$times} {incr i} {
=>       puts $text
=>     }
=>     return $times
=>   }
tcl> printText 2
hello word
hello word
tcl> printText 1 "hello Monika"
hello Monika
```

Each procedure has a local scope for variables. But you can use the ’global’ command in a procedure to access global variables.

```
tcl> proc putsnames {} {global name1; puts $name1; puts $name2}
tcl> putsnames
can't read "name1": no such variable
tcl> set name1 Tanja
tcl> set name2 Petra
tcl> putsnames
Tanja
can't read "name2": no such variable
```

## Control flow

```
tcl> if {$i > 0} {puts "1"} else {puts "0"}
tcl> if {"$name" == "Tilo"} {
=>     #
=>     #do something here
=>     #
=>   }
tcl> for {set i 0} {$i < 10} {incr i} {puts $i}
tcl> foreach value {1 2 3 5} {puts stdout "$value"}
tcl> while {$i>0} {incr i -1}
tcl> switch $i {
=>     1         {puts "i = 1"}
=>     "hello"   {puts "hi"}
=>     default   {puts "?"}
=>   }
```

You can exit a loop with 'break' or 'continue' with the next iteration.

## Errors

With 'catch' errors can be trapped.

```
tcl> if [catch {expr 1.0 / $a} result ] {
=>      puts stderr $result
=>   } else {
=>      puts "1 / $a = $result"
=>   }
```

## File I/O

```
tcl>  set FP [open $fileName r]
tcl>  set found 0
tcl>  while {[gets $FP line] >= 0} {
=>        if {[string compare "ABC" $line] == 0} {set found 1; break}
             # found exactly "ABC"
=>        if ![string compare "XYZ" $line] {set found 2; break}
             # found exactly "XYZ"
=>        if [string match ABC*XYZ $line] {set found 3; break}
             # found "ABC..something..XYZ"
=>    }
tcl>  close $FPI

tcl>  set FP [open $fileName r]
tcl>  set first100bytes  [read $FP 100]
tcl>  set rest           [read $FP]
tcl>  close $FPI
```

### The string command

Strings are the basic data items in Tcl. The general syntax of the Tcl `string` command is

`string /operation stringvalue otherargs/`.

```
tcl> string length abc
3
tcl> string index abc 1
b
tcl> string range abcd 1 end
bcd
```

To compare two strings you can also use `==`. But that might not work as you wanted with strings containing digits because 1 equals 1.00 (but not in a string sense).

```
   if ![string compare $a $b] {puts "$a and $b differ"}
```

Use 'first' or 'last' to look for a substring. The return value is the index of the first character of the substring within the string.

```
tcl> string first abc xxxabcxxxabcxx
3
tcl> string last abc xxxabcxxxabcxxx
9
tcl> string last abc xxxxxx
-1
```

The 'string match' command uses the glob-style pattern matching like many UNIX shell commands do (Glob-style syntax):

***** Matches any number of any character.

**?** Matches any single character.

**[ ]** One of a set of characters like [a-z].

```
tcl> string match {a[0-9]bc?def\?ghi*} a5bcYdef?ghixxx
1

tcl> set a [string tolower abcXY]
abcxy
tcl> string toupper $a
ABCXY
tcl> string trim "  abc  "
abc
tcl> string trimright "xxabcxxxx" x
xxabc
tcl> string trimleft "  a  bc"
a  bc
```

Here comes a small example that finds the word with 'x' in a sentence.

```
tcl> set s {abc dexfgh ijklm}
tcl> string first x $s
6
tcl> set start  [string wordstart $s 6]  ;# start position
4
tcl> set end    [string wordend $s 6]    ;# position after word
10
tcl> string range $s $start [expr $end - 1]
dexfgh
```

## More commands dealing with strings

```
tcl> set a abc
tcl> append a def
abcdef

tcl> puts [format "%8s\t%8.4f" $a -12.7]
  abcdef        -12.7000
tcl> scan "distance 12.34m" "%s%f%c" what value unit
3
```

## Regular Expressions

Regular expression syntax. Matches any character.

***** Matches zero or more.

**?** Matches zero or one.

( ) Groups a sub-pattern.

— Alternation.

[ ] Set of characters like [a-z]. [0̂-9] means that numbers are excluded.

ˆ␣ Beginning of the string.

**$** End of string.

```
tcl> regexp {hello|Hello} Hello
1
tcl> regexp {[hH]ello} Hello
1
tcl> regexp {[0-9]\.([a-z])([a-wyz]*)} "xxx8.babcxxxxxx" match s1 s2
1
tcl> puts "$match $s1 $s2"
8.babc b abc

tcl> regsub {[0-9]\.([a-z])([a-wyz]*)} "xxx8.babcxxxxxx" {__\1__\2__&__}  var
tcl> puts $var
xxx__b__abc__8.babc__xxxxxx
```

## Lists

Tcl lists are just strings with a special interpretation. Separated by white space or grouped with braces or quotes.

```
tcl> set mylist "a b {c d}"
tcl> set mylist [list a b {c d}]         ;# same as above
tcl> foreach element $mylist {puts $element}
a
b
c d
```

Here several Tcl commands related to lists:

```
tcl> lindex $mylist 1         ;# note the index starts with 0
b
tcl> llength $mylist          ;# 'c d' is only one element
3

tcl> lappend mylist {g h}     ;# this time the list name 'mylist' is used
a b {c d} {g h}
tcl> lrange $mylist 2 end
{c d} {g h}
tcl> linsert $mylist 3 E x    ;# note that we don't give the list name here!
a b {c d} E x {g h}
tcl> set mylist [linsert $mylist 3 E x];# to change the list we have to use 'set'
a b {c d} E x {g h}

tcl> lsearch -exact $mylist E ;# other modes are the default '-glob' and '-regexp'
3
tcl> lreplace $mylist 3 5 e f {g h i}
a b {c d} e f {g h i}
tcl> lreplace $mylist 3 3     ;# delete element 3

tcl> lsort "-1.2 -1 -900 -90 1e-3 10"
-1 -1.2 -90 -900 10 1e-3
tcl> lsort -real "-1.2 -1 -900 -90 1e-3 10"
     # other flags are '-ascii','-integer','-increasing','-decreasing'
-900 -90 -1.2 -1 1e-3 10

tcl> list "a b" c
{a b} c
tcl> concat "a b" c
a b c

tcl> join "{} usr local bin" /
/usr/local/bin
tcl> split /usr/my-local/bin /-
{} usr my local bin
```

## Arrays

```
tcl> array exists a
0
tcl> set a(0) 0.12;   set a(1) 1.23;   set a(name) hello

tcl> array size a
3
tcl> array names a
0 name 1
tcl> array get a
0 0.12 name hello 1 1.23
```

The initialization could have been done with:

```
tcl> array set a "0 0.12 name hello 1 1.23"
tcl> array set b [array get a]     ;# Copy array b from a:
```

Other array commands are startsearch, nextelement, anymore, donesearch.

## 3.2 Janus Objects

JANUS was designed to be programmable. The programming language is Tcl/Tk, expanded by some object classes and their methods. Object classes are things like dictionaries, codebooks, but also the decoder itself is an object class. Every object class has its methods (operations that can be done with objects of that class). Objects can have subobjects and can be hierarchically organized. The object oriented programming paradign allows, at least in principle, to plug in and out objects as one wishes. Simply change the dictionary by assigning a new one, copy codebooks as easily as ”cb1 := cb2”, add distribution accumulators as easily as ”ds1.accu += ds2.accu”, etc.

### Create JANUS Objects

Objects are meant to hold data but also provide methods to manipulate that data. To define an object you have to specify its *type*. The convention is that type names start with capital letters and objects with small letters. You can define as many objects of one type as you like. To see what types exist just type (one of the few) JANUS command ’types’.[1]

```
% types
FlatFwd ModelArray DurationSet Word Cbcfg SampleSet DVector PTree HMM
Vocab FMatrix CodebookMapItem PTreeSet MLNorm Dscfg PathItemList
CodebookAccu HypoList DBaseIdx SampleSetClass SenoneTag Feature
PhonesSet DCovMatrix Phone FBMatrix Phones DistribAccu IMatrix
TopoSet SVector XWModel IArray DMatrix FVector StateGraph FCovMatrix
Duration PTreeNode LatNode Hypo Senone TreeFwd LDA Topo MLNormClass
Codebook Tags LDAClass FeatureSet Tree PhoneGraph CodebookMap Path
Search AModelSet RewriteSet
```

The list you get here depends on the version and compile options. You create an object when you enter a *type name* followed by the *name of the new object*. Some types require additional arguments, like subobjects. As an example we define an object (let’s call it ’ps’) of type PhonesSet. You can get a list of all objects you have defined with the command ’objects’. One object name can only be used once (also for different types) but you can ’destroy’ objects. ’destroy’ is a standard method of every object (s.b.).

```
% PhonesSet ps
ps
% PhonesSet ps
WARNING itf.c(0287)   Object ps already exists.
% PhonesSet ps2
ps2
% objects
ps ps2
```

[1] By the way ’%’ is the prompt of the JANUS shell. You can also use any Tcl and Tk commands.

summary: JANUS commands *types * list all object types *objects * list all objects defined by user

## Standard Methods

As soon as you have defined an object you can use its *name* followed by a *method* and arguments. The different object types have their own methods of course but at least a few are standard methods that exist for every object. These are

**type** gives the type of the object

**puts** print contents of the object

**configure** configure the object

**:** allow access to list element

**.** allow access to subobjects

**destroy** destroy object

To find out what other methods exist we enter eather the *type name* without any object name or the object name follwed by '-help'. To get more information about a specific method we enter the object name, the method and '-help'.

```
% ps -help

DESCRIPTION

A 'PhonesSet' object is a set of 'Phones' objects.

METHODS

puts               displays the contents of a set of phone-sets
add                add new phone-set to a set of phones-set
delete             delete phone-set(s) from a set of phone-sets
read               read a set of phone-sets from a file
write              write a set of phone-sets into a file
index              return index of named phone-set(s)
name               return the name of indexed phone-set(s)

% ps add -help
Options of 'add' are:
 < name>    name of list (string:"NULL")
 < phone*>  list of phones
% ps add VOWEL A E I O U
```

We just added the element 'VOWEL' to the PhonesSet object ps. To see the contents of the object we can use the method 'puts' or just the object name which is the same in most cases.

```
% ps puts
VOWEL
% ps
VOWEL
```

## Access to Elements and Subobjects

The standard methods ':' and '.' allow access to elements and subobjects respectively. *Elements* of an objects have the same kind of structure like the words of a dictionary or phone groups (like the 'VOWEL') in the PhonesSet. Nevertheless they can also be objects and most time they are. *Subobjects* are more unique, like the Phones of a dictionary as we will see immedeately. For these two methods and only for them you can omit the spaces between the object and also between the single argument which is the name of the element or the subobject. If you don't give a name you will obtain a list of the choices.[2]

```
% ps:
VOWEL
% ps:VOWEL
A E I O U
% ps type
PhonesSet
% ps:VOWEL type
Phones
```

Let's assume we have also defined a phone group 'PHONES' in the PhonesSet 'ps' that contains all the Phones of a dictionary. Then we can create a dictionary object that needs the name of a *Phones object* and of a *Tags object* as arguments. Both have to be created before.

```
% Tags ts
ts
% Dictionary d ps:PHONES ts
d
% d add DOG "D O G"
% d add DUCK "D U CK"
% d.
phones tags item(0..1) list
% d:
DOG DUCK
```

With 'd.phones' for example you have access to the object 'ps:PHONES'. Although there is a method for PhonesSet to delete elements like 'PHONES' you will get an error if you try that because it was locked as you defined the dictionary. This prevents objects from being deleted while they are used by other objects.

## Configuration

Sometimes it might be necessary to configure *objects* or *object types*. You can get all configure items, get a specific one or set one or more items. In the latter case only if they are writable.

```
% ps configure
{-useN 1} {-commentChar {;}} {-itemN 2} {-blkSize 20}
% ps configure -commentChar
{;}
% ps configure -commentChar #
```

[2] In case of ':' you again get a list of the elements like with 'puts' or just the object name with no method.

Note: The old comment sign ';' was protected with curly braces  because it is the command separator in Tcl.

Can you explain the following line and its return value.

```
% ps configure -commentChar ;
#
```

## 3.3 The Janus Library: “tcl-lib” and “gui-tcl”

The Tcl-library is a set of procedures the user can invoke and which provide a number of “convenience” functions. The scripts in the Janus Scripts Collection (`~/janus/scripts`, see chapter `scripts`) use the Tcl-library extensively. The Tcl-library can be found in `~/janus/tcl-lib` and `~/janus/gui-tcl`. To auto-load the functions, the Tcl-variable `auto_path` has to be set correctly, i.e. to the value of these two directories. Also, a file `tclIndex` has to exist in these directories. You should not need to worry, if you follow the standard install instructions, otherwise refer to any Tcl manual for a description of the auto-loading mechanism. The functions available in the tcl-lib are described in chapter `lib`.

## 3.4 The Janus Scripts Collection

The directory `~/janus/scripts` contains a number of scripts, which we normally use to train and test systems. These scripts are often modified and copied in a system directory for documentation purposes.

If you have access to an example system (i.e. IslData, IslSystem), we suggest that you have a look at it to see how data and scripts are typically organized in a JRTk project. Usually, the structure of a project looks as follows (this project would be called the “M1” system):

```
M1/
|
+--master.log
|
+--Log/
|  '--makeCI.log
|
+--desc/
|  +--desc.tcl
|  +--codebookSet
|  +--distribSet
|  +--distribTree
|  +--featAccess
|  +--featDesc
|  +--phonesSet
|  +--tags
|  +--tmSet
|  +--topoSet
|  '--topoTree
|
+--train/
|  +--ldaM1.bmat
|  +--ldaM1.counts
|  +--convList
|  |
|  +--Log/
|  |  +--lda.log
|  |  +--samples.log
|  |  +--kmeans.log
|  |  '--train.log
|  |
|  +--Accus/
|  |  +- 1.cba.gz
|  |  '--1.dsa.gz
|  |
|  '--Weights
|     +--0.cbs.gz
|     '--0.dss.gz
|
+--test/
   +--convList
   |
   +--Log/
   |  '--test.log
   |
   '--hypos/
      '--H_kottan_z26_p0_LV.ctm
```

Typically, a system directory (here: "M1") contains a number of sub-directories, each for different phases (label writing, cepstral mean computation, model training ("train"), polyphone training/ clustering, testing ("test"), ...). Each directory then contains the data resulting from this step and log-files.

The scripts who perform the operations can be left under `janus/scripts`, only `desc.tcl` is a configuration file specific to this project and is therefore copied into the project directory along with the other description files.

### 3.4.1 Available Scripts

The following scripts are available in the Janus Scripts Collection (in the order in which they are usually called):

`genDBase.tcl` This script can be used to create a database, which is necessary for all further steps. Look at the resulting database files (they are called `db-{spk|utt}.{idx|dat}` to see what information can and needs to be defined in the database.

Depending on your needs and the format, in which you have the information available, you will need to modify this script to suit your needs.

`makeCI.tcl` Creates the following description files for a context-independent (CI) system:

- codebookSet
- distribSet
- distribTree

You'll need to have all the other description files in place, namely the phonesSet. If you want to use a different architecture (i.e. semi-tied, or non-tri-state architectures), you can edit this file according to your needs.

`means.tcl` This script will create the cepstral means needed for the standard preprocessing of the Janus-based recognizers.

`lda.tcl` This script computes an LDA matrix, used for the standard pre-processing.

`samples.tcl` This script extracts samples for further clustering with `kmeans.tcl`.

`kmeans.tcl` Performs KMeans clustering on data extracted with `samples.tcl`.

`train.tcl` Performs EM-training on initial codebooks from `kmeans.tcl`. Can be used for training of a context-independent (CI), a context-dependent (CD), or a polyphone (PT) system. Normally, we do label training although it is also possible to do viterbi- or forward-backward-training by replacing `viterbiUtterance` by for example `viterbiUtterance`.

`makePT.tcl` Creates a polyphone (PT) system from the CI description files.

`cluster.tcl` Clusters the contexts from PT training.

`split.tcl` Creates the context-dependent (CD) models after PT training, creates the following CD description files (with $N > 0$):

- codebookSet.N.gz
- distribSet.Np.gz
- distribTree.Np.gz

`createBBI.tcl` Creates a BBI (bucket-box intersection) tree for a codebook.

`test.tcl` Tests a system.

`score.tcl` Scores a system, i.e. computes word error rates.

`ana_time.tcl` Allows to measure the CPU-time spent in pre-defined sections of a script. You can find more details about timing analysis and how to use this script in section 6.1.6.

`labels.tcl` Writes new labels with an existing system. Can also be used to bootstrap a new system using the acoustic models from another system.

An example `desc.tcl` file is also included in the script collection. All "working" scripts source `../desc/desc.tcl` and load the settings (paths, ...) from there, although these can be overridden at the command line or in the script themself. `janusrc` is an example configuration file for janus, which is best adapted and copied into your home directory as `.janusrc`.

### 3.4.2 Working with master.tcl

We assume you have a system directory setup correctly, including pre-computed time-alignments ("labels"). When working with the example system "IslSystem", you have a `desc` directory which contains an appropriate `desc.tcl` file. In the "system home directory" ("M1" in the above example), you can now enter

```
janus <janus>/scripts/master.tcl -do init means lda samples kmeans train
```

and the master script will create a context-independent (CI) system in the `M1/train` directory. <`janus`> refers to your Janus installation directory.[3] You'll find logfiles in your system's `Log` subdirectory. The following steps were performed, calling the following scripts:

init (`makeCI.tcl`) to create the codebookSet, distribSet, distribTree definition files for the CI system

means (`means.tcl`) to compute the cepstral means for this preprocessing. This can be re-used for a CD-system

[3] Usually this will be `~/janus`.

lda (lda.tcl) to compute the LDA (Linear Discriminant Analysis) matrix for this system.

samples (samples.tcl) to extract samples for the CI models

kmeans (kmeans.tcl) to create initial codebooks from the samples, written into Weights/0. Once this step is completed, you can remove the data subdirectory.

train (train.tcl) to perform several iterations of EM-training on the initial codebooks. At the end of this step, you can remove the contents of the Accus subdirectory as well as intermediate codebooks in Weights, to save space.

master.tcl will show you the command lines it executes, if you want to parallelize your training, you can copy the output lines `exec janus lda.tcl ...` (omitting the `exec` and changing the log file name) and run the same script on several machines.

To run the polyphone training, enter

```
janus <janus>/scripts/master.tcl -do makePT trainPT cluster split
```

This will create the description files for a context-dependent system.

To run the training for the context-dependent system, enter

```
janus <janus>/scripts/master.tcl -do lda samples kmeans train test score
```

assuming that you have created a new directory and set up the paths for codebookSetParam and distribSetParam accordingly. To create initial time alignments for a new system, edit the description file (probably you'll have to change most of the files usually in the "desc" directory to match your old system and your desired new setup) and execute:

```
janus <janus>/scripts/master.tcl -do labels
```

If you type

```
janus <janus>/scripts/master.tcl -h
```

you'll get a list of all command line options for `master.tcl`.

### 3.4.3 Extra scripts

The `scripts` directory also contains a number of scripts, which can not (currently) be called through master.tcl. They can however serve as example scripts, which can be adapted to specific problems.

`map.tcl` An example script to perform *MAP* adaptation.

`mllr.tcl` An example script to perform *MLLR* adaptation.

# Chapter 4

# Pre-Processing with JRTk

Janus can use just about any conceivable recognizer front-end. Most “standard” ways of doing pre-processing, such as mel frequency cepstral coefficients (MFCC)s or perceptual linear prediction, are almost certainly already implemented in the FeatureSet. FIR-Filters can be applied to features with the filter method, and so on. Have a look at the FeatureSet and example featDescs to see what’s already available. In the remainder of this chapter we will give more details about some of the features which are available in the FeatureSet module and might require a more detailed explanation.

Various sample scripts including MFCC and warped-minimum variance distortionless response (MVDR) front-ends as well as reverberation compensation by multi-step linear prediction (MSLP), non-stationary noise compensation by particle filter (PF)s and joint compensation of both distortions can be found in the scripts directory.

## 4.1 Spectral Estimation

Spectral analysis is a fundamental part of speech feature extraction for automatic recognition and many other speech processing algorithms. Janus contains a broad variety of spectral estimation techniques to adjust for spectral resolution, variance of the estimated spectra, and to model the frequency response function of the vocal tract during voiced speech. In the following example the Fourier spectrum <FFT>, the warped MVDR spectral envelope <MVDR> —mel frequency for a 16 kHz signal— and the scaling of the spectral envelope <sMVDR> is demonstrated:

```
set order 30
set windowsize 16ms

$fes spectrum  FFT    ADC         $windowsize
$fes adc2spec  ADC                $windowsize -win hamming -adc SPADC
$fes specest   MVDR   SPADC       $order -type MVDR \
                                         -lpmethod warp -warp 0.4595
$fes specadj   sMVDR  MVDR   FFT  -smooth 2
```

NOTE: For a 8 kHz signal the warp factor has to be replaced by 0.3624.

Different spectral estimation techniques within the Janus framework are compared and explained in [19, 17, 20].

## 4.2 VTLN

Vocal track length normalization (VTLN) can be applied either in the linear or in the warped (mel) domain. The domain mainly depends on the used spectral estimation method as described in section 4.1. While the implementation in the linear domain is not able to reduce the number of spectral bins (can for example be implemented by the mel filterbank), the implementation in the warped domain can provide a reduced number of spectral bins. The two different implementations can be called as follows:

- In the linear domain

  ```
  $fes   VTLN <TO> <FROM> $WARP -mod lin -edge 0.8
  ```

- In the warped (mel) domain

  ```
  set warp [expr round(200-$WARP*100)/100.0]

  if { $warp < 0.75 } { set warp 0.75 }
  if { $warp == 1.0 } { set warp 1 }
  if { $warp > 1.35 } { set warp 1.35 }

  if {[llength [objects FBMatrix VTLN${warp}]] != 1} {
      writeLog stderr "LOAD filterbank 16 kHz, 16 ms, 129 bins, \
          ${warp} warp"
      source ${path}/filterbanks/Filterbanks16kHz16ms129bins/ \
          VTLN.filterbank.${warp}
  }

  $fes   filterbank <TO> <FROM> VTLN${warp}
  ```

  NOTE: Different pre-calculated filterbank can be loaded for 8 and 16 kHz.

## 4.3 Feature Enhancement

To cope well with the non-stationary behavior of additive and convolutive distortions Janus contains different feature enhancement techniques which can be used in addition to other adaptation methods as described in section 5.3. In the remainder of this section we present a generic compensation framework to jointly compensate for additive and reverberant distortion. The framework can be easily adjusted to compensate for additive or reverberant distortion only as will be discussed.

Different feature enhancement techniques within the Janus framework are compared and explained in [18, 20].

In **featAccess.tcl** we read the distorted wave file, adjust the segment length, estimate late reflections <fADC> and subtract the energy of the late reflections <fMVDR> to get a dereverberant frame-by-frame speech estimate <subMVDR>:

```
# delay in seconds
set delay 0.06
set size 1000

# delermine var. automatically
set delayBins  [expr round(16000 * $delay)]
set delayFrames [expr round(100 * $delay)]
```

```
$fes readADC ADC16 $adcFile
$fes cut ADC ADC16 [expr $arg(FROM) - $delay -2.0]s $arg(TO)s

$fes multisteplp FILTER ADC -delay $delayBins -order $size
set FILTER "[$fes:FILTER.data]"
$fes filter fADC ADC "0 $FILTER"

# estimate spectra ADC -> sMVDR and fADC -> fMVDR

# adjust frames accordingly
$fes cut sMVDR sMVDR $delayFrames end
set frames [$fes:fMVDR configure -frameN]
$fes cut fMVDR fMVDR 0 [expr $frames-1-$delayFrames]

# dereverberant speech features
$fes specsub subMVDR sMVDR fMVDR -a 1.0 -b 0.1
```

In **featDesc.tcl** we initialize the particle filter as described in more detail in SpeechGMM.tcl, learn a GMM for noise as well as fnoise, and apply the particle filter to get the cleaned estimate <SPEC_cleaned> from the noisy frames <SPEC>:

```
if { ![info exists AMINIT] } {
    writeLog stderr "=====> INIT Particle Filter <========"
    source SpeechGMM.tcl
    set AMINIT change
    initAM $SID $fes $AM $WARP
    writeLog stderr "===================================="
}

if { $USEPF > 0.5 && [file exists ${spectra}/$arg(UTT)_cleaned.smp] } {
  $fes FMatrix SPEC_cleaned
  $fes:SPEC_cleaned.data bload ${spectra}/$arg(UTT)_cleaned.smp
  puts "Loaded spectrum: ${spectra}/$arg(UTT)_cleaned.smp"
} else {

  # reduce spectral dimension sMVDR -> SPEC and subMVDR -> diffSPEC
  $fes specsublog diffSPEC SPEC subSPEC

  if { $USEPF > 0.5 } {
    # train new noise GMM ---------------------------------------------
    $fes lin SILENCE SPEECH -1 1
    $fes cut NSPEC SPEC 0 last -select SILENCE
    set noiseN [$fes:NSPEC configure -frameN]
    set inputN [$fes:SPEC configure -frameN]
    writeLog stderr "INFO noise frames: $noiseN of $inputN detected"

    if {$noiseN > 9} {
        trainCB distribSet$AM codebookSet$AM noise
    } else {
        writeLog stderr "INFO < 10 noise frames, noise GMM not updated"
    }

    # shift means of codebook (do not use for additive compensation only)
    trainCB distribSet$AM codebookSet$AM fnoise
    subtractCB distribSet$AM codebookSet$AM noise fnoise

    # ------------------------------------------------------------------
    # use Particle Filter
    # ------------------------------------------------------------------
    $fes particlefilter SPEC_cleaned SPEC distribSet$AM \
      -variance PREDICTVAR                            \
      -refresh 1E-40                                  \
      -nio 0.0                                        \
      -ARsmoothing 1                                  \
      -type sia                                       \
      -init 0                                         \
      -delayspec diffSPEC

    # save spectra -----------------------------------------------------
    $fes:SPEC_cleaned.data bsave ${spectra}/$arg(UTT)_cleaned.smp
  }
}

# additional processing

# cut final feature length
$fes cut LDA LDA 200 last
```

NOTE: If we are interested in compensating for additive distortions only we can remove "-delayspec diffSPEC" from the PF setting. If init is set to 1 the PF is reinitialized: new samples are drawn from the noise GMM and the AR matrix is set to diagonal. Reinitialization is necessary if an environment change is expected, e.g. for a new recording.

In **SpeechGMM.tcl** necessary procedures are defined which are called by featDesc.tcl:

```
# load a codebook containing a clean speech GMM
set ${AM}(codebookSetDesc)  ${pathPFAM}/final.cbsDesc.gz
set ${AM}(codebookSetParam) ${pathPFAM}/final.cbs.gz
set ${AM}(distribSetDesc)   ${pathPFAM}/final.dssDesc.gz
set ${AM}(distribSetParam)  ${pathPFAM}/final.dss.gz

# ----------------------------------------
#  procedures
# ----------------------------------------
proc initAM {SID fes AM warp} {
    codebookSetInit $AM  -featureSet featureSet$SID
    distribSetInit  $AM

    # add noise and fnoise model
    codebookSet$AM add noise NSPEC 1 20 DIAGONAL
    codebookSet$AM:noise createAccu
    distribSet$AM add noise noise

    codebookSet$AM add fnoise diffSPEC 1 20 DIAGONAL
    codebookSet$AM:fnoise createAccu
    distribSet$AM add fnoise fnoise

    if { [llength [objects FMatrix $fes:PREDICTVAR]] != 1} {
        $fes FMatrix PREDICTVAR
        $fes:PREDICTVAR.data := "10 10 10 10 10 10 10 10 10 10 10 10 \
                             10 10 10 10 10 10 10 10 0.001 0.001"
    }
}

# procedure to train a noise codebook
proc trainCB {dss cbs class} {
    set fe [$cbs.featureSet name [$cbs:$class configure -featX]]
    set frameN [$cbs.featureSet : $fe configure -frameN]
    $dss clearAccus
    $cbs clearAccus
    for {set i 0} {$i < $frameN} {incr i} {
        $dss accuFrame $class $i
    }
    $dss update
    puts "trained new $class model"
}

# subtract noise codebooks
proc subtractCB {dss cbs class1 class2} {
    set m1 [lindex [$cbs:$class1.mat] 0]
    set m2 [lindex [$cbs:$class2.mat] 0]

    set count 0
    set mean_values "{"
    foreach m $m1 {
        set temp [expr pow(10.0,0.1*[lindex $m1 $count])
            -pow(10.0,0.1*[lindex $m2 $count])]
        if { $temp < 10.0 } { set temp 10.0 }
        set temp [expr 10.0*log10($temp)]
        incr count
        set mean_values "$mean_values $temp"
    }
    set mean_values "$mean_values }"
    $cbs:$class1 set $mean_values
}
```

# Chapter 5

# Training with JRTk

## 5.1 Basic Training

“Basic training” refers to the `Training` of a complete context-dependent (CD) system. The Tcl-scripts residing in the `scripts` subdirectory of the JRTk distribution, the so-called “Janus Scripts Collection”, can be studied and used as a basis for experiments. In this section, whenever a Tcl-script is referred to, it can be found in this directory. You can copy these scripts to your systems directory and use them on their own, or you can call them through the script `master.tcl`. The Janus Scripts Collection in turn uses the procedures defined in the Tcl-library (`janus/tcl-lib` and `janus/gui-tcl`), which are described in section 9. Using `master.tcl` it is possible to easily train different systems. Other, more complex training schemes are however possible, see 5.2.

The basic training scheme (possible using `master.tcl`), looks as follows:

1. Create various description files.

   This is usually done by manually changing existing files (“desc.tcl”) to your needs. Additionally, you can use the scripts

   **genDBase.tcl** to create a new database from free-format information. A Janus database holds all the information related to a specific task, i.e. the transcriptions for an utterance, the appropriate audio file, the utterances for a speaker ...

   **makeCI.tcl** to create the codebook and distribution descriptions for a CI system from information supplied

   **makePT.tcl** to create the description files for the polyphone training (PT)

   If you want to use pre-compute cepstral means during your pre-processing, look at “means.tcl”.

2. Build and train a context-independent system.

   This is done by calling lda.tcl, samples.tcl, kmeans.tcl, and train.tcl in that order.

3. Cluster a context-independent system, i.e. do “polyphone-training”.

   Use makePT.tcl, train.tcl, cluster.tcl

4. Build and train a context-dependent system using the results form the polyphone-training

   Using split.tcl you can create new description files (for codebooks and distributions) using the results from a polyphone training. The remaining steps are the same as for CI training: lda.tcl, samples.tcl, kmeans.tcl, and train.tcl

5. More: build a BBI, write labels or test a system.

   BBI (Bucket-Box-Intersection) is a speed-up algorithm. Look at `createBBI.tcl` to see how a BBI tree is computed for an existing codebook. However, you do not need this step, if you don't want to speed up your system, but `test.tcl` can read a bbi tree during testing. `score.tcl` demonstrates how to score the results of a test run. Labels can be written with the example `labels.tcl` file.

This section will first focus on the training scheme, and the concepts behind the JRTk training environment. Step-by-step instructions for training a new system follow in sub-section 5.1.5, although the exact arguments to use for `master.tcl` and the example system are described in the documentation for IslSystem.

If you want to write labels with an existing system in order to bootstrap a new system, go to sub-section 5.1.8.

### 5.1.1 Description Files

No matter whether you train a context independent or dependent system, you need a few description files to define your front-end, size and number of acoustic models and so on. The system description file `desc.tcl`, which is usually created by hand, plays a central role here. The file `desc.tcl` from the example system "ISLci" or the `scripts/desc.tcl` file might serve as a template for you. This file provides pointers to the description files for each module. Typically you need to provide the following information:

1. Phonology : `phonesSet`, `tags`
   defines a set of phones, phone-classes, tags (e.g. word boundaries)

2. Front-End : `featDesc`, `featAccess`
   access to the audio data, definition of the preprocessing steps

3. Codebooks : `codebookSet`
   defines a set of Gaussian mixture densities, link to the underlying feature space

4. Distributions : `distribSet` defines a set of mixture weights, link to the underlying codebooks
   The mixture weights together with the codebooks define probability density functions (pdf). A fully continuous system is obtained by a one by one mapping of codebooks to distributions.

5. Polyphone Tree : `distribTree`
   context decision tree, attach pdfs to HMM states with a given phonetic or dynamic context (modalities). Even for context independent systems, you will need to define such a tree.

6. HMM : `topoSet`, `topoTree`, `tmSet`
   defines HMM topologies and transition models

7. Pronunciation Dictionary `dictionary`

8. Database
   Typically 2-level, provides speaker- and utterance-specific information; `scripts/genDBase.tcl` is an example script which creates a DBase from information available in other formats. Usually, a "speaker database" contains at least a list of all utterances pertaining to this speaker. The "utterance database" then contains, for every utterance, the speaker, the transcription, the gender, ...
   It's easy to build a database using the provided methods and then save it in the Janus DBase file format.

### 5.1.2 Module Initialization

To run a training, you first have to initialize all modules needed to create a training environment. Given some inital acoustic models (e.g. created by the k-means algorithm), a database, and a suitable system description, the following lines will create a training environment under the system ID 'X3'. The module initialization functions will read all relevant parameters from the system description, read from `../desc/desc.tcl`. Optional arguments might be used to overwrite these variables.

```
source ../desc/desc.tcl

phonesSetInit   X3
tagsInit        X3
featureSetInit  X3 -lda ldaX3.bmat
codebookSetInit X3 -param Weights/0.cbs.gz
distribSetInit  X3 -param Weights/0.dss.gz
distribTreeInit X3
senoneSetInit   X3  distribStreamX3
topoSetInit     X3
ttreeInit       X3
dictInit        X3
trainInit       X3
dbaseInit       X3 dbaseSWB
```

Have a look at the scripts in the `scripts` directory, to see how this initialization is done.

### 5.1.3 General Training Procedure

Now, if all modules are initialized, we can start a training experiment. There are basically two phases. In phase 1, the statistics for all training speaker will be accumulated. In phase 2, the acccumulated statistics will be used to find a ML estimation of the model parameters. Phase 1 can be parallelized, so you can use a number of machines to speed up the training. Each client job dumps partial accumulators which will be read by the server process, which will then estimate new models. The process can be repeated for several iterations.

The following procedures are used frequently during standard training:

- *doParallel*
  create semaphore files and synchronize the client jobs
- *fgets* and *foreachSegment*
  loop over all training data, fgets uses a file locking mechanism to read the speaker from the conversation list
- *viterbiUtterance* and *senoneSet accu path*
  do the preprocessing (evaluate `FeatureSet`), build a `HMM` using the training transcription from the `DBase`, computes a forced alignment (stored in `Path`), and accumulate the statistics in `SenoneSet` using the state probabilities
- *senoneSet update*
  read the statistics from the clients and do the parameter update in `SenoneSet`, the default configuration is to do a Maximum-Likelihood update.

```
codebookSetX3 createAccus
distribSetX3  createAccus
doParallel {
  while {[fgets $convLst spk] >= 0} {
    foreachSegment utt uttDB $spk {
      viterbiUtterance X3 $spk $utt
      senoneSetX3 accu pathX3
    }
  }
  codebookSetX3 saveAccus Accus/clientID.cba
  distribSetX3  saveAccus Accus/clientID.dsa
} {
  codebookSetX3 clearAccus
  distribSetX3  clearAccus
  foreach file [glob Accus/*cba] {codebookSetX3 readAccus $file}
  foreach file [glob Accus/*dsa] {distribSetX3  readAccus $file}
  senoneSetX3 update
  codebookSetX3 save Weights/new.cbs.gz
  distribSetX3  save Weights/new.dss.gz
} {} {} {}
```

### 5.1.4 Forced Alignments

Besides the viterbi algorithm, the full forward-backward algorithm might be used to accumulate the training statistics. JANUS provides the Path object to compute and maintain state alignments. By using precomputed alignments (called labels), the training procedure can be speed up drastically, since the viterbi or forward-backward based alignments are computed only once and not in each training iteration.

1. labelUtterance
   training using precomputed aligments
2. viterbiUtterance
   compute alignment using the Viterbi algorithm
3. fwdBwdUtterance
   compute alignment using the forward-backward algorithm

The Tcl-Library provides functions to generate forced aligments which might be used in a later training experiment using the *labelUtterance* scheme. Addionally, you can also use a method called “label-boosting” to generate speaker dependent alignments by using MLLR transformed acoustic models. This method can be seen as an efficient variant of speaker adaptive training.

1. labelsWrite
   compute speaker independent viterbi aligments for a list of speakers
2. labelsMLAdaptWrite
   compute speaker dependent viterbi aligments for a list of speakers; this needs a MLAdapt object and allocated accumulators for the codebooks to compute MLLR transforms.

If you want to bootstrap a new system, you usually write labels with an existing system (for example with one in a different language, with different acoustic conditions but the same topology), at least to create initial codebooks using samples.tcl and kmeans.tcl. You can then replace labelUtterance in “train.tcl” with viterbiUtterance and train your system without labels, because these will be of poor quality.

### 5.1.5 Train a context-independent system

This is the first step in training a new system. We assume you have the following ready:

- Dictionary and PhonesSet
- Labels (even if they stem from a bad system)
- Database, speaker list
- FeatureSet description and access files
- Tags, Transition Models, Topology Set, Topology Tree

You can now create a new directory, where you want to create the system in, let's assume it's called `M1`. Create a subdirectory `desc` and copy the template file `desc.tcl` in it. Edit it according to your needs, the `desc` directory usually also holds the files **`devTrain, featAccess, featDesc*, phonesSet, tags, tmSet, topoSet,`** and **`ttree`**.

If you don't yet have description files for codebooks and distributions, you can create them with "makeCI.tcl". If you need to pre-compute vectors for cepstral mean subtraction, "means.tcl" can do that for you. If you want to write labels (time-alignments) with another existing system, look at 5.1.8 first.

The first real step during acoustic training is the computation of an LDA matrix using lda.tcl. Although not strictly necessary, most Janus systems use an LDA during preprocessing. Also, calling "lda.tcl" extracts the number of occurences for every codebook in the file "lda$SID.counts". This file is read by "samples.tcl" in the next steps to extract an evenly distributed number of example vectors, which are then combined into an initial codebook by "kmeans.tcl". The actual EM training is then performed by "train.tcl". Typically, the size of the (gzipped) codebooks increases with every iteration (a factor of 2 between 0i and 1i, less afterwards) and the counts you can find with "dss:ds configure -count" should be equivalent to those you find in the counts file produced by lda.tcl.

### 5.1.6 Polyphone training

You'll need a completed CI training for this step. In the standard setup, we suggest that you run the polyphone training in the same system directory as the CI-training, but in a "pt" subdirectory (instead of "train").

The first step, makePT, creates the necessary description file for polyphone training: keeping the CI codebookSet, we create separate distributions for every polyphone context (distribTree.PT, distribSet.PT). Usually, there will be several millions of them. Then, a few iterations of EM training will be performed. The thus trained CD distributions will then be clustered according to an entropy criterion. Finally, you can create a codebook of a given size by taking the "N" most important contexts and creating separate codebooks and distributions for them (split.tcl).

### 5.1.7 Train a context-dependent system

Using the output from the polyphone training, e.g. the files codebookSet.N.gz, distribSet.Np.gz, and distribTree.Np.gz which were created by split.tcl[1], you can train a full context-dependent system. You can call the same scripts as in the CI case, but we suggest you create a new directory for the CD training.

[1] "N" refers to the desired size of the CD-codebook, e.g. 4000.

### 5.1.8 Write labels

You can write labels with any existing system. Usually you set up your system description files so that they match the system you want to build (database, dictionary, topology, ...). The only information you take from an “old” system are the acoustic models (codebooks). Therefore, the featDesc (feature description file), which describes how to preprocess the input data (ADCs) to make it compatible with the codebook, has to be adapted to match the old codebook and the new data, on which you write labels on. If the phones and codebooks don’t match between the old and new system, you can load both codebooks and copy them as we do here:

```
# We hope it's ok to load these (old) codebooks/ distribs
printDo [CodebookSet cbs featureSet$SID] read otherCodebookSet.desc
printDo [DistribSet  dss cbs]            read otherDistribSet.desc
printDo cbs load otherCodebookSet.param
printDo dss load otherDistribSet.param

# Create the new codebooks/ distribs
codebookSetInit $SID
distribSetInit  $SID

# Read the set, copy the codebooks/ distribs
set fp [open rewriteRules r]
while {[gets $fp line] != -1} {
    if {[regexp "^;" $line]} { continue }
    set from [lindex $line 0]; set to [lindex $line 1]
    puts stderr "    ReWriting $from -> $to"
    catch { codebookSet$SID:$to := cbs:$from }
    catch { distribSet$SID:$to  := dss:$from }
}
close $fp
```

The file “rewriteRules” might look like that:

```
; -------------------------------------------------------
;  Name            : rewriteSet
;  Type            : RewriteSet
;  Number of Items : n
;  Date            : Thu Jul 11 14:59:49 2002
; -------------------------------------------------------
AA-b    A-b
AA-e    A-e
AA-m    A-m
AE-b    AEH-b
AE-e    AEH-e
AE-m    AEH-m
AH-b    AH-b
AH-e    AH-e
AH-m    AH-m
AY-b    AI-b
AY-e    AI-e
AY-m    AI-m
AX-b    AU-b
AX-e    AU-e
AX-m    AU-m
...
```

This means that, e.g. the codebook “AX-m” of the new system (this is a context-independent system) is to be modeled by the old “AU-m”.

## 5.2 Advanced Training

In this section, we assume that you already have some experience with the JANUS object interface and the Tcl-Library. To run some more advanced experiments you will probably use funtions from the library directly without making use of the script collection as it was the case in the previous section.

### 5.2.1 Flexible Transcriptions

Given a transcription for a utterance, a corresponding HMM can be build, e.g.

```
% hmmX3 make "OH I SEE UH-HUH"
% hmmX3.phoneGraph puts
% OW AY S IY AH HH AH
```

However, if you have to deal with conversational speech, your training transcriptions might be not accurate, or background noises occur. To deal with such effects, you can insert optional words into the HMM, skip certain words, or even allow alternative words or pronunciations. By running the Viterbi algorithm, the best path will be computed according to the flexible transcription network. Flexible transcriptions can be computed via the TextGraph object. The following lines will create a HMM with an optional *NOISE* between each regular word, allowing alternative words *SEE / SAW*, skipping *UH-HUH* optionally, and allowing pronunciation variants.

```
% Textgraph textGraphX3
% textGraphX3 make "OH I {SEE/SAW} {UH-HUH/@}" -optWord NOISE
% array set  HMM [textGraphX3]
% set words $HMM(STATES)
% set trans $HMM(TRANS)
% set init  $HMM(INIT)
% hmmX3 make $words -trans $trans -init $init -variants 1
```

### 5.2.2 Vocal Tract Length Normalization

VTLN is a known technique to compensate variations across speaker by warping the frequencies. There are several ways to train VTLN models in JANUS. In the following, we describe a Maximum-Likelihood based variant. The object FeatureSet has a method *VTLN* to transform the short-term power-spectrum features. Given a certain *warpfactor*, the function call in your feature description may look like:

```
$fes VTLN  WFFT  FFT $warpfactor -mod lin -edge 0.8
```

To train VTLN acoustic models, the Tcl-Library provides functions to estimate warpfactors based on viterbi alignents. Assuming you have a basic system with a VTLN-capable front-end, the following lines will estimate a warpfactor for each training speaker.

```
vtlnInit X3
while {[fgets $convLst spk] >= 0} {
  set w [findViterbiWarp X3 $spk -warp 1.0 -window 8 -delta 0.02]
  puts "VTLN-Estimation for speaker $spk: $w"
}
```

It's straightforward to integrate the VTLN estimation with the standard training procedure. Instead of using findViterbiWarp, there is also a label based variant findLabelWarp. If there are warpfactors already available and you don't want to reestimate the factors, you can simply just load these warpfactors from the file by given an argument to vtlnInit and train with fixed warpfactors. The file should contain two words per line, the first one being a speaker-id, the second one being the warp-factor. To avoid common problems with training VTLN models, please note the following points:

1. voiced phones
   The VTLN estimation in the functions findViterbiWarp and findLabelWarp rely only a certain class of phones. The default configuration use *voiced* phones. To provide this information, you need to specify a class *voiced* in your PhonesSet.
2. Cepstral Mean Substraction
   If you use speaker based cepstral mean and variance normalization, the means and variances depend on the warpfactor should therefore be jointly estimated.

### 5.2.3 Model space based Speaker Adaptive Training

Similar to VTLN, the goal of SAT is to compensate speaker variations during training. SAT uses linear transforms of the acoustic models to explicitely model variations across speakers. Since the computational and memory resources needed to train SAT model are much higher than during standard training, we start with initial models and refine them by SAT. First, we need to create a MLAdapt object to estimate MLLR transforms.

```
set mode      2     ; # use full transforms
set minCount  2500  ; # minimum threshold to update regresion class
set depth     7     ; # depth of regression tree

codebookSetX3 createAccus
distribSetX3  createAccus

MLAdapt mlaX3 codebookSetX3 -mode $mode -bmem 1
foreach cb [codebookSetX3:] { mlaX3 add $cb }
mlaX3 cluster : [mlaX3 cluster -depth $depth]"
```

Now, the next step is to accumulate the SAT statistics. The following procedure will do this for one speaker via labelUtterance. First, the speaker independent models will be reset, and the speaker independent statistics are accumulated to estimate MLLR transforms. Since only transforms for the means are computed, statistics for the distributions are not needed at this point. In a second loop over all segments, the speaker dependent statistics will be accumulated, followed the accumulation of the SAT statistics.

```
proc doAccu {spk labelPath cbsfile minCount} {

  # load SI models and clear accus
  codebookSetX3 load $cbsfile
  codebookSetX3 clearAccus
  distribSetX3  clearAccus
  mlaX3         clearSAT

  # accumulate SI statistics
  Dscfg configure -accu n
  foreachSegment utt uttDB $spk {
    eval set label $labelPath
    labelUtterance X3 $spk $utt $label
    senoneSetX3 accu pathX3
  }

  # compute MLLR transforms
  mlaX3 update -minCount $minCount

  # accumulate SD statistics
  codebookSetX3 clearAccus
  Dscfg configure -accu y
  foreachSegment utt uttDB $spk {
    eval set label $labelPath
    labelUtterance X3 $spk $utt $label
    senoneSetX3 accu pathX3
  }

  # accumulate SAT statistics
  mlaX3 accuSAT
}
```

Now, we have to build the loop over all speakers and do the ML estimation of the SAT models. The loop over the speaker can be parallelized as usual. To write the SAT accumulators for each client, you need to store a full matrix for each component. For example, if you have 150$k$ Gaussians with a feature dimension of 24, you need to store 691 MB. To reduce the computional and memory load, you can use diagonal transforms instead of full transforms. If you have enough memory, you can store and restore the SI models to avoid the reloading of the SI models from disc. Additionally, we recommend that you organize your database to group all conversations for the same speaker together. By doing this, you get more robust estimates for the MLLR transforms and speed up the training drastically.

```
doParallel {
  while {[fgets $convLst spk] >= 0} {
    doAccu $spk $labelPath $cbsfile $minCount
   }
   mlaX3        saveSAT   Accus/clientID.sat.gz
   distribSetX3 saveAccus Accus/clientID.dsa.gz
} {
  codebookSetX3 clearAccus
  distribSetX3  clearAccus
  mlaX3          clearSAT
  foreach file [glob Accus/*saa.gz] {mlaX3        readSAT   $file}
  foreach file [glob Accus/*dsa.gz] {distribSetX3 readAccus $file}
  mlaX3        updateSAT
  distribSetX3 update
  codebookSetX3 save Weights/new.cbs.gz
  distribSetX3  save Weights/new.dss.gz
} {} {} {}
```

The decoding of SAT models should rely on *adapted* models of course, otherwise you will observe poor recognition rate due to unmatched model and test data conditions.

The Adaptation to the test data is done analogous to the first part of the *doAccu* routine and can be refined using confidence measures.

### 5.2.4 Feature space based Speaker Adaptive Training

Instead of transforming the models, the features might be transformed using linear transforms during training. The advantage is, that the statistics to accumulate rely on the same models and the SAT training becomes much more efficient. Feature space adaptation (FSA) might be viewed as a constrained model based transform, where the same transform is used to transform means and covariances. However, the ML estimatation process of the transforms uses an integrated Jacobi normalization which results in a true feature space transform. Since feature space adaptation is much faster than model based transforms and can be combined with Gaussians selection methods (e.g. Bucket Box Intersection BBI), incremental FSA is very well suited to real-time systems.

```
SignalAdapt SignalAdaptX3 senoneSetX3
SignalAdaptX3 configure -topN 1 -shift 1.0
foreach ds [distribSetX3] { SignalAdaptX3 add $ds }
```

The Creation of a SignalAdapt object is based directly on the set of senones. To apply the transforms, your feature desciption files needs a line like this, where *transIndex* is an index of transform:

```
SignalAdaptX3 adapt $fes:LDA.data $fes:LDA.data $transIndex
```

The estimation process consists of an accumulation and update phase. The accumulation for one speaker using a forced alignment procedure can be written as follows:

```
proc doAccu {spk labelPath accuIndex} {
  foreachSegment utt uttDB $spk {
    eval set label $labelPath
    labelUtterance X3 $spk $utt $label
    SignalAdaptX3 accu pathX3 $accuIndex
  }
}
```

Given sufficient statistics stored in the accumulator *accuIndex*, you can now find an iterative solution for the ML estimate of the transform. Usually, 10 iterations are enough to reach convergence. The transform is then stored in *transIndex*.

```
SignalAdaptX3 compute $iterations $accuIndex $transIndex
```

These routines can be integrated in the standard training procedure to simultaneously update the model and adaptation parameters. Furthermore, you can combine FSA and MLLR to adapt to test data. In that case, a feature space transform can be estimated at first, and the model based transforms then rely on the adapted feature space.

In an incremental adaptation scheme, you might like to enhance the robustness by combining the adaptation data with preaccumulated statistics. This can be achieved by using the *addAccu*, *readAccu*, *writeAccu*, and *scaleAccu* methods of the SignalAdapt object. For example, you can generate gender and channel dependent statistics from the training data which are then combined on the fly with the test speaker statistics. To find the right modality, a ML criterium using a forced aligment procedure can be applied.

### 5.2.5 Incremental growing of Gaussians

In the previous sections, we discussed training schemes which start with some initial models (typically generated by the k-means algorithm). An alternative approach is to start with one component only, and incrementally add parameters by splitting components according along the largest covariances. As a result of this training procedure, the gaussians are more evenly distributed and the parameters cover the acoustic space more efficiently. However, the convergence of that procedure is slower and more training iterations are needed. The training can be optimized if fixed forced aligments are used. In that case, a full sample extraction dumps all data for each state and the data has to be loaded only once during the training, which reduces the disc I/O drastically. The sample extraction can be done using *samples.tcl* from the script collection and setting the *maxCount* variable approriate.

```
phonesSetInit   X3
tagsInit        X3
featureSetInit  X3 -desc ""
codebookSetInit X3 -desc ""
distribSetInit  X3 -desc ""

featureSetX3 FMatrix LDA
featureSetX3:LDA.data resize 1 42
```

The module initialization now becomes much simpler, since we don't have to load any description files for the codebooks and distributions anymore. To add new codebooks in CodebookSet, we have to provide an underlying feature in FeatureSet, in the example we use a feature *LDA* with a dimension of 42. Instead of parallelizing the training over the speaker, we can now run a loop over all *codebooks*.

```
proc estimateState {cb samplePath} {
  # codebook to distrib mapping
  set ds $cb

  # load training samples
  set smp  featureSet$SID:LDA.data
  $smp bload $samplePath/$cb.smp
  $smp resize [$smp configure -m] [expr [$smp configure -n] -1]

  # create codebook and distrib
  codebookSetX3 add $cb LDA 1 42 DIAGONAL
  distribSetX3  add $ds $cb

  # max. nr. of components
  codebookSetX3:$cb     configure -refMax 24

  # mincount per component
  codebookSetX3:$cb.cfg configure -mergeThresh 50

  # step size for spliting components
  codebookSetX3:$cb.cfg configure -splitStep 0.001

  codebookSetX3:$cb createAccu
  distribSetX3:$ds  createAccu

  # main iterations with increasing components
  for {set i 0} {$i < 7} { incr i} {
    # accumulate data
    codebookSetX3:$cb.accu clear
    distribSetX3:$ds.accu  clear
    for {set frX 0} {$frX <  [$smp configure -m]} {incr frX} {
       distribSetX3 accuFrame $cb $frX
    }

    # update, split and merge
    distribSetX3 update
    distribSetX3 split
    distribSetX3 merge

    # small iterations without increasing components
    for {set j 0} {$j < 3} { incr j} {
      codebookSetX3:$cb.accu clear
      distribSetX3:$ds.accu  clear
      for {set frX 0} {$frX <  [$smp configure -m]} {incr frX} {
        distribSet$X3 accuFrame $ds $frX
      }
      distribSetX3 update
    }
  }
  codebookSetX3:$cb freeAccu
  distribSetX3:$ds  freeAccu
}
```

The remaining part synchronizes the clients and saves acoustic models and description files. To match the new description files with the original distribution tree, missing distributions will be added in the final phase. Untrained distributions occur due to the backoff to context independent nodes in the tree.

```
doParallel {
    while { [fgets $stateLst cb] >= 0} { estimateState $cb $samplePath}
    codebookSet$SID write Weights/clientID.cbsDesc
    codebookSet$SID save  Weights/clientID.cbs.gz
    distribSet$SID  write Weights/clientID.dssDesc
    distribSet$SID  save  Weights/clientID.dss.gz
} {
    CodebookSet cbs featureSetX3
    DistribSet  dss cbs
    foreach f [glob Weights/*.*.cbsDesc] { cbs read $f }
    foreach f [glob Weights/*.*.dssDesc] { dss read $f }
    foreach f [glob Weights/*.*.cbs.gz]  { cbs load $f }
    foreach f [glob Weights/*.*.dss.gz]  { dss load $f }

    # read missing distribs
    set fp [open distribSet.org]
    while { [gets $fp line] >= 0} {
        set ds [lindex $line 0]
        set cb [lindex $line 1]
        if {[dss index $ds] < 0} { dss add $ds $cb }
    }
    close $fp
    cbs write Weights/final.cbsDesc.gz
    cbs save  Weights/final.cbs.gz
    dss write Weights/final.dssDesc.gz
    dss save  Weights/final.dss.gz
} { } { }
```

The training procedure described here is well suited to train fully continuous systems. If you'd like to train semi continous systems, where you have more than one distribution for each codebook, we recommend to start the training with the full continous setup and use the trained codebooks as seed models to start the training of the distributions.

### 5.2.6 Semi-tied full Covariances

Although JANUS supports Gaussian densities with radial, diagonal or even full covariances, normally only models with diagonal covariances are trained due to lack of training data or cpu and memory restrictions. On the other hand, a linear transform of the covariance corresponds to an inverse transform of the means and features. However, LDA or PCA transforms aren't optimized according to the ML criterium as is the case for all other model parameters. The concept of semi-tied full covariances (STC) introduces full transforms for the diagonal covariances. These transforms might be shared for several components and trained in a ML fashion. During decoding, the inverse transforms are applied to the features, which results in multiple feature spaces. Therefore, this technique is also called *Optimal Feature Space, OFS*. To train semi-tied covariance, we start with some initial models as usual and refine them. There are basically 4 steps:

1. create description files for the covariance classes
2. convert acoustic models to OFS format
3. train OFS models
4. convert OFS models back to standard codebooks/distributions

#### Create description files

We need to create description files for CBNewParMatrixSet and CBNewSet. CBNewParMatrixSet describe the linear transforms associated to the covariances, while CBNewSet describe the densities itself. Assuming a feature *LDA* with a dimension of 42 and full transforms for each basephone, the following lines will create description files, given an appropriate definition of the helper function *map_distrib_to_basephone* to map the model names.

```
parmatSetInit X3 -desc "" -dimN 42
cbnewSetInit  X3 -desc ""

foreach phone [phonesSetX3:PHONES] {
  parmatSetX3 add $phone 1 {42}
}
foreach ds [distribSetX3] {
  set phone [map_distrib_to_basephone $ds]
  cbnewSetX3 add  $ds LDA [distribSetX3:$ds configure -valN]
  cbnewSetX3 link $phone  [cbnewSetX3 index $ds] all
}
parmatSetX3 save        desc/paramSet
parmatSetX3 saveWeights Weights/0.pms.gz
cbnewSetX3  save        desc/cbnewSet
```

### Convert acoustic models to OFS format

The means, covariances, and mixture weights will now be converted into a format suitable for the CBNew objects. Please note, that the covariances stored in Codebook are *inverse* while CBNewParMatrixSet store them directly.

```
foreach ds [distribSetX3] {
  set cbX  [distribSetX3:$ds configure -cbX]
  set refN [distribSetX3:$ds configure -valN]
  # means
  cbnewSetX3:$ds set mean codebookSetX3.item($cbX).mat

  # covariances
  FMatrix m1 $refN $dimN
  for {set i 0} {$i < $refN} {incr i} {
    set cvL [lindex [codebookSetX3.item($cbX).cov($i)] 0]
    for {set j 0} {$j < $dimN} {incr j} {
      m1 set $i $j [expr 1.0 / [lindex $cvL $j]]
    }
  }
  cbnewSetX3:$ds set diag m1
  m1 destroy

  # mixture weights
  FMatrix m2 1 $refN
  m2 := [distribSetX3:$ds configure -val]
  cbnewSetX3:$ds set distrib m2
  m2 destroy
}
cbnewSetX3 saveWeights Weights/0.cbns.gz
```

### Train OFS models

The training of the models can be done using the already known *labelUtterance* (or other forced aligment procedures). We describe here an approach using a full sample extraction, similar to the procedure used in the section *Incremental Growing of Gaussians*. You can reuse the samples extracted from there.

```
set protPath prot

cbnewSetX3 phase work
doParallel {
  FMatrix smp
  while {[fgets $stateLst cb] >= 0} {
    smp bload $samplePath/$cb.smp
    smp resize [smp configure -m] [expr [smp configure -n] -1]
    cbnewSetX3 accuMatrix [cbnewSetX3 index $cb] smp
  }
  cbnewSetX3 saveAccusDep Accus/clientID.cbna
  storeLH Accus/clientID.lha
} {
  set sum 0
  foreach f [glob Accus/*.*.lha] {set sum [expr $sum + [loadLH $f]]}
  storeLHProt $sum
  foreach f [glob Accus/*.*.lha] {rm $f }
  cbnewSetX3  clearAccus
  foreach f [glob Accus/*.*.cbna] {cbnewSetX3 loadAccusDep $f}

  senoneSetX3 update
  parmatSetX3 update -stepN 100 -smallSteps 20 -firstSmall 40 -deltaThres 0.05

  calcProts
  parmatSetX3 saveWeights Weights/new.pms.gz
  cbnewSetX3  saveWeights Weights/new.cbns.gz
}
```

This excerpt shows the general procedure to train semi-tied full covariances. Similar to the standard training procedure, you will probably repeat the training several iterations to reach convergence. Please note, that the server process needs enough memory to store full covariances during the update of the covariance transforms.

### Convert OFS models back to standard codebooks/distributions

Before we can start the decoding, we have to convert the acoustic models back to the standard format. The following lines will create new description and parameter files.

```
parmatSetInit X3 -desc desc/paramSet -param Weights/new.pms.gz
cbnewSetInit  X3 -desc desc/cbnewSet -param Weights/new.cbns.gz
[CodebookSet cbs featureSetX3] read cbs.orig.desc
[DistribSet  dss cbs]          read dss.orig.desc
foreach cb [cbs] { cbs:$cb alloc }
cbnewSetX3 convert cbs dss
cbs write desc/cbs.new.desc
dss write desc/dss.new.desc
cbs save  Weights/new.cbs.gz
dss save  Weights/new.dss.gz
```

The last thing to do is to modify our feature description to apply the OFS transform to the features. You can combine this technique together with MLLR or FSA. In the latter case, the feature space adaptation should rely on the transformed OFS features. If you use MLLR, the same regression tree should be used for MLLR and STC. Otherwise, the adaptation transforms will be computed over different feature spaces, resulting in inconsistent ML estimates for the transforms.

```
foreach p [parmatSetX3] { $fes matmul OFS-$p LDA $pms:$p.item(0) }
```

### 5.2.7 MMIE-Training

Janus can be used to perform *MMIE* training. Example scripts can be found in `/project/ears4/D2` at UKA. Be warned that this type of training is computationally very expensive.

## 5.3 Adaptation

Janus contains several `Adaptation` methods, subsumed under "training". Please note that the examples here can only serve as a rough guide to your own experiments.

### 5.3.1 MLLR Adapation in feature space

This type of adaptation (often referred to as *CMLLR*, *FMLLR*) has already been described in the context of Feature-space based Speaker Adaptive Training. It represents the easiest way to adapt a system to new conditions. It uses the `SignalAdapt` module.

Basically, you use `adapt` to collect statistics and `compute` to compute an adaptation matrix, which you can then use to transform your features. Interestingly, it seems more effective to collect adapted features then unadapted ones, which would be theoretically correct. The `SignalAdapt` allows you to keep several accumulators at the same time, so you can adapt to several targets at the same time. This approach is also very useful for incremental adaptation and in cases where only little adaptation data is available.

For more information on this adaptation scheme, which can also be used for speaker-adaptive training, look at section 5.2.4, too.

### 5.3.2 MLLR Adapation in model space

*MLLR* adaptation can be used to adapt models to new test conditions, be it a particular speaker or a new channel. It is more powerful than feature-space MLLR (see 5.3.1), because it uses a regression tree and therefore more and a variable amount of parameters can be adapted.

The first step to perform MLLR adaptation is to collect statistics on the adaptation data. To do this, use the standard `train.tcl` script as if you wanted to continue training your model on your adaptation data. All you have to do is make sure you call the procedure `doAccu` only once by setting the begin and end iterations accordingly and you also probably don't want to execute the `update` on the `SenoneSet` and save the generated `Codebook`. Note that there are two ways of doing this:

- **`Supervised`** when you have true transcripts of the adaptation data; you can now use labels or viterbi to align the data
- **`Unsupervised`** when you do not have transcripts. Usually, you then use the confidence-annotated output of a previous recognition run and viterbi to align the data

Depending on which way suits you more, you might have to change the calls in `doAccu` to `labelUtterance` to `viterbiUtterance`. Now you can use your models and the collected statistics (found in the accu) to update the parameters of the model using the script `mllr.tcl`.

If you have a lot of adaptation data, you can also update the covariances of your models by setting `adaptVar` to `1` in `mllr.tcl` and executing this script a second time. You would then use the models with adapted means (the result of executing `mllr.tcl` once) to collect statistics a second time and adapt the covariances of the models in a second run of `mllr.tcl`.

The optimal settings of the `depth` and `minCount` parameters depend on your task, so feel free to experiment with these.

### 5.3.3 MAP Adaptation

*MAP* adaptation uses the same *MLE* criterion employed during training. Adaptation therefore consists of loading the accumulated statistics of the "background" system, weighting them with a weighting factor and then adding the weighted statistics accumulated with the same models on the adaptation data before doing the normal ML-update. To collect the statistics, use the `train.tcl` script on both data sets as described in chapter 5.3.2, doing only one iteration and leaving out the update; the script `map.tcl` will then perform the adaptation.

### 5.3.4 MAM Adaptation

*MAM* is an adaptation/ normalization method developed by Martin Westphal in his thesis. The current Janus contains all the code he wrote, an experiment with MAM can for example be found in `/project/nespole/sys/mam` at UKA.

# Chapter 6

# Decoding with Ibis

## 6.1 Basic Decoding

This section describes how to setup Janus to produce hypotheses and lattices from ADC data using the *Ibis* decoder. `Decoding` can also be done using Janus' three-pass decoder, which is however not documented here.

### 6.1.1 Module list

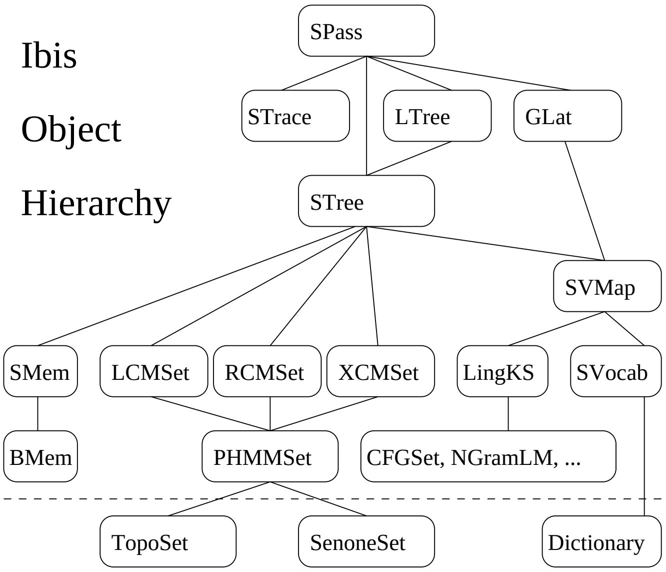
You'll need the following modules (most of these modules were already introduced in the description of training procedures):

1. A `PhonesSet`, describing the phones you want to use. An example description file can be found in `phonesSet`.
2. A set of `Tags`, containing the tags (phone modifiers) in your `Dictionary`; typically "WB" for word boundaries only. An example description file can be found in `tags`.
3. A `FeatureSet`, which contains your ADC data and derived features (MCEP, LDA, ...). You normally use `featAccess` and `featDesc` to locate and process the ADC file.
4. A `CodebookSet`, which contains acoustic models (Gaussians). An example description file can be found in `codebookSet`. You'll also need a parameter file.
5. A `DistribSet`, which contains the mixture weights for the Gaussians. An example description file can be found in `distribSet`. You'll also need a parameter file.
6. A DistribTree, an instance of `Tree`, which defines which distribution to use for which model. An example description file can be found in `distribTree`.
7. A `SenoneSet`, which is a set of (context-dependent) sub-phone model states.
8. A `TmSet`, which describes the allowed transitions between states. An example description file can be found in `tmSet`.
9. A `TopoSet`, containing the useable topologies. An example description file can be found in `topoSet`.
10. A TopoTree. This module of type `Tree` describes which topologies to use for which phones. An example description file can be found in `topoTree`.
11. A `Dictionary`, see `dictionary`.
12. A `SVocab`, the search vocabulary. An example file can be found in `svocab`.

13. A LingKS, a language model. The standard language models (NGramLM) are in ARPABO format and can for example be generated with the CLAUSI tools, the CMU-SLMT or the SRI toolkit.

14. A DBase, the Janus database module, is optional. The standard scripts however make use of the database. For an example of how to generate the DBase, have a look at the file scripts/genDBase.tcl.

The object hierarchy for the Ibis decoder looks as follows:



The file ibis.tcl might save you a lot of trouble. Also, consider looking at the example script master.tcl, which in turn calls test.tcl. This file will start up a standard decoder, which you can then modify or re-configure according to your needs. The initialization of the decoder can be done automatically by calling ibisInit from ibis.tcl and setting the values in desc.tcl correctly.

You can also use lksInit to set up the language model independently from the decoder, this makes it a bit easier to handle dumps generated by STree dump (remember, when you read a dump, you cannot load the contents of Dictionary and SVocab and you can also store the dump for the language model in the same file.

### 6.1.2 Example decoding script

A simple decoding script looks as follows:

```
source ../desc/desc.tcl

# ------------------------------------------------------
#  Init Modules
# ------------------------------------------------------

phonesSetInit   $SID
tagsInit        $SID
featureSetInit  $SID
codebookSetInit $SID
distribSetInit  $SID
distribTreeInit $SID -ptree ""
```

```
senoneSetInit   $SID  distribStream$SID
topoSetInit     $SID
ttreeInit       $SID
dictInit        $SID
trainInit       $SID
dbaseInit       $SID  [set ${SID}($dbaseName)]
ibisInit        $SID

# ---------------------------------
#   Here we go...
# ---------------------------------

while {[fgets $spkLst spk] != -1} {

    # --------------------------------------------------
    #   Loop over all utterances
    # --------------------------------------------------

    foreachSegment utt uttDB $spk {

        # preprocess audio data
        set uttInfo [dbaseUttInfo db$SID $spk $utt]
        featureSet$SID eval $uttInfo

        # run decoder
        spass$SID run

        # get hypothesis
        set hypo [spass$SID.stab trace]
        set text  [lrange $hypo 3 [expr [llength $hypo] -2]]
        puts "$utt: $text"

    }
}
```

The DBase allows you to use this double loop construct **while fgets** and **foreachUtterance** easily. The outer loop ("while fgets") loops over all speakers and allows to parallelize the work over several machines, by assigning different speakers to different machines. The inner loop works over all utterances of this particular speaker. Other loop constructions are possible, of course. Choose whichever is appropriate for your needs.

Even though the decoder can be initialized by calling ibisInit, you might want to have a look at the following sections on decoder initialization and beam settings to get a better understanding on how things work, although the complexity can be hidden from view by using lksInit and ibisInit.

### 6.1.3 Beam Settings

The tuning of beam thresholds to speed up the decoding without increasing search errors is dependent on the task and the effectivity of the language model lookahead. In [11], a background about the decoding technology can be found. In principle, there are three thresholds to control the beam search. The *stateBeam* controls the number of active states, while the *wordBeam* controls the number of active word ends. The number of linguistically morphed instances, which depends on your language model history, is pruned using the *morphBeam* threshold. To reduce high peaks in memory consumption, a cutting of active word ends and instances is performed with *transN* and *morphN*. For an effective pruning, the following rules can be applied:

- $morphBeam \ll wordBeam \ll stateBeam$
- $morphN \ll transN$

It should be noted that the language model weight has a high impact on the pruning process. That means, the optimization of pruning thresholds must be done with respect to the language model weights. Examples for the beam settings for different tasks are given here:

```
# read Broadcast News
spassX3 configure -stateBeam 130
spassX3 configure -wordBeam   90
spassX3 configure -morphBeam  80
spassX3 configure -transN 35 -morphN 8

# conversational telephony speech
spassX3 configure -stateBeam 195
spassX3 configure -wordBeam  135
spassX3 configure -morphBeam 120
spassX3 configure -transN 80 -morphN 20
```

The Ibis decoder's performance with a given set of acoustic and language models is mainly governed by the beam settings. These directly influence the search space and the tradeoff between speed and accuracy.

If you need faster recognition time, there are two main techniques implemented in JRTk to help you achieve this goal: Gaussian Selection (BBI) reduces the number of Gaussians evaluated during score computation and Phonetic Fast Match (LookAhead) prunes unlikely hypotheses away much earlier, therefore reducing the search space.

Also, Feature Space Adaptation is a useful technique that improves both speed and accuracy for a given system if you have enough data to adapt on. These techniques are described in the following sections.

### 6.1.4 Gaussian Selection (BBI)

Gaussian Selection using the Bucket-Box-Intersection algorithm [4] is the most popular speed-up algorithm for Janus. The script `createBBI.tcl` (3.4.1) can be used to compute boxes for a given depth and cut-off value. The resulting description and parameter files can then be loaded into the `CodebookSet` using `bbiSetInit`, as shown in the script `test.tcl` (3.4.1).

Usually, you can achieve a speed-up by a factor of two with only minor degradation in performance ($< 1\%$).

### 6.1.5 Phonetic Fast Match (LookAhead)

A Phonetic Fast Match consists of an additional `SenoneSet`, which is based on context-independent codebooks. These are evaluated with a fixed topology up to (normally) 5 frames "in the future" and their score is added to the normal acoustic model score. Unlikely hypotheses ("this does not look like an `/s/` in the next 5 frames") can therefore be pruned away much earlier, which speeds up decoding despite the overhead of extra score computations.

An example script looks like this:

```
[...]

ibisInit        $SID -streeDump ../test6-b-6/streeC.gz                     \
                     -lmDesc    ../bigVocab/isip_ntp10+ch+cell+swb3.ibis.gz \
                     -lmType PhraseLM  -xcm 1

[...]

[FMatrix ldaLA]                  bload /project/ears2/X7/trainCI/ldaX7.bmat
[CodebookSet cbs featureSet$SID] read  /project/ears2/X7/desc/codebookSetLA
[DistribSet  dss cbs]            read  /project/ears2/X7/desc/distribSet

cbs load /project/ears2/X7/trainCI/Weights/4.cbs.gz
dss load /project/ears2/X7/trainCI/Weights/4.dss.gz

Tree dst phonesSet$SID:PHONES phonesSet$SID tags$SID dss
dst configure -padPhone @
dst read  /project/ears2/X7/desc/distribTreeLA

DistribStream str dss dst
SenoneSet     sns str -phones phonesSet$SID:PHONES -tags tags$SID

spass$SID fmatch sns
spass$SID configure -fastMatch 1.0

proc featureSetEvalLA {fes uttInfo} {
    global SID
    $fes matmul LDA-LA FEAT+ ldaLA -cut 42
}
FeatureSet method evalLA featureSetEvalLA  -text ""

[...]

while {[fgets $uttLst utt] >= 0} {

    [...]

    featureSet$SID eval   $uttInfo
    featureSet$SID evalLA $uttInfo

    [...]

    TIME START run $utt; spass$SID run; TIME STOP run $utt

    [...]
```

You can also initialize a LookAhead system by specifying the -fastMatch option to ibisInit. Be aware that the Fast Match does not have any topology information. You therefore have to make sure you have models for all sub-phonetic units in the distribution Tree. Typically you would specify ROOT-m or ROOT-b ROOT-m to the fmatch method via the -snTag option, which allows some flexibility in which models to use for the Fast Match. In some cases it might be necessary to manually add SIL-m to the ROOT-b part of the distribution Tree.

### 6.1.6 Timing

Speed-up methods such as the ones just discussed always trade off speed against accuracy. How accuracy can be measured is discussed in section 2.5, of course JRTk also contains mechanisms to estimate the speed of a process. The relevant script is called `scripts/ana_time.tcl` (see section 3.4.1).

You'll need to define the following settings in your decoding (or any other) script, to use `ana_time.tcl`:

`...` `(CODE)` Some code

`set ModulStatus(Timing) 1` to switch on timing analysis mode. If this variable is undefined or st to 0, nothing will happen.

`...` `(CODE)` follows, no timing analysis

`TIME START total $utt` To start a block called "total" which you want to time for all utterances

`...` `(CODE)` for "total" follows

`TIME START run $utt` To start another block called "run", it is also part of "total"

`...` `(CODE)` for "run" and "total"

`TIME STOP run $utt` "run" ends here

`...` `(CODE)` More "total" code

`TIME STOP total $utt` "total" ends here, too

`...(CODE)` Cleaning up

`saveTIME "time.[pid]"` generates a file containing the timing information, which can be used for further analysis

`...`

The script `scripts/test.tcl` contains the necessary code, although it is switched off by default. Once you ran a script with these commands, you have two options:

1. Look at the output (the log-file). It will now contain lines such as

   ```
   ...
   START total tobias_breyer.99 (1699.580000, 13.810000)
       featureSetISLcd ADCfile tobias_breyer.99 WARP 1.060
   START run tobias_breyer.99 (1699.620000, 13.820000)
   STOP run tobias_breyer.99 (1701.180000, 13.820000)
   ...
   ```

   which tell you that decoding the utterance "tobias_breyer.99" took 1701.18-1699.62=1.56 seconds of CPU (user-) time.

2. If you have `time.*` files, you can use `ana_time.tcl` and generate a summary report by calling `ana_time.tcl` as follows in the test directory, i.e. the directory in which you ran the test script and which contains the `time.*` files:

   ```
   janus ~/janus/scripts/ana_time.tcl /project/njd/IslData/scoring/utts 5319.16
   ```

   The first parameter is the list of utterances to time. If you don't have it, it is normally very easy to generate this from the log-files or the data base. The second parameter is the total time of these utterances in seconds. Running this script should generate the following output:

```
ReadTime: time.16274
/project/njd/IslData/scoring/utts -> 831 utterances
duration        = 5319.16s
RTF run         : 0.29442806759
RTF total       : 0.318975176532
RTF warp        : 0.0174689236646
```

> The first line contains the name(s) of the time dump files read, the second line the number of utterances taken into account and the third line the total time used. The next lines contain the real-time factors (xRTF) calculated for the different blocks. This example is from decoding the IslSystem on a 3GHz P4 machine.

The times reported by this method are generated by the `times` C-library function and returned as `tms_utime` and `tms_stime`. Be warned that the values you get will frequently be meaningless on SMP machines and if the machine was running two processes (for a single-CPU system) or even swapping.

### 6.1.7 Feature Space Adaptation (FSA)

The ML adaptation technique described in 5.2 can also be applied during decoding. In this case, you simply compute an adaptation matrix to transform the features to better match your models. The advantage in transforming the input features is of course, that you don't have to touch the models, making it possible to retain BBI and Fast Match models.

Initialization of the necessary objects can be as simple as:

```
set sas signalAdapt${SID}
SignalAdapt $sas senoneSet$SID
$sas configure -topN 1 -shift 1.0
foreach ds [distribSet$SID] { $sas add $ds }
```

Assuming you have created a `Path` object containing your current hypothesis and your codebooks are based on the "LDA" feature, the following code will accumulate sufficient statistics:

```
    # Feature space adaptation
    $sas clear      0
    $sas clearAccu 0
    $sas accu       path$SID 0
    $sas compute    10 0 0
    $sas adapt featureSet${SID}:LDA.data featureSet${SID}:LDA.data 0
```

You can now decode again (without re-evaluating the feature set). The resulting hypothesis should have a better score than before and, on average, the error rate will be lower.

It is often a good idea to only adapt the features, once you have accumulated enough data (say 10 seconds of speech). Assuming you have more utterances from one speaker or from one channel, you can of course accumulate the data over several utterances, compute the matrix and re-decode. The `SignalAdapt` object can hold several accumulators and adaptation matrices at the same time. It is also possible to load and save them. Another idea is to compute Viterbi scores for several adaptation matrices in order to find the "best" one and use it for further adaptation. In a demo system, you could compute in advance adaptation matrices for several speakers or channels and, during showtime, decode the first utterance once without adaptation, pick the best adaptation matrix and use if for subsequent utterances, further adapting it to the current situation. This is called "delayed adaptation".

## 6.2 Advanced Decoding

In this section, we assume that you already have some experience with the JANUS object interface and the Tcl-Library. To run some more advanced experiments you will probably initialize the decoding engine by yourself without making use of the `ibisInit` function. The decoder works in a single pass using all available acoustic and linguistic information. A full language model lookahead is implemented based on the concept of linguistic polymorphism. The search vocabulary is organized as a compact network sharing both prefixes and suffixes. The active search space will be dynamically allocated on demand using a block memory management. The decoder can handle virtually unlimited vocabularies with higher order n-gram language models. Context free grammars as well as decoding along word graphs are supported.

### 6.2.1 Decoder Initialization

To setup the central search object, called `SPass`, you will create several objects along the module hierarchy shown above. The interface to the training objects is the `SenoneSet`, which provides access to a set of probability density functions (pdf) for each HMM state for a given left and right phonetic context. Each pdf itself might consist of streams using statistical models as gaussian mixtures or neural networks. The `TopoSet` defines the HMM topologies used to model the base phones. Both `SenoneSet` and `TopoSet` are needed to build a `PHMMSet` object which serves as the basic acoustic model interface for the decoder. Left and right context dependent models cab then be built on top of these basic acoustic models. If you have a statistical n-gram language model *mylm.arpabo.gz* together with a vocabulary *myvocab*, the decoder initialization may look like this.

```
# context dependent phonetic hidden markov models
PHMMSet phmmSetX3  ttreeX3    ROOT
LCMSet  lcmSetX3   phmmSetX3
RCMSet  rcmSetX3   phmmSetX3

# language model
[LingKS lmX3 NGramLM] load mylm.arpabo.gz

# Search Vocabulary, Vocabulary Mapper
SVocab  svocabX3   dictX3
SVMap   svmapX3    svocabX3  lmX3

svocabX3 read myvocab
svmapX3  map  base

# Search Network, Linguistic Tree, Single Pass Decoder
STree streeX3 svmapX3 lcmSetX3 rcmSetX3
LTree ltreeX3 streeX3
SPass spassX3 streeX3 ltreeX3
```

A few configuration options for the language model cache and the beam search will complete the startup. A word penalty and a language model weight can be configured in the `SVMap` object.

```
# configure LanguageModel cache
ltreeX3 configure -cacheN 200 -ncacheN 10
```

```
# configure Vocabulary Mapper
svmapX3 configure -phonePen  0.0 -wordPen 0 -silPen 10 -filPen 0 -lz 30

# configure Single Pass Decoder
spassX3 configure -stateBeam 130
spassX3 configure -wordBeam   90
spassX3 configure -morphBeam  80
spassX3 configure -transN 35 -morphN 8
```

### 6.2.2 Lattices

By default, lattices (defined by the object GLat) will not be generated at all, since all acoustic and linguistic information is truly used in the first pass and a second rescoring pass is not necessary. However, for several tasks like MMIE training, Consensus decoding, or acoustic rescoring, lattices might be wanted.

A lattice node ("GNode" in GLat) represents a word with start and end time together with the phonetic context, while the linguistic context is excluded. Lattice links ("GLink" in GLat) store the acoustic scores for the right context dependent models. The lattice generation works in two phases. In the first phase, lattice nodes and links will be created on the fly during decoding directly from the active search space by bypassing the back-pointer table. Since, we bypass the back-pointer table, several lattices nodes might be unconnected. Therefore, a second phase will add lattice links with respect to their a-posteriori probabilities. This approach allows to extract more information when compared to a lattice generation based on a back-pointer table.

```
# configure thresholds for lattice generation
spassX3.glat configure -alphaBeam 200 -topN 150

# preprocess audio data
set uttInfo [dbaseUttInfo dbX3 $spk $utt]
featureSetX3 eval $uttInfo

# run decoder
spassX3 run

# connect lattice nodes and prune
spassX3.glat connect -beam 200
spassX3.glat prune -nodeN [expr 100 * [llength $hypo]]
spassX3.glat write myLat.gz
```

To apply different language model weights and penalties, the method rescore might be used to get the n-best hypotheses. Word posteriori based confidences can be extracted using the method confidence. There are many manipulation functions to add or delete nodes and links. You can also create lattices by adding word sequences with addPath. Lattice error rates can be computed by align a sequence of reference words to the lattice with align.

```
GLat glatX3 svmapX3

glatX3 read mylat.gz
set output [glatX3 rescore -map svmapX3 -topN 1]
set hypo   [lrange [lindex $output 0] 2 end]
set conf   [glatX3 confidence $hypo -scale [expr 1.0 / $lz]]
```

### 6.2.3 Vocabulary Mapper

A SVMap defines a map function to map words from the search vocabulary SVocab to the vocabulary defined by a linguistic knowledge source LingKS. The search vocabulary consists of all words to be recognized potentially while the vocabulary from the LingKS contains those words for which linguistic information, e.g. a language model probability, is available. For example, a pronunciation variant belongs to the SVocab, but only the base-form occurs in the language model. The SVMap will define a mapping between pronunciation variant and base-form, potentially including a pronunciation probability. The same concept can be used to define class-based language models, e.g. pronunciation variants can be seen as a special case of a class based language model, which is shown in the following lines.

```
# class based language model
[LingKS lmX3 NGramLM] load classLM.arpabo.gz

# Search Vocabulary, Vocabulary Mapper
SVocab  svocabX3   dictX3
SVMap   svmapX3    svocabX3  lmX3

svocabX3 read myvocab

# define basic map
svmapX3  map  base

# read substitution section from a class based language model
svmapX3 readSubs -lks lmX3
```

The SVMap allows great flexibility in combining vocabularies and languages models. You can define your own mapping easily by using the add, delete or readMapFile functions. Pronunciation probabilities can be modified dynamically during decoding by changing the SVMap entries. If you want to exclude a word from the search vocabulary, just delete the corresponding map entry. No restructuring of the search network is necessary.

A frequent use of the vocabulary mapper is to quickly and easily add a few words to an existing recognizer. If you don't have any language model information, you just create dictionary entries and map them to some existing language model word in the SVMap. The probabilities accepted add method are log-probs, i.e. if you add a word using `-prob 0.0` it will have an intra-class probability of 1, while a word added using `-prob -2.3` will have an intra-class probability of 0.5. Note that it is also to possible to have "probabilities" $p > 1$ using for example `-prob 1.0`. This is a useful hack if you find that some words "just don't get recognized properly".

### 6.2.4 Interpolation of Language Models

Now, let's interpolate some linguistic knowledge sources. The interpolation object itself is again a linguistic knowledge source, but of type *MetaLM*. By doing this, you can create a hierarchy of interpolated language models. You can also combine statistical n-gram models with context free grammars. Global or context dependent interpolation weights might be used. Here is an example of interpolating a Switchboard and a Broadcast-News language model.

```
# basis language models
[LingKS lks_SWB NGramLM] load switchboard.lm.gz
[LingKS lks_BN  NGramLM] load broadcast.lm.gz
```

```
# interpolated LM
LingKS lks MetaLM
lks.data LMadd lks_SWB
lks.data LMadd lks_BN

# interpolation weights
lks.data loadWeights interpol.weights
```

In principle, you can use the interpolated language model as it is. However, the interpolation causes millions of *exp* and *log* computations and therefore the decoding time will increase significantly. To speed up the decoding, we recommend to use a simplified language model as a lookahead instead of the full model. In particular, you can use one of the basis models to that end.

```
# Search Vocabulary, Vocabulary Mapper
SVocab  svocabX3   dictX3
SVMap   svmapX3    svocabX3  lks

svocabX3 read myvocab
svmapX3   map  base

# Search Network, Linguistic Tree, Single Pass Decoder
STree streeX3 svmapX3 lcmSetX3 rcmSetX3

# Simplified lookahead
SVMap svmapLA svocabX3 lks_SWB
svmapLA map base
LTree ltreeX3 streeX3 -map svmapLA

# Decoder
SPass spassX3 streeX3 ltreeX3

# configure LTree to use svmap's score function for the leafs
ltreeX3 configure -cacheN 1 -ncacheN 500 -mode single
svmapX3 configure -cacheN 30
```

### 6.2.5 Modeling of Word Phrases

A linguistic knowledge source from type *PhraseLM* can be used to model word phrases (aka multi-words). This object type defines mappings between sequences of words. In particular, the substitutions of a class based language model can be handled by a `LingKS` of type *PhraseLM*. Map files can be read in by using the method `readMapFile`. A line of the map file might looks like this "aboutit(2) {about it} -prob -1.06", which maps the word *aboutit* to the sequence *about it* with a negative logarithmic probability of -1.06. The following lines shows the construction of an linguistic knowledge source by interpolating a 3-gram with a class based 5-gram language model and each is using word phrases.

```
# 3gram swb LM
[LingKS lm1 NGramLM] load swb.3gram.gz

# 5gram class based swb LM
[LingKS lm2 NGramLM] load swb.5gram.gz
[[LingKS lm3 PhraseLM].data base lm2].data readSubs
```

```
# interpolated LM
LingKS lm4 MetaLM
lm4.data LMadd lm1
lm4.data LMadd lm3
lm4.data loadWeights interpol.weights

# multiwords for the final LM
[[LingKS lmX3 PhraseLM].data base lm4].data readMapFile swb.dict03.map

# lookahead LM : phraseLM over swb lm1
[[LingKS lmLA PhraseLM].data base lm1].data readMapFile swb.dict03.map

# Search Vocabulary, Vocabulary Mapper
SVocab  svocabX3  dictX3
SVMap   svmapX3   svocabX3  lmX3
svmapX3 map base
svmapLA readMapFile swb.dict03.map

# Simplified lookahead svmap
SVMap svmapLA svocabX3 lm1
svmapLA map base
svmapLA readMapFile swb.dict03.map

# linguistic tree
LTree ltreeX3 streeX3 -map svmapLA

# Decoder
SPass spassX3 streeX3 ltreeX3

# configure LTree to use svmap's score function for the leafs
ltreeX3 configure -cacheN 1 -ncacheN 500 -mode single
svmapX3 configure -cacheN 30
```

### 6.2.6 Context Free Grammars

IBIS allows to decode also along context free grammars (CFG) in addition to the classical statistical n-gram language models. This is especially an advantage in small domains, where less domain dependent training data is available for n-gram language models. Rather than compiling one network or a finite state graph out of the grammar description files, we use a more dynamic approach. Several rule based recursive transition networks (RTNs) are linked together by their non-terminal symbols. During decoding, a rule stack gives us the ability to enter or leave the linked networks. This kind of network organization gives us high flexibility when used in combination with dialog managers. Furthermore, it enables us to work with real context-free grammars.

The CFG implementation supports currently the following grammar description formats:

SOUP The grammar format of the CMU SOUP-Parser [6], which is used for many IF-based translation systems. It is an expansion of the CMU Phoenix-Parser [16] grammar format. An example grammar can be found in the section `ContextFreeGrammars`.

JSGF The Java Speech Grammar Format, whereby import statements are currently not supported. Further documentation can be found in [15].

FSM The AT&T FSM (finite state machine) text file format [9].

PFSG The probabilistic finite state graph (PFSG) format, which is used by the SRI language model toolkit [14].

In most cases, we are working with non-statistical semantic grammars, i.e. each transition to the next word has the same language model score[1]. But there is also support for all other kinds of probabilistic recurrent transition networks, whereby the probabilities can either be specified in the grammar file or equally distributed during the build process.

The context free grammar implementation in Janus can not only be used to do speech recognition, but also for parsing sentences and returning the parse trees for a given hypothesis. Currently there is only one major disadvantage compared to the SOUP or Phoenix parser, skipping of words is not possible.

A demonstration system for grammar based speech recognition with IBIS and for building small demo applications using speech recognition, named One4All was written by Christian Fügen and is maintained at the ISL by Sebastian. Feel free to ask him, if you want to have that system, or if there are any questions about it.

### Initialization

The initialization of the CFGs can be done either automatically by using `cfgInit` together with some settings in `desc.tcl` or manually. `cfgInit` has to be called somewhere before `ibisInit`, because the linguistic knowledge source has to be given as LM parameter to `ibisInit`. After setting the appropriate values in `desc.tcl`, the differing two lines from the standard start-up given above looks as follows:

```
cfgInit         $SID
ibisInit        $SID -lm cfgSet$SID
```

After decoding you can get the resulting parse tree by calling `parseTree`, which has also the ability to map terminal classes (see 6.2.6) back to their class members, by using the corresponding `SVMap` as additional argument. This function is case sensitive and can also be used for parsing and afterwards returning the parse tree for any other text.

```
# e.g. get hypothesis
set hypo [spass$SID.stab trace]
set text [lrange $hypo 2 end]

# get parse tree
set parseTree [cfgSet$SID.data parseTree $text -svmap svmap$SID]
```

Initializing grammars or grammar sets manually goes e.g. as follows:

```
# grammar set for decoding
LingKS cfgSet CFGSet

CFG cfg1 -lks cfgSet
cfg1 load grammar1

CFG cfg2 -lks cfgSet
cfg2 load grammar2
```

[1] Therefore a score of -1.0 (=0.1) is used for all transitions to terminals and a score of 0.0 (=1.0) is used for $\epsilon$-transitions and all transitions to non-terminals.

```
cfgSet.data build

# single grammar for parsing
CFG cfg
cfg load grammar1
cfg load grammar2
cfg build
```

As mentioned already above several grammar file formats are supported. They can be specified, if the automatic format detection fails. To change the manner of how the initial transition probabilities are set, a mode can be given to the build command. Following are a few example commands:

```
cfg load grammar -format pfsg
cfg build -mode equal              ;# equally distributed scores
cfg build -mode equal -overwrite 1 ;# also overwrites given probs from file
```

### Sub-Grammars and Grammar Domains

Several domain dependent sub-grammars can be activated/deactivated and loaded at run time by using the CFGSet object. The activation/deactivation mechanism goes all the way to the rule level, giving the dialogue management system the full control over the speech recognizer. Furthermore, it is also allowed to penalize grammars or rules, by giving them a penalty factor.

When working with domain dependent grammars we support also a so-called shared grammar, which includes domain independent concepts, to eliminate the overhead of defining the same concepts in different grammars. Therefore you can assign domain tags to grammars, with which grouping of several grammars to one domain is possible (see also desc.tcl). Grammars can now be activated or deactivated by using their domain tags instead of switching each grammar in the set directly. The tag SHARED is reserved for the shared grammar, which is always activated and with the tag all given as argument to the activation/deactivation function all grammars are switched. Deactivated grammars are excluded from the next decoding or parsing process.

```
# activates only grammars of the navigation domain
cfgSet$SID.data deactivate all
cfgSet$SID.data activate NAV

# deactivates a rule in a grammar
cfgSet$SID.data.cfg(0):greeting configure -status Inactive
```

In the resulting parse tree, the domain tags are separated from the non terminal symbols by a colon, which makes it easy to see directly the matching domain of a query.

### Tight coupling of Speech Recognition and Dialogue Management

When developing human-machine interfaces consisting of speech recognition and dialogue management, the context free grammar implementation of IBIS allows for a tight coupling between speech recognition and dialogue management. Therefore the same linguistic knowledge sources should be used in both components, so that IBIS can be used as a parser for the user queries. The parsed output of IBIS can be directly used by the dialogue manager to determine the user intention. Furthermore, especially when using clarification questions, the current dialogue context can be used to predict the context of the next user utterance and can be directly passed to the speech recognizer in form of top-level rule names to restrict the search space for the next decoding step[5].

The following script excerpts are showing the weighting mechanism of rules in IBIS to restrict the search space. All the entry rules of the used grammars are divided into two sets, a responseSet and a querySet. The responseSet consists only of rules, which are less likely to be used at the beginning of a dialogue, i.e. consists mainly of rules which cover all responses to clarification questions. The querySet contains all the rules which are most likely used at the beginning of the dialog.

```
proc weightRules { mode rules } {
    global par agent SID

    switch $mode {
        responseSet {
            # reset weights and clear cache
            cfgSet$SID.data weightRules _ALL_ -weight 0.000
            cfgSet$SID.data clear
            cfgSet$SID.data build -verbose 0

            set par(responseSet) $rules
            cfgSet$SID.data weightRules _ALL_ -weight -1.000
            set ret [cfgSet$SID.data weightRules $rules -weight -2.000]

            putsInfo "disabling rules ($ret): $rules"
        }
        enable {
            if { ![info exists par(responseSet)] } {
                set par(responseSet) _ALL_
            }
            weightRules responseSet $par(responseSet)

            set par(enableRules) $rules
            set ret [cfgSet$SID.data weightRules $rules -weight 0.000]

            putsInfo "enabling rules ($ret): $rules"
        }
    }
}
```

Before starting the decoding, i.e. during the initialization phase of the recognizer, a predefined set of rules can be loaded from a file, which consist of rules, which are usually not used at the beginning of a dialogue. These rules are used to define the responseSet and are penalized per default.

```
set fp [open $disableLst r]
while { [gets $fp rule] >= 0 } { lappend rules [string trim $rule {[]}] }
close $fp

weightRules responseSet $rules
```

Depending on the dialogue context, rules of that responseSet can be selected by the dialogue manager and given to the speech recognizer. These rules can now be enabled, i.e. preferred against all other rules.

```
if { [info exists infoA(ENRULES)] && [llength $infoA(ENRULES)] } {
    weightRules enable      $infoA(ENRULES)
} else {
    weightRules responseSet $par(responseSet)
}
```

### Expanding the Grammar on the fly

Another feature is, that grammars can be expanded on the fly by new rules or terminals without restarting the recognizer. Even new words can be added to the grammar and the search network on the fly.

```
# adding of a few new paths together with some new rules
# this does not add new words to the search network
cfg addPath {[_NT_last]} {( last but not least )}
cfg addPtah {s[test]} {( this is the first  sentence )}
cfg addPath {s[testSuite]} {( this is the second sentence )}
cfg addPath {s[testSuite]} {( *BLA the third )}
cfg addPath {s[testSuite]} {( *BLA fourth )}
cfg addPath {s[testSuite]} {( *BLA [_NT_last] the fifth )}
```

### Starting Over

By default, it is not possible to walk through the grammar more than once, when decoding a sentence. This might be okay for most applications, but for some others, it might restrict the way to communicate with the system too much. In these cases, you can reconfigure the parsing process, so that it will be possible to start again with the top level rules, when a final terminal in the grammar is reached. However, due to the extended search space, the recognition accuracy might get worse. To have an influence on this, it is possible to set a penalty for starting over. An example looks like:

```
# enable startover for all grammars with a penalty factor of 2.0
cfgInit         $SID -startover 2.0

# disable startover for one grammar in the set
cfgSet$SID.data.cfg(0) configure -startover -1.0
```

### Top Level Rules

In some cases it might be useful to allow the parsing to start at every rule defined in the grammar and not only at the top level rules. This can be done for e.g. the first grammar in the set by

```
cfgSet$SID.data.cfg(0) configure -allPublic 1
```

### Synchronize Dictionary

Using the functions defined in cfg.tcl it is possible to bring the dictionary in synchronization with the grammars, so that the words defined in the dictionary are limited to the grammar vocabulary. Therefore you should define at least the following variables in desc.tcl:

```
set ${SID}(cfg,dict)      $dictPath/nav.dict
set ${SID}(cfg,baseDict)  $dictPath/baseDict
```

With `basedict` a large background dictionary is defined, in which all words in the grammars have to be defined. The result of the synchronization can be found in `dict`. The initialization of the decoder then looks as follows:

```
cfgInit         $SID -makeDict 1
dictInit        $SID -desc [set ${SID}(cfg,dict)]
ibisInit        $SID -lm cfgSet$SID
```

### Out-Sourcing of Terminal Classes to SVMap

When working with large classes of terminals, like in the navigation domain a large number of street names, it is often helpful to out-source them from the grammar to the search vocabulary mapper (`SVMap`). This reduces the number of grammar accesses and therefore speeds up the recognition process. To use this functionality you have to use the initialization given in section 6.2.6 and should additionally define the following variable in `desc.tcl`:

```
set ${SID}(cfg,classes)   [list $dictPath/nav.classes]
```

The referred file defines a mapping between a terminal and its class identifier. An example of a mapping between street names looks as follows.

```
acherstra~se                   @street
adalbert-stifter-stra~se       @street
adenauerring                   @street
adlerstra~se                   @street
agathenstra~se                 @street
ahaweg                         @street
ahornweg                       @street
```

You have to use `@` as a class identifier.

### Handling of Noises

To cope with spontaneous non-verbal speech events and non-human noises, we are using the mechanism of filler words in the decoder. Filler words can potentially occur between any two terminals. Instead of asking the language model for their score, a predefined filler penalty is applied. A complete set of variables defined in `desc.tcl` together with the handling of noises as filler words looks then as follows (the variable `fillers` is added):

```
set ${SID}(cfg,grammars)  [list [list NAV \
                                     $cfgPath/cfg.ka.nav \
                                     $cfgPath/cfg.base.nav] \
                                [list SHARED \
                                     $cfgPath/cfg.shared]]
set ${SID}(cfg,dict)      $dictPath/nav.dict
set ${SID}(cfg,baseDict)  $dictPath/baseDict
set ${SID}(cfg,classes)   [list $dictPath/nav.classes]
set ${SID}(cfg,fillers)   [list $dictPath/nav.fillers]
```

The initialization differs only in one point from the initialization in section 6.2.6:

```
set dict [set ${SID}(cfg,dict)]
cfgInit         $SID -makeDict 1
dictInit        $SID -desc $dict
ibisInit        $SID -lm cfgSet$SID \
                     -vocabDesc $dict.v -mapDesc $dict.m
```

In the fillers file all noises are defined which should occur during decoding as filler words. An example looks as follows:

```
+click+
+interjection+
+interjection+(ah)
+pause+
```

To not loose too much in recognition accuracy, you need to tune the filler penalty on a development set. The configuration can be done as follows:

```
svmap$SID configure -filPen 60
```

### Visualization and Generation

The recursive transitions networks for a specific rule can be printed out and visualized using the AT&T FSM-Toolkit [9], e.g. for debugging purposes. Therefore the FSM-Toolkit has to be installed and all the executables has to be added to the PATH environment variable. Furthermore, 'dot' has to be installed. The following command produces the FSM description files for the given rule and uses the FSM-Toolkit and 'dot' to generate a postscript file (in this case rqPathDscrFSM.ps).

```
cfg fsm request-path-description rqPathDscrFSM
```

Also the terminal sequences produced by a specific rule can be printed out. Therefore the generation functions can be used. The terminal sequences can either be generated randomly according to the stored probability distribution in the transitions or fixed, by traversing all the transitions in a fixed order. In the latter case, no recursions are supported. Following are a few example commands.

```
cfg generate 10 -mode random
cfg generate 10 -mode random -recurse 1
cfg generate 10 -mode fixed

# generates 10 terminal sequences randomly with no recursions (default)
cfg:request-path-description generate 10

# generation can also be used in combination with starting over
cfg configure -startover 1
cfg generate 10
```

## 6.2.7 Decoding along Lattices

A Lattice can be seen as a constrain of your search space. This allows you to rescore lattices with new better acoustic modes without a full decoding. To that end, a lattice can be attached to a LTree. To allow a more flexible word graph, the lattice might be optimized with compress. After attaching the lattice, the decoding can be done as usual.

```
GLat glatX3 svmapX3
ltreeX3 constraint glatX3 -mode exact -type SVX
```

### 6.2.8 Run-On Recognition, partial traceback

For practical applications, the decoding should be run while receiving audio data and output partial results immediately. It is straightforward to write a Tcl loop for such purposes. The only thing to care, is to tell the decoder to not start from the beginning each time. Assuming a audio interface function *getAudio* is provided, the loop will look like this:

```
set myinit 1
while { [getAudioData] != 0 } {
  featureSet eval $uttInfo
  spass run -init $myinit
  set hypo   [spass.stab trace]
  set frameX [spass configure -frameX]
  puts "processed $frameX frames, got partial hypo $hypo"
  set myinit 0
}
```

### 6.2.9 Network Optimization

The default construction of the search network builds a tree structure. However, a more compact network can be obtained by using the method compress, which exploits redundancies in a more general way. Additionally, the whole search network might be dumped into a single file, allowing a faster startup of the decoder. If you load a dump file, you don't have to read other description files for the dictionary, vocabulary, mapper or even language model. At startup, you load the dump file by adding an option "-dump filename" at creating of the STree object.

```
streeX3 compress
streeX3 dump mydump.stree.gz
```

### 6.2.10 Dynamic Vocabularies

The IBIS decoder is designed to handle vocabularies dynamically, e.g. it is possible to add or delete words at runtime without reconstruction of the search network. To delete a word, it's actually not necessary to delete the word from the search network. You can also simply deactivate the word by removing the corresponding map entry from the SVMap object.

```
# add word
dictX3   add $newWord $newPron
svocabX3 add $newWord
svmapX3  add $newWord $lmClass -prob $classProb
streeX3  add $newWord
spassX3  reinit

# delete word
streeX3  delete $newWord
svmapX3  delete $newWord
svocabX3 delete $newWord
spassX3  reinit

# deactivation instead of deletion
svmapX3  delete $newWord
```

In particular, you can combine these techniques with run-on recognition to add unknown words on the fly by defining a time offset for the decoder reinitialization. This will allow the decoder to process that audio excerpt again to consider the added word at the correct time. The offset will be configured with "-START" option at the reinit method from the SPass object..

The default configuration of the IBIS decoder will allows you to process a vocabulary of 64k words. However, if you want to use larger vocabularies, you can simply change the defines for SVX and SVX_MAX in src/ibis/slimits.h and recompile.

```
typedef UINT     SVX;
#define SVX_MAX  UINT_MAX
```

### 6.2.11 Consensus Decoding

When doing ASR, what you really are interested in is word error rate (WER), not sentence error rate (SER), which however is what the standard beam search optimizes. Several approaches exist which do not try to minimize the overall score, but instead try to optimize the word error rate via confidence measures or introduce some kind of clustering between competing hypotheses in a lattice.

One such approach was developed by Lidia Mangu, when she was at John's Hopkins. Lidia Mangu's code can read our lattices when you write them with -format slf, implemented and documented by Florian.

IBIS implements this approach to "Consensus Lattice Processing", which allows you to decode, produce a lattice, compute confidence measures on it and the convert it into a confusion network, which you can then rescore for the most likely hypo. The sequence looks as follows:

```
set nodeDens  20
set postScale 2.0
set clpBeam   5.0
set silScale  1.0
set cutoff    0.1

...

lat$SID read $latIn/$utt.lat.gz
set  hypo [lindex [lat$SID rescore -v 1] 0]

svmap$SID load svmapCLP
svmap$SID  configure -wordPen $lp -lz $lz

lat$SID prune -nodeN [expr $nodeDens * [llength $hypo]]
lat$SID splitMW

lat$SID posteriori -scale [expr $postScale/$lz]
set  cons [lat$SID consensus -v 1 -beam $clpBeam -silScale $silScale -cutoff $cutoff]
```

As pronunciation probabilities need to be regarded differently during confidence computation (here, they are real probabilities, which sum up to 1, while during decoding they are mere scores), you might want to use a separate vocabulary mapper (and maybe LM for multi-words) for a-posteriori generation. It is usually a good idea to prune a lattice before computing posteriors. The consensus method computes the consensus on the probabilities filled in by posteriori, you can also compute a confusion network on several lattices at the same time by adding the -lats option.

The other parameters to consensus should be set with care for performance and time consumption.

Usually, the word-posteriors (confidence scores) generated using Consensus are superior to those generated by other methods (i.e. posteriori alone). If pruning takes too long, try using a simpler svmapLA. If it fails with interpolated LMs, try:

```
# Configure LM
printDo mlm1.MetaLM configure -mlctMax 1000000
printDo mlm2.MetaLM configure -mlctMax 1000000
printDo mlm2.MetaLM configure -lvxCache 100000
```

If computing the consensus takes too long, try reducing nodeDens or clpBeam. The resulting confusion networks can be converted into lattices, HMMs, ... and can be used for MMIE training, and many other purposes.

# Chapter 7

# Tips and Trouble-shooting

## 7.1 General

If you don't find the information you need in this documentation, there might be more information available on-line at http://isl.ira.uka.de/~jrtk/janus-doku.html. A recent addition to the JRTk documentation is the `Wiki` page available at http://www.is.cs.cmu.edu/janus/moin.cgi.[1] As this is meant to be a "discussion white-board", you might also find help for your problem there. If you do not find an answer to your problem there, please send e-mail to jrtk@ira.uka.de or directly to one of the maintainers, but be sure to add sufficient debugging information, so that others have a chance to trace the problem. The more (useful) information you provide and the better you can describe the problem, the more people will be able to help you ;-)

## 7.2 Installation

On Unix boxes, first make sure the `janus` binary is in your search `PATH`. If you can't run Janus by simply typing `janus` at the shell prompt, try:

```
(i13pc33:/home/metze) setenv PATH /home/metze/janus/scr/Linux.gcc/janus:${PATH}
```

If you do not use `tcsh` or your Janus binary is not in the above directory, you'll have to change nomenclature or path accordingly. Janus can be compiled with or without support for X11, so in some cases you may need to set the `DISPLAY` environment variable:

```
(i13pc33:/home/metze) setenv DISPLAY i13pc33:0.0
```

Note that e.g. `ssh -XC i13pc33` does not work properly when you redefine the `DISPLAY` environment variable, for example in your `.tcshrc`, if you're using this it is usually best to leave `DISPLAY` as it is.

This is the output of an interactive example trouble-shooting session under Linux fixing several common installation difficulties:

```
i13pc33 /home/data> janus
application-specific initialization failed: no display name and no
$DISPLAY environment variable
```

[1] Currently, this is accessible at http://penance.is.cs.cmu.edu/janus/moin.cgi.

```
% exit
i13pc33 /home/data> setenv DISPLAY i13pc33:0.0
i13pc33 /home/data> janus
# ======================================================================
#  ____  ____  _____ _
# |__  ||  _ \|_   _| | _     V5.1 P001 [Apr  9 2003 11:21:47]
#    | || |_| | | | | | |/ |  --------------------------------------------
#    | ||  _ <  | | |   <     University of Karlsruhe, Germany
#    | ||_| |_| |_| |_|\_|    Carnegie Mellon University, USA
#   _| | JANUS Recognition
#  \__/        Toolkit        (c) 1993-2002 Interactive Systems Labs
#
# ======================================================================
application-specific initialization failed: Can't find a usable init.tcl in the
following directories:
    /home/data/janus/src/../../library /home/data/janus/src/../../../library
This probably means that JanusRTk wasn't installed properly.
% exit
i13pc33 /home/data> setenv JANUS_LIBRARY /home/data/janus/library
i13pc33 /home/data> setenv   TCL_LIBRARY /usr/lib/tcl8.3
i13pc33 /home/data> setenv    TK_LIBRARY /usr/lib/tk8.3
i13pc33 /home/data> janus
# ======================================================================
#  ____  ____  _____ _
# |__  ||  _ \|_   _| | _     V5.1 P001 [Apr  9 2003 11:21:47]
#    | || |_| | | | | | |/ |  --------------------------------------------
#    | ||  _ <  | | |   <     University of Karlsruhe, Germany
#    | ||_| |_| |_| |_|\_|    Carnegie Mellon University, USA
#   _| | JANUS Recognition
#  \__/        Toolkit        (c) 1993-2002 Interactive Systems Labs
#
# ======================================================================
% puts $auto_path
/home/data/janus/library /home/data/janus/tcl-lib
/home/data/janus/gui-tcl /usr/lib/tcl8.3 /usr/lib
/home/data/janus/src/lib /usr/lib/tk8.3 /home/data/janus/library
%
```

If you encounter one of the above errors, you can add the problem-solving line to your start-up scripts. The Tcl-variable `auto_path` can also be changed in `.tcshrc`. As Janus is a Tcl/Tk application, you might also need to install the relevant libraries in the correct version and set up the environment variables `TCL_PATH` and `TK_PATH` accordingly (in the above example, the first of the three "setenv" lines will often suffice). Some versions of Janus might also be dynamically linked against `libreadline, libtermcap,` and `libcurses`.

## 7.3 Tcl-library problems

Normally, Tcl/Tk will automatically source the files in the "tcl-lib" and "gui-tcl" directories, when functions which are defined in those scripts are called. If you define new functions, you have to add them to the index file `tclIndex`, which you'll find in both directories. The standard way to recreate this file is to issue the following commands to an instance of janus started in the "tcl-lib" or the "gui-tcl" directory:

```
file delete tclIndex
auto_mkindex $JANUSLIB *.tcl
```

## 7.4 Object problems

Janus (more exactly: the Tcl/ Tk interface) uses ":" to access list elements and "." to access sub-objects. For example, the dictionary word `HAS` can be accessed as `dict$SID:HAS` and `dict$SID.item(100)`. Now it is perfectly legal to have subobjects, whose name contains a ".", although you won't be able to access it via ":" now, because the "." in the list item name will be interpreted as a subobject. The solution is to avoid the "." altogether or to access your list item as `dict$SID.item([dict$SID.list index HAS])`.

## 7.5 The fgets-problem

If Janus blocks (hangs) as soon as it tries to lock access to a file via `fgets`, the best solution is to set up an NGets-server by editing the following lines in your `.janusrc` (see the example in `~/janus/scripts/janusrc`)

```
set NGETS(HOST)         ""
set NGETS(PORT)         63060
```

to look somewhat like this:

```
set NGETS(HOST)         i13s8
set NGETS(PORT)         63050
```

You can choose any combination of HOST and PORT you like, but the HOST should be a reliable machine (SUNs are great) and the PORT should not be used for system services or somebody else's NGets-server. You should now start the server on the reliable machine using

```
(i13s8:/home/metze) janus janus/tcl-lib/ngetsGUI.tcl -server
# ======================================================================
#  ____  ____  _____ _
# |__  ||  _ \|_   _| | _     V5.0 P011 [Nov 13 2002 17:17:29]
#    | || |_| | | | | |/ |  --------------------------------------------
#    | ||  _ <  | | |   <     University of Karlsruhe, Germany
#    | ||_| |_| |_| |_|\_|    Carnegie Mellon University, USA
#   _| | JANUS Recognition
#  \__/        Toolkit        (c) 1993-2002 Interactive Systems Labs
#
# ======================================================================
Server accepting connection on 63060 ...
CurrentSock: sock5
```

or, if you don't want the graphical interface,

```
(i13s8:/home/metze) janus
# ======================================================================
#  ____  ____  _____ _
# |__  ||  _ \|_   _| | _     V5.0 P012 [Nov 27 2002 14:43:58]
#    | || |_| | | | | |/ |  --------------------------------------------
#    | ||  _ <  | | |   <     University of Karlsruhe, Germany
#    | ||_| |_| |_| |_|\_|    Carnegie Mellon University, USA
#   _| | JANUS Recognition
#  \__/        Toolkit        (c) 1993-2002 Interactive Systems Labs
#
# ======================================================================
% ngetsServerStart
Server accepting connection on 63060 ...
CurrentSock: sock5
0
%
```

This NGets-server process will now handle all calls to `fgets` and `glob` for all other processes. You can test this setup by generating a simple file `/home/metze/x` containing a few lines of text in your home directory and then executing Janus in your home directory (assuming you started a server as above):

```
(i13pc33:/home/metze) echo "Line" >> x
(i13pc33:/home/metze) echo "Line two" >> x
(i13pc33:/home/metze) janus
# ======================================================================
#  ____  ____  _____ _
# |__  ||  _ \|_   _| | _     V5.0 P012 [Nov 27 2002 14:40:14]
#    | || |_| | | | | |/ |  --------------------------------------------
#    | ||  _ <  | | |   <     University of Karlsruhe, Germany
#    | ||_| |_| |_| |_|\_|    Carnegie Mellon University, USA
#   _| | JANUS Recognition
#  \__/        Toolkit        (c) 1993-2002 Interactive Systems Labs
#
# ======================================================================
INFO: Using NGETS on i13s10:63060!
% fgets x line
4
% puts $line
Line
% fgets x line
10
% puts $line
{Line two}
% fgets x line
-1
%
```

Be aware that this server variant reads the file in memory once and will only write it back when all the entries have been processed by client processes. If your jobs die and you want to restart the jobs, you can simply select the file and click on "Clear" in the `ngetsGUI.tcl` interface window. You can check if a process is using an NGets server by looking for the line

```
INFO: Using NGETS on i13s10:63060!
```

at startup.

## Background

"fgets" is an important Tcl-function, which is used in most parts of Janus to parallelize jobs on different machines. The Janus Library (described in chapter lib) makes extensive use of it, as do our standard testing scripts.

"fgets" is implemented in C (~/janus/src/itf/itf.c in case you want to have a look). If you run JANUS on a single machine, using

```
while {[fgets spkList spk] != -1} {
    puts $spk
}
```

is equivalent to

```
set fp [open spkList r]
while {[gets $fp line] != -1} {
    puts $spk
}
close $fp
```

Both scripts will print out the contents of the file spkList. If, however you run the same script on different machines on the same file and at the same time, you will notice the difference: The first version will "divide" the list between the different machines, while the second version will print the whole list on every machine. Also, if you have a look at the file after you ran the first script, you will notice that the first character of every line is no "#". Running this script on such a file will produce no output, because it "believes" that all "keys" (lines) have already been processed (output) by another machine. It is therefore a good idea to keep backup copies of speaker lists etc. around.

On some machines or operating systems (e.g. Linux with certain nfs implementations), this mechanism does not work reliably, because exclusive file locking cannot be guaranteed, e.g. two machines can read and write to one file at the same time. The easiest solution to this problem is to re-define "fgets" in Tcl and replace this mechanism by something else, i.e. a server that reads files and listens on ports. Such an approach is implemented in ~/janus/tcl-lib/ngets.tcl, and ~/janus/tcl-lib/ngetsGUI.tcl.

## 7.6 Catching aborts

Janus is implemented in C. Some program faults will therefore be caused by segmentation violations. C has handlers to catch a seg-fault signal and execute specific code. The relevant procedure is called ~/janus/src/itf/itc.c:janusSignalHandler and can be used to send mail or do something else if you define a procedure "janusErrorHandler" at Tcl-level.

Code like this (in combination with other approaches) can be very useful in maximising CPU load during evaluation times, while it will not improve the quality of the code. If you get aborts, it will be best to debug the code .

An example procedure that will send e-mail if Janus crashes inexpectedly looks like this:

```
proc janusErrorHandler { sig } {
    global errorInfo errorCode argv argv0 env

    set sigN [lindex "NONE SIGHUP SIGINT SIGQUIT SIGILL 5 SIGABRT 7 \
                     SIGFPE SIGKILL 10 SIGSEGV 12 SIGPIPE SIGALRM SIGTERM" $sig]
    regsub   "\\..*" [info hostname] "" host
    set exe  [info nameofexecutable]
    set cmd  "$argv0 $argv"
    set pwd  $env(PWD)
    set mail $env(USER)@ira.uka.de
    switch $sigN {
        SIGABRT -
        SIGFPE  -
        SIGSEGV {
            janusSendMail $mail \
                "$sigN $host $pwd: $argv0" \
                "$host.[pid] $pwd:\n$exe $cmd\n[string repeat - 72]\n$errorInfo"
        }
        default {
            puts stderr "\nReceived signal $sig ([lindex $sigL $sig]).\n"
        }
    }
}

proc janusSendMail { address subject body } {
    exec echo $body | mailx -s $subject $address
}
```

Define these procedures in your `.janusrc` and you'll receive e-mails when janus seg-faults. A system to notify the user of all possible errors is however difficult to realise :-(.

## 7.7 Filesystem issues

Janus can read compressed files transparently. In some cases the piping mechanism used however causes problems, so if you see an I/O-related problem on compressed files, try working with uncompressed (or local) files first.

Accumulating and particularly combining ML accumulators can pose a heavy burden on distributed filesystems. If you want to guarantee the execution of the server part of the doParallel loop on a particular machine (i13pc44 in this case), for example because this machine holds the data locally or has a very fast network connection, you can include the following code in your `.janusrc`:

```
proc doParallelServer { } {
    set SERVER [lindex [glob -nocomplain "i13pc44.*.INIT"] 0]
    if {$SERVER == ""} {
        set SERVER [lindex [glob -nocomplain "i13pc4\[0-6\].*.INIT"] 0]
    }
    if {$SERVER == ""} {
        set SERVER [lindex [lsort -decreasing [glob -nocomplain "i13pc3*.*.INIT"]] 0]
    }
    if {$SERVER == ""} {
        set SERVER [lindex [glob -nocomplain "i13pc5\[0-1\].*.INIT"] 0]
    }
    if {$SERVER == ""} {
        set SERVER [lindex [glob -nocomplain "i13pc2*.*.INIT"] 0]
    }
    if {$SERVER == ""} {
        set SERVER [lindex [lsort [glob "*.INIT"]] end]
    }
    return [string range $SERVER 0 [expr [string length $SERVER]-6]]
}
```

If i13pc44 is not available, this procedure will choose the next-best machine and so on.

## 7.8 featAccess and featDesc

The `featAccess` and `featDesc` file serve to define where to find acoustic data and how to process it. They are in fact Tcl scripts evaluated in a separate interpreter. The reason to hold them separately is to allow for greater flexibility when porting systems between tasks, architectures, or sites.

The fact that these scripts are evaluated as Tcl-scripts in a separate interpreter limits the scope of variables; if you're experiencing error messages stemming from featAccess or featDesc, debugging can be a bit tedious, because you cannot run the scripts interactively and determine which variables are visible or which commands fail (and for what reason).

```
% featureSet$SID eval $uttInfo
  warp /project/MT/data/ESST/cd28/e044a/e044ach2_039.16.shn with factor
1.000
ERROR   matrix.c(2080)      expected ’float’ type elements.
ERROR   itf.c(0359) <ITF,FCO>  Failed to create ’dummyS’ object.
ERROR   itf.c(0720)         featureSetEval<featureSetQ4g> featureSetQ4g
{{spk MBB_e044ach2} {utt e044ach2_039_MBB} MBB_e044ach2 {EDUCATION
graduate} {PROFESSION student} {NATIVE_LANG e} {SEX m} {ID MBB} {KEY
MBB_e044ach2} {DIALECT American English} {DATE_OF_BIRTH 710808}
{PRIMARY_SCHOOL Louisville, KY} {SEGS e044ach2_001_MBB e044ach2_003_MBB
e044ach2_150_MBB e044ach2_152_MBB} e044ach2_039_MBB {ADC e044ach2_039.16}
{ID MBB} {LM yeah #NIB_H## #NIB_H## though it is #NIB_GE# what #NIB_UM#
seven hours #NIB_GE#} {TEXT yeah #NIB_H## #NIB_H## though it is #NIB_GE#
what #NIB_UM# seven hours #NIB_GE#} {CHANNEL e044ach2} {PATH cd28/e044a}
{KEY e044ach2_039_MBB} {TIME 4.181} {SPEAKER MBB_e044ach2}}:
```

In this example, you can determine that the error occured during the evaluation of featDesc (`featureSetEval<featureSetQ4g> featureSetQ4g`); the exact kind and location of the error (the subtraction of spectral means failed because none were loaded) is usually determined by the insertion of several `puts ‘‘Now I’m here ...’’` and `puts ‘‘WARPFACTOR=$WARP’’` lines in `featDesc`.

## 7.9 Score functions

Janus spends most of its time doing score computations. This is usually done in a highly optimized routine called `ssa_opt`. This routine makes a few assumptions makes a few assumptions on the underlying acoustic models, namely:

- They are fully continuous
- All codebooks share the same feature

So, if you use multiple STC classes or you're building a semi-continuous system, be sure to add the following line to your decoding script:

```
senoneSet$SID setScoreFct base
```

Also, if your acoustic scores look really bad, you might try this line, too.

## 7.10 Labels and Dictionaries

Labels store pre-computed time-alignments as computed by the Viterbi or Forward-Backward algorithm. If you're using labels and you get error messages stating

```
Couldn't map 234 of 1234 path items.
```

or the results from training are unreasonable, usually your `Path` (labels) and your current `HMM` construction don't match. Labels store state indices, i.e. "frame X occupies the HMM state(s) Y (and Z)". If the HMM object associated with the Path object when reading the labels was built differently from the one used during label writing, the indexing will be different (i.e. skewed in time) and the labels are essentially useless. Typical culprits changed during HMM construction are:

- A modified `Dictionary`
- The `-optWord` and `-variants` flags to HMM `make`
- Different transcriptions (filtered differently, more pauses, etc.)

If you want to change any of the above, your time alignments will change anyway, so you'll need to write new labels. In some cases it is possible to re-use old (Viterbi) labels by creating the old and new HMMs side by side and re-configuring the path items' `-stateX` by hand (you'll have to create them by `bload` or some other method), but you better know exactly what you're doing or your results will be bogus.

## 7.11 Language Models

Janus can read in ARPABO-format language models. However, depending on the sorting order of n-grams, this can take a long time or simply fail. In this case, try to re-sort the file so that the first column is sorted first and/ or try to load the uncompressed file. If you experience trouble reading a compressed LM file (as in dump files), read on with section 7.12.

## 7.12 Defines and Dump Files

The Ibis decoder stores files in a compressed binary format for reasons of I/O speed and memory consumption on disk. Particularly, the STree `dump` contains a `Dictionary`, `SenoneSet`, `SVocab`, `PHMMSet`, `LCMSet`, `RCMSet`, and possibly an `XCMSet` as well as an `LingKS`/ `SVMap`.

If you're trying to read a dump file and you're getting an error message, chances are the system the dump was written with a system which differs from the system you're reading the dump with in some aspect. Favourites are:

**Objects:** used in one system, but not the other, particularly the `XCMSet`. Also, when loading a dump, the object reading the dump has to be empty, i.e. it has to exist, but cannot contain data.

**Configuration:** some objects are configured differently or contain a different number of items.

**Language Models:** the `MetaLM` relies on other, lower-level models and only stores the top-level information in its dump file; therefore the lower LMs have to be initialized properly before reading a dump into a `MetaLM`.

**Executable:** Janus and the Ibis decoder depend on a few compile-time `#define`s and `typedef`s set in `src/ibis/slimits.h`. See section 2.6 for information on how to compile Janus and the meaning of these flags. If you have two executables compiled with different settings, you will most likely not be able to exchange dump-files between the two, because the internal representation of data is different.

## 7.13 Speed

If you find your training or decoding is taking too long, chances are you're working on an ill-conditioned problem, e.g. you'll have to think about a more intelligent setup.

The first step in finding out why your jobs run so slowly is to pinpoint the part of the code which takes up most of the time. Frequently, your job is just to big (takes up too much memory) and the machine starts "swapping". If this is the case, try moving to another machine or look at section 7.14. If this is not the case, usual suspects are:

**I/O:** it can take a long time to load data into memory, if it's stored on network disks or is distributed over many small files. Try to move the data to a local disk or reduce the number of individual files by using a differently organized `DBase`, a more intelligent `featDesc` and `featAccess` or by using a `Labelbox` (c.f. sections 7.8 and 7.7).

**Decoding:** Try reducing the beams used in `SPass` or use a different scoring function by using `setScoreFct`. Popular speed-up techniques include the use of a `BBITree` and `fmatch`. Sometimes it is necessary to use profiling to see where exactly the time is spent.

If you're using the "base" scoring function (`setScoreFct`), try to configure the Gaussian evaluation threshold differently, i.e. set `-expT` to a higher cutoff such as in

```
codebookSet$SID:SIL-m.cfg configure -expT -10.0.
```

This should speed up your decoding by up to 20% without loss in accuracy. Also see section 7.9 on scoring functions.

**Language Models:** Complicated language models can take a long time to evaluate, try using a simpler language model for decoding, use a simple n-gram `LingKS` as look-ahead by using the `-map` option when creating an `LTree`, and/ or play with the `-cache` and `-mode` settings in `LTree`.

You can also check the compile-time settings of Janus. You might find a setting which optimizes speed for your particular hardware.

## 7.14 Memory Consumption

If you find your training or decoding uses too much memory, chances are you're working on an ill-conditioned problem, e.g. you'll have to think about a more intelligent setup.

You can reduce the memory footprint of the executable by compiling Janus as "Ibis" only (c.f. section 2.6) or by using different `#defines` and `typesets` for the `LVX` and `SVX` types used in the decoder (see sections 2.6 and 7.12).

# Chapter 8

# Modules

The structure of this section is according to the organization of the source code.

## 8.1 Base modules (src/base)

### 8.1.1 `CMatrix`

This section describes the '*CMatrix*': *Matrix of char values*

**Creation:** `CMatrix` <`name`>

`name` name of the object

**Configuration:** `cmatrix configure`

`-m` = 1
`-n` = 1

**Methods:** `cmatrix`

`puts`

print matrix contents as TCL list

### 8.1.2 `DBase`

This section describes the '*DBase*': *DBase*

**Creation:** `DBase` <`name`>

`name` name of the object

**Methods:** `dbase`

`add` <`key`> <`list`>

add record to database

`key` key
`list` list of varName varValue

`close`

close database

`delete <key>`
: delete record from database

| | |
|---|---|
| `key` | key |

`first`
: get first key in database

`get <key>`
: get record from database

| | |
|---|---|
| `key` | key |

`list`
: list all keys in database

`next`
: get next key in database

`open <file> <index> [-ptrSize ptrsize] [-mode mode]`
: open database

| | |
|---|---|
| `file` | name of database file |
| `index` | name of index file |
| `ptrsize` | size of pointer (use when creating only, global parameter) |
| `mode` | r — rw — rwc |

`read <filename>`
: add records from file to database

| | |
|---|---|
| `filename` | name of file to read |

`uttFilter <dbase> <uttID>`
: filter utterance in foreachSegment (dbaseUttFilter)

| | |
|---|---|
| `dbase` | database name (not object) |
| `uttID` | utterance ID |

`uttInfo <dbase> <spkID> <uttID>`
: find utterance information (dbaseUttInfo1)

| | |
|---|---|
| `dbase` | database name (not object) |
| `spkID` | speaker ID |
| `uttID` | utterance ID |

`write <filename>`
: write records from database to file

| | |
|---|---|
| `filename` | name of file to read |

**Subobjects:**

`dbaseIdx` (`DBaseIdx`)

### 8.1.3 DBaseIdx

This section describes the '*DBaseIdx*': *DBase Index Object*

**Creation:** `DBaseIdx` cannot be created directly.
It is accessible as a sub-object of `DBase`!

**Configuration:** `dbaseidx configure`

`-hashSizeX` = 2

**Methods:** `dbaseidx`

`add <key> <offset> <size>`

add record to index

| | |
|---|---|
| `key` | key |
| `offset` | offset |
| `size` | size |

`close`

close index database

`delete <key>`

delete record from index

| | |
|---|---|
| `key` | key |

`first`

get first key in index file

`get <key>`

get record from index

| | |
|---|---|
| `key` | key |

`list`

list all keys in index file

`next`

get next key in index file

`open <filename> [-mode mode]`

open index file

| | |
|---|---|
| `filename` | name of index file |
| `mode` | r — rw — rwc |

### 8.1.4 DMatrix

This section describes the '*DMatrix*': *Matrix of double values*

**Creation:** `DMatrix <name> <matrix>`

| | |
|---|---|
| `name` | name of the object |
| `matrix` | @filename or name or definition |

**Configuration:** `dmatrix configure`

| | |
|---|---|
| `-count` | = 0.000000 |
| `-m` | = 1 |
| `-n` | = 1 |

**Methods:** `dmatrix`

`:=`

assign matrix (equiv. to 'copy')

`FMatrix`

convert from a FMatrix

`bload <filename>`

load matrix

`filename`

`bsave` <`filename`>

save matrix

`filename`

`clear`

set all matrix values to 0

`copy`

copy matrix

`det` [`-format format`]

compute determinant

| | |
|---|---|
| `format` | format string |

`eigen` <`matrix`> [`-iter iter`] [`-thresh thresh`] [`-clean clean`] [`-sort sort`]

eigenvalues and vectors of symmetric matrix

| | |
|---|---|
| `matrix` | matrix to hold eigenvectors |
| `iter` | max. number of iterations |
| `thresh` | threshold for max. non diagonal element |
| `clean` | clean up eigenvalue matrix |
| `sort` | sort eigenvalues |

`get` <`1st index`> <`2nd index`>

get a single entry from a matrix

| | |
|---|---|
| `1st index` | first index |
| `2nd index` | second index |

`inv` <`matrix`>

inverse of matrix using svd

`matrix`

`mul` <`matrix`> <`matrix`>

matrixA * matrixB

| | |
|---|---|
| `matrix` | matrix A |
| `matrix` | matrix B |

`mulot` <`matrix`> <`matrix`>

matrixA * matrixB'

| | |
|---|---|
| `matrix` | matrix A |
| `matrix` | matrix B |

`puts`

print matrix contents as TCL list

`resize`

resize matrix

`set` <`1st index`> <`2nd index`> <`value`>

set a single entry in a matrix

| | |
|---|---|
| `1st index` | first index |
| `2nd index` | second index |
| `value` | value |

`simdiag` <`matrix`> <`matrix`> <`matrix`> [`-iter iter`] [`-thresh thresh`]

simultaneous diagonalisation

| | |
|---|---|
| `matrix` | matrix with eigenvalues |
| `matrix` | total scatter matrix |
| `matrix` | within scatter matrix |
| `iter` | max. number of iterations |
| `thresh` | threshold for max. non diagonal element |

`svd <matrix> <matrix> [-clean clean]`

singular value decomposition

| | |
|---|---|
| `matrix` | matrix W to hold singular values |
| `matrix` | matrix V to hold basis of nullspace |
| `clean` | clean up singular values |

`trans`

transpose matrix

`unity`

make matrix a unity matrix

### 8.1.5 DVector

This section describes the 'DVector': *Vector of double values*

**Creation:** `DVector <name> <vector>`

| | |
|---|---|
| `name` | name of the object |
| `vector` | @filename or name or definition |

**Configuration:** `dvector configure`

| | |
|---|---|
| `-count` | = 0.000000 |
| `-n` | = 1 |

**Methods:** `dvector`

`:= <dvector>`

assign vector (equiv. to 'copy')

`dvector`

`copy <dvector>`

copy vector

`dvector`

`puts`

print vector as TCL list

`resize <dimension>`

resize vector

`dimension`

### 8.1.6 FBMatrix

This section describes the 'FBMatrix': *Band matrix of float values*

**Creation:** `FBMatrix <name> <matrix>`

| | |
|---|---|
| `name` | name of the object |
| `matrix` | @filename or name or definition |

**Methods:** `fbmatrix`

`display <canvas> [-width width] [-height height] [-x x] [-y y] [-min min] [-max max] [-tag tag]`

display fbmatrix

`canvas`
`width`
`height`
`x`
`y`
`min`
`max`
`tag`

`linear [-N n] [-p p] [-rate rate] [-low low] [-up up]`

linear filterbank

| | |
|---|---|
| `n` | number of filters |
| `p` | number of (power) points |
| `rate` | sampling rate in Hz |
| `low` | lowest frequency in Hz |
| `up` | highest frequency in Hz, 0 means rate/2 |

`mel [-N n] [-p p] [-rate rate] [-low low] [-up up]`

melscale filterbank

| | |
|---|---|
| `n` | number of filters |
| `p` | number of (power) points |
| `rate` | sampling rate in Hz |
| `low` | lowest frequency in Hz |
| `up` | highest frequency in Hz, 0 means rate/2 |

`meltra [-rate rate] [-p p]`

trapezoid shaped melscale filterbank

| | |
|---|---|
| `rate` | sampling rate in Hz |
| `p` | number of (power) points |

`meltri`

triangular shaped melscale filterbank

`puts`

print matrix contents as TCL list

### 8.1.7 FCovMatrix

This section describes the '*FCovMatrix*': *Covariance matrix type (float)*

**Creation:** `FCovMatrix` cannot be created directly.

It is accessible as a sub-object of `Codebook`!

**Configuration:** `fcovmatrix configure`

| | |
|---|---|
| `-det` | = 0.000000 |
| `-type` | = DIAGONAL |
| `-useN` | = 0 |

**Methods:** `fcovmatrix`

`+=` `<source>` [`-scale scale`] [`-alpha alpha`]

add two scaled covariance matrices

| | |
|---|---|
| `source` | source covariance matrix (`FCovMatrix`) |
| `scale` | scaling of the destination |
| `alpha` | scaling of the source |

`:=` `<source>`

copy covariance matrix

| | |
|---|---|
| `source` | source covariance matrix (`FCovMatrix`) |

`clear`

clear the contents of an covariance accumulator

`set` `<matrix>`

set the covariance matrix

| | |
|---|---|
| `matrix` | matrix of covariance vectors |

`variances`

returns a list of the variances along the axis

## 8.1.8 FMatrix

This section describes the '*FMatrix*': *Matrix of float values*

**Creation:** `FMatrix <name> <matrix>`

| | |
|---|---|
| `name` | name of the object |
| `matrix` | @filename or name or definition |

**Configuration:** `fmatrix configure`

| | |
|---|---|
| `-count` | = 0.000000 |
| `-m` | = 1 |
| `-n` | = 1 |

**Methods:** `fmatrix`

`:=`

assign matrix (equiv. to 'copy')

`DMatrix`

convert from a DMatrix

`add` `<a>` `<fmatrixA>` `<b>` `<fmatrixB>`

a * matrixA + b * matrixB

| | |
|---|---|
| `a` | |
| `fmatrixA` | |
| `b` | |
| `fmatrixB` | |

`addvec` `<fmatrixA>` `<a>` `<fvectorV>` `<b>`

a * matrixA + b * vectorB

| | |
|---|---|
| `fmatrixA` | |
| `a` | |
| `fvectorV` | |
| `b` | |

`append` `<matrix>` `<matrix>` `<where>`

append matrixB to matrixA

| | |
|---|---|
| `matrix` | matrix A |
| `matrix` | matrix B |
| `where` | in above, below, left, right |

`bappend <filename>`

append matrix to binary file

| | |
|---|---|
| `filename` | |

`bic <clusterN> [-lambda lambda] [-iter iter] [-eps eps]`

Bayesian Information Criterion

| | |
|---|---|
| `clusterN` | number of cluster |
| `lambda` | penalty term |
| `iter` | maximal iteration for kmeans |
| `eps` | minimal distortion |

`bload <filename> [-im im] [-append append]`

load matrix from binary file

| | |
|---|---|
| `filename` | |
| `im` | ignore m in file header |
| `append` | append file to matrix |

`bmulot`

matrixA * bandmatrixB'

`bsave <filename>`

save matrix to binary file

| | |
|---|---|
| `filename` | |

`clear`

set all matrix values to 0

`cload <filename> [-append append]`

load matrix from compressed file

| | |
|---|---|
| `filename` | |
| `append` | append file to matrix |

`cluster [-minM minm] [-maxM maxm] [-variance variance]`

create optimal codebook

| | |
|---|---|
| `minm` | minimal size of output matrix |
| `maxm` | maximal size of output matrix |
| `variance` | maximal variance when clustering |

`copy`

copy matrix

`cosine <m> <n> [-type type]`

create cosine transformation matrix

| | |
|---|---|
| `m` | |
| `n` | |
| `type` | |

`csave <filename> [-mode mode]`

save matrix to compressed file

| | |
|---|---|
| `filename` | filename |
| `mode` | extra compression modes: rl, none |

`det [-format format]`

compute determinant

| | |
|---|---|
| `format` | format string |

`dev <matrix> <matrix>`

matrixA * matrixB

| | |
|---|---|
| `matrix` | mean values |
| `matrix` | squared mean values |

`display` <canvas> [-width width] [-height height] [-borderwidth borderwidth] [-dx dx] [-dy dy] [-space space] [-x x] [-y y] [-from from] [-to to] [-mode mode] [-grey grey] [-min min] [-max max] [-tag tag] [-outline outline]

display matrix

`canvas`
`width`
`height`
`borderwidth`
`dx`
`dy`
`space`
`x`
`y`
`from`
`to`
`mode`
`grey`
`min`
`max`
`tag`
`outline`

`fromSample` <fmatrix> <a>

convert sample to kmeans'able FMatrix

`fmatrix`
`a`

`get` <1st index> <2nd index>

get a single entry from a matrix

| | |
|---|---|
| `1st index` | first index |
| `2nd index` | second index |

`iload` <filename>

load matrix from IBM file

`filename`

`isave` <filename>

save matrix to IBM file

`filename`

`load`

load matrix from file

`minmax`

gives minimum and maximum

`mload` <filename> [-idx idx] [-append append]

load matrix from Matlab file

| | |
|---|---|
| `filename` | |
| `idx` | index of matrix to load |
| `append` | append file to matrix |

`modulo` [-mod mod] [-max max] [-start start]

modulo matrix

| | |
|---|---|
| `mod` | modulo factor |
| `max` | maximum count |
| `start` | don't module first data |

`mul` <`matrix`> <`matrix`>

matrixA * matrixB

| | |
|---|---|
| `matrix` | matrix A |
| `matrix` | matrix B |

`mulcoef` <`fmatrixA`> <`fmatrixB`> [`-a a`] [`-div div`] [`-mode mode`]

multiply each coefficient

| | |
|---|---|
| `fmatrixA` | |
| `fmatrixB` | |
| `a` | |
| `div` | division instead multiplication |
| `mode` | mode 0, 1 or -1 for dimesion(result) =, max or min of input |

`mulot`

matrixA * matrixB'

`neuralGas` <`matrix`> [`-maxIter maxiter`] [`-tempS temps`] [`-tempF tempf`] [`-counts counts`] [`-step step`] [`-init init`]

neural gas clustering

| | |
|---|---|
| `matrix` | matrix of sample vectors |
| `maxiter` | number of iterations |
| `temps` | start temperature (0=k-means) |
| `tempf` | temperature multiplyer |
| `counts` | vector with counts |
| `step` | only take every Nth sample |
| `init` | initialize with random samples |

`perceptron` [`-bias bias`] [`-log log`] [`-type type`]

perform perceptron operations for MLP

| | |
|---|---|
| `bias` | the bias FVector |
| `log` | return logarithmic output |
| `type` | type: softmax or sigmoid |

`puts` [`-ib ib`] [`-ie ie`] [`-jb jb`] [`-je je`] [`-format format`] [`-left left`] [`-right right`] [`-middle middle`]

print matrix contents as TCL list

| | |
|---|---|
| `ib` | start row |
| `ie` | end row |
| `jb` | start column |
| `je` | end column |
| `format` | format string |
| `left` | left side |
| `right` | right side |
| `middle` | between coefficients |

`resize` <`1st dimension`> <`2nd dimension`>

resize matrix

| | |
|---|---|
| `1st dimension` | first index |
| `2nd dimension` | second index |

`scatterPlot` <`canvas`> [`-width width`] [`-height height`] [`-x x`] [`-y y`] [`-xindex xindex`] [`-yindex yindex`] [`-from from`] [`-to to`] [`-xmin xmin`] [`-xmax xmax`] [`-ymin ymin`] [`-ymax ymax`] [`-tag tag`] [`-line line`] [`-p p`]

scatter plot

| | |
|---|---|
| `canvas` | |
| `width` | |
| `height` | |
| `x` | left side |
| `y` | upper side |
| `xindex` | |
| `yindex` | |
| `from` | |
| `to` | |
| `xmin` | |
| `xmax` | |
| `ymin` | |
| `ymax` | |
| `tag` | |
| `line` | draw lines |
| `p` | point size |

`set <1st index> <2nd index> <value>`

set a single entry in a matrix

| | |
|---|---|
| `1st index` | first index |
| `2nd index` | second index |
| `value` | value |

`square <FMatrix> [-index index]`

convert a row to a square matrix or vice-versa

| | |
|---|---|
| `FMatrix` | float matrix (`FMatrix`) |
| `index` | index |

`trans`

transpose matrix

`vts <a> <fmatrixA> <b> <fmatrixB>`

a * matrixA + b * matrixB

`a`
`fmatrixA`
`b`
`fmatrixB`

`window <FMatrix> <1st index> <2nd index>`

window matrix (into other matrix at offset)

| | |
|---|---|
| `FMatrix` | float matrix (`FMatrix`) |
| `1st index` | first index |
| `2nd index` | second index |

### 8.1.9 FVector

This section describes the '*FVector*': *Vector of float values*

**Creation:** `FVector <name> <vector>`

| | |
|---|---|
| `name` | name of the object |
| `vector` | @filename or name or definition |

**Configuration:** `fvector configure`

| | |
|---|---|
| `-count` | = 0.000000 |
| `-n` | = 1 |

**Methods:** `fvector`

`:=` `<fvector>`

assign vector (equiv. to 'copy')

`fvector`

`add` `<a>` `<fvectorA>` `<b>` `<fvectorB>` `[-mode mode]`

add two vectors

| | |
|---|---|
| `a` | |
| `fvectorA` | |
| `b` | |
| `fvectorB` | |
| `mode` | mode 0, 1 or -1 for dimension(result) =, max or min of input |

`bload` `<filename>`

load vector from binary file

`filename`

`bsave` `<filename>`

save vector to binary file

`filename`

`copy` `<fvector>`

copy vector

`fvector`

`norm`

norm of the vector

`puts` `[-format format]` `[-middle middle]`

print vector as TCL list

| | |
|---|---|
| `format` | format string |
| `middle` | between coefficients |

`resize` `<dimension>`

resize vector

`dimension`

### 8.1.10 IMatrix

This section describes the '*IMatrix*': *Matrix of integer values*

**Creation:** `IMatrix <name> <matrix>`

| | |
|---|---|
| `name` | name of the object |
| `matrix` | @filename or name or definition |

**Configuration:** `imatrix configure`

| | |
|---|---|
| `-m` | = 1 |
| `-n` | = 1 |

**Methods:** `imatrix`

`:=`

assign matrix (equiv. to 'copy')

`bload` `<filename>` `[-im im]`

load matrix from binary file

| | |
|---|---|
| `filename` | |
| `im` | ignore m in file header |

`bsave` <filename>

save matrix to binary file

| | |
|---|---|
| `filename` | |

`clear`

set all matrix values to 0

`copy`

copy matrix

`get` <1st index> <2nd index>

get a single entry from a matrix

| | |
|---|---|
| `1st index` | first index |
| `2nd index` | second index |

`puts`

print matrix contents as TCL list

`resize`

resize matrix

`set` <1st index> <2nd index> <value>

set a single entry in a matrix

| | |
|---|---|
| `1st index` | first index |
| `2nd index` | second index |
| `value` | value |

### 8.1.11 List

This section describes the '*List*': *List of indexed items*

**Creation:** `List` cannot be created directly.

It is accessible as a sub-object of `QuestionSet`!

**Configuration:** `list configure`

| | |
|---|---|
| `-blkSize` | = 50 |
| `-itemN` | = 0 |

**Methods:** `list`

`delete` <item>

remove distribution from the set

| | |
|---|---|
| `item` | name of item in list |

`index` <names*>

translate names to indices

| | |
|---|---|
| `names*` | list of names |

`name` <idx*>

translate indices to names

| | |
|---|---|
| `idx*` | list of indices |

**Subobjects:**

| | |
|---|---|
| `list` | (`List`) |

### 8.1.12 SVector

This section describes the '*SVector*': *Vector of short values*

**Creation:** `SVector <name> <vector>`

| | |
|---|---|
| `name` | name of the object |
| `vector` | @filename or name or definition |

**Methods:** `svector`

`:= <svector>`
assign vector (equiv. to 'copy')
`svector`

`add <a> <svectorA> <b> <svectorB> [-mode mode]`
a * vectorA + b * vectorB
`a`
`svectorA`
`b`
`svectorB`
`mode` mode 0, 1 or -1 for dimension(result) =, max or min of input

`append <SVector> <SVector>`
appends two svector
`SVector` SVector A
`SVector` SVector B

`copy <svector>`
copy vector
`svector`

`display <canvas> [-height height] [-from from] [-to to] [-step step] [-scale scale] [-tag tag]`
display vector
`canvas`
`height`
`from`
`to`
`step`
`scale`
`tag`

`lin <a> <b>`
a * vector + b
`a`
`b`

`mean`
gives the mean value

`minmax`
gives minimum and maximum

`mul <svectorA> <svectorB> [-a a] [-div div] [-mode mode]`
vector multiplication
`svectorA`
`svectorB`
`a`
`div` division instead multiplication
`mode` mode 0, 1 or -1 for dimension(result) =, max or min of input

`power`

gives the power value

`puts` [`-index index`]

print vector as TCL list

`index`

`resize` <`dimension`>

resize vector

`dimension`

`set` <`index`> <`value`>

set single coefficient

`index`
`value`

`swap`

swap byte order of short vector values

## 8.2 Feature stuff (src/features)

### 8.2.1 FeatureSet

This section describes the '*FeatureSet*': *set of features*

**Creation:** `FeatureSet` <`name`>

| | |
|---|---|
| `name` | name of the object |

**Configuration:** `featureset configure`

| | |
|---|---|
| `-adcByteOrder` | = auto |
| `-adcHeader` | = auto |
| `-byteModeIn` | = 1 |
| `-byteModeOut` | = 1 |
| `-fadeIn` | = 0 |
| `-frameShift` | = 10.000000 |
| `-from` | = 0 |
| `-name` | = featureSetISLci |
| `-offset` | = 0 |
| `-ready` | = 1 |
| `-runon` | = 0 |
| `-samplingRate` | = 16.000000 |
| `-to` | = -1 |
| `-trans` | = 0 |
| `-useN` | = 6 |
| `-verbosity` | = 0 |
| `-writeHeader` | = 1 |

**Methods:** `featureset`

`CholeskyDecomp` <`feature`> <`source`>

calculates the Cholesky Decomposition

| | |
|---|---|
| `feature` | name of the new feature |
| `source` | adjust this feature |

`EFVR <feature> <source_feature> <threshold> [-weight weight] [-boost boost] [-decrease decrease] [-shrink shrink] [-maxboost maxboost] [-thresweight thresweight]`

Early Feature Vector Reduction

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `threshold` | threshold level for feature reduction |
| `weight` | weight feature to be written |
| `boost` | boost factor for the weights |
| `decrease` | decrease of influence of each dimension of feature vector (0=all equal,1= 1/n) |
| `shrink` | !=0 means no shrinking of the feature vector, merged fv are duplicated |
| `maxboost` | maximum boost factor |
| `thresweight` | vector of factors for dynamic threshold level |

`FMatrix`

insert FMatrix type object into feature set

`MVN <feature> <source> <mean> <smean> <alpha> [-a a] [-weight weight]`

mean and variance normalisation with exponential weighting (by FF)

| | |
|---|---|
| `feature` | name of the new feature |
| `source` | name of the source feature |
| `mean` | update mean in FVector object |
| `smean` | update mean of squares in FVector object |
| `alpha` | exponential weighting factor |
| `a` | if (a > 0) a * standard deviation is normalised to 1.0 |
| `weight` | feature that weights each frame when mean is calculated |

`QWarp <feature> <source_feature> [-WinSize winsize]`

feature warping based on CDF matching

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `winsize` | window size |

`SVector`

insert SVector type object into feature set

`VTLN <feature> <source> <ratio> [-min min] [-max max] [-edge edge] [-mod mod]`

Vocal Tract Length Normalization (VTLN)

| | |
|---|---|
| `feature` | name of the new feature |
| `source` | name of the source feature |
| `ratio` | warping factor |
| `min` | max warping factor |
| `max` | min warping factor |
| `edge` | edge point for piecewise warping |
| `mod` | warping modus: lin, nonlin |

`access`

preprocess feature evaluation parameters (featureSetAccess)

`accumulatematrix <feature> <feature> <source> [-silence silence]`

accumulate matrix

| | |
|---|---|
| `feature` | name of the new feature |
| `feature` | name of the new feature |
| `source` | adjust this feature |
| `silence` | silence frames marked with 1, otherwise 0 |

`adc2mel <feature> <source_feature> <win> [-shift shift]`

16 framebased melscale coefficients, 8 and 16 kHz only

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `win` | window size |
| `shift` | shift |

`adc2pow <feature> <source_feature> <win> [-shift shift]`

framebased power

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `win` | window size |
| `shift` | shift |

`adc2spec <source_feature> <win> [-shift shift] [-win win] [-rea rea] [-ima ima] [-mag mag] [-pha pha] [-pow pow] [-adc adc] [-D d]`

framebased spectral analysis

| | |
|---|---|
| `source_feature` | name of the source feature |
| `win` | window size |
| `shift` | shift |
| `win` | window type [hamming—hanning—tukey—rect] |
| `rea` | feature with real part spectrum |
| `ima` | feature with complex part spectrum |
| `mag` | feature with magnitude |
| `pha` | feature with phase |
| `pow` | feature with power spectrum |
| `adc` | feature with windowed audio signal |
| `d` | |

`add <new_feature> <a> <featureA> <b> <featureB> [-mode mode]`

add two features: a * featureA + b * featureB

| | |
|---|---|
| `new_feature` | <a> * <featureA> + <b> * <featureB> |
| `a` | |
| `featureA` | name of source feature 1 |
| `b` | |
| `featureB` | name of source feature 2 |
| `mode` | mode 0, 1 or -1 for dimension(result) =, max or min of input |

`adjacent <feature> <source_feature> [-delta delta]`

put adjacent frames together: x(t-delta), x(t+1-delta), ..., x(t+delta)

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `delta` | delta (in time format) |

`alog <new_feature> <source_feature> <m> <a>`

m * log(source_feature + b) with b=max/10â

| | |
|---|---|
| `new_feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `m` | |
| `a` | |

`append <feature> <feature> <mode>`

append frames/ coefficients to the source feature

| | |
|---|---|
| `feature` | name of feature to which the new data is appended |
| `feature` | name of appending feature |
| `mode` | append frames/ coeffs (i.e. rows/ columns) |

`appendAESVM <feature> <msg> <header> <trailer>`

append an ASCII encoded short vector message to an SVector feature

| | |
|---|---|
| `feature` | name of an SVector feature |
| `msg` | ascii encoded short vector messgage |
| `header` | message header |
| `trailer` | message trailer |

`aspike <destin> <source> [-window window] [-width width] [-maxslope maxslope] [-meanslope meanslope] [-thresh thresh] [-alpha alpha] [-v v]`

remove spikes from signal

| | |
|---|---|
| `destin` | name of the new feature |
| `source` | name of the source feature |
| `window` | window width of median filter (<3 = off) |
| `width` | max spike width of slope filter (<1 = off) |
| `maxslope` | max slope of slope filter |
| `meanslope` | start mean value of slope filter |
| `thresh` | thresh of slope filter |
| `alpha` | adaption factor of slope filter |
| `v` | verbosity |

`audioInit [-sr sr] [-gain gain]`

init audio device

| | |
|---|---|
| `sr` | sampling rate |
| `gain` | microphon gain |

`auditory <feature> <source_feature> [-nf nf]`

auditory filterbank

| | |
|---|---|
| `feature` | name of new feature |
| `source_feature` | name of source feature |
| `nf` | number of filters |

`autocorr <feature> <source_feature> <coeffN> <win> [-shift shift]`

auto correlation

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `coeffN` | coeffN |
| `win` | window size |
| `shift` | shift |

`avMagnitude <feature> <source_feature> <win> [-shift shift] [-mean mean] [-log log] [-abs abs]`

frame based average magnitude

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `win` | window size |
| `shift` | shift |
| `mean` | mean of source feature |
| `log` | compute log magnitude |
| `abs` | compute absolute value (useful before taking log) |

`beepSeg <feature> [-from from] [-to to] [-band band] [-thresh thresh] [-minDur mindur] [-maxInt maxint]`

segment (spectral) feature at beeper positions

| | |
|---|---|
| `feature` | (spectral) source feature |
| `from` | starting frame |
| `to` | final frame |
| `band` | index of frequency band |
| `thresh` | energy threshold value |
| `mindur` | minimum duration |
| `maxint` | maximum interruption |

`changesub <feature> <adjustto> <adjustfrom>`

change scale of noise for spectral subtraction

| | |
|---|---|
| `feature` | name of the new feature |
| `adjustto` | adjust this feature |
| `adjustfrom` | adjust from this feature |

`compress <new_feature> <source_feature> <codebookSet> [-verbose verbose] [-trainMode trainmode]`

compress float features to 8bit values

| | |
|---|---|
| `new_feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `codebookSet` | will need a cbs after a couple of beers (`CodebookSet`) |
| `verbose` | verbose |
| `trainmode` | store compressed values in orginal feature |

`concat`

concat frames (or samples) of features

`conv <source_feature> <fmatrix> [-gain gain]`

convolution with an impulse response

| | |
|---|---|
| `source_feature` | name of the source feature |
| `fmatrix` | impulse resp |
| `gain` | gain |

`corr <new_feature> <featureA> <featureB> [-from from] [-to to] [-step step] [-samplestep samplestep] [-pad pad]`

correlation of two features

| | |
|---|---|
| `new_feature` | correlation of <featureA> and <featureB> |
| `featureA` | name of source feature 1 |
| `featureB` | name of source feature 2 |
| `from` | |
| `to` | |
| `step` | |
| `samplestep` | |
| `pad` | pad with 0 |

`corrMatrix <feature> <source> [-normalize normalize]`

correlation matrix of features

| | |
|---|---|
| `feature` | name of the new feature |
| `source` | adjust this feature |
| `normalize` | normalize correlation |

`cut <feature> <source_feature> <from> <to> [-select select]`

take frames <from> .. <to> of source feature

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `from` | start |
| `to` | end |
| `select` | 1-dimensional FMatrix feature that selects the parts to be taken |

`delete`

delete a feature

`delta <feature> <source_feature> [-delta delta]`

symmetrical delta coefficients: x(t+delta) - x(t-delta)

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `delta` | delta (in time format) |

`display`

displays a feature

`distance <feature> <source_feature>`

frame based distance

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |

`downsample <feature> <source_feature>`

downsample from 16kHz to 8kHz telephone quality

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |

`dtcwt <feature> <source_feature> <filter0a> <filter0b> <filter1a> <filter1b> <level> [-useLowpass uselowpass]`

perform the dual-tree complex wavelet transform according to Kingsbury

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `filter0a` | the lowpass filters for the first stage of the |
| `filter0b` | / transform in trees a,b |
| `filter1a` | the lowpass filters for all subsequent stages of the |
| `filter1b` | / transform, trees a,b |
| `level` | decomposition level |
| `uselowpass` | number of low-pass coefficients to use (0 .. <level>) |

`eval`

run feature description script (featureSetEval)

`exp <new_feature> <source_feature> <m> <a>`

m * exp(a * source_feature)

| | |
|---|---|
| `new_feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `m` | |
| `a` | |

`fft2 <feature> <source_feature> <win> [-shift shift]`

framebased complex spectrum

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `win` | window size |
| `shift` | shift |

`filter <feature> <source_feature> <filter> [-pad pad]`

filter a feature

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `filter` | @filename, name or definition of a filter |
| `pad` | =0 pad with 0.0, !=0 pad with first & last value |

`filterbank <feature> <source_feature> <bmatrix>`

multiply band matrix A with each frame x of feature: A * x

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `bmatrix` | float band matrix |

`findpeaks <new_feature> <feature> [-hz_min hz_min] [-hz_max hz_max] [-sr sr]`

framebased peak tracker

| | |
|---|---|
| `new_feature` | peaks of <feature> |
| `feature` | name of the source feature |
| `hz_min` | |
| `hz_max` | |
| `sr` | |

`flip <feature> <source_feature>`

take last frames first

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |

`formants <feature> <source_feature> [-N n] [-fMin fmin] [-fMax fmax] [-bMax bmax]`

extract fromants from lpc

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `n` | max. number of formants |
| `fmin` | min. formant frequency |
| `fmax` | max. formant frequency |
| `bmax` | max. formant bandwidth |

`frame <source_feature> <frame> [-format format]`

return frame of a feature given a featureSet frame index

| | |
|---|---|
| `source_feature` | name of the source feature |
| `frame` | featureSet frame index |
| `format` | format string |

`frameN <feature*>`

return featureSet frame number given a list of features

| | |
|---|---|
| `feature*` | list of features |

`fwt <feature> <source_feature> <filter> <level> [-useLowpass uselowpass]`

perform the fast wavelet tranformation, real decimated case

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `filter` | the filter (i.e. the scaling function's coefficients) tho be used |
| `level` | decomposition level |
| `uselowpass` | number of low-pass coefficients to use (0 .. <level>) |

`getdeltasamples <feature> <source> <matrix>`

get delta of samples 4 correlated random variables

| | |
|---|---|
| `feature` | name of the new feature |
| `source` | adjust this feature |
| `matrix` | LP-COEFF-Matrix |

`getgaussian <feature_out> <feature_in> <distribSet> [-gmm gmm]`

return the gaussian with the highest likelihood

| | |
|---|---|
| `feature_out` | name of the new feature |
| `feature_in` | feature to be cleaned |
| `distribSet` | enter the DistribSet here (CodebookSet is also loaded) (`DistribSet`) |
| `gmm` | name of gmm to be used |

`gradient <feature> <source_feature> [-win win]`

compute gradients for a given window length

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `win` | number of Frames in window |

`impulse <feature> <source_feature> <win> [-shift shift]`

framebased impulse reponse

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `win` | window size |
| `shift` | shift |

`index <names*>`

get feature index for a given name

| | |
|---|---|
| `names*` | list of feature names |

`lin <new_feature> <source_feature> <m> <a>`

m * source_feature + a

| | |
|---|---|
| `new_feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `m` | |
| `a` | |

`log <new_feature> <source_feature> <m> <a>`

m * log(source_feature + a)

| | |
|---|---|
| `new_feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `m` | |
| `a` | |

`lpc <feature> <source_feature> <order> [-a0 a0]`

linear predictive coding

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `order` | order |
| `a0` | include a0 |

`map <new feature> <featureA> <featureB> <matrix>`

acoustic mapping

| | |
|---|---|
| `new feature` | estimate for environment 2 |
| `featureA` | features from environment 1 |
| `featureB` | probs for each class |
| `matrix` | FMatrix with shift vectors |

`matmul <feature> <source_feature> <matrix> [-cut cut]`

multiply matrix A with each frame x of feature: A * x

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `matrix` | FMatrix |
| `cut` | take first n coefficients |

`maxarg <feature> <source_feature> [-abs abs]`

index of maximum value per frame

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `abs` | 1 for absolute value or 0 for signed values |

`maxpeak <feature> <source_feature> <win> [-shift shift]`

framebased maximum of peak to peak

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `win` | window size |
| `shift` | shift |

`mean <matrix> <source> [-weight weight] [-dev dev] [-smean smean] [-count count] [-update update]`

calculate mean and variance

| | |
|---|---|
| `matrix` | mean vector(s) of type FMatrix |
| `source` | name of the source feature |
| `weight` | weight frames when calculate mean vector |
| `dev` | deviation vector(s) of type FMatrix |
| `smean` | mean of squares vector(s) of type FMatrix |
| `count` | counts |
| `update` | update mean and smean with using counts |

`meanarg <feature> <source_feature>`

mean index per frame

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |

`meansub <feature> <source> [-a a] [-mean mean] [-dev dev] [-smean smean] [-upMean upmean] [-upSMean upsmean] [-useup useup] [-weight weight] [-factor factor] [-alpha alpha]`

meansubtraction and variance normalisation

| | |
|---|---|
| `feature` | name of the new feature |
| `source` | name of the source feature |
| `a` | if (a > 0) a * standard deviation is normalised to 1.0 |
| `mean` | mean vector of type FVector |
| `dev` | deviation vector of type FVector |
| `smean` | mean of squares vector of type FVector |
| `upmean` | update mean in FVector object |
| `upsmean` | update mean of squares in FVector object |
| `useup` | 1 for: "use updated vectors" or 0 for: "current" |
| `weight` | feature that weights each frame when mean is calculated |
| `factor` | feature that weights each frame when mean is subtracted, a:=0! |
| `alpha` | adaptation factor |

`melscale <feature> <source_feature>`

melscale from power spectrum

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |

`merge <new_feature> <names*>`

merge coefficients (interleave samples) of features

| | |
|---|---|
| `new_feature` | name of the new feature |
| `names*` | list of source features |

`mix` <`source_feature`> <`source_feature`> [`-shift shift`] [`-gain gain`]

mix with a new signal

| | |
|---|---|
| `source_feature` | name of the source feature |
| `source_feature` | name of the source feature |
| `shift` | shift between features |
| `gain` | gain |

`mixn` <`feature`> <`source_features`> [`-filters filters`] [`-ignoreN ignoren`] [`-normalize normalize`] [`-gain gain`]

mix n signals

| | |
|---|---|
| `feature` | name of the new feature |
| `source_features` | list of source features |
| `filters` | filters (objects) to apply to signals |
| `ignoren` | samples to ignore left/ right |
| `normalize` | normalize with number of features |
| `gain` | gain |

`mul` <`new_feature`> <`featureA`> <`featureB`> [`-a a`] [`-div div`] [`-mode mode`]

multiply two features: a * featureA * featureB

| | |
|---|---|
| `new_feature` | <a> * <featureA> * <featureB> |
| `featureA` | name of source feature 1 |
| `featureB` | name of source feature 2 |
| `a` | factor a |
| `div` | division instead multiplication |
| `mode` | mode 0, 1 or -1 for dimesion(result) =, max or min of input |

`multisteplp` <`feature`> <`source`> [`-delay delay`] [`-order order`]

calculate multi step linear prediction coefficients

| | |
|---|---|
| `feature` | name of the new feature |
| `source` | adjust this feature |
| `delay` | delay of linear prediction |
| `order` | model order |

`name` <`idx*`>

get feature name for a given index

| | |
|---|---|
| `idx*` | list of feature indices |

`noise` <`feature`> <`length`> [`-type type`] [`-sr sr`] [`-mean mean`] [`-dev dev`] [`-dim dim`]

create noise signal

| | |
|---|---|
| `feature` | name of the new feature |
| `length` | length in time format |
| `type` | "uniform" or "normal" distribution |
| `sr` | sampling rate in kHz |
| `mean` | mean value |
| `dev` | deviation value |
| `dim` | 0 to create a SVector noise feature, >0 to create a FMatrix noise feature with <dim> |

`noiseest` <`feature`> <`source_feature`> [`-method method`] [`-time time`] [`-nrOfBestMins nrofbestmins`] [`-alpha alpha`] [`-overEstimation overestimation`] [`-debug debug`]

estimate the noise in a given signal

| | |
|---|---|
| feature | name of the new feature |
| source_feature | name of the source feature |
| method | method for noise estimation |
| time | time length of the window in seconds |
| nrofbestmins | number of mini-windows |
| alpha | memory factor for minimum statistic |
| overestimation | over-estimation factor for minimum statistic |
| debug | 0: no debugging output, 1: print debugging output |

noisered <feature> <source_feature> <noise_feature> [-alpha alpha] [-Rprio_min rprio_min] [-rprio rprio] [-rpost rpost] [-debug debug]

Ephraim and Malah Noise Reduction (additive noise reduction)

| | |
|---|---|
| feature | name of the new feature |
| source_feature | name of the source feature |
| noise_feature | estimated noise |
| alpha | weight for calculation of the a priori SNR |
| rprio_min | min. value for Rprio to adjust residual noise level |
| rprio | feature with Rprio (in dB) |
| rpost | feature with Rpost (in dB) |
| debug | 0: no debugging output, 1: print debugging output |

normalize <feature> <source_feature> [-min min] [-max max]

normalize coefficients to range <min> .. <max>

| | |
|---|---|
| feature | name of the new feature |
| source_feature | name of the source feature |
| min | |
| max | |

normframe <feature> <source_feature> [-L l] [-n n] [-add add]

normalize each frame

| | |
|---|---|
| feature | name of the new feature |
| source_feature | name of the source feature |
| l | Lp norm = (SUM —x[i]—p̂)1̂/p |
| n | feature to hold norm |
| add | 1: take norm as additional coefficient to new feature |

offset <destin> <source> [-alpha alpha] [-count count] [-offset offset] [-mean mean] [-smean smean] [-a a] [-mindev mindev] [-delta delta] [-upMean upmean] [-upSMean upsmean]

remove offset adaptively from signal

| | |
|---|---|
| destin | name of the new feature |
| source | name of the source feature |
| alpha | adaption factor of offset filter |
| count | if not 0 then calculate alpha using count of the mean vectors |
| offset | see -mean (old flag!) |
| mean | start value for mean |
| smean | start value for smean |
| a | a * standard deviation is normalised to 1.0 (if a > 0 and smean given) |
| mindev | minimal deviation |
| delta | calculate mean <delta> frames/samples ahead |
| upmean | update mean in FVector object |
| upsmean | update mean of squares in FVector object |

paGetRecorded

get recorded PortAudio data

**paInfo** <option>

info on status, devices, ...

| | |
|---|---|
| option | one of "devicesIn/Out", "defaultIn/Out", "status", "?" |

**paPlay** <feature> [-device device] [-from from] [-to to]

start PortAudio playing

| | |
|---|---|
| feature | name of the feature |
| device | audio device number |
| from | from (sample no.) |
| to | to (sample no.) |

**paStartRecording** [-device device] [-sr sr] [-chN chn] [-buf buf] [-feature feature] [-file file]

start PortAudio recording

| | |
|---|---|
| device | audio device number |
| sr | sampling rate in kHz |
| chn | number of channels |
| buf | buffer length (in seconds) |
| feature | name of the feature(s) |
| file | name of the file to write (Windows and WAV only) |

**paStop**

stop PortAudio playing

**particlefilter** <feature_out> <feature_in> <distribSet> [-number number] [-fast fast] [-variance variance] [-refresh refresh] [-nio nio] [-ARsmoothing arsmoothing] [-transcription transcription] [-type type] [-speech speech] [-init init] [-delayspec delayspec]

partilce filter spectral enhancement

| | |
|---|---|
| feature_out | cleaned feature |
| feature_in | feature to be cleaned |
| distribSet | enter the DistribSet here (CodebookSet is also loaded) (DistribSet) |
| number | number of particles |
| fast | fast version (skipping every second frame) |
| variance | variance of the noise propagation |
| refresh | cut of the likelihood |
| nio | noise intensity offset |
| arsmoothing | determines smoothing over frames for the dynamic AR matrix |
| transcription | acoustic model trancription |
| type | "sia" or "vts" |
| speech | speech frames marked with 1, otherwise 0 |
| init | initialize particle filter |
| delayspec | list of delay spectra |

**peak** <feature> <source_feature> <win> [-shift shift]

framebased peak distance

| | |
|---|---|
| feature | name of the new feature |
| source_feature | name of the source feature |
| win | window size |
| shift | shift |

**play** <src_feature> [-sr sr]

play audio

| | |
|---|---|
| src_feature | feature to play |
| sr | sampling rate in kHz |

**plp** <feature> <source_feature> [-o o] [-n n]

perceptual linear prediction

| | |
|---|---|
| feature | name of new feature |
| source_feature | name of source feature |
| o | filter order |
| n | number of output coefficients, 0 means order+1 |

**postaud** <feature> <source_feature>

post processing for auditory filterbank

| | |
|---|---|
| feature | name of new feature |
| source_feature | name of source feature |

**pow** <new_feature> <source_feature> <m> <a>

m * (source_feature â)

| | |
|---|---|
| new_feature | name of the new feature |
| source_feature | name of the source feature |
| m | |
| a | |

**power** <feature> <source_feature> <win> [-shift shift] [-mean mean]

frame based power

| | |
|---|---|
| feature | name of the new feature |
| source_feature | name of the source feature |
| win | window size |
| shift | shift |
| mean | mean of source feature |

**predictionmatrix** <feature> <source> [-weight weight]

linear prediction matrix

| | |
|---|---|
| feature | name of the new feature |
| source | name of the source feature |
| weight | feature that weights each frame when mean is calculated |

**predictionvariance** <feature> <source> [-silence silence]

calculate variance

| | |
|---|---|
| feature | name of the new feature |
| source | adjust this feature |
| silence | silence frames marked with 1, otherwise 0 |

**puls** <feature> <from> <to> [-value value]

create puls in signals

| | |
|---|---|
| feature | name of the new feature |
| from | start in time format |
| to | length in time format |
| value | value of puls |

**rdwt** <feature> <source_feature> <filter> <level> [-useLowpass uselowpass]

perform the redundant discrete wavelet transform

| | |
|---|---|
| feature | name of the new feature |
| source_feature | name of the source feature |
| filter | the filter (i.e. the scaling function's coefficients) tho be used |
| level | decomposition level |
| uselowpass | number of low-pass coefficients to use (0 .. <level>) |

**read**

read feature file

`readADC <feature> <filename> [-hm hm] [-bm bm] [-f f] [-chX chx] [-chN chn] [-from from] [-to to] [-sr sr] [-offset offset] [-fadeIn fadein] [-v v] [-startFile startfile] [-readyFile readyfile] [-sleep sleep] [-rmFiles rmfiles]`

read ADC file

| | |
|---|---|
| `feature` | name of the new feature |
| `filename` | name of ADC file |
| `hm` | header mode, kind or size in byte |
| `bm` | byte mode |
| `f` | 1 => skip unnecessary bytes when reading |
| `chx` | selected channel: 1..chN |
| `chn` | number of channels |
| `from` | from |
| `to` | to |
| `sr` | sampling rate in kHz |
| `offset` | subtract offset |
| `fadein` | fade in |
| `v` | verbosity |
| `startfile` | runon: name of start file |
| `readyfile` | runon: name of ready file |
| `sleep` | runon: time to wait before next try |
| `rmfiles` | runon: remove files |

`readhtk`

read HTK feature file

`recordGet <feature> [-stop stop] [-device device]`

get new audio data after starting with 'recordStart'

| | |
|---|---|
| `feature` | name of the new (recorded) feature |
| `stop` | stop recording |
| `device` | audio device |

`recordStart <feature> [-sr sr]`

start audio recording (see also 'recordGet')

| | |
|---|---|
| `feature` | name of the new (recorded) feature |
| `sr` | sampling rate in HZ |

`reorder <feature> <source_feature> [-nextDestin nextdestin] [-nextSource nextsource]`

reorder entries in feature

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `nextdestin` | name of the new feature |
| `nextsource` | name of the source feature |

`replace <feature> <feature> <from>`

replace frames starting at <from> of source feature

| | |
|---|---|
| `feature` | name of feature to replace |
| `feature` | name of replacing feature |
| `from` | start |

`resample <feature> <source_feature> <rate/shift> [-style style] [-order order]`

resample audiosignal changing sampling rate

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `rate/shift` | new sampling rate in kHz for SVector or new shift in ms for FMatrix |
| `style` | 'lin' or 'si' (short only!) |
| `order` | order for 'si' |

`setAccess @<filename>|<command>`

read a 'File Access Description'

| | |
|---|---|
| `@filename|command` | @ and name of 'Feature Description File' or Tcl command |

`setDesc @<filename>|<command>`

read a 'Feature Description'

| | |
|---|---|
| `@filename|command` | @ and name of 'Feature Description' or Tcl command |

`shift <feature> <source_feature> [-delta delta]`

shift frames: x(t+delta)

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `delta` | delta (in time format) |

`show <FeatureSet> <Feature> [-width width] [-height height]`

show feature set (featshow)

| | |
|---|---|
| `FeatureSet` | FeatureSet to use (`FeatureSet`) |
| `Feature` | name of feature to display |
| `width` | width of window |
| `height` | height of window |

`silDetCF <feature> <magnitude> <zeroX> [-magnitudeFactor magnitudefactor] [-zeroFactor zerofactor] [-smoothPasses smoothpasses] [-smoothLength smoothlength] [-minSilLength minsillength] [-minZeroLength minzerolength]`

Christian Fuegen's Silence Detection

| | |
|---|---|
| `feature` | name of the new feature |
| `magnitude` | log magnitude (magnitude or power see -feType) |
| `zeroX` | zero crossing feature |
| `magnitudefactor` | factor to multiply the magnitude threshold |
| `zerofactor` | factor to multiply the zeroX threshold |
| `smoothpasses` | passes to smooth |
| `smoothlength` | smooth length (odd number) |
| `minsillength` | minimum number of frames for a silence |
| `minzerolength` | minimum number of frames for which the zeroX must exceed the threshold |

`silSeg <feature> [-from from] [-to to] [-band band] [-thresh thresh] [-minDur mindur] [-maxInt maxint]`

segment (spectral) feature at silence positions

| | |
|---|---|
| `feature` | (spectral) source feature |
| `from` | starting frame |
| `to` | final frame |
| `band` | index of frequency band |
| `thresh` | energy threshold value |
| `mindur` | minimum duration |
| `maxint` | maximum interruption |

`silTK <feature> <power> <ptp> [-minPower minpower] [-maxPower maxpower]`

T.Kemp's silence feature

| | |
|---|---|
| `feature` | name of the new feature |
| `power` | name of power feature |
| `ptp` | name of ptp feature |
| `minpower` | mean of the most silent frames |
| `maxpower` | mean of loudest frames |

`snr <source_feature> <silence_feature> [-silSub silsub] [-mean mean]`

signal to noise ratio of feature

| | |
|---|---|
| `source_feature` | name of the source feature |
| `silence_feature` | silence feature (1/0) |
| `silsub` | subtract the silence from speech Energy |
| `mean` | mean of source feature |

`snrK <source_feature> <win> [-shift shift] [-mean mean] [-kmeansIterN kmeansitern]`

signal to noise ratio of feature (kmeans)

| | |
|---|---|
| `source_feature` | name of the source feature |
| `win` | window size |
| `shift` | shift |
| `mean` | mean of source feature |
| `kmeansitern` | number of iterations of kmeans |

`spec2adc <feature> <source_feature1> <source_feature2> [-win win] [-sr sr] [-D d]`

audio signal reconstruction from spectrum

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature1` | magnitude |
| `source_feature2` | phase |
| `win` | window type [tukey—none] |
| `sr` | sampling rate in kHz |
| `d` | |

`specadj <feature> <adjustto> <adjustfrom> [-smooth smooth] [-show show]`

adjust first spectrum to max of second spectrum

| | |
|---|---|
| `feature` | name of the new feature |
| `adjustto` | adjust this feature |
| `adjustfrom` | adjust from this feature |
| `smooth` | smooth the adjust from feature (0,1,2,3,4) |
| `show` | ”on” or ”off” |

`specest <feature> <source_feature> <order> [-type type] [-warp warp] [-sensibility sensibility] [-lpmethod lpmethod] [-correlate correlate] [-compensate compensate]`

spectral estimation: lp wlp, mvdr or wmvdr

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `order` | order |
| `type` | ”LP” or ”MVDR” or ”MVDR.rewarp” |
| `warp` | warp |
| `sensibility` | sensibility |
| `lpmethod` | ”autocorrelation” or ”modcovarianz” or ”burg” or ”warp” |
| `correlate` | needed for burg and modcovariance |
| `compensate` | compensate for the amplitute change in rewarp |

`specsub <new feature> <featureA> <featureB> [-a a] [-b b]`

Spectral Subtraction after Boll (additive noise reduction)

| | |
|---|---|
| `new feature` | spectral subtraction after Boll with estimated noise |
| `featureA` | spectral feature |
| `featureB` | estimated noise |
| `a` | overestimation factor alpha |
| `b` | spectral floor beta |

`specsublog <feature> <adjustto> <adjustfrom>`

logarithmic spectral subtraction (log10)

| | |
|---|---|
| `feature` | name of the new feature |
| `adjustto` | adjust this feature |
| `adjustfrom` | adjust from this feature |

`specsublog1 <feature> <adjustto> <adjustfrom>`

logarithmic spectral subtraction (log1)

| | |
|---|---|
| `feature` | name of the new feature |
| `adjustto` | adjust this feature |
| `adjustfrom` | adjust from this feature |

`spectrum <feature> <source_feature> <win> [-shift shift]`

framebased power spectrum

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `win` | window size |
| `shift` | shift |

`speechDetect`

speech detector based on gaussian mixture (speechDetect)

`split <feature> <source_feature> <from> <to>`

take coefficients <from> .. <to> of source feature

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `from` | |
| `to` | |

`thresh <feature> <source_feature> <value> <thresh> <mode>`

set coefficients to a specified value if they exceed a threshold

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `value` | |
| `thresh` | |
| `mode` | |

`tone <feature> <vector> [-g g] [-sr sr] [-attack attack] [-peak peak] [-decay decay] [-release release] [-amA ama] [-amF amf] [-fmA fma] [-fmF fmf] [-sound sound]`

create audio signals

| | |
|---|---|
| `feature` | name of the new feature |
| `vector` | vector with "<length_ms> <pitch_Hz> ..." |
| `g` | gain |
| `sr` | sampling rate in kHz |
| `attack` | attack time in ms |
| `peak` | relative peak |
| `decay` | decay time in ms |
| `release` | release time in ms |
| `ama` | AM amplitude in %% |
| `amf` | AM frequency in Hz |
| `fma` | FM frequency shift in 0.01%% |
| `fmf` | FM frequency in Hz |
| `sound` | sound |

`traceScatter <feature> <source_feature> <class> [-numberofclasses numberofclasses]` return the trace of the scatter matrix

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `class` | frame belongs to class |
| `numberofclasses` | define number of classes |

`varss <source_feature>`

variance of the speech signal

| | |
|---|---|
| `source_feature` | name of the source feature |

`write`

write feature file

`writeADC <source_feature> <filename> [-hm hm] [-bm bm] [-from from] [-to to] [-append append] [-v v]`

write ADC file

| | |
|---|---|
| `source_feature` | name of the source feature |
| `filename` | file to write |
| `hm` | header kind or "" for no header |
| `bm` | byte mode |
| `from` | from |
| `to` | to |
| `append` | append to file |
| `v` | verbosity |

`xtalk <new_feature> <channelA> <channelB> [-L l] [-shift shift] [-u u] [-sf sf] [-alpha alpha] [-thr1 thr1] [-thr2 thr2] [-xpow1 xpow1] [-xpow2 xpow2] [-pshift pshift] [-forget forget] [-min min] [-ac ac] [-adap adap] [-infA infa] [-infF inff]`

remove crosstalk with an adaptive filter

| | |
|---|---|
| `new_feature` | name of filtered channel A |
| `channelA` | channel with xtalk |
| `channelB` | channel causing xtalk |
| `l` | number of filter weights |
| `shift` | shift of the input samples |
| `u` | filter convergence factor |
| `sf` | adaptiv shift factor |
| `alpha` | power estimate factor |
| `thr1` | power ratio activating the adaptation |
| `thr2` | power ratio deactivating the adaptation |
| `xpow1` | xtalk power threshold activating the adaptation |
| `xpow2` | xtalk power threshold deactivating the adaptation |
| `pshift` | shift of the power window |
| `forget` | forget weights with (1.0 - forget) when not adapted |
| `min` | take minimum(original,filter) as output, boolean |
| `ac` | adaption counter |
| `adap` | feature telling when to do adaptation |
| `infa` | feature showing when was adapted |
| `inff` | feature showing filter coefficients |

`zero <feature> <source_feature> <win> [-shift shift]`

framebased zero crossing rate / sec

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `win` | window size |
| `shift` | shift |

`zeroX <feature> <source_feature> <win> [-shift shift] [-mean mean] [-log log]`

frame based zero crossing

| | |
|---|---|
| `feature` | name of the new feature |
| `source_feature` | name of the source feature |
| `win` | window size |
| `shift` | shift |
| `mean` | mean of source feature |
| `log` | compute log magnitude |

### 8.2.2 LDA

This section describes the '*LDA*': *LDA*

**Creation:** `LDA <name> <featureSet> <feature> <dimN>`

| | |
|---|---|
| `name` | name of the LDA object |
| `featureSet` | name of the feature set (`FeatureSet`) |
| `feature` | feature name |
| `dimN` | input dimension |

**Configuration:** `lda configure`

| | |
|---|---|
| `-blkSize` | = 100 |
| `-dimN` | = 4 |
| `-featX` | = 0 |
| `-featureSet` | = featureSetISLci |
| `-indexN` | = 0 |
| `-itemN` | = 0 |
| `-name` | = ldaISLci |
| `-useN` | = 1 |

**Methods:** `lda`

`accu <path> [-factor factor] [-from from] [-to to]`

accumulate samples from a path object

| | |
|---|---|
| `path` | name of the path object (`Path`) |
| `factor` | training factor |
| `from` | from frameX |
| `to` | to frameX |

`add <name>`

add a new LDA class to the set

| | |
|---|---|
| `name` | name of the class |

`clear`

clear means

`delete <item>`

remove LDA class from the set

| | |
|---|---|
| `item` | name of item in list |

`index <names*>`

returns indices of named LDA classes

| | |
|---|---|
| `names*` | list of names |

`loadMeans <filename>`

load means from a file

`filename`

`loadScatter <filename>`

load scatter matrix from a file

| | |
|---|---|
| `filename` | filename |

`map <index> [-class class]`

add/get index to class mapping information

| | |
|---|---|
| `index` | index to map |
| `class` | name of the class |

`name <idx*>`

returns names of indexed LDA classes

| | |
|---|---|
| `idx*` | list of indices |

`saveMeans <filename>`

save means to a file

| | |
|---|---|
| `filename` | filename for means |

`saveScatter <filename>`

save scatter matrix to a file

| | |
|---|---|
| `filename` | filename |

`update`

update the scatter matrices

**Subobjects:**

| | |
|---|---|
| `featureSet` | (`FeatureSet`) |
| `list` | (`List`) |
| `matrixS` | (`DMatrix`) |
| `matrixT` | (`DMatrix`) |
| `matrixW` | (`DMatrix`) |
| `mean` | (`DVector`) |

## 8.3 Hidden Markov Models (src/hmm)

### 8.3.1 HMM

This section describes the '*HMM*': *An 'HMM' object contains states, transitions and acoustic references*

**Creation:** `HMM <name> <dictionary> <amodelset>`

| | |
|---|---|
| `name` | name of the HMM |
| `dictionary` | name of the Dictionary object (`Dictionary`) |
| `amodelset` | name of the AmodelSet object (`AModelSet`) |

**Configuration:** `hmm configure`

| | |
|---|---|
| `-full` | = 1 |
| `-logPen` | = 1 |
| `-rcmSdp` | = 0 |
| `-xwmodels` | = 1 |

**Methods:** `hmm`

`convert <GLat>`

convert GLat into HMM object (hmmConvertGLat)

| | |
|---|---|
| `GLat` | (`GLat`) |

`lattice <lattice>`

create full detail HMM from a lattice

| | |
|---|---|
| `lattice` | Verbmobil style lattice |

`make <words> [-trans trans] [-init init] [-optWord optword] [-variants variants]`

create full detail HMM

| | |
|---|---|
| `words` | list of word nodes |
| `trans` | transition model |
| `init` | initial states |
| `optword` | optional word |
| `variants` | pronunciation variants |

`makeUtterance <text> [-optWord optword] [-variants variants]`

create utterance HMM (hmmMakeUtterance)

| | |
|---|---|
| `text` | transcription |
| `optword` | optional word |
| `variants` | variants 0/1 |

`modMakeUtterance <speaker> <uttID> [-text text] [-modalitySet modalityset] [-distribTree distribtree] [-amodelSet amodelset] [-senoneSet senoneset] [-textTag texttag] [-frameN framen] [-optWord optword] [-variants variants]`

create utterance HMM with modalities (hmmModMakeUtterance)

| | |
|---|---|
| `speaker` | speaker ID |
| `uttID` | utterance ID |
| `text` | text to align |
| `modalityset` | name of ModalitySet |
| `distribtree` | name of DistribTree |
| `amodelset` | name of AmodelSet |
| `senoneset` | name of SenoneSet |
| `texttag` | text tag in uttInfo |
| `framen` | number of frames |
| `optword` | optional word |
| `variants` | variants 0/1 |

`puts`

displays the contents of an HMM

`resetModTags`

reset modality Tags for hmm

`setModTags <path> <modalitySet>`

set modality Tags for hmm

| | |
|---|---|
| `path` | name of reference path object (`Path`) |
| `modalitySet` | set of modalities (`ModalitySet`) |

**Subobjects:**

| | |
|---|---|
| `dict` | (`Dictionary`) |
| `phoneGraph` | (`PhoneGraph`) |
| `stateGraph` | (`StateGraph`) |
| `wordGraph` | (`WordGraph`) |

## 8.3.2 Path

This section describes the '*Path*': *A 'Path' object is filled by a forced alignment function and is used by training functions*

**Creation:** `Path <name>`

| | |
|---|---|
| `name` | name of the object |

**Configuration:** `path configure`

| | |
|---|---|
| `-firstFrame` | = 0 |
| `-lastFrame` | = 0 |
| `-name` | = pathISLci |
| `-phoneMissPen` | = 0.000000 |
| `-senoneMissPen` | = 0.000000 |
| `-useN` | = 1 |
| `-wordMissPen` | = 0.000000 |

**Methods:** `path`

`:=` `<path>`

copy path objects

| | |
|---|---|
| `path` | source path object (`Path`) |

`alignGlat` `<hmm> <glat> <pathTmp> [-variants variants] [-modtags modtags] [-thresh thresh] [-mode mode] [-verbose verbose]`

compute forced alignment by Lattice constraint

| | |
|---|---|
| `hmm` | Hidden Markov Model (`HMM`) |
| `glat` | IBIS Lattice object (`GLat`) |
| `pathTmp` | Temporary path variable (`Path`) |
| `variants` | pronunciation variants |
| `modtags` | modality tags |
| `thresh` | minimum posteriori threshold |
| `mode` | alignment mode, 0=viterbi, 1=fwdbwd |
| `verbose` | verbosity |

`bload` `<file> [-hmm hmm]`

binary load of path items

| | |
|---|---|
| `file` | filename |
| `hmm` | HMM object used for mapping (`HMM`) |

`bsave` `<file>`

binary save of path items

| | |
|---|---|
| `file` | filename |

`durentropy` `<hmm>`

Compute durational entropy of a path

| | |
|---|---|
| `hmm` | the underlying HMM (`HMM`) |

`fwdBwd` `<hmm> [-eval eval] [-from from] [-to to] [-skipl skipl] [-skipt skipt] [-topN topn] [-width width] [-label label]`

compute a forward backward path for a HMM

| | |
|---|---|
| `hmm` | name of the HMM object (`HMM`) |
| `eval` | feature set eval string |
| `from` | frame where to start alignment |
| `to` | frame where to end alignment |
| `skipl` | leading frames to skip |
| `skipt` | trailing frames to skip |
| `topn` | topN pruning |
| `width` | maximal width of the path |
| `label` | viterbi follows labels in paht |

`labels <hmm> [-what what]`

displays the contents of a path as labels

| | |
|---|---|
| `hmm` | the underlying HMM (`HMM`) |
| `what` | list of what to display |

`lscore <hmm> [-eval eval] [-from from] [-to to] [-gamma gamma]`

compute the local scores

| | |
|---|---|
| `hmm` | name of the HMM object (`HMM`) |
| `eval` | feature set eval string |
| `from` | start frame |
| `to` | end frame |
| `gamma` | use gamma values |

`make <senoneSet> [-eval eval] [-from from] [-to to] [-skipl skipl] [-skipt skipt]`

creates a path

| | |
|---|---|
| `senoneSet` | name of the SenoneSet object (`SenoneSet`) |
| `eval` | feature set eval string |
| `from` | frame where to start alignment |
| `to` | frame where to end alignment |
| `skipl` | leading frames to skip |
| `skipt` | trailing frames to skip |

`map <hmm> [-senoneSet senoneset] [-stream stream] [-codebookX codebookx]`

map senone indices

| | |
|---|---|
| `hmm` | name of the HMM object (`HMM`) |
| `senoneset` | name of the SenoneSet object (`SenoneSet`) |
| `stream` | index of stream |
| `codebookx` | want codebook instead of distrib indices (0/1) |

`phoneMatrix <FMatrix> [-from from] [-to to] [-first first] [-last last]`

matrix of cum. phone gamma scores

| | |
|---|---|
| `FMatrix` | float matrix (`FMatrix`) |
| `from` | first frame of matrix |
| `to` | last frame to include in matrix |
| `first` | first phone index to include |
| `last` | last phone index to include |

`phones <hmm> [-from from] [-to to]`

displays the phones labels

| | |
|---|---|
| `hmm` | name of the HMM object (`HMM`) |
| `from` | start frame |
| `to` | end frame |

`pjkdmc <hmm> <sns> <wghts> <pjks> [-from from] [-to to] [-idx idx] [-zero zero]`

compute pjk-dmc for AFs

| | |
|---|---|
| `hmm` | name of the HMM object (`HMM`) |
| `sns` | name of the SenoneSet object (`SenoneSet`) |
| `wghts` | name of the wghts FMatrix object (in) (`FMatrix`) |
| `pjks` | name of the pjks FMatrix object (out) (`FMatrix`) |
| `from` | start frame |
| `to` | end frame |
| `idx` | use entry in pjks matrix |
| `zero` | train streams with 0 prob |

`puts` [`-from from`] [`-to to`]

displays the contents of a path

| | |
|---|---|
| `from` | frame where to start output |
| `to` | frame where to end output |

`reset`

remove all items from a path

`senoneMatrix` <`FMatrix`> [`-from from`] [`-to to`] [`-first first`] [`-last last`]

matrix of senone gamma scores

| | |
|---|---|
| `FMatrix` | float matrix (`FMatrix`) |
| `from` | first frame of matrix |
| `to` | last frame to include in matrix |
| `first` | first senone index to include |
| `last` | last senone index to include |

`stateMatrix` <`FMatrix`> [`-from from`] [`-to to`] [`-first first`] [`-last last`]

matrix of state gamma scores

| | |
|---|---|
| `FMatrix` | float matrix (`FMatrix`) |
| `from` | first frame of matrix |
| `to` | last frame to include in matrix |
| `first` | first state index to include |
| `last` | last state index to include |

`viterbi` <`hmm`> [`-eval eval`] [`-from from`] [`-to to`] [`-skipl skipl`] [`-skipt skipt`] [`-beam beam`] [`-topN topn`] [`-label label`] [`-bpMod bpmod`] [`-bpMul bpmul`]

compute a Viterbi path for a given HMM

| | |
|---|---|
| `hmm` | name of the HMM object (`HMM`) |
| `eval` | feature set eval string |
| `from` | frame where to start alignment |
| `to` | frame where to end alignment |
| `skipl` | leading frames to skip |
| `skipt` | trailing frames to skip |
| `beam` | constant beam size |
| `topn` | topN pruning |
| `label` | viterbi follows labels in path |
| `bpmod` | after every X frames clean up bpTable (<0 never) |
| `bpmul` | go Y * X frames back during cleanup (<1 start at first frame) |

`wordMatrix` <`FMatrix`> [`-from from`] [`-to to`] [`-first first`] [`-last last`]

matrix of cum. word gamma scores

| | |
|---|---|
| `FMatrix` | float matrix (`FMatrix`) |
| `from` | first frame of matrix |
| `to` | last frame to include in matrix |
| `first` | first word index to include |
| `last` | last word index to include |

`words` <`hmm`> [`-from from`] [`-to to`]

displays the word/variant labels

| | |
|---|---|
| `hmm` | name of the HMM object (`HMM`) |
| `from` | start frame |
| `to` | end frame |

**Subobjects:**

| | |
|---|---|
| `itemList(0..0)` | () |

### 8.3.3 `PathItem`

This section describes the '*PathItem*': *PathItem*

**Creation:** `PathItem` cannot be created directly.

It is accessible as a sub-object of `PathItemList`!

**Configuration:** `pathitem configure`

| | |
|---|---|
| `-alpha` | = 0.000000 |
| `-beta` | = 0.000000 |
| `-gamma` | = 0.000000 |
| `-lscore` | = 0.000000 |
| `-phoneX` | = -1 |
| `-senoneX` | = -1 |
| `-stateX` | = -1 |
| `-wordX` | = -1 |

### 8.3.4 `PathItemList`

This section describes the '*PathItemList*': *PathItemList*

**Creation:** `PathItemList` cannot be created directly.

It is accessible as a sub-object of `Path`!

**Configuration:** `pathitemlist configure`

| | |
|---|---|
| `-beam` | = 0.000000 |
| `-best` | = 0.000000 |
| `-itemN` | = 1 |
| `-logScale` | = 0.000000 |
| `-score` | = 0.000000 |

**Methods:** `pathitemlist`

`add` <n> [`-stateX statex`] [`-senoneX senonex`] [`-phoneX phonex`] [`-wordX wordx`]

add items to the path list

| | |
|---|---|
| `n` | number of pathItems to add |
| `statex` | state index |
| `senonex` | relative senone index |
| `phonex` | relative phone index |
| `wordx` | relative word index |

`clear`

remove all items from the path list

**Elements:** are of type `PathItem`.

### 8.3.5 PhoneGraph

This section describes the '*PhoneGraph*': *PhoneGraph*

**Creation:** `PhoneGraph` cannot be created directly.
It is accessible as a sub-object of `HMM`!

**Configuration:** `phonegraph configure`

`-modTags` = 0

**Methods:** `phonegraph`

`build` <`wordGraph`> [`-logPen logpen`] [`-full full`] [`-xwmodels xwmodels`] [`-rcmSdp rcmsdp`]

create PhoneGraph from WordGraph

| | |
|---|---|
| `wordGraph` | word graph (`WordGraph`) |
| `logpen` | log penalties |
| `full` | full PGhraph to PGraph transitions |
| `xwmodels` | xword models |
| `rcmsdp` | right context models for single phone words |

`make` <`phones`> [`-trans trans`] [`-init init`]

create PhoneGraph

| | |
|---|---|
| `phones` | list of phone nodes |
| `trans` | transition model |
| `init` | initial states |

**Subobjects:**

| | |
|---|---|
| `amodel(0..2)` | (`???`) |
| `stateGraph(0..2)` | (`???`) |

### 8.3.6 StateGraph

This section describes the '*StateGraph*': *StateGraph*

**Creation:** `StateGraph` cannot be created directly.
It is accessible as a sub-object of `HMM`!

**Methods:** `stategraph`

`build` <`phoneGraph`> [`-logPen logpen`]

create StateGraph from PhoneGraph

| | |
|---|---|
| `phoneGraph` | phone graph (`PhoneGraph`) |
| `logpen` | log penalties |

**Subobjects:**

| | |
|---|---|
| `senoneSet` | (`SenoneSet`) |

### 8.3.7 TextGraph

This section describes the '*TextGraph*': *Text Graph*

**Creation:** `TextGraph` <`name`>

| | |
|---|---|
| `name` | name of the TextGraph object |

### 8.3.8 WordGraph

This section describes the '*WordGraph*': *WordGraph*

**Creation:** `WordGraph` cannot be created directly.
It is accessible as a sub-object of `HMM`!

**Methods:** `wordgraph`

`lattice <lattice>`
create WordGraph from lattice

| | |
|---|---|
| `lattice` | Verbmobil style lattice |

`make <words> [-trans trans] [-init init] [-optWord optword] [-variants variants]`
create WordGraph

| | |
|---|---|
| `words` | list of word nodes |
| `trans` | transition model |
| `init` | initial states |
| `optword` | optional word |
| `variants` | pronunciation variants |

**Subobjects:**

| | |
|---|---|
| `amodelSet` | (`AModelSet`) |
| `dictionary` | (`Dictionary`) |
| `phoneGraph(0..2)` | (`???`) |

## 8.4 Ibis decoder (src/ibis)

### 8.4.1 BMem

This section describes the '*BMem*': *Block Memory*

**Creation:** `BMem` cannot be created directly.
It is accessible as a sub-object of `GLat`!

**Methods:** `bmem`

`puts [-v v]`
displays the allocation status

| | |
|---|---|
| `v` | verbose output |

### 8.4.2 CFG

This section describes the '*CFG*': *A 'CFG' object is a context free grammar.*

**Creation:** `CFG <name> [-cfgSet cfgset] [-lks lks] [-tag tag]`

| | |
|---|---|
| `name` | name of the object |
| `cfgset` | context free grammar set (`CFGSet`) |
| `lks` | linguistic knowledge source (`LingKS`) |
| `tag` | tag of grammar |

**Configuration:** `cfg configure`

| | |
|---|---|
| `-allPublic` | = 0 |
| `-arcN` | = 5 |
| `-buildMode` | = fixed |
| `-built` | = 1 |
| `-name` | = cfg |
| `-nodeN` | = 8 |
| `-ruleN` | = 3 |
| `-startover` | = -1.000000 |
| `-status` | = Active |
| `-tag` | = cfg |
| `-weight` | = 0.000000 |

**Methods:** `cfg`

`addPath` <`rule`> <`line`> [`-format format`]

adds a path to a CFG

| | |
|---|---|
| `rule` | rule to add path |
| `line` | path to add |
| `format` | grammar format |

`build` [`-mode mode`] [`-overwrite overwrite`] [`-verbose verbose`]

builds a context free grammar

| | |
|---|---|
| `mode` | score definition mode (null—fixed—equal—default=use stored build mode) |
| `overwrite` | overwrite predefined scores from grammar file |
| `verbose` | verbosity |

`clear` [`-free free`]

clears a context free grammar

| | |
|---|---|
| `free` | free items instead of clearing |

`compress` [`-level level`] [`-unfold unfold`] [`-matchFile matchfile`] [`-verbose verbose`]

compress a context free grammar

| | |
|---|---|
| `level` | compress level |
| `unfold` | unfold grammar in new top level rule |
| `matchfile` | file with matching terminals |
| `verbose` | verbosity |

`fsm` <`rule`> <`fsm`>

write a FSM for a given rule (cfgSetWriteFSM)

| | |
|---|---|
| `rule` | rule to print as FSM |
| `fsm` | fsm file |

`generate` <`seqN`> [`-mode mode`] [`-recurse recurse`] [`-file file`]

generate terminal sequences (cfgGenerate)

| | |
|---|---|
| `seqN` | number of terminal sequences |
| `mode` | generation mode (random—fixed) |
| `recurse` | follow recursions |
| `file` | file name to write output |

`load` <`fileName`> [`-format format`] [`-verbose verbose`]

loads a context free grammar

| | |
|---|---|
| `fileName` | file name |
| `format` | grammar format (soup—jsgf—fsm—pfsg—dump) |
| `verbose` | verbosity level |

`parse <text> [-verbose verbose]`

parse a sentence

| | |
|---|---|
| `text` | text to path |
| `verbose` | verbosity |

`parseTree <text> [-svmap svmap] [-format format] [-auxNT auxnt]`

returns the parse tree of a given text string (cfgGetParseTree)

| | |
|---|---|
| `text` | text string to parse |
| `svmap` | use SVMap to map SVX<->LVX (`SVMap`) |
| `format` | output format (soup—jsgf) |
| `auxnt` | print also auxilliary NTs |

`puts [-format format]`

display the contents of CFG

| | |
|---|---|
| `format` | output format (short, long) |

`reduce <matchFile> [-verbose verbose]`

reduces a context free grammar

| | |
|---|---|
| `matchFile` | file with matching terminals |
| `verbose` | verbosity |

`save <fileName> [-pt pt] [-format format]`

saves a context free grammar

| | |
|---|---|
| `fileName` | file name |
| `pt` | dump also parse tree and rule stack |
| `format` | grammar format |

**Subobjects:**

| | |
|---|---|
| `arc(0..4)` | () |
| `bos` | (`CFGNode`) |
| `lex` | (`CFGLexicon`) |
| `node(0..7)` | () |
| `pt` | (`CFGParseTree`) |
| `root` | (`CFGRule`) |
| `rs` | (`CFGRuleStack`) |
| `rule(0..2)` | () |
| `set` | (`CFGSet`) |

### 8.4.3 CFGArc

This section describes the '*CFGArc*': *A 'CFGArc' object is an arc between two nodes of a context free grammar.*

**Creation:** `CFGArc` cannot be created directly.
It is accessible as a sub-object of `CFG`!

**Configuration:** `cfgarc configure`

| | |
|---|---|
| `-lvX` | = 0 |
| `-score` | = 0.000000 |
| `-type` | = T_Arc |

**Methods:** `cfgarc`

`puts [-format format]`

display the contents of CFG arc

| | |
|---|---|
| `format` | output format |

**Subobjects:**

| | |
|---|---|
| `node` | (`CFGNode`) |
| `rule` | (`CFGRule`) |

### 8.4.4 `CFGLexicon`

This section describes the '*CFGLexicon*': *A 'CFGLexicon' object is a lexicon of a Context Free Grammar.*

**Creation:** `CFGLexicon` cannot be created directly.

It is accessible as a sub-object of `CFG`!

**Configuration:** `cfglexicon configure`

| | |
|---|---|
| `-NTN` | = 3 |
| `-TN` | = 2 |
| `-beginOS` | = <s> |
| `-endOS` | = </s> |

**Methods:** `cfglexicon`

`add` <`word`> [`-type type`]

adds an item to the CFG lexicon

| | |
|---|---|
| `word` | word to add |
| `type` | type of arc |

`index` <`word`> [`-type type`]

get lvX of item with given name

| | |
|---|---|
| `word` | word |
| `type` | type of arc |

`name` <`lvX`> [`-type type`]

get name of item with given lvX

| | |
|---|---|
| `lvX` | vocabulary index |
| `type` | type of arc |

`puts` [`-type type`] [`-format format`]

display the contents of CFG lexicon

| | |
|---|---|
| `type` | type of arc |
| `format` | output format |

`write` <`filename`> [`-type type`]

writes a lexicon to file

| | |
|---|---|
| `filename` | file to write into |
| `type` | type of arc |

**Subobjects:**

| | |
|---|---|
| `NT(0..2)` | () |
| `T(0..1)` | () |

**Elements:** are of type `CFGLexiconItem`.

### 8.4.5 CFGLexiconItem

This section describes the '*CFGLexiconItem*': *A 'CFGLexiconItem' object is a item of a CFG lexicon.*

**Creation:** `CFGLexiconItem` cannot be created directly.
It is accessible as a sub-object of `CFGLexicon`!

**Configuration:** `cfglexiconitem configure`

```
-name    = <s>
```

**Methods:** `cfglexiconitem`

```
puts  [-format format]
    display the contents of CFG lexicon
      format    output format
```

### 8.4.6 CFGNode

This section describes the '*CFGNode*': *A 'CFGNode' object is a node in a context free grammar.*

**Creation:** `CFGNode` cannot be created directly.
It is accessible as a sub-object of `CFG`!

**Configuration:** `cfgnode configure`

```
-arcN    = 1
-type    = Root_Node
```

**Methods:** `cfgnode`

```
puts  [-format format]
    display the contents of CFG node
      format    output format
```

**Subobjects:**

```
arc(0..0)   ()
```

### 8.4.7 CFGPTNode

This section describes the '*CFGPTNode*': *A 'CFGPTNode' object is a node of a parse tree.*

**Creation:** `CFGPTNode` cannot be created directly.
It is accessible as a sub-object of `CFGParseTree`!

**Configuration:** `cfgptnode configure`

```
-bestScore   = 0.000000
-bestX       = 0
-itemN       = 1
-lvX         = 0
```

**Methods:** `cfgptnode`

`puts` [`-format format`]
display the contents of parse tree node

| | |
|---|---|
| `format` | output format (SHORT, LONG) |

`trace` [`-auxNT auxnt`] [`-topN topn`] [`-format format`]
returns parse tree by tracing back node

| | |
|---|---|
| `auxnt` | print also auxilliary NTs |
| `topn` | print the topN parse trees |
| `format` | output format (jsgf, soup) |

**Subobjects:**

| | |
|---|---|
| `child` | (`???`) |
| `next` | (`???`) |
| `parent` | (`???`) |

**Elements:** are of type `CFGPTItem`.

### 8.4.8 CFGPTItem

This section describes the '*CFGPTItem*': *A 'CFGPTItem' object is a item in a parse tree node.*

**Creation:** `CFGPTItem` cannot be created directly.
It is accessible as a sub-object of `CFGPTNode`!

**Configuration:** `cfgptitem configure`

| | |
|---|---|
| `-offset` | = 0.000000 |
| `-parentX` | = -1 |

**Methods:** `cfgptitem`

`puts` [`-format format`]
display the contents of parse tree item

| | |
|---|---|
| `format` | output format (SHORT, LONG) |

**Subobjects:**

| | |
|---|---|
| `arc` | (`CFGArc`) |
| `rsitem` | (`CFGRSItem`) |

### 8.4.9 CFGParseTree

This section describes the '*CFGParseTree*': *A 'CFGParseTree' object is a parse tree.*

**Creation:** `CFGParseTree` cannot be created directly.
It is accessible as a sub-object of `CFG`!

**Configuration:** `cfgparsetree configure`

| | |
|---|---|
| `-nodeN` | = 1 |

**Methods:** `cfgparsetree`

`puts [-format format]`

display the contents of parse tree

| | |
|---|---|
| `format` | output format (SHORT, LONG) |

`trace <spass> [-auxNT auxnt] [-topN topn] [-format format]`

returns parse tree by tracing back

| | |
|---|---|
| `spass` | single pass (`SPass`) |
| `auxnt` | print also auxilliary NTs |
| `topn` | print the topN parse trees |
| `format` | output format (jsgf, soup) |

**Subobjects:**

| | |
|---|---|
| `node(0..0)` | () |
| `root` | (`CFGPTNode`) |

### 8.4.10 CFGRSItem

This section describes the '*CFGRSItem*': *A 'CFGRSItem' object is an item in the stack of CFG rules.*

**Creation:** `CFGRSItem` cannot be created directly.

It is accessible as a sub-object of `CFGRuleStack`!

**Methods:** `cfgrsitem`

`puts [-format format]`

display the contents of this item

| | |
|---|---|
| `format` | output format (SHORT, LONG) |

**Subobjects:**

| | |
|---|---|
| `arc` | (???) |
| `child` | (???) |
| `next` | (???) |
| `parent` | (???) |

### 8.4.11 CFGRule

This section describes the '*CFGRule*': *A 'CFGRule' object is a rule of a context free grammar.*

**Creation:** `CFGRule` cannot be created directly.

It is accessible as a sub-object of `CFG`!

**Configuration:** `cfgrule configure`

| | |
|---|---|
| `-lvX` | = 0 |
| `-status` | = Active |
| `-type` | = Root_Rule |
| `-weight` | = 0.000000 |

**Methods:** `cfgrule`

`addPath <line> [-format format]`

adds a path to a rule

| | |
|---|---|
| `line` | path to add |
| `format` | grammar format |

`generate <seqN> [-mode mode] [-recurse recurse] [-file file] [-append append]`

generates sentences starting with rule

| | |
|---|---|
| `seqN` | number of terminal sequences |
| `mode` | generation mode (random—fixed) |
| `recurse` | follow recursions |
| `file` | file to write output |
| `append` | append to file |

`puts [-format format]`

display the contents of CFG rule

| | |
|---|---|
| `format` | output format (short, long) |

**Subobjects:**

| | |
|---|---|
| `cfg` | (`CFG`) |
| `leaf` | (`CFGNode`) |
| `root` | (`CFGNode`) |

### 8.4.12 CFGRuleStack

This section describes the '*CFGRuleStack*': *A 'CFGRuleStack' object is a stack of CFG rules.*

**Creation:** `CFGRuleStack` cannot be created directly.

It is accessible as a sub-object of `CFG`!

**Configuration:** `cfgrulestack configure`

`-itemN` = 1

**Methods:** `cfgrulestack`

`puts [-format format]`

display the contents of CFG rule stack

| | |
|---|---|
| `format` | output format (SHORT, LONG) |

**Subobjects:**

| | |
|---|---|
| `root` | (`CFGRSItem`) |

**Elements:** are of type `CFGRSItem`.

### 8.4.13 CFGSet

This section describes the '*CFGSet*': *A 'CFGSet' object is a set of context free grammar.*

**Creation:** `CFGSet` cannot be created directly.

It is accessible as a sub-object of `LingKS`!

**Configuration:** `cfgset configure`

| | |
|---|---|
| `-built` | = 1 |
| `-cfgN` | = 1 |
| `-name` | = c |

**Methods:** `cfgset`

`activate` <`tag`>

activates a grammar given by tag (cfgActivate)

| | |
|---|---|
| `tag` | tag of the grammar |

`build` [`-mode mode`] [`-overwrite overwrite`] [`-verbose verbose`]

builds a context free grammar set

| | |
|---|---|
| `mode` | score definition mode (null—fixed—equal—default=use stored build mode) |
| `overwrite` | overwrite predefined scores from grammar file |
| `verbose` | verbosity |

`clear` [`-free free`]

clears a context free grammar set

| | |
|---|---|
| `free` | free items instead of clearing |

`compress` [`-level level`] [`-unfold unfold`] [`-matchFile matchfile`] [`-verbose verbose`]

compress a context free grammar set

| | |
|---|---|
| `level` | compress level |
| `unfold` | unfold grammar in new top level rule |
| `matchfile` | file with matching terminals |
| `verbose` | verbosity |

`deactivate` <`tag`>

deactivates a grammar given by tag (cfgDeactivate)

| | |
|---|---|
| `tag` | tag of the grammar |

`fsm` <`rule`> <`fsm`>

write a FSM for a given rule (cfgSetWriteFSM)

| | |
|---|---|
| `rule` | rule to print as FSM |
| `fsm` | fsm file |

`generate` <`seqN`> [`-mode mode`] [`-recurse recurse`] [`-file file`]

generate terminal sequences (cfgSetGenerate)

| | |
|---|---|
| `seqN` | number of terminal sequences |
| `mode` | generation mode (random—fixed) |
| `recurse` | follow recursions |
| `file` | file name to write output |

`load` <`fileName`>

loads a context free grammar set

| | |
|---|---|
| `fileName` | file name |

`makeDict` <`baseDict`> <`dict`> [`-vocab vocab`] [`-map map`] [`-classes classes`] [`-fillers fillers`]

makes a dictionary out of a base dictionary limited to the word entries of the CFG (cfgMakeDict)

| | |
|---|---|
| `baseDict` | base dict for lookup |
| `dict` | resulting new dict |
| `vocab` | resulting search vocab |
| `map` | resulting mapping file |
| `classes` | mapping of classes |
| `fillers` | list of filler words |

`parse` <`text`> [`-verbose verbose`]

parse a sentence

| | |
|---|---|
| `text` | text to path |
| `verbose` | verbosity |

`parseTree` <`text`> [`-svmap svmap`] [`-format format`] [`-auxNT auxnt`]

returns the parse tree of a given text string (cfgGetParseTree)

| | |
|---|---|
| `text` | text string to parse |
| `svmap` | use SVMap to map SVX<->LVX (`SVMap`) |
| `format` | output format (soup—jsgf) |
| `auxnt` | print also auxilliary NTs |

`puts` [`-format format`]

display the contents of CFG set

| | |
|---|---|
| `format` | output format (short, long) |

`reduce` <`matchFile`> [`-verbose verbose`]

reduces a context free grammar set

| | |
|---|---|
| `matchFile` | file with matching terminals |
| `verbose` | verbosity |

`save` <`fileName`> [`-pt pt`]

saves a context free grammar set

| | |
|---|---|
| `fileName` | file name |
| `pt` | dump also parse tree and rule stack |

`weightRules` <`rules`> [`-weight weight`]

applies a weight to the given list of entry rules (cfgSetWeightRules)

| | |
|---|---|
| `rules` | list of rules |
| `weight` | weight apllied to all rules |

**Subobjects:**

| | |
|---|---|
| `cfg(0..0)` | () |
| `lex` | (`CFGLexicon`) |
| `list` | (`List`) |
| `pt` | (`CFGParseTree`) |
| `rs` | (`CFGRuleStack`) |

**Elements:** are of type `CFG`.

### 8.4.14 GLat

This section describes the '*GLat*': *Generic Lattice (pronounced 'Gillette, everything a man ...')*

**Creation:** `GLat` <`name`> <`SVMap`> [`-spass spass`]

| | |
|---|---|
| `name` | name of the lattice |
| `SVMap` | Search Vocabulary Mapper (`SVMap`) |
| `spass` | Search Pass Decoder (`SPass`) |

**Configuration:** `glat configure`

| | |
|---|---|
| `-alphaBeam` | = 150.000000 |
| `-expert` | = 0 |
| `-frameShift` | = 0.010000 |
| `-linkN` | = 0 |
| `-name` | = glatISLci |
| `-nodeN` | = 0 |
| `-singularLCT` | = 0 |
| `-status` | = INIT |
| `-topN` | = 0 |
| `-useN` | = 1 |

**Methods:** `glat`

`addLink` <start> <end> [-score score]

add a link to a lattice

| | |
|---|---|
| `start` | start node |
| `end` | end node |
| `score` | acoustic (delta) score |

`addNode` <word> <start> <end> [-nodeX nodex] [-score score] [-alpha alpha] [-beta beta] [-gamma gamma] [-beam beam]

add a node to a lattice

| | |
|---|---|
| `word` | search word |
| `start` | start frame |
| `end` | end frame |
| `nodex` | don't add, but configure nodeX |
| `score` | acoustic score |
| `alpha` | forward probability |
| `beta` | backward probability |
| `gamma` | a posteriori probability |
| `beam` | beam to re-use existing node |

`addPath` <path>

add a path to a lattice

| | |
|---|---|
| `path` | the path to add |

`align` <ref> [-ignoreFtag ignoreftag] [-v v]

align lattice with reference

| | |
|---|---|
| `ref` | sequence of words |
| `ignoreftag` | treat filler words as regular words |
| `v` | verbose |

`clear`

clear lattice

`clearAddon`

clear glat addon vars

`compress` [-iter iter] [-delFil delfil] [-ignoreLCT ignorelct] [-adjustTime adjusttime]

compress lattice

| | |
|---|---|
| `iter` | nr. of iterations |
| `delfil` | delete filler words |
| `ignorelct` | ignore linguistic context |
| `adjusttime` | adjust start and end points |

`confidence` <ref> [-map map] [-sum sum] [-tie tie] [-scale scale] [-norm norm] [-v v] compute confidence measure

| | |
|---|---|
| ref | sequence of words |
| map | Vocabulary Mapper (SVMap) |
| sum | sum or max over prob's |
| tie | node tying: none, svX, lvX |
| scale | mystic scaling factor |
| norm | puts real probabilities instead of negative log |
| v | puts time information |

connect [-map map] [-beam beam] [-factor factor] [-filler filler] [-sum sum]

connect matching nodes

| | |
|---|---|
| map | Vocabulary Mapper (SVMap) |
| beam | lattice beam |
| factor | multiplication factor for beam to make beam settings in old scripts compatible with bu |
| filler | connect filler words |
| sum | sum the probabilities |

consensus [-lats lats] [-map map] [-beam beam] [-scale scale] [-silScale silscale] [-cutoff cutoff] [-silWord silword] [-intra intra] [-inter inter] [-verbose verbose] [-dictWords dictwords]

find consensus in lattice(s)

| | |
|---|---|
| lats | extra list of lattices |
| map | Vocabulary Mapper (SVMap) |
| beam | pruning beam |
| scale | score scaling factor |
| silscale | silence prob scaling factor |
| cutoff | cutoff probability for output |
| silword | word to use for missed words |
| intra | intra-class merging method (max or avg) |
| inter | inter-class merging method (max, avg, old, or time) |
| verbose | verbosity |
| dictwords | output dictionary words instead of lm words |

createCN <GLat> [-optWord optword] [-factor factor] [-beam beam]

convert lattice into confusion network (createCNet)

| | |
|---|---|
| GLat | (GLat) |
| optword | optional word |
| factor | mystic scaling factor |
| beam | posteriori beam |

delLink <start> <end>

delete a link from a lattice

| | |
|---|---|
| start | start node |
| end | end node |

delNode <nodeX>

delete a node from a lattice

| | |
|---|---|
| nodeX | node index |

initAddon <hmm> <pathTmp> [-variants variants] [-modtags modtags] [-mode mode] [-verbose verbose]

initialize glat addon vars

| | |
|---|---|
| `hmm` | Hidden Markov Model (`HMM`) |
| `pathTmp` | Temporary path variable (`Path`) |
| `variants` | pronunciation variants |
| `modtags` | modality tags |
| `mode` | alignment mode, 0=viterbi, 1=fwdbwd |
| `verbose` | verbosity |

`map` [`-map map`]

map vocabulary words in lattice nodes

| | |
|---|---|
| `map` | Vocabulary Mapper (`SVMap`) |

`posteriori` [`-map map`] [`-scale scale`] [`-sum sum`] [`-tie tie`] [`-tieFiller tiefiller`] [`-mmie mmie`]

compute a-posteriori probabilities

| | |
|---|---|
| `map` | Vocabulary Mapper (`SVMap`) |
| `scale` | mystic scaling factor |
| `sum` | sum or max over prob's |
| `tie` | node tying: none, svX, lvX |
| `tiefiller` | include filler words for clustering |
| `mmie` | use functions for MMIE training |

`prune` [`-beam beam`] [`-factor factor`] [`-scale scale`] [`-sum sum`] [`-nodeN noden`] [`-link link`] [`-map map`]

prune lattice nodes

| | |
|---|---|
| `beam` | lattice beam |
| `factor` | multiplication factor for beam to make beam settings in old scripts compatible with bugfixes |
| `scale` | scaling factor |
| `sum` | sum the probabilities |
| `noden` | prune to absolute nr. of nodes |
| `link` | prune lattice links |
| `map` | Vocabulary Mapper (`SVMap`) |

`purify`

delete non-terminating nodes and links

`puts`

displays the contents of a lattice

`read` <`file`> [`-garbageWord garbageword`]

read a lattice from file

| | |
|---|---|
| `file` | file to read from |
| `garbageword` | garbage word |

`recombine` [`-map map`] [`-connect connect`] [`-verbose verbose`]

recombine lattice nodes with equal LCT

| | |
|---|---|
| `map` | mapper object (`SVMap`) |
| `connect` | connect nodes |
| `verbose` | verbosity |

`rescore` [`-map map`] [`-conf conf`] [`-topN topn`] [`-maxN maxn`] [`-beam beam`] [`-v v`]

rescore a lattice using svMap

| | |
|---|---|
| `map` | mapper object between svocab and language model (`SVMap`) |
| `conf` | do posteriori rescoring |
| `topn` | how many hypotheses do we want |
| `maxn` | size of hypotheses stack |
| `beam` | beam threshold to prune hypotheses stack |
| `v` | verbose output (-1 = index only, 0 = name only, 1 = name, pos, and score, 2 = gamma) |

`singularLCT <lattice> [-map map] [-verbose verbose]`

expand the lattice with respect to LCT

| | |
|---|---|
| `lattice` | Lattice to process (GLat) |
| `map` | Vocabulary Mapper to use (SVMap) |
| `verbose` | verbosity |

`splitMW [-map map]`

split nodes which contain multiwords

| | |
|---|---|
| `map` | Vocabulary Mapper (SVMap) |

`warp [-shift shift] [-factor factor] [-frameN framen] [-scores scores]`

warp (scale) time axis

| | |
|---|---|
| `shift` | frame shift after warping |
| `factor` | relative scaling factor |
| `framen` | number of frames after warping |
| `scores` | scale scores |

`write <file> [-format format] [-utt utt] [-mode mode] [-map map] [-from from]`

write a lattice to file

| | |
|---|---|
| `file` | file to write to |
| `format` | file format (njd or slf) |
| `utt` | utterance ID (optional) |
| `mode` | mode |
| `map` | Vocabulary Mapper (for SLF) (SVMap) |
| `from` | Start time of this utterance |

`writeCTM <speaker> <uttID> [-field field] [-conv conv] [-channel channel] [-from from] [-file file] [-result result] [-map map] [-topX topx] [-topN topn] [-maxN maxn] [-beam beam] [-rate rate] [-warpA warpa] [-conf conf] [-lz lz] [-silsmb silsmb] [-v v]`

write hypo in CTM format (glatWriteHypo)

| | |
|---|---|
| `speaker` | speaker ID |
| `uttID` | utterance ID |
| `field` | conversation field in DB |
| `conv` | conversation or episode |
| `channel` | channel |
| `from` | start point |
| `file` | filename |
| `result` | result from rescoring |
| `map` | SVMap (SVMap) |
| `topx` | topX |
| `topn` | topN |
| `maxn` | maxN |
| `beam` | beam |
| `rate` | rate |
| `warpa` | warpA |
| `conf` | write confidences instead of scores |
| `lz` | lz for confidences |
| `silsmb` | symbol for opt sil |
| `v` | verbose |

`writeSRT <speaker> <uttID> [-from from] [-file file] [-result result] [-map map] [-topX topx] [-topN topn] [-maxN maxn] [-beam beam] [-rate rate] [-warpA warpa] <>`

write hypo in SRT format (glatWriteSRT)

| | |
|---|---|
| `speaker` | speaker ID |
| `uttID` | utterance ID |
| `from` | start point |
| `file` | filename |
| `result` | result from rescoring |
| `map` | SVMap (`SVMap`) |
| `topx` | topX |
| `topn` | topN |
| `maxn` | maxN |
| `beam` | beam |
| `rate` | rate |
| `warpa` | warpA |

**writeTRN** `<speaker> <uttID> [-from from] [-file file] [-result result] [-map map] [-topX topx] [-topN topn] [-maxN maxn] [-beam beam] [-rate rate] [-warpA warpa] [-time time]`

write hypo in TRN format (glatWriteTRN)

| | |
|---|---|
| `speaker` | speaker ID |
| `uttID` | utterance ID |
| `from` | start point |
| `file` | filename |
| `result` | result from rescoring |
| `map` | SVMap (`SVMap`) |
| `topx` | topX |
| `topn` | topN |
| `maxn` | maxN |
| `beam` | beam |
| `rate` | rate |
| `warpa` | warpA |
| `time` | include time information |

**Subobjects:**

| | |
|---|---|
| `lctMem` | (`BMem`) |
| `linkMem` | (`BMem`) |
| `nodeMem` | (`BMem`) |
| `pathMem` | (`BMem`) |
| `rcmMem` | (`BMem`) |

### 8.4.15 LCMSet

This section describes the '*LCMSet*': *set of left context models*

**Creation:** `LCMSet <name> <PHMMSet>`

| | |
|---|---|
| `name` | name of the LCM set |
| `PHMMSet` | phone HMM Set (`PHMMSet`) |

**Methods:** `lcmset`

`load <filename>`

load a set of left context models

| | |
|---|---|
| `filename` | file to load from |

`puts`

displays the set of left context models

`save <filename>`

save a set of left context models

| | |
|---|---|
| `filename` | file to load from |

**Subobjects:**

| | |
|---|---|
| `phmmSet` | (`PHMMSet`) |

### 8.4.16 LingKS

This section describes the '*LingKS*': *Generic Linguistic Knowledge Source:*

**Creation:** `LingKS <name> <type>`

| | |
|---|---|
| `name` | name of the linguistic knowledge source |
| `type` | Kind of LingKS: NGramLM—PhraseLM—MetaLM—CFG—CFGSet |

**Configuration:** `lingks configure`

| | |
|---|---|
| `-dirty` | = 1 |
| `-gInterpol` | = 0 |
| `-name` | = c |
| `-type` | = CFGSet |
| `-useN` | = 1 |
| `-weight` | = 0.000000 |

**Methods:** `lingks`

`index <word>`

return the internal index of an LingKSItem

| | |
|---|---|
| `word` | word you want the index for |

`load <fileName>`

loads an LM-file (dump and generic files)

| | |
|---|---|
| `fileName` | file name |

`name <index>`

return the name of an LingKSItem

| | |
|---|---|
| `index` | index of element to print |

`puts [-format format]`

display the contents of an LingKS

| | |
|---|---|
| `format` | output format (short, long) |

`save <fileName> [-pt pt]`

create binary dump of LM

| | |
|---|---|
| `fileName` | file name |
| `pt` | dump also parse tree and rule stack |

`score <word sequence> [-idx idx] [-array array] [-usehistory usehistory] [-map map] [-startString startstring] [-ignoreS ignores]`

return the score of a text string

| | |
|---|---|
| `word sequence` | sequence of words |
| `idx` | start index for conditional probabilities |
| `array` | use ScoreArray, implies idx == n-1 |
| `usehistory` | use the stored reduced history |
| `map` | use vocab mapper (`SVMap`) |
| `startstring` | different start string than <s> |
| `ignores` | ignore initial start string in scoring |

**Subobjects:**

| | |
|---|---|
| `CFGSet` | (`CFGSet`) |
| `data` | (`CFGSet`) |

### 8.4.17 LTree

This section describes the '*LTree*': *Language-Model Look-Ahead object (Lexical tree)*

**Creation:** `LTree <name> <SearchTree> [-map map] [-depth depth] [-reduced reduced]`

| | |
|---|---|
| `name` | name of the LTree |
| `SearchTree` | Search tree (`STree`) |
| `map` | Vocabulary mapper to use for LookAhead only (`SVMap`) |
| `depth` | Maximum depth of LookAhead tree |
| `reduced` | Set 'reduce' flag for LookAhead nodes |

**Configuration:** `ltree configure`

| | |
|---|---|
| `-cacheN` | = 100 |
| `-cachehits` | = 0 |
| `-depth` | = 5 |
| `-expert` | = 0 |
| `-lctMax` | = 100000 |
| `-lm(leafs)` | = lmISLci |
| `-lm(nodes)` | = lmISLci |
| `-map(leafs)` | = svmapISLci |
| `-map(nodes)` | = svmapISLci |
| `-mode` | = array |
| `-name` | = ltreeISLci |
| `-ncacheN` | = 10 |
| `-nodecachehits` | = 0 |
| `-pcacheN` | = 0 |
| `-queries` | = 0 |
| `-reduced` | = 0 |
| `-svxHash` | = 1 |
| `-svxMax` | = 100000 |
| `-useN` | = 2 |

**Methods:** `ltree`

`constrain <GLat> [-mode mode] [-type type] [-padX padx]`

create GLat constraint for LTree

| | |
|---|---|
| `GLat` | GLat (or NULL to deactivate constraint) |
| `mode` | flat—weak—exact—time |
| `type` | SVX—LVX |
| `padx` | padding for time based constraints |

`fillCtx <w1> <w2>`

fills a LTree object with scores for a specific lct

| | |
|---|---|
| `w1` | w1 context |
| `w2` | w2 context |

`puts`

displays the contents of a LTree

**Subobjects:**

| | |
|---|---|
| `cachehits` | (???) |
| `latlmM` | (BMem) |
| `latlmN` | (???) |
| `nodecachehits` | (???) |
| `queries` | (???) |

### 8.4.18 MetaLM

This section describes the '*MetaLM*': *Meta language model: flexible LM using sub-LMs.*

**Creation:** `MetaLM` cannot be created directly.

It is accessible as a sub-object of `LingKS`!

**Configuration:** `metalm configure`

| | |
|---|---|
| `-blkSize` | = 1000 |
| `-elemN` | = 2 |
| `-itemN` | = 1 |
| `-lvxCache` | = 0 |
| `-lvxCacheN` | = 0 |
| `-mlctMax` | = 200000 |
| `-mlctN` | = 0 |
| `-order` | = -1 |

**Methods:** `metalm`

`LMadd` <`LingKS`> [`-weight weight`]

add a language model for usage with metaLM

| | |
|---|---|
| `LingKS` | Linguistic Knowledge Source (`LingKS`) |
| `weight` | weight |

`LMindex` <`names*`>

return the internal index of an atomic LM

| | |
|---|---|
| `names*` | list of names |

`LMname` <`idx*`>

return the name of an element (atomic LM)

| | |
|---|---|
| `idx*` | list of indices |

`add` <`LM word`> [`-lksA lksa`] [`-lksB lksb`] [`-nameA namea`] [`-nameB nameb`] [`-prob prob`]

add an item (using atomic LMs)

| | |
|---|---|
| `LM word` | LM word in this model |
| `lksa` | Language Model A |
| `lksb` | Language Model B |
| `namea` | corresponding word in LM A |
| `nameb` | corresponding word in LM B |
| `prob` | probability |

`cover` [`-lksA lksa`] [`-lksB lksb`] [`-prob prob`]

cover an element (read all words from it)

| | |
|---|---|
| `lksa` | index of atomic LM to read words from |
| `lksb` | index of atomic LM to connect with |
| `prob` | probability |

`get <word>`
: get the parameters for one item
  `word` item

`isStopWord <word>`
: check if a word is a stopword
  `word` a word

`list`
: list the currently available LMs

`loadStopList <filename>`
: loads a list of stopwords (must be called after LM is loaded)
  `filename` path to the stopword list

`loadWeights <file>`
: load interpolation weights (metaLMloadWeights)
  `file` weight file

`puts`
: display the contents of a MetaLM

`scoreFct <function>`
: change the score function
  `function` score function

**Subobjects:**

| | |
|---|---|
| `list` | (`List`) |
| `lm(0..1)` | (`???`) |

**Elements:** are of type `MetaLMItem`.

### 8.4.19 `MetaLMElem`

This section describes the '*MetaLMElem*': *Meta language model element (sub-LM).*

**Creation:** `MetaLMElem` cannot be created directly.
It is accessible as a sub-object of `MetaLM`!

**Configuration:** `metalmelem configure`

```
-name    = lmISLci
-weight  = 1.000000
```

### 8.4.20 `MetaLMItem`

This section describes the '*MetaLMItem*': *Meta language model item.*

**Creation:** `MetaLMItem` cannot be created directly.
It is accessible as a sub-object of `MetaLM`!

**Configuration:** `metalmitem configure`

```
-idxA  = 0
-idxB  = 0
-lmA   = 0
-lmB   = 0
-name  = <UNK>
-prob  = 0.000000
```

### 8.4.21 NGramLM

This section describes the '*NGramLM*': *N-gram Language Model*

**Creation:** `NGramLM` cannot be created directly.

It is accessible as a sub-object of `LingKS`!

**Configuration:** `ngramlm configure`

| | |
|---|---|
| `-blkSize` | = 1000 |
| `-hashLCT` | = 0 |
| `-history` | = 0 |
| `-itemN` | = 3 |
| `-log0` | = -99.000000 |
| `-log0Val` | = -5.000000 |
| `-order` | = 1 |
| `-segSize` | = 6 |

**Methods:** `ngramlm`

`connectSriServer <host> <port> <order> [-vocabFile vocabfile]`

Connect to a running SRI LM Server

| | |
|---|---|
| `host` | host that the server runs on |
| `port` | port that the server runs on |
| `order` | port that the server runs on |
| `vocabfile` | file to read vocab from |

`disconnectSriServer`

disconnect from the SRI LM Server

`readVocab <filename>`

fill NGram item list from vocab file, e.g. for SRI LM usage

| | |
|---|---|
| `filename` | file to read the vocab from |

**Subobjects:**

| | |
|---|---|
| `backOffA(1..0,0..N)` | (???) |
| `idA(2..1,0..N)` | (???) |
| `linkA(1..0,0..N)` | (???) |
| `list` | (`List`) |
| `mgramN(1..1)` | (3) |
| `probA(1..1,0..N)` | (???) |
| `subslist` | (`List`) |

**Elements:** are of type `NGramLMItem`.

### 8.4.22 NGramLMItem

This section describes the '*NGramLMItem*': *N-gram Language Model Item*

**Creation:** `NGramLMItem` cannot be created directly.

It is accessible as a sub-object of `NGramLM`!

**Configuration:** `ngramlmitem configure`

| | |
|---|---|
| `-linkX` | = 0 |
| `-name` | = <UNK> |

### 8.4.23 PHMMSet

This section describes the '*PHMMSet*': *set of phone hidden markov models*

**Creation:** `PHMMSet <name> <TTree> <TTreeRoot> [-useCtx usectx]`

| | |
|---|---|
| `name` | name of the PHMM set |
| `TTree` | topology tree (`Tree`) |
| `TTreeRoot` | root name in TTree |
| `usectx` | use HMM context table 0/1 |

**Methods:** `phmmset`

`add <states> <trans>`

Add a PHMM by specifying a state graph

| | |
|---|---|
| `states` | list of states |
| `trans` | list of transitions for each state |

`load <filename>`

load a set of Phone models

| | |
|---|---|
| `filename` | file to load from |

`puts`

displays the set of Phone models

`save <filename>`

save a set of Phone models

| | |
|---|---|
| `filename` | file to save to |

**Subobjects:**

| | |
|---|---|
| `senoneSet` | (`SenoneSet`) |
| `tmSet` | (`TmSet`) |
| `tree` | (`Tree`) |

### 8.4.24 PhraseLM

This section describes the '*PhraseLM*': *This module takes a LM and adds phrases (aka. multi-words) to it.*

**Creation:** `PhraseLM` cannot be created directly.
It is accessible as a sub-object of `LingKS`!

**Configuration:** `phraselm configure`

| | |
|---|---|
| `-baseLM` | = lmISLci |
| `-baseN` | = 3 |
| `-bias` | = 0.000000 |
| `-history` | = 1 |
| `-itemN` | = 0 |
| `-order` | = 1 |

**Methods:** `phraselm`

`add <search word> <LM word string> [-prob prob] [-v v]`

add a mapping for a phrase

| | |
|---|---|
| `search word` | search vocabulary word |
| `LM word string` | language-model word(s) |
| `prob` | probability |
| `v` | verbose |

`base <LingKS>`

define the base LingKS

| | |
|---|---|
| `LingKS` | Base Linguistic Knowledge source (`LingKS`) |

`puts`

display the contents of a PhraseLM

`readMapFile <file> [-mode mode] [-verbose verbose] [-base base]`

read multi-words from an existing JANUS-Format map file (phraseLM-ReadMap)

| | |
|---|---|
| `file` | map-file to read in |
| `mode` | add which entries (base, multi, all) |
| `verbose` | verbose |
| `base` | underlying lm |

`readSubs [-lks lks]`

read map-table from 'NGramLM' object

| | |
|---|---|
| `lks` | Linguistic Knowledge Source (`LingKS`) |

**Subobjects:**

| | |
|---|---|
| `list` | (`List`) |

### 8.4.25 RCMSet

This section describes the '*RCMSet*': *set of right context models*

**Creation:** `RCMSet <name> <PHMMSet>`

| | |
|---|---|
| `name` | name of the RCM set |
| `PHMMSet` | phone HMM Set (`PHMMSet`) |

**Methods:** `rcmset`

`load <filename>`

load a set of right context models

| | |
|---|---|
| `filename` | file to load from |

`puts`

displays the set of right context models

`save <filename>`

save a set of right context models

| | |
|---|---|
| `filename` | file to load from |

**Subobjects:**

| | |
|---|---|
| `phmmSet` | (`PHMMSet`) |

### 8.4.26 SMem

This section describes the '*SMem*': *Search Memory Manager*

**Creation:** `SMem` cannot be created directly.

It is accessible as a sub-object of `STree`!

**Configuration:** `smem configure`

```
-level      = -1
-morphBlk   = 2
-smemFree   = 1
```

**Methods:** `smem`

`puts`
displays the contents of a memory manager

**Subobjects:**

| | |
|---|---|
| c | (BMem) |
| f | (BMem) |
| li | (BMem) |
| n | (BMem) |
| ni | (BMem) |
| p | (BMem) |
| r | (BMem) |
| ri | (BMem) |

### 8.4.27 SPass

This section describes the '*SPass*': *Single Pass Decoder*

**Creation:** `SPass <name> <STree> <LTree>`

| | |
|---|---|
| name | name of the search pass objects |
| STree | Search Tree (STree) |
| LTree | LM Tree (LTree) |

**Configuration:** `spass configure`

```
-fastMatch   = 0.000000
-frameX      = 0
-morphBeam   = 80.000000
-morphN      = 8
-name        = spassISLci
-stateBeam   = 130.000000
-transN      = 35
-useN        = 1
-wordBeam    = 90.000000
```

**Methods:** `spass`

`fmatch <senoneSet> [-frameN framen] [-factor factor] [-snTag sntag]`
initialize fast match module

| | |
|---|---|
| senoneSet | set of senones (SenoneSet) |
| framen | nr. of fast match frames |
| factor | weighting factor for fast match models |
| sntag | sequence of senone tags |

`pathSub <val>`
subtract a constant score value from all active paths

| | |
|---|---|
| val | value subtracted from all active paths |

`puts`
puts information

`reinit` [`-start start`]

reinit decoder after changes in search network

| | |
|---|---|
| `start` | frameX for restart |

`run` [`-to to`] [`-init init`]

run decoder using the underlying search network

| | |
|---|---|
| `to` | frameN |
| `init` | initialize search tree |

`traceStable` [`-fromX fromx`] [`-toX tox`] [`-bpL bpl`] [`-v v`]

trace back stable hypothesis

| | |
|---|---|
| `fromx` | start frame for trace back |
| `tox` | final frame for trace back |
| `bpl` | list of backpointers to end trace back |
| `v` | verbose output |

`writeCTM` <`speaker`> <`uttID`> [`-field field`] [`-conv conv`] [`-channel channel`] [`-from from`] [`-file file`] [`-rate rate`] [`-silsmb silsmb`] [`-warpA warpa`]

write hypo in CTM format (spassWriteHypo)

| | |
|---|---|
| `speaker` | speaker ID |
| `uttID` | utterance ID |
| `field` | conversation field in DB |
| `conv` | conversation or episode |
| `channel` | channel |
| `from` | start point |
| `file` | filename |
| `rate` | rate |
| `silsmb` | symbol for opt sil |
| `warpa` | warpA |

**Subobjects:**

| | |
|---|---|
| `glat` | (`GLat`) |
| `stab` | (`STab`) |
| `stree` | (`STree`) |

### 8.4.28 STab

This section describes the '*STab*': *Backpointer table*

**Creation:** `STab` cannot be created directly.

It is accessible as a sub-object of `SPass`!

**Methods:** `stab`

`puts` [`-fromX fromx`] [`-toX tox`]

displays the contents of a backpointer table

| | |
|---|---|
| `fromx` | from frame |
| `tox` | to frame |

`trace` [`-bpIdx bpidx`] [`-frameIdx frameidx`] [`-fromX fromx`] [`-bpL bpl`] [`-v v`]

trace back from final state

| | |
|---|---|
| `bpidx` | final state for trace back |
| `frameidx` | final frame for trace back |
| `fromx` | trace back until frame fromX |
| `bpl` | list of backpointers to end trace back |
| `v` | verbose output |

### 8.4.29 STree

This section describes the '*STree*': *Search Tree*

**Creation:** `STree <name> <SVMap> <LCMSet> <RCMSet> [-XCMSet xcmset] [-dump dump] [-level level] [-morphBlk morphblk] [-smemFree smemfree] [-v v]`

| | |
|---|---|
| `name` | name of the search tree |
| `SVMap` | Vocabulary Mapper (`SVMap`) |
| `LCMSet` | Set of left context models (`LCMSet`) |
| `RCMSet` | Set of right context models (`RCMSet`) |
| `xcmset` | Set of left and right context models (`XCMSet`) |
| `dump` | Search Tree dump file |
| `level` | tree level for memory management |
| `morphblk` | block size for memory management |
| `smemfree` | memory management mode |
| `v` | verbose tree dump |

**Configuration:** `stree configure`

| | |
|---|---|
| `-compress` | = 0 |
| `-leafN` | = 0 |
| `-name` | = streeISLci |
| `-nodeN` | = 0 |
| `-rootN` | = 0 |
| `-sdpN` | = 0 |
| `-sipN` | = 3 |
| `-useN` | = 3 |

**Methods:** `stree`

`add <word> [-phmmX phmmx]`

add word to search tree

| | |
|---|---|
| `word` | word |
| `phmmx` | PHMM index |

`compress [-v v]`

compress search tree, convert tree into generalized graph structure

| | |
|---|---|
| `v` | verbose output |

`delete <word>`

delete word from search tree

| | |
|---|---|
| `word` | word |

`dump <filename> [-dumpLM dumplm]`

dump search tree

| | |
|---|---|
| `filename` | file to dump |
| `dumplm` | dump lm |

`puts`

puts information

`trace <word>`

trace search tree

| | |
|---|---|
| `word` | word |

**Subobjects:**

| | |
|---|---|
| `lcmSet` | (`LCMSet`) |
| `rcmSet` | (`RCMSet`) |
| `root(0..-1)` | (`???`) |
| `smem` | (`SMem`) |
| `svMap` | (`SVMap`) |
| `xcmSet` | (`XCMSet`) |

### 8.4.30 SVMap

This section describes the '*SVMap*': *Search Vocabulary Mapper*

**Creation:** `SVMap <name> <SVocab> <LingKS>`

| | |
|---|---|
| `name` | name of the SVMap |
| `SVocab` | Search Vocabulary (`SVocab`) |
| `LingKS` | Linguistic Knowledge Source (`LingKS`) |

**Configuration:** `svmap configure`

| | |
|---|---|
| `-baseLM` | = lmISLci |
| `-baseVocab` | = svocabISLci |
| `-cacheN` | = 0 |
| `-calls` | = 0 |
| `-dirty` | = 0 |
| `-endString` | = </s> |
| `-filPen` | = 10.000000 |
| `-hits` | = 0 |
| `-lalz` | = 32.000000 |
| `-lvN` | = 0 |
| `-lz` | = 32.000000 |
| `-name` | = svmapISLci |
| `-phonePen` | = 0.000000 |
| `-startString` | = <s> |
| `-svN` | = 4 |
| `-unkString` | = <UNK> |
| `-useN` | = 5 |
| `-wordPen` | = 3.000000 |
| `-xN` | = 0 |

**Methods:** `svmap`

`add <search word> <LM word> [-prob prob]`

add or alter map entry

| | |
|---|---|
| `search word` | search vocabulary word |
| `LM word` | language-model word |
| `prob` | 'probability' (>0 is higher) |

`delete <word>`

delete map entry

| | |
|---|---|
| `word` | vocabulary word |

`get <search word>`

prints out mapping for vocabulary word

| | |
|---|---|
| `search word` | the search word |

`index <n>`

show mapping entry

| | |
|---|---|
| `n` | index |

`load <filename>`

load Mapping from binary file

| | |
|---|---|
| `filename` | file name |

`map <mapType> [-verbose verbose]`

map SVocab indices to LM indices

| | |
|---|---|
| `mapType` | id, base, class |
| `verbose` | verbosity |

`mappedto <word>`

list words mapped to a particular word

| | |
|---|---|
| `word` | word to search for, empty string for filler words |

`match <lm> <words> <text> [-variants variants]`

find best match for word (ngramLMMatch)

| | |
|---|---|
| `lm` | lm to use (`LingKS`) |
| `words` | words to find |
| `text` | text to use in file |
| `variants` | include variants in search |

`puts ( <s> 0.000000 ) </s> 0.000000`

prints out map table

```
( s 0.000000
) /s 0.000000
```

`read <filename>`

read an LMMap file

| | |
|---|---|
| `filename` | file to read from |

`readMapFile <file> [-verbose verbose] [-lm lm]`

read mappings from an existing JANUS-Format map file (svmapReadMap)

| | |
|---|---|
| `file` | map-file to read in |
| `verbose` | verbosity |
| `lm` | underlying lm |

`readSubs [-lks lks]`

read map-table from 'NGramLM' object

| | |
|---|---|
| `lks` | Linguistic Knowledge Source (`LingKS`) |

`save <filename>`

save Mapping to binary file

| | |
|---|---|
| `filename` | file name |

**Subobjects:**

| | |
|---|---|
| `lingks` | (`LingKS`) |
| `svocab` | (`SVocab`) |

### 8.4.31 SVocab

This section describes the '*SVocab*': *Search Vocabulary*

**Creation:** `SVocab <name> <Dictionary>`

| | |
|---|---|
| `name` | name of the vocabulary |
| `Dictionary` | Dictionary (`Dictionary`) |

**Configuration:** `svocab configure`

| | |
|---|---|
| `-blkSize` | = 500 |
| `-endString` | = ) |
| `-itemN` | = 4 |
| `-name` | = svocabISLci |
| `-nilString` | = IamtheNILword |
| `-startString` | = ( |
| `-svxMax` | = -1 |
| `-useN` | = 2 |

**Methods:** `svocab`

`add <word> [-ftag ftag] [-fTag ftag] [-pron pron]`

add a word to the vocabulary

| | |
|---|---|
| `word` | name |
| `ftag` | filler tag |
| `ftag` | filler tag (too) |
| `pron` | pronunciation |

`delete <word>`

delete a word from the vocabulary

| | |
|---|---|
| `word` | word to delete |

`index`

return the internal index of a search vocab word

`load <filename>`

load Vocabulary from binary file

| | |
|---|---|
| `filename` | file name |

`puts`

displays the contents of a search vocabulary

`read <filename>`

read Vocabulary from file

| | |
|---|---|
| `filename` | file name |

`save <filename>`

save Vocabulary to binary file

| | |
|---|---|
| `filename` | file name |

`sync [-f f] [-v v]`

synchronize vocabulary with dictionary

| | |
|---|---|
| `f` | force update for word candidates |
| `v` | verbose output |

**Subobjects:**

| | |
|---|---|
| `dict` | (`Dictionary`) |
| `list` | (`List`) |

**Elements:** are of type `SWord`.

### 8.4.32 SWord

This section describes the '*SWord*': *Search Vocabulary Word*

**Creation:** `SWord` cannot be created directly.
It is accessible as a sub-object of `List`!

**Configuration:** `sword configure`

| | |
|---|---|
| `-dictX` | = 0 |
| `-fTag` | = 1 |

**Methods:** `sword`

`puts`
displays the contents of a search vocabulary word

### 8.4.33 XCMSet

This section describes the '*XCMSet*': *set of left/right context models*

**Creation:** `XCMSet <name> <PHMMSet> [-ignoreRCM ignorercm]`

| | |
|---|---|
| `name` | name of the XCM set |
| `PHMMSet` | phone HMM Set (`PHMMSet`) |
| `ignorercm` | ignore right context dependency |

**Methods:** `xcmset`

`load <filename>`
load a set of left/right context models

| | |
|---|---|
| `filename` | file to load from |

`puts`
displays the set of left/right context models

`save <filename>`
save a set of left/right context models

| | |
|---|---|
| `filename` | file to load from |

**Subobjects:**

| | |
|---|---|
| `phmmSet` | (`PHMMSet`) |

## 8.5 Acoustic models (src/models)

### 8.5.1 AModel

This section describes the '*AModel*': *acoustic model*

**Creation:** `AModel` cannot be created directly.
It is accessible as a sub-object of `PhoneGraph`!

**Configuration:** `amodel configure`

| | |
|---|---|
| `-durX` | = -1 |
| `-topoX` | = 0 |

**Methods:** `amodel`

`puts`
displays the contents of an amodel

### 8.5.2 AModelSet

This section describes the '*AModelSet*': *set of acoustic models*

**Creation:** `AModelSet <name> <TTree> <TTreeRoot> [-durationTree durationtree] [-durationRoot durationroot] [-contextCache contextcache]`

| | |
|---|---|
| `name` | name of the amodel set |
| `TTree` | topology tree (`Tree`) |
| `TTreeRoot` | root name in TTree |
| `durationtree` | duration tree (`Tree`) |
| `durationroot` | duration tree root |
| `contextcache` | 1 = create context cache |

**Configuration:** `amodelset configure`

| | |
|---|---|
| `-durRoot` | = -1 |
| `-durTree` | = (null) |
| `-name` | = amodelSetISLci |
| `-senoneSet` | = senoneSetISLci |
| `-tmSet` | = tmSetISLci |
| `-tree` | = ttreeISLci |
| `-treeRoot` | = 0 |
| `-useN` | = 7 |

**Methods:** `amodelset`

`add <senones> <trans>`
add a state graph to a set

| | |
|---|---|
| `senones` | list of senones |
| `trans` | list of transition models |

`get <tagged phones> <leftContext> <rightContext>`
find acoustic model given a phonetic context

| | |
|---|---|
| `tagged phones` | list of tagged phones |
| `leftContext` | left context |
| `rightContext` | right context |

`puts`
displays the contents of an amodel set

`reset`
remove all amodels from the set

`scale <scale>`
scale transition penalties

| | |
|---|---|
| `scale` | scale factor |

`skip <skip>`
switch to 3state skip topologies

| | |
|---|---|
| `skip` | 0/1 use skip architecture |

**Subobjects:**

| | |
|---|---|
| `senoneSet` | (`SenoneSet`) |
| `tmSet` | (`TmSet`) |
| `tree` | (`Tree`) |

### 8.5.3 `BBINode`

This section describes the '*BBINode*': *node in a BBI search tree*

**Creation:** `BBINode` cannot be created directly.

It is accessible as a sub-object of `BBITree`!

**Configuration:** `bbinode configure`

| | |
|---|---|
| `-h` | = 0.000000 |
| `-k` | = 0 |

### 8.5.4 `Cbcfg`

This section describes the '*Cbcfg*': *configuration of a codebook*

**Creation:** `Cbcfg <name>`

| | |
|---|---|
| `name` | name of the object |

**Configuration:** `cbcfg configure`

| | |
|---|---|
| `-E` | = 1.000000 |
| `-H` | = 0.800000 |
| `-I` | = 0.000000 |
| `-accu` | = y |
| `-bbiOn` | = 1 |
| `-beta` | = -1.000000 |
| `-expT` | = -100.000000 |
| `-mergeThresh` | = 10.000000 |
| `-method` | = m |
| `-minCv` | = 6.000000 |
| `-minRv` | = 1.000000 |
| `-momentum` | = 0.000000 |
| `-momentumCv` | = -1.000000 |
| `-name` | = cbcfg |
| `-rdimN` | = 0 |
| `-rhoGlob` | = 1.000000 |
| `-splitStep` | = 0.010000 |
| `-topN` | = 0 |
| `-update` | = y |
| `-useN` | = 3 |
| `-weight` | = 1.000000 |

### 8.5.5 `Codebook`

This section describes the '*Codebook*': *Codebook*

**Creation:** `Codebook` cannot be created directly.

It is accessible as a sub-object of `CodebookSet`!

**Configuration:** `codebook configure`

| | |
|---|---|
| `-bbiX` | = -1 |
| `-bbiY` | = 0 |
| `-cfg` | = default |
| `-count(0..3)` | = 0.000000 |
| `-dimN` | = 4 |
| `-featX` | = 0 |
| `-featY` | = -1 |
| `-name` | = SIL |
| `-refMax` | = 0 |
| `-refN` | = 4 |
| `-type` | = DIAGONAL |
| `-useN` | = 2 |

**Methods:** `codebook`

`:=` <`source`>

copy the parameters of one codebook into another

| | |
|---|---|
| `source` | name of the source codebook (`Codebook`) |

`accuMatrix` <`fmatrix`>

accumulate data from fmatrix

`fmatrix`

`add` <`codebook`> <`count`> <`codebook`> <`count`>

add two one-dimensional codebooks

| | |
|---|---|
| `codebook` | first codebook (`Codebook`) |
| `count` | count for first codebook |
| `codebook` | second codebook (`Codebook`) |
| `count` | count for second codebook |

`alloc` [`-compress compress`] [`-mode mode`]

allocate the codebook

| | |
|---|---|
| `compress` | compressed codebook |
| `mode` | mode for compressed codebooks (ask Hagen at soltau@ira.uka.de) |

`bhattacharyaMatrix` <`codebook`> <`fmatrix`>

compute pairwise bhattacharya distances of codebook components; store result in FMatrix

| | |
|---|---|
| `codebook` | Codebook to calculate the distance to (`Codebook`) |
| `fmatrix` | FMatrix to store the distances in (`FMatrix`) |

`covarShift` <`shift`>

add a constant value to all variances

| | |
|---|---|
| `shift` | shift value to be added |

`covarTie` <`indexList`>

tie covariance matrices together

| | |
|---|---|
| `indexList` | indices of matrices to be tied |

`covarTie?`

show which covariance matrices are tied together

`covarType` <`n`> <`type`>

modify the type of covariance matrix

| | |
|---|---|
| `n` | index of the reference vector |
| `type` | desired type of the covariance matix |

`covarUntie` <`indexList`>

untie covariance matrices

| | |
|---|---|
| `indexList` | indices of matrices to get their own copy |

`createAccu` [`-subN subn`]

create an accumulator

| | |
|---|---|
| `subn` | number of subaccumulators |

`createMap` <`n`>

create a codebook map

| | |
|---|---|
| `n` | length of map |

`extMhnMatrix` <`codebook`> <`fmatrix`>

compute pairwise extended Mahanalobis distances of codebook components; store result in FMatrix

| | |
|---|---|
| `codebook` | Codebook to calculate the distance to (`Codebook`) |
| `fmatrix` | FMatrix to store the distances in (`FMatrix`) |

`freeAccu`

remove an accumulator

`freeMap`

remove a codebook map

`invert` [`-updateDet updatedet`]

invert covariance matrix to get original one

| | |
|---|---|
| `updatedet` | update log(det(covar)) before inversion |

`klmatrix` <`codebook`> <`fmatrix`>

compute pairwise KL-distances of codebook components; store result in FMatrix

| | |
|---|---|
| `codebook` | Codebook to calculate the distance to (`Codebook`) |
| `fmatrix` | FMatrix to store the distances in (`FMatrix`) |

`lin2log`

transformation into log domain

`log2lin`

transformation into linear domain

`noise` <`codebook`> [`-s s`] [`-n n`]

adding of a noise cb (lin domain!)

| | |
|---|---|
| `codebook` | noise codebook (`Codebook`) |
| `s` | weight for speech |
| `n` | weight for noise |

`set` <`matrix`> [`-refX refx`] [`-dimX dimx`]

set reference vectors in the codebook

| | |
|---|---|
| `matrix` | matrix of reference vectors |
| `refx` | index of the reference vector |
| `dimx` | index of the dimension |

`split` [`-max max`] [`-beam beam`]

split codebook (create map)

| | |
|---|---|
| `max` | splitting beam |
| `beam` | max. number of splits |

`splitList`

codebook split candidates

`stepdiag` <modulo> [-mode mode]
: create step-diagonal covariances

| | |
|---|---|
| `modulo` | Modulo |
| `mode` | 0=dimensions, 1=sorted individually, 2=sorted by average cov |

`update`
: update one codebook

**Subobjects:**

| | |
|---|---|
| `accu` | (CodebookAccu) |
| `cfg` | (Cbcfg) |
| `cov(0..3)` | (???) |
| `map` | (CodebookMap) |
| `mat` | (FMatrix) |
| `ref(0..3)` | (???) |

### 8.5.6 CodebookAccu

This section describes the '*CodebookAccu*': *a single codebook's accumulator*

**Creation:** `CodebookAccu` cannot be created directly.
It is accessible as a sub-object of `Codebook`!

**Configuration:** `codebookaccu configure`

| | |
|---|---|
| `-count` | = 0.0000e+00 |
| `-distortion` | = 0.000000 |
| `-maxDistance` | = 0.000000 |
| `-minDistance` | = 0.000000 |
| `-score` | = 0.000000 |
| `-subN` | = 1 |

**Methods:** `codebookaccu`

`*=` <factor>
: multiplies an accumulator with a factor

| | |
|---|---|
| `factor` | multiplication factor |

`+=` <source> [-factor factor] [-refX refx]
: adds one accumulator to another

| | |
|---|---|
| `source` | source accumulator (CodebookAccu) |
| `factor` | scaling factor |
| `refx` | add accus to reference refX |

`:=` <source>
: copies one accumulator into another

| | |
|---|---|
| `source` | source accumulator (CodebookAccu) |

`clear` [-subX subx]
: reset a single codebook's accumulator to zero

| | |
|---|---|
| `subx` | sub-accumulator, -1 to clear all |

`set` <matrix> [-subX subx] [-refX refx] [-dimX dimx]
: set reference vectors in the accumulator

| | |
|---|---|
| `matrix` | matrix of reference vectors |
| `subx` | index of the subaccu |
| `refx` | index of the reference vector |
| `dimx` | index of the dimension |

`subspace`

define the accumulator subspacing

**Subobjects:**

| | |
|---|---|
| `cov(0..0,0..3)` | () |
| `mat(0..0)` | (???) |
| `priorCV` | (???) |
| `priorRV` | (???) |

### 8.5.7 CodebookMap

This section describes the '*CodebookMap*': *CodebookMap*

**Creation:** `CodebookMap` cannot be created directly.

It is accessible as a sub-object of `Codebook`!

**Configuration:** `codebookmap configure`

`-itemN` = 1

**Methods:** `codebookmap`

`add <n> [-from from] [-to to]`

add items to the map

| | |
|---|---|
| `n` | number of pathItems to add |
| `from` | map from index |
| `to` | map to index |

`clear`

remove all items from the map

**Elements:** are of type `CodebookMapItem`.

### 8.5.8 CodebookMapItem

This section describes the '*CodebookMapItem*': *CodebookMapItem*

**Creation:** `CodebookMapItem` cannot be created directly.

It is accessible as a sub-object of `CodebookMap`!

**Configuration:** `codebookmapitem configure`

| | |
|---|---|
| `-alpha` | = 1.000000 |
| `-alpha0` | = 0.000000 |
| `-beta` | = 1.000000 |
| `-beta0` | = 0.000000 |
| `-from` | = -1 |
| `-subX` | = -1 |
| `-to` | = -1 |

### 8.5.9 CodebookSet

This section describes the '*CodebookSet*': *Set of codebooks*

**Creation:** `CodebookSet <name> <featureset> [-bmem bmem]`

| | |
|---|---|
| `name` | name of the codebook set |
| `featureset` | name of the feature set (`FeatureSet`) |
| `bmem` | bmem option |

**Configuration:** `codebookset configure`

| | |
|---|---|
| `-blkSize` | = 1000 |
| `-commentChar` | = ; |
| `-defaultBbiOn` | = 1 |
| `-defaultExpT` | = 0 |
| `-defaultRdimN` | = 0 |
| `-defaultTopN` | = 0 |
| `-featureSet` | = featureSetISLci |
| `-itemN` | = 1 |
| `-name` | = codebookSetISLci |
| `-offset` | = 0.000000 |
| `-rewriteSet` | = (null) |
| `-scaleCV` | = 1.000000 |
| `-scaleRV` | = 1.000000 |
| `-subX` | = -1 |
| `-swc-hits` | = 0 |
| `-swc-queries` | = 0 |
| `-swc-width` | = 8 |
| `-useN` | = 3 |

**Methods:** `codebookset`

`add <name> <feat> <refN> <dimN> <type>`

add a new codebook to the set

| | |
|---|---|
| `name` | name of the codebook |
| `feat` | name of the feature space |
| `refN` | number of reference vectors |
| `dimN` | dimension of feature space |
| `type` | type of covariance matrix NO,RADIAL,DIAGONAL,FULL |

`addBBI <codebook> <bbiTree>`

add new (or link to existing) BBI tree

| | |
|---|---|
| `codebook` | name of codebook |
| `bbiTree` | name of BBI tree |

`clearAccus [-subX subx]`

clear accumulators for all codebooks

| | |
|---|---|
| `subx` | sub-accumulator, -1 to clear all |

`compress [-underflowRV underflowrv] [-overflowRV overflowrv] [-overflowCV overflowcv] [-classRV classrv] [-classCV classcv] [-compressCV compresscv] [-resortFeat resortfeat] [-deallocCB deallocb] [-trainMode trainmode]`

compress means/covars to 8bit values

| | |
|---|---|
| `underflowrv` | underflow threshold |
| `overflowrv` | overflow threshold |
| `overflowcv` | overflow threshold |
| `classrv` | number of quantization classes (max 255) |
| `classcv` | number of quantization classes (max 255) |
| `compresscv` | covariance compression mode 1,2 |
| `resortfeat` | resort feature dimensions |
| `dealloccb` | deallocate orginal codebooks |
| `trainmode` | store compressed values in orginal codebooks |

`createAccus` [`-subN subn`]

creates accumulators for all codebooks

| | |
|---|---|
| `subn` | number of subaccumulators |

`createMaps` <`n`>

creates maps for all codebooks

| | |
|---|---|
| `n` | size of maps |

`delete` <`item`>

remove codebook from the set

| | |
|---|---|
| `item` | name of item in list |

`freeAccus`

removes accumulators of all codebooks

`freeBBI`

free all BBI trees

`freeMaps`

removes maps of all codebooks

`index` <`names*`>

returns indices of named codebooks

| | |
|---|---|
| `names*` | list of names |

`load` <`filename`>

load codebook weights

| | |
|---|---|
| `filename` | file to read from |

`loadAccus` <`filename`> [`-factor factor`]

loads codebook accumulators from a file

| | |
|---|---|
| `filename` | file to read from |
| `factor` | multiplicator before adding |

`loadBBI` <`filename`>

load BBI tree parameters

| | |
|---|---|
| `filename` | name of param file |

`makeBBI` [`-depth depth`] [`-gamma gamma`] [`-verbose verbose`]

make new BBI trees

| | |
|---|---|
| `depth` | depth of trees |
| `gamma` | Gaussian box threshold |
| `verbose` | verbose level |

`map`

map all codebooks to new codebooks

`name` <`idx*`>

returns names of indexed codebooks

| | |
|---|---|
| `idx*` | list of indices |

`pruneBBI <levelN>`

prune BBI trees

| | |
|---|---|
| `levelN` | number of levels |

`read <filename>`

read codebook definitions from file

| | |
|---|---|
| `filename` | file to read from |

`readBBI <filename>`

read BBI description file

| | |
|---|---|
| `filename` | name of desc file |

`save <filename> [-mode mode]`

save codebook weights

| | |
|---|---|
| `filename` | file to write to |
| `mode` | compression mode (-1,1) |

`saveAccus <filename>`

saves codebook accumulators into a file

| | |
|---|---|
| `filename` | file to write |

`saveBBI <filename>`

save BBI tree parameters

| | |
|---|---|
| `filename` | name of param file |

`set [-topN topn] [-rdimN rdimn] [-bbiOn bbion] [-expT expt]`

set and propagate defaultTopN or defaultRdimN

| | |
|---|---|
| `topn` | set topN scoring |
| `rdimn` | reduce dimensionality |
| `bbion` | enable/disable BBI scoring |
| `expt` | threshold for evaluating exp() |

`split [-beam beam] [-max max]`

split all codebooks

| | |
|---|---|
| `beam` | splitting beam |
| `max` | max. number of splits |

`update`

update all codebooks

`write <filename>`

write codebook definitions to file

| | |
|---|---|
| `filename` | file to write to |

**Subobjects:**

| | |
|---|---|
| `featureSet` | (`FeatureSet`) |
| `list` | (`List`) |
| `swCache` | (`FMatrix`) |
| `swIndex` | (`IMatrix`) |

**Elements:** are of type `Codebook`.

### 8.5.10 `DictWord`

This section describes the '*DictWord*': *Word with tagged phone transcription*

**Creation:** `DictWord` cannot be created directly.

It is accessible as a sub-object of `Dictionary`!

**Configuration:** `dictword configure`

| | |
|---|---|
| `-itemN` | = 1 |
| `-name` | = $ |
| `-variant` | = -1 |

### 8.5.11 Dictionary

This section describes the '*Dictionary*': *Set of words*

**Creation:** `Dictionary <name> <Phones> <Tags>`

| | |
|---|---|
| `name` | name of the dictionary |
| `Phones` | phones (`Phones`) |
| `Tags` | tags (`Tags`) |

**Configuration:** `dictionary configure`

| | |
|---|---|
| `-blkSize` | = 5000 |
| `-commentChar` | = ; |
| `-itemN` | = 3 |
| `-phones` | = PHONES |
| `-tags` | = tagsISLci |
| `-useN` | = 4 |
| `-wbTags` | = WB |
| `-weTags` | = WB |
| `-xwTags` | = |

**Methods:** `dictionary`

`add <name> <pronunciation>`

add a new word to the set

| | |
|---|---|
| `name` | name (spelling) of the word |
| `pronunciation` | pronunciation of the word |

`delete <item>`

remove word from the set

| | |
|---|---|
| `item` | name of item in list |

`index <names*>`

return the internal index of a word

| | |
|---|---|
| `names*` | list of names |

`load <filename>`

load a dictionary from a binary file

| | |
|---|---|
| `filename` | file to read from |

`name <idx*>`

return the spelled word given the index

| | |
|---|---|
| `idx*` | list of indices |

`puts`

display the contents of a dictionary

`read <filename>`

reads a dictionary file

| | |
|---|---|
| `filename` | file to read from |

`save <filename>`

save a dictionary file into a binary file

| | |
|---|---|
| `filename` | file to write into |

`write <filename> [-format format]`

writes a dictionary file

| | |
|---|---|
| `filename` | file to write to |
| `format` | file format (janus or htk) |

**Subobjects:**

| | |
|---|---|
| `list` | (`List`) |
| `phones` | (`Phones`) |
| `tags` | (`Phones`) |

**Elements:** are of type `DictWord`.

### 8.5.12 Distrib

This section describes the '*Distrib*': *A single distribution*

**Creation:** `Distrib` cannot be created directly.
It is accessible as a sub-object of `DistribSet`!

**Configuration:** `distrib configure`

| | |
|---|---|
| `-cbX` | = 0 |
| `-cfg` | = default |
| `-count` | = 0.000000 |
| `-name` | = SIL-b |
| `-val` | = 2.5000e-01 2.5000e-01 2.5000e-01 2.5000e-01 |
| `-valN` | = 4 |

**Methods:** `distrib`

`:= <source>`

copies distribution weights

| | |
|---|---|
| `source` | name of the source distribution (`Distrib`) |

`createAccu [-subN subn]`

create a single distribution's accumulator

| | |
|---|---|
| `subn` | number of subaccumulators |

`freeAccu`

remove a single distribution's accumulator

**Subobjects:**

| | |
|---|---|
| `cfg` | (`Dscfg`) |

### 8.5.13 DistribAccu

This section describes the '*DistribAccu*': *a single distribution's accumulator*

**Creation:** `DistribAccu` cannot be created directly.
It is accessible as a sub-object of `Distrib`!

**Configuration:** `distribaccu configure`

| | |
|---|---|
| `-count` | = 4.0000e-06 |
| `-subN` | = 1 |

**Methods:** `distribaccu`

`*=` `<factor>`

multiplies an accumulator with a factor

| | |
|---|---|
| `factor` | multiplication factor |

`+=` `<source>` `[-factor factor]` `[-valX valx]`

adds one accumulator to another

| | |
|---|---|
| `source` | source accumulator (`DistribAccu`) |
| `factor` | scaling factor |
| `valx` | add accus to valX component |

`:=` `<source>`

copies one accumulator into another

| | |
|---|---|
| `source` | source accumulator (`DistribAccu`) |

`>=` `<shift>`

increase an accumulator's counts by a number

| | |
|---|---|
| `shift` | value by which to increment every count |

`clear` `[-subX subx]`

reset a single distribution's accumulator to zero

| | |
|---|---|
| `subx` | sub-accumulator, -1 to clear all |

### 8.5.14 DistribSet

This section describes the '*DistribSet*': *Set of distributions*

**Creation:** `DistribSet <name> <CodebookSet> [-bmem bmem]`

| | |
|---|---|
| `name` | name of the distrib set |
| `CodebookSet` | set of codebooks (`CodebookSet`) |
| `bmem` | use block memory management |

**Configuration:** `distribset configure`

| | |
|---|---|
| `-blkSize` | = 5000 |
| `-codebookSet` | = codebookSetISLci |
| `-distance` | = e |
| `-dummyName` | = dummyDs |
| `-dummyStart` | = -1 |
| `-itemN` | = 3 |
| `-minCount` | = 0.000000 |
| `-name` | = distribSetISLci |
| `-normDistance` | = 0 |
| `-rewriteSet` | = (null) |
| `-stateTable` | = (null) |
| `-subX` | = -1 |
| `-useN` | = 2 |

**Methods:** `distribset`

`accuFrame` `<distrib>` `<frame>` [`-factor factor`] [`-toframe toframe`]

accumulates sufficient statistic from frame

`distrib` name of the distribution
`frame` index of the requested frame
`factor` training factor
`toframe` first frame not to train

`accuPath` `<path>` [`-factor factor`]

accumulates sufficient statistic from path

`path` name of the path object (`Path`)
`factor` training factor

`add` `<name>` `<codebook>`

add a new distribution to the set

`name` name of the distribution
`codebook` name of the codebook

`clearAccus` [`-subX subx`]

clears accumulators for all distributions

`subx` sub-accumulator, -1 to clear all

`createAccus` [`-subN subn`]

creates accumulators for all distributions

`subn` number of subaccumulators

`createLh` `<lh>`

fill the lh fields of the accumulators

`lh` source likelihood accumulator (`Lh`)

`delete` `<item>`

remove distribution from the set

`item` name of item in list

`dist` `<ModelArray P>` `<ModelArray Q>` `<ModelArray R>`

measure distance between distributions

`ModelArray P` model array (`ModelArray`)
`ModelArray Q` model array (`ModelArray`)
`ModelArray R` model array (`ModelArray`)

`freeAccus`

frees accumulators for all distributions

`index` `<names*>`

returns indices of named distributions

`names*` list of names

`kldist` `<distribution 1>` `<distribution 2>`

computes the symmetrized Kullback-Leibler distance of two distribs

`distribution 1` name of first distribution (`Distrib`)
`distribution 2` name of second distribution (`Distrib`)

`load` `<filename>`

loads distribution weights from a file

`filename` file to read from

`loadAccus` `<filename>` [`-factor factor`]

loads distribution accumulators from a file

`filename` file to read from
`factor` multiplicator before adding

`map`

map all distributions

`merge`

merge distributions and codebooks

`multiVar <Distrib> [-samples samples] [-first first] [-last last] [-mode mode] [-verbosity verbosity]`

perform Multivar algorithm on codebook/ distribution given samples

| | |
|---|---|
| `Distrib` | Distribution |
| `samples` | Number of random samples drawn (-1 for linear mode) |
| `first` | First sample |
| `last` | Last sample |
| `mode` | Univar, Multivar, TiedRho, TiedSelf mode? |
| `verbosity` | Verbosity of output |

`multiVarInit <Distrib> [-seed seed] [-rhoRel rhorel]`

initialize Multivar algorithm

| | |
|---|---|
| `Distrib` | Distribution |
| `seed` | seed for random number generator |
| `rhorel` | initialize univariate distribution |

`name <idx*>`

returns names of indexed distributions

| | |
|---|---|
| `idx*` | list of indices |

`read <filename>`

reads a distribution description file

| | |
|---|---|
| `filename` | file to read from |

`save <filename>`

saves distribution weights into a file

| | |
|---|---|
| `filename` | file to read from |

`saveAccus <filename>`

saves distribution accumulators into a file

| | |
|---|---|
| `filename` | file to write |

`score <distrib> <frame>`

computes the score of a mixture distribution

| | |
|---|---|
| `distrib` | name of the distribution |
| `frame` | index of the requested frame |

`scoreFeature <distrib> <matrix> [-row row] [-from from] [-to to] [-offset offset] [-factor factor] [-anti anti]`

store contribution of a feature in a matrix

| | |
|---|---|
| `distrib` | distribution name |
| `matrix` | FMatrix |
| `row` | row |
| `from` | start frame |
| `to` | end frame |
| `offset` | offset |
| `factor` | factor for this contribution |
| `anti` | name of the anti-distribution |

`scoreMatrix <distrib> <matrix> [-from from] [-to to]`

store contribution of distrib in a matrix

| | |
|---|---|
| `distrib` | name of the distribution |
| `matrix` | FMatrix |
| `from` | start frame |
| `to` | end frame (needed!) |

`scoreNBest <n> <frame>`

computes the n-best mixtures mixtures

| | |
|---|---|
| `n` | length of list |
| `frame` | index of the requested frame |

`split`

split distributions and codebooks

`update`

update distributions and codebooks

`write <filename>`

writes a distribution description file

| | |
|---|---|
| `filename` | file to read from |

**Subobjects:**

| | |
|---|---|
| `codebookSet` | (`CodebookSet`) |
| `distrib(0..2)` | () |
| `list` | (`List`) |
| `rewriteSet` | (`???`) |

**Elements:** are of type `Distrib`.

### 8.5.15 DistribStream

This section describes the '*DistribStream*': *Distribution based stream*

**Creation:** `DistribStream <name> <DistribSet> <Tree>`

| | |
|---|---|
| `name` | name of the distrib stream |
| `DistribSet` | set of distributions (`DistribSet`) |
| `Tree` | distribution tree (`Tree`) |

**Configuration:** `distribstream configure`

| | |
|---|---|
| `-distribSet` | = distribSetISLci |
| `-name` | = distribStreamISLci |
| `-tree` | = distribTreeISLci |
| `-useN` | = 1 |

**Methods:** `distribstream`

`accu <distrib> <frame> [-factor factor] [-toframe toframe]`

accumulate sufficient statistic

| | |
|---|---|
| `distrib` | name of the distribution |
| `frame` | index of the requested frame |
| `factor` | training factor |
| `toframe` | first frame not to train |

`get <node> <tagged phones> <leftContext> <rightContext> [-node node]`

returns a distribution given a tagged phone sequence

| | |
|---|---|
| `node` | root node |
| `tagged phones` | list of tagged phones |
| `leftContext` | left context |
| `rightContext` | right context |
| `node` | want node name (0/1) |

`index <names*>`

returns indices of named distributions

| | |
|---|---|
| `names*` | list of names |

`name <idx*>`

returns names of indexed distributions

| | |
|---|---|
| `idx*` | list of indices |

`score <distrib> <frame>`

compute distribution score

| | |
|---|---|
| `distrib` | name of the distribution |
| `frame` | index of the requested frame |

`update`

update distributions/codebook

**Subobjects:**

| | |
|---|---|
| `distribSet` | (`DistribSet`) |
| `list` | (`List`) |
| `tree` | (`Tree`) |

**Elements:** are of type `Distrib`.

### 8.5.16 Dscfg

This section describes the '*Dscfg*': *configuration of a distribution*

**Creation:** `Dscfg <name>`

| | |
|---|---|
| `name` | name of the object |

**Configuration:** `dscfg configure`

| | |
|---|---|
| `-accu` | = y |
| `-floor` | = 0.000001 |
| `-method` | = m |
| `-minCount` | = 1.000000 |
| `-momentum` | = 0.000000 |
| `-name` | = dscfg |
| `-shift` | = 0.000000 |
| `-update` | = y |
| `-useN` | = 5 |

### 8.5.17 DurationSet

This section describes the '*DurationSet*': *A 'DurationSet' object is an array of explicite duration models.*

**Creation:** `DurationSet <name> <map>`

| | |
|---|---|
| `name` | name of the object |
| `map` | duration to histogram mapping |

**Configuration:** `durationset configure`

| | |
|---|---|
| -blkSize | = 5000 |
| -commentChar | = ; |
| -floor | = 0.000000 |
| -itemN | = 0 |
| -map | = 1 |
| -minCount | = 5.000000 |
| -momentum | = 0.000000 |
| -useN | = 1 |

**Methods:** durationset

accu <path> <hmm> [-factor factor]

accumulate training data

| | |
|---|---|
| path | name of the path object (Path) |
| hmm | name of the HMM object (HMM) |
| factor | training factor |

add <durModel> <probs> [-count count]

add new duration model(s) to a duration set

| | |
|---|---|
| durModel | name of duration models |
| probs | array of probabilities |
| count | total occurence count |

clearAccus

clear training data accumulators

createAccus

allocate training data accumulators

delete <durModel*>

delete duration model(s) from a duration set

| | |
|---|---|
| durModel* | list of duration models |

dist <ModelArray P> <ModelArray Q> <ModelArray R>

measure distance between duration models

| | |
|---|---|
| ModelArray P | model array (ModelArray) |
| ModelArray Q | model array (ModelArray) |
| ModelArray R | model array (ModelArray) |

freeAccus

allocate training data accumulators

index <names*>

return index of named duration model(s)

| | |
|---|---|
| names* | list of names |

loadAccus <filename>

load training data accumulators from file

| | |
|---|---|
| filename | file to write |

name <idx*>

return the name of indexed duration model(s)

| | |
|---|---|
| idx* | list of indices |

prob <durationModel> <durationFrameN>

return the duration probability for a named duration model

| | |
|---|---|
| durationModel | name of duration model |
| durationFrameN | duration in frames |

`puts`
displays the contents of a duration set

`putsAccu <durationModel*>`
display training data accumulator

| | |
|---|---|
| `durationModel*` | duration models |

`read <filename>`
read a duration set from a file

| | |
|---|---|
| `filename` | name of DurationSet file |

`saveAccus <filename>`
save training data accumulators to file

| | |
|---|---|
| `filename` | file to write |

`scale <factor>`
multiply all log-probs with given value

| | |
|---|---|
| `factor` | multiplicative factor for log-probs |

`update`
update the duration probabilities

`write <filename>`
write a duration set into a file

| | |
|---|---|
| `filename` | name of DurationSet file |

**Subobjects:**

| | |
|---|---|
| `list` | (`List`) |

### 8.5.18 Labelbox

This section describes the '*Labelbox*': *Labelbox*

**Creation:** `Labelbox <name>`

| | |
|---|---|
| `name` | name of the object |

**Methods:** `labelbox`

`add <name>`
add a new path to the Labelbox

| | |
|---|---|
| `name` | name of the Path |

`clear`
clear Labelbox

`delete <item>`
remove a path from the Labelbox

| | |
|---|---|
| `item` | name of item in list |

`load <file>`
load Labelbox

| | |
|---|---|
| `file` | filename |

`puts`
puts Labelbox

`save <file>`
save Labelbox

| | |
|---|---|
| `file` | filename |

**Subobjects:**

| | |
|---|---|
| `list` | (`List`) |

### 8.5.19 Lh

This section describes the '*Lh*': *a codebook-likelihoods accumulator*

**Creation:** `Lh <name>`

| | |
|---|---|
| `name` | name of the object |

**Methods:** `lh`

`clear`
clear likelihoods

`like <probs>`
compute likelihood

| | |
|---|---|
| `probs` | array of mixture weights |

`load <file>`
load likelihoods

| | |
|---|---|
| `file` | feature name |

### 8.5.20 MLAdapt

This section describes the '*MLAdapt*': *Maximum Likelihood Adaptation*

**Creation:** `MLAdapt <name> <CodebookSet> [-mode mode] [-bmem bmem] [-thread thread]`

| | |
|---|---|
| `name` | name of MLAdapt object |
| `CodebookSet` | name of the codebook set (`CodebookSet`) |
| `mode` | 0=mean 1=diagonal 2=full |
| `bmem` | use block memory management for SAT |
| `thread` | use multiple threads for MLLR/SAT |

**Configuration:** `mladapt configure`

| | |
|---|---|
| `-dimN` | = 0 |
| `-featX` | = -1 |
| `-itemN` | = 0 |
| `-name` | = mlAdaptISLci |
| `-useN` | = 0 |

**Methods:** `mladapt`

`accuSAT [-file file]`
SAT accu means

| | |
|---|---|
| `file` | SAT accu file |

`accuTree`
accu MLAdapt information for optimizing tree

`add <Codebook>`
add items to the adaptation

| | |
|---|---|
| `Codebook` | codebook |

`clear`
remove all items from the adaptation list

`clearSAT`
clear SAT accus

`clearTree`

clear MLAdapt tree accus

`cluster` [`-depth depth`] [`-maxIter maxiter`] [`-tempS temps`] [`-tempF tempf`]

cluster items in the list

| | |
|---|---|
| `depth` | maximum depth of tree |
| `maxiter` | number of iterations |
| `temps` | start temperature (0=k-means) |
| `tempf` | temperature decay |

`load` <`filename`>

load MLAdapt tree/accus from file

| | |
|---|---|
| `filename` | file to save accumulators |

`loadSAT` <`filename`>

load SAT accus from file

| | |
|---|---|
| `filename` | file accumulators |

`optTree`

optimize tree based on accus

`restore` [`-covar covar`]

restore means from MLAdapt

| | |
|---|---|
| `covar` | restore (diagonal) covariances |

`restoreAccu` [`-covar covar`]

restore accumulators from MLAdapt

| | |
|---|---|
| `covar` | restore (diagonal) covariances |

`save` <`filename`>

save MLAdapt tree/accus to file

| | |
|---|---|
| `filename` | file to save accumulators |

`saveSAT` <`filename`>

save SAT accus to file

| | |
|---|---|
| `filename` | file to save accumulators |

`store` [`-covar covar`]

save current means to MLAdapt

| | |
|---|---|
| `covar` | store (diagonal) covariances |

`storeAccu` [`-covar covar`]

save accumulators of the gaussians to MLAdapt

| | |
|---|---|
| `covar` | store (diagonal) covariances |

`transform` <`matrixX`>

print transformation matrix

| | |
|---|---|
| `matrixX` | print which transformation matrix |

`update` [`-minCount mincount`]

update codebook means

| | |
|---|---|
| `mincount` | minimal splitting count |

`updateSAT` [`-file file`] [`-updateMean updatemean`] [`-updateCV updatecv`]

SAT update codebook means

| | |
|---|---|
| `file` | SAT accu file |
| `updatemean` | update means |
| `updatecv` | update covariances |

`variance [-minCount mincount] [-minAdapt minadapt]`

update codebook variances

| | |
|---|---|
| `mincount` | minimal splitting count |
| `minadapt` | minimal total count for adaptation |

### 8.5.21 ModelArray

This section describes the '*ModelArray*': *Array of models.*

**Creation:** `ModelArray <name> <ModelSet>`

| | |
|---|---|
| `name` | name of the object |
| `ModelSet` | model set |

**Configuration:** `modelarray configure`

| | |
|---|---|
| `-itemN` | = 0 |

**Methods:** `modelarray`

`add <model> <count>`

add another model to the array

| | |
|---|---|
| `model` | name of the model |
| `count` | count |

`clear`

remove all entries from the array

`puts`

print model array

### 8.5.22 Modality

This section describes the '*Modality*': *A 'Modality' object answers a question about the modality of a recording.*

**Creation:** `Modality <name> <updateProc> <tagName> [-mode mode] [-limit limit]`

| | |
|---|---|
| `name` | name of the modality |
| `updateProc` | TCL modality update proc |
| `tagName` | Name for tag |
| `mode` | update mode (ALL, GIVEN, RUNON, NOT) |
| `limit` | update only if intervall greater limit |

**Configuration:** `modality configure`

| | |
|---|---|
| `-endFrameX` | = -1 |
| `-name` | = modality |
| `-startFrameX` | = -1 |
| `-tagName` | = WB |
| `-timeInfo` | = 0 |
| `-updateLimit` | = -1 |
| `-updateMode` | = GIVEN |
| `-updateProc` | = putsInfo |
| `-useN` | = 1 |
| `-yesN` | = 0 |

**Methods:** `modality`

`answer <startFrameX> <endFrameX>`

get anser for modality

| | |
|---|---|
| `startFrameX` | start frame for answer |
| `endFrameX` | end frame for answer |

`majorityAnswer [-startFrameX startframex] [-endFrameX endframex]`

get the majority of the answers

| | |
|---|---|
| `startframex` | start frame for answer |
| `endframex` | end frame for answer |

`puts`

display the contents of the modality

`reset`

reset modality

`update <startFrameX> <endFrameX>`

update modality

| | |
|---|---|
| `startFrameX` | start frame for update |
| `endFrameX` | end frame for update |

### 8.5.23 ModalitySet

This section describes the '*ModalitySet*': *A 'ModalitySet' object is a set of modalities.*

**Creation:** `ModalitySet <name> <tags> [-addTags addtags]`

| | |
|---|---|
| `name` | name of the modality set |
| `tags` | tags object (`Tags`) |
| `addtags` | add tag names to tags-object |

**Configuration:** `modalityset configure`

| | |
|---|---|
| `-addTags` | = 0 |
| `-dummyStart` | = -1 |
| `-endFrameX` | = -1 |
| `-itemN` | = 0 |
| `-name` | = modalitySetISLci |
| `-startFrameX` | = -1 |
| `-tags` | = tagsISLci |
| `-tree` | = (null) |

**Methods:** `modalityset`

`add <name> <updateProc> <tagName>`

add a new modality to the set

| | |
|---|---|
| `name` | name of modality |
| `updateProc` | TCL modality update proc |
| `tagName` | Name for tag |

`addTags`

add tags to tags-object

`answer <startFrameX> <endFrameX>`

get answer of all modalities in the set

| | |
|---|---|
| `startFrameX` | start frame for answer |
| `endFrameX` | end frame for answer |

`answer2codedTags` <answer> [-tags tags]

coded tags for answer

| | |
|---|---|
| answer | answer (majority) |
| tags | tags-object (Tags) |

`answer2tags` <answer>

get a list of tags for an answer

| | |
|---|---|
| answer | answer for modalities (binary coded) |

`delete`

delete a modality from the set

`deleteTags`

delete tags from tags-object

`getRootNodes`

get root nodes of tree

`majorityAnswer` [-startFrameX startframex] [-endFrameX endframex]

get the majority of the answers

| | |
|---|---|
| startframex | start frame for answer |
| endframex | end frame for answer |

`puts`

display the contents of the modality-set

`reset`

reset set

`trace` <rootX> <answer>

trace given subtree with given answers

| | |
|---|---|
| rootX | root node index of subtree |
| answer | answers for modalities (coded as int) |

`update`

update all modalities in the set

`updateUtterance`

update modality for the whole utterance (modalityUpdateUtterance)

**Subobjects:**

| | |
|---|---|
| list | (List) |
| localTags | (Tags) |
| tags | (Tags) |
| tree | (???) |

### 8.5.24 Phone

This section describes the '*Phone*': *Phone*

**Creation:** `Phone` cannot be created directly.

It is accessible as a sub-object of Phones!

### 8.5.25 Phones

This section describes the '*Phones*': *A 'Phones' object is an array of strings, each of which is a phoneme.*

**Creation:** `Phones <name>`

| | |
|---|---|
| `name` | name of the object |

**Configuration:** `phones configure`

| | |
|---|---|
| `-blkSize` | = 10 |
| `-commentChar` | = ; |
| `-itemN` | = 0 |
| `-useN` | = 1 |

**Methods:** `phones`

`add <phone*>`

add new phone(s) to a phone-set

| | |
|---|---|
| `phone*` | list of phones |

`delete <phone*>`

delete phone(s) from a phone-set

| | |
|---|---|
| `phone*` | list of phones |

`index <names*>`

return index of named phone(s)

| | |
|---|---|
| `names*` | list of names |

`name <idx*>`

return the name of indexed phone(s)

| | |
|---|---|
| `idx*` | list of indices |

`puts`

displays the contents of a phone-set

`read <filename>`

read a phone-set from a file

| | |
|---|---|
| `filename` | name of phones file |

`write <filename>`

write a phone-set into a file

| | |
|---|---|
| `filename` | name of phones file |

**Subobjects:**

| | |
|---|---|
| `list` | (`List`) |

### 8.5.26 PhonesSet

This section describes the '*PhonesSet*': *A 'PhonesSet' object is a set of 'Phones' objects.*

**Creation:** `PhonesSet <name>`

| | |
|---|---|
| `name` | name of the object |

**Configuration:** `phonesset configure`

| | |
|---|---|
| `-blkSize` | = 20 |
| `-commentChar` | = ; |
| `-itemN` | = 1 |
| `-useN` | = 5 |

**Methods:** `phonesset`

`add <name> <phone*>`

add new phone-set to a set of phones-set

| | |
|---|---|
| `name` | name of list |
| `phone*` | list of phones |

`delete <phoneSet*>`

delete phone-set(s) from a set of phone-sets

| | |
|---|---|
| `phoneSet*` | list of phone sets |

`index <names*>`

return index of named phone-set(s)

| | |
|---|---|
| `names*` | list of names |

`name`

return the name of indexed phone-set(s)

`puts`

displays the contents of a set of phone-sets

`read <filename>`

read a set of phone-sets from a file

| | |
|---|---|
| `filename` | name of phone set file |

`write <filename>`

write a set of phone-sets into a file

| | |
|---|---|
| `filename` | name of phones file |

**Subobjects:**

`list` (`List`)

**Elements:** are of type `Phones`.

### 8.5.27 PTree

This section describes the '*PTree*': *Polyphonic Tree*

**Creation:** `PTree <name> <phones> <tags> <modelSet> [-addProc addproc]`

| | |
|---|---|
| `name` | name of the object |
| `phones` | set of phones (`Phones`) |
| `tags` | set of tags (`Tags`) |
| `modelSet` | set of models |
| `addproc` | TCL add model proc |

**Configuration:** `ptree configure`

| | |
|---|---|
| `-addProc` | = (null) |
| `-count` | = 0.000000 |
| `-maxContext` | = -1 |
| `-name` | = ptree |

**Methods:** `ptree`

`add` <`tagged phones`> <`leftContext`> <`rightContext`> [`-count count`] [`-model model`]

adds another polyphone to the tree

| | |
|---|---|
| `tagged phones` | list of tagged phones |
| `leftContext` | left context |
| `rightContext` | right context |
| `count` | count |
| `model` | model |

`get` <`tagged phones`> <`leftContext`> <`rightContext`>

find polyphone in the tree

| | |
|---|---|
| `tagged phones` | list of tagged phones |
| `leftContext` | left context |
| `rightContext` | right context |

`models` <`modelArray`> [`-minCount mincount`]

returns a model array of models in the tree

| | |
|---|---|
| `modelArray` | model array (`ModelArray`) |
| `mincount` | minimum count |

`question` <`questionSet`> [`-minCount mincount`]

find a question for splitting

| | |
|---|---|
| `questionSet` | question set (`QuestionSet`) |
| `mincount` | minimum count |

`split` <`questionSet`> <`question`> [`-minCount mincount`]

split a tree by asking a question

| | |
|---|---|
| `questionSet` | question set (`QuestionSet`) |
| `question` | question |
| `mincount` | minimum count |

**Subobjects:**

| | |
|---|---|
| `modelSet` | (`DistribSet`) |

### 8.5.28 PTreeSet

This section describes the '*PTreeSet*': *A 'PTreeSet' object is a set of polyphone context trees.*

**Creation:** `PTreeSet` <`name`> <`phones`> <`tags`> <`modelSet`>

| | |
|---|---|
| `name` | name of the object |
| `phones` | set of phones (`Phones`) |
| `tags` | set of tags (`Tags`) |
| `modelSet` | set of models |

**Configuration:** `ptreeset configure`

| | |
|---|---|
| `-blkSize` | = 100 |
| `-commentChar` | = ; |
| `-itemN` | = 0 |
| `-name` | = ptreeSet |
| `-useN` | = 1 |

**Methods:** `ptreeset`

`add <name> <polyphone>`

adds another polyphonic tree

| | |
|---|---|
| `name` | name of polyphonic tree |
| `polyphone` | polyphone description |

`index <names*>`

find index of a polyphone tree

| | |
|---|---|
| `names*` | list of names |

`name <idx*>`

find name of a polyphone tree

| | |
|---|---|
| `idx*` | list of indices |

`puts`

displays the contents of a PTreeSet object

`read <filename>`

reads polyphone tree from a file

| | |
|---|---|
| `filename` | name of PTreeSet file |

`write <filename> [-minCount mincount]`

writes polyphone tree to a file

| | |
|---|---|
| `filename` | name of tree file |
| `mincount` | minimum count |

**Subobjects:**

| | |
|---|---|
| `list` | (`List`) |
| `modelSet` | (`DistribSet`) |

### 8.5.29 QuestionSet

This section describes the '*QuestionSet*': *A 'QuestionSet' object is a set of characteristic function definitions and a set of questionSet.*

**Creation:** `QuestionSet <name> <phones> <phonesSet> <tags> [-padPhone padphone]`

| | |
|---|---|
| `name` | name of the question set |
| `phones` | set of phones (`Phones`) |
| `phonesSet` | set of phone set (`PhonesSet`) |
| `tags` | set of tags (`Tags`) |
| `padphone` | padding phone index |

**Configuration:** `questionset configure`

| | |
|---|---|
| `-blkSize` | = 50 |
| `-commentChar` | = ; |
| `-itemN` | = 0 |
| `-padPhone` | = -1 |
| `-phones` | = PHONES |
| `-phonesSet` | = phonesSetISLci |
| `-tagOperation` | = 1 |
| `-tags` | = tagsISLci |
| `-useN` | = 1 |

**Methods:** `questionset`

`add <question>`

add a new question to a questionSet object

| | |
|---|---|
| `question` | question string |

`delete <item>`

remove a question from a questionSet object

| | |
|---|---|
| `item` | name of item in list |

`index <names*>`

return the index of a named question

| | |
|---|---|
| `names*` | list of names |

`name <idx*>`

return the name of an indexed question

| | |
|---|---|
| `idx*` | list of indices |

`puts`

displays the contents of a questionSet object

`read <filename>`

read questionSet from a file

| | |
|---|---|
| `filename` | name of question set file |

`write <filename>`

write questionSet into a file

| | |
|---|---|
| `filename` | name of questionSet file |

**Subobjects:**

| | |
|---|---|
| `list` | (`List`) |
| `phones` | (`Phones`) |
| `phonesSet` | (`PhonesSet`) |
| `tags` | (`Tags`) |

## 8.5.30 RewriteSet

This section describes the '*RewriteSet*': *Set of rewrite rules*

**Creation:** `RewriteSet <name>`

| | |
|---|---|
| `name` | name of the object |

**Configuration:** `rewriteset configure`

| | |
|---|---|
| `-blkSize` | = 100 |
| `-itemN` | = 0 |
| `-useN` | = 1 |

**Methods:** `rewriteset`

`add <from> <to>`

add a new rewrite rule to the set

| | |
|---|---|
| `from` | left side of the rewrite rule |
| `to` | right side of the rewrite rule |

`delete <item>`

remove rewrite rule from the set

| | |
|---|---|
| `item` | name of item in list |

`read <filename>`

reads a rewrite rules file

| | |
|---|---|
| `filename` | file to read from |

`write <filename>`

writes a rewrite rules file

| | |
|---|---|
| `filename` | file to write to |

**Subobjects:**

| | |
|---|---|
| `list` | (`List`) |

### 8.5.31 SampleSet

This section describes the '*SampleSet*': *containers for samples*

**Creation:** `SampleSet <name> <featureSet> <feature> <dimN>`

| | |
|---|---|
| `name` | name of the SampleSet object |
| `featureSet` | name of the feature set (`FeatureSet`) |
| `feature` | feature name |
| `dimN` | input dimension |

**Configuration:** `sampleset configure`

| | |
|---|---|
| `-blkSize` | = 100 |
| `-dimN` | = 4 |
| `-featX` | = 0 |
| `-featureSet` | = featureSetISLci |
| `-indexN` | = 0 |
| `-itemN` | = 0 |
| `-name` | = sampleSetISLci |
| `-useN` | = 1 |

**Methods:** `sampleset`

`accu <path> [-factor factor] [-lh lh] [-from from] [-to to]`

accumulate samples from a path object

| | |
|---|---|
| `path` | name of the path object (`Path`) |
| `factor` | training factor |
| `lh` | distribSet for lh accumulation (`DistribSet`) |
| `from` | start frame |
| `to` | end frame |

`add <name> [-filename filename] [-featX featx] [-dimN dimn] [-size size] [-mod mod] [-lhdss lhdss]`

add a new SampleSet class to the set

| | |
|---|---|
| `name` | name of the class |
| `filename` | name of the dump file |
| `featx` | index of the feature to use |
| `dimn` | this feature's number of dimensions |
| `size` | use buffer of the given size |
| `mod` | use only every -mod-th vector |
| `lhdss` | distrib set for likelihood accumulation (`DistribSet`) |

`clear`

clear accumulation buffers

`delete <item>`

remove SampleSet class from the set

| | |
|---|---|
| `item` | name of item in list |

`flush`

flush accumulation buffers to file

`index <names*>`

returns indices of named SampleSet classes

| | |
|---|---|
| `names*` | list of names |

`map <index> [-class class]`

add/get index to class mapping information

| | |
|---|---|
| `index` | index to map |
| `class` | name of the class |

`name <idx*>`

returns names of indexed SampleSet classes

| | |
|---|---|
| `idx*` | list of indices |

`showmap`

display class mapping information

**Subobjects:**

| | |
|---|---|
| `featureSet` | (`FeatureSet`) |
| `list` | (`List`) |

## 8.5.32 Senone

This section describes the '*Senone*': *Senone*

**Creation:** `Senone` cannot be created directly.

It is accessible as a sub-object of `SenoneSet`!

**Configuration:** `senone configure`

| | |
|---|---|
| `-name` | = SIL-b |
| `-snX` | = -1 |
| `-streamN` | = 1 |

**Methods:** `senone`

`setAccu [-accu accu]`

set/get stream weights accu

| | |
|---|---|
| `accu` | array of stream accu values |

`setWeights [-weight weight]`

set stream weights

| | |
|---|---|
| `weight` | array of stream weights |

## 8.5.33 SenoneSet

This section describes the '*SenoneSet*': *Set of senones*

**Creation:** `SenoneSet <name> <streamArray> [-phones phones] [-tags tags]`

| | |
|---|---|
| `name` | name of the senones set |
| `streamArray` | list of stream [-streamType ST] [-weight W] |
| `phones` | set of phones (`Phones`) |
| `tags` | set of tags (`Tags`) |

**Configuration:** `senoneset configure`

| | |
|---|---|
| `-blkSize` | = 500 |
| `-commentChar` | = ; |
| `-featSetN` | = 0 |
| `-itemN` | = 3 |
| `-likelihood` | = 228895450841486969006328485837565558188250792269271278873354681566845682 |
| `-mixMode` | = 0 |
| `-normalize` | = 0 |
| `-scoreScale` | = 1.000000 |
| `-useN` | = 7 |

**Methods:** `senoneset`

`accu <path> [-factor factor] [-random random] [-from from] [-to to]`

accumulate training data for the given path

| | |
|---|---|
| `path` | name of the path object (`Path`) |
| `factor` | training factor |
| `random` | random frame presentation |
| `from` | start frame |
| `to` | end frame |

`accuWeights <path> [-from from] [-to to] [-accu accu] [-v v] [-zeroMode zeromode]`

accu statistics to train stream weights (MMIE)

| | |
|---|---|
| `path` | path object (`Path`) |
| `from` | first frame |
| `to` | last frame |
| `accu` | for MMI: accu into 'num' (ref) or 'den' (hyp) |
| `v` | verbose information |
| `zeromode` | don't train streams with weight 0 |

`accuWeightsMLE <path> [-zeroMode zeromode] [-v v]`

accumulate MLE statistics to train stream weights

| | |
|---|---|
| `path` | path object (`Path`) |
| `zeromode` | update streams with weight=0 |
| `v` | verbose information |

`add <senone> [-name name]`

add a new senone to the set

| | |
|---|---|
| `senone` | list of score names |
| `name` | name of the senone |

`addNorm <name> <streamX> [-histN histn] [-minmaxN minmaxn]`

add a stream normalizer item

| | |
|---|---|
| `name` | name of stream normalizer |
| `streamX` | stream index |
| `histn` | resolution of histogram |
| `minmaxn` | number of samples for min/max computation |

`clearAccuWeights`

clear update stream weights accu

`clearMix <streamN> <frameN>`

clear dynamic stream mixer

| | |
|---|---|
| `streamN` | number of streams |
| `frameN` | number of frames |

`clearNorm [-name name]`

clear stream normalizer

| | |
|---|---|
| `name` | name of stream normalizer |

`clearStreamCache [-frameN framen]`

clear stream cache (opt_str score fct)

| | |
|---|---|
| `framen` | number of frames to clear |

`get <senone tag> <tagged phones> <leftContext> <rightContext>`

find a senone given phonetic context

| | |
|---|---|
| `senone tag` | tag |
| `tagged phones` | list of tagged phones |
| `leftContext` | left context |
| `rightContext` | right context |

`index`

returns indices of named senones

`labelMix <path> [-soft soft] [-smooth smooth]`

compute mixing weights based on labels

| | |
|---|---|
| `path` | path object (`Path`) |
| `soft` | soft targets? |
| `smooth` | size of smoothing window |

`load <filename>`

load a senone binary file

| | |
|---|---|
| `filename` | file to load from |

`loadAccuWeights <filename>`

load MLE- or MMIE-update stream weights accu

| | |
|---|---|
| `filename` | file to load from |

`loadNorm <filename> [-name name]`

load stream normalizer

| | |
|---|---|
| `filename` | name of file |
| `name` | name of stream normalizer |

`name <index>`

returns names of indexed senones

| | |
|---|---|
| `index` | index to look up |

`read <filename>`

reads a senone description file

| | |
|---|---|
| `filename` | file to read from |

`reset`

reset senoneSet

`save <filename>`

save a senone binary file

| | |
|---|---|
| `filename` | file to save to |

`saveAccuWeights <filename>`

save MLE- or MMIE-update stream weights accu

| | |
|---|---|
| `filename` | file to save to |

`saveNorm` <`filename`> [`-name name`]

save stream normalizer

| | |
|---|---|
| `filename` | name of file |
| `name` | name of stream normalizer |

`score` <`senone`> <`frame`>

compute the score for a senone and a frame

| | |
|---|---|
| `senone` | senone index |
| `frame` | index of the requested frame |

`scoreMatrix` <`matrix`> [`-topN topn`] [`-mode mode`]

compute the scores for senones in a matrix

| | |
|---|---|
| `matrix` | matrix to score results in |
| `topn` | topN value for score function |
| `mode` | cb or ds |

`setScoreFct` <`name`>

set score function (interface to Ibis)

| | |
|---|---|
| `name` | one of (base, opt, opt_thread, opt_semCont, opt_str, compress, old_base, old_opt) |

`setWeights` [`-global global`] [`-local local`] [`-weight weight`]

set stream weights

| | |
|---|---|
| `global` | set weight global |
| `local` | set weights for each senone |
| `weight` | array of stream weights |

`update`

update the underlying acoustic parameters

`updateMix` [`-smooth smooth`]

update dynamic stream mixer

| | |
|---|---|
| `smooth` | size of smoothing window |

`updateWeights` [`-mode mode`] [`-tie tie`] [`-mass mass`] [`-contrast contrast`] [`-minCount mincount`] [`-start start`] [`-clear clear`] [`-iter iter`] [`-min min`] [`-v v`]

update stream weights (MMIE)

| | |
|---|---|
| `mode` | dmc or mmi |
| `tie` | global, phone, state, or senone-based smoothing |
| `mass` | probability mass to assign |
| `contrast` | contrast |
| `mincount` | min count for update |
| `start` | first stream index to touch (0 or 1) |
| `clear` | clear accus after update |
| `iter` | loops |
| `min` | min weight, >=1 no control, <0 auto to -1*val |
| `v` | verbosity |

`updateWeightsMLE` [`-minCnt mincnt`] [`-M m`] [`-K k`] [`-zeroMode zeromode`] [`-noiseMode noisemode`] [`-mode mode`] [`-startIdx startidx`]

MLE-update stream weights

| | |
|---|---|
| `mincnt` | min. count to update |
| `m` | M-norm, M > 1 |
| `k` | normalizer constant |
| `zeromode` | update streams with weight=0 |
| `noisemode` | zero noises before update? |
| `mode` | global, phone, state, or senone-based smoothing |
| `startidx` | start index (0 or 1 is useful) |

`write <filename>`

writes a senone description file

| | |
|---|---|
| `filename` | file to read from |

**Subobjects:**

| | |
|---|---|
| `stream(0..0)` | (???) |
| `tagList` | (List) |

**Elements:** are of type `Senone`.

### 8.5.34 SenoneTag

This section describes the '*SenoneTag*': *SenoneTag*

**Creation:** `SenoneTag` cannot be created directly.

It is accessible as a sub-object of `List`!

### 8.5.35 SignalAdapt

This section describes the '*SignalAdapt*': *Signal Adaption*

**Creation:** `SignalAdapt <name> <SenoneSet> [-stream stream] [-maxAccu maxaccu] [-maxTran maxtran]`

| | |
|---|---|
| `name` | name of SignalAdapt object |
| `SenoneSet` | name of the senone set (`SenoneSet`) |
| `stream` | stream to use |
| `maxaccu` | max number of accus |
| `maxtran` | max number of transformations |

**Configuration:** `signaladapt configure`

| | |
|---|---|
| `-beta(0..5)` | = |
| `-name` | = signalAdaptISLci |
| `-shift` | = 1.000000 |
| `-stream` | = 0 |
| `-topN` | = 1 |
| `-useN` | = 0 |

**Methods:** `signaladapt`

`accu <path> <accuX> [-match match] [-from from] [-to to] [-stream stream] [-gamma gamma] [-conf conf]`

accu path for signal adaption

| | |
|---|---|
| `path` | name of the path object (`Path`) |
| `accuX` | accu to be used |
| `match` | only accu senones that match this string |
| `from` | start frame |
| `to` | end frame (-1 = last frame) |
| `stream` | stream to accumulate |
| `gamma` | scaling factor |
| `conf` | Confidence values (FVector) (`FVector`) |

`adapt <src> <dst> <tranX>`

adapt feature

| | |
|---|---|
| `src` | source feature, FMatrix (`FMatrix`) |
| `dst` | dst feature, FMatrix (`FMatrix`) |
| `tranX` | transformation index |

`add <Distribution>`
add distribution for signal adaption

| | |
|---|---|
| `Distribution` | distribution |

`addAccu <accuX> <accuY> [-factor factor]`
accuX += factor *accuY

| | |
|---|---|
| `accuX` | accuX |
| `accuY` | accuY |
| `factor` | weighting factor |

`clear <tranX>`
clear parameter matrix (will not be done automatically!)

| | |
|---|---|
| `tranX` | transformation index |

`clearAccu <accuX>`
clear accu's

| | |
|---|---|
| `accuX` | accu index |

`combine <tranX1> <tranX2>`
combine two transforms

| | |
|---|---|
| `tranX1` | transformation 1 |
| `tranX2` | transformation 2 |

`compare <tranX> <tranY>`
compare two transforms (sum of squares)

| | |
|---|---|
| `tranX` | transformation index |
| `tranY` | transformation index |

`compute <iter> <accuX> <tranX>`
compute adaption matrix

| | |
|---|---|
| `iter` | Number of iterations |
| `accuX` | accu index |
| `tranX` | transformation index |

`load <filename> <tranX>`
load parameter matrix

| | |
|---|---|
| `filename` | filename |
| `tranX` | transformation index |

`puts`
puts distributons

`readAccu <filename> <accuX> [-factor factor]`
read accu's

| | |
|---|---|
| `filename` | filename |
| `accuX` | accu index |
| `factor` | weighting factor |

`save <filename> <tranX>`
save parameter matrix

| | |
|---|---|
| `filename` | filename |
| `tranX` | transformation index |

`scaleAccu <factor> <accuX>`
scale accu's

| | |
|---|---|
| `factor` | scaling factor for accu's |
| `accuX` | accu index |

`writeAccu <filename> <accuX>`
write accu's

| | |
|---|---|
| `filename` | filename |
| `accuX` | accu index |

**Subobjects:**

| | |
|---|---|
| `g(0..4,0..3)` | () |
| `w(0..0)` | (???) |
| `z(0..4)` | (???) |

### 8.5.36 `StateTable`

This section describes the '*StateTable*': *A 'StateTable' object is a matrix for looking up distribution indices.*

**Creation:** `StateTable <name> <modalitySet> [-compress compress]`

| | |
|---|---|
| `name` | name of the state table |
| `modalitySet` | modality set (`ModalitySet`) |
| `compress` | compress stateTable |

**Configuration:** `statetable configure`

| | |
|---|---|
| `-commentChar` | = 59 |
| `-compress` | = 0 |
| `-dummyStart` | = -1 |
| `-endFrameX` | = -1 |
| `-modXN` | = 0 |
| `-name` | = stateTableISLci |
| `-startFrameX` | = -1 |
| `-timeInfo` | = 0 |
| `-treeXN` | = 0 |
| `-useN` | = 1 |

**Methods:** `statetable`

`copy`

copy state table

`create`

create new matrix

`get <treeX> <modalityX>`

get a single entry of the state table

| | |
|---|---|
| `treeX` | index of subtree |
| `modalityX` | index of modality combination |

`lookup <dsX> <frameX>`

make a table lookup

| | |
|---|---|
| `dsX` | index of distribution |
| `frameX` | index of frame |

`puts`

displays the contents of the state table

`read <fileName>`

read state table from file

| | |
|---|---|
| `fileName` | Name of file |

`reset`

reset state table and modalitySet

`resize`

resize state table

`set <treeX> <modalityX> <dsX>`

set a single entry in the state table

| | |
|---|---|
| `treeX` | index of subtree |
| `modalityX` | index of modality combination |
| `dsX` | index of distribution |

`update <startFrameX> <endFrameX>`

update state-table

| | |
|---|---|
| `startFrameX` | start frame for answer |
| `endFrameX` | end frame for answer |

`updateUnsupervised`

update stateTable and modalities for the whole utterance (unsupervised) (stateTableUpdateUnsupervised)

`updateUtterance`

update stateTable and modalities for the whole utterance (stateTableUpdateUtterance)

`write <fileName>`

write state table to file

| | |
|---|---|
| `fileName` | Name of file |

**Subobjects:**

| | |
|---|---|
| `matrix` | (`IMatrix`) |
| `modalitySet` | (`ModalitySet`) |

### 8.5.37 Tag

This section describes the '*Tag*': *Tag*

**Creation:** `Tag` cannot be created directly.

It is accessible as a sub-object of `Tags`!

**Methods:** `tag`

`puts`

print information about tag

### 8.5.38 Tags

This section describes the '*Tags*': *A 'Tags' object is an array of strings.*

**Creation:** `Tags <name>`

| | |
|---|---|
| `name` | name of the object |

**Configuration:** `tags configure`

| | |
|---|---|
| `-blkSize` | = 10 |
| `-commentChar` | = ; |
| `-itemN` | = 1 |
| `-modMask` | = 1 |
| `-useN` | = 11 |
| `-wordBeginTag` | = WB |
| `-wordEndTag` | = WE |

**Methods:** `tags`

`add` <`tag*`>

add new tag(s) to a tags-set

| | |
|---|---|
| `tag*` | list of tags |

`delete` <`tag*`>

delete tag(s) from a tags-set

| | |
|---|---|
| `tag*` | list of tags |

`index`

return index of named tag(s)

`name`

return the name of indexed tag(s)

`puts`

displays the contents of a tags-set

`read` <`filename`>

read a tag-set from a file

| | |
|---|---|
| `filename` | name of tags file |

`write` <`filename`>

write a tag-set into a file

| | |
|---|---|
| `filename` | name of tags file |

**Subobjects:**

| | |
|---|---|
| `list` | (`List`) |

**Elements:** are of type `Tag`.

### 8.5.39 TmSet

This section describes the '*TmSet*': *A TmSet is a set of state transition model objects (Tm)*

**Creation:** `TmSet` <`name`>

| | |
|---|---|
| `name` | name of the tmset |

**Configuration:** `tmset configure`

| | |
|---|---|
| `-blkSize` | = 20 |
| `-commentChar` | = ; |
| `-itemN` | = 1 |
| `-useN` | = 3 |

**Methods:** `tmset`

`add` <`name`> <`tm`>

add a Tm to the list

| | |
|---|---|
| `name` | name of the transition model |
| `tm` | transition model description |

`index` <`names*`>

return index of named Tm(s)

| | |
|---|---|
| `names*` | list of names |

`name <idx*>`

return the name of indexed Tm(s)

| | |
|---|---|
| `idx*` | list of indices |

`puts`

displays the contents of a transition model

`read <filename>`

reads a TmSet from a file

| | |
|---|---|
| `filename` | name of transition model description file |

`write <filename>`

writes a TmSet to a file

| | |
|---|---|
| `filename` | file to read from |

**Subobjects:**

| | |
|---|---|
| `list` | (`List`) |

**Elements:** are of type `Tm`.

### 8.5.40 Topo

This section describes the '*Topo*': *A 'Topo' object is a definition of a single topology description.*

**Creation:** `Topo` cannot be created directly.

It is accessible as a sub-object of `TopoSet`!

**Configuration:** `topo configure`

| | |
|---|---|
| `-name` | = topo |

**Methods:** `topo`

`puts`

display one single topo

### 8.5.41 TopoSet

This section describes the '*TopoSet*': *A 'TopoSet' object is a set of different topologies.*

**Creation:** `TopoSet <name> <SenoneSet> <TmSet>`

| | |
|---|---|
| `name` | name of the topo set |
| `SenoneSet` | senone set (`SenoneSet`) |
| `TmSet` | set of transition models (`TmSet`) |

**Configuration:** `toposet configure`

| | |
|---|---|
| `-blkSize` | = 20 |
| `-commentChar` | = ; |
| `-itemN` | = 1 |
| `-senoneSet` | = senoneSetISLci |
| `-tmSet` | = tmSetISLci |
| `-useN` | = 1 |

**Methods:** `toposet`

`add <name> <senoneTag*> <tmSet*>`

add a new topo to a TopoSet object

| | |
|---|---|
| `name` | name of topology |
| `senoneTag*` | sequence to senonic tree nodes |
| `tmSet*` | sequence to transitions |

`delete <item>`

remove a topo from a TopoSet object

| | |
|---|---|
| `item` | name of item in list |

`index <names*>`

return the index of a named topo

| | |
|---|---|
| `names*` | list of names |

`name <idx*>`

return the name of an indexed topo

| | |
|---|---|
| `idx*` | list of indices |

`puts`

displays the contents of a TopoSet object

`read <filename>`

read TopoSet from a file

| | |
|---|---|
| `filename` | name of topo set file |

`write <filename>`

write TopoSet into a file

| | |
|---|---|
| `filename` | name of topoSet file |

**Subobjects:**

| | |
|---|---|
| `list` | (`List`) |
| `senoneSet` | (`SenoneSet`) |
| `tmSet` | (`TmSet`) |

**Elements:** are of type `Topo`.

### 8.5.42 Tree

This section describes the '*Tree*': *A 'Tree' object is an allophone clustering tree.*

**Creation:** `Tree <name> <phones> <phonesSet> <tags> <modelSet> [-padPhone padphone]`

| | |
|---|---|
| `name` | name of the tree |
| `phones` | set of phones (`Phones`) |
| `phonesSet` | set of phone set (`PhonesSet`) |
| `tags` | set of tags (`Tags`) |
| `modelSet` | model set |
| `padphone` | padding phone index |

**Configuration:** `tree configure`

| | |
|---|---|
| `-blkSize` | = 5000 |
| `-commentChar` | = ; |
| `-itemN` | = 0 |
| `-name` | = cbsdt |
| `-padPhone` | = -1 |
| `-phones` | = PHONES |
| `-phonesSet` | = phonesSetISLci |
| `-ptreeAdd` | = 0 |
| `-tags` | = tagsISLci |
| `-useN` | = 2 |

**Methods:** `tree`

`add <nodeName> <question> <noNode> <yesNode> <undefNode> <model> [-ptree ptree]`

add a new node to the tree

| | |
|---|---|
| `nodeName` | name of the node |
| `question` | question string |
| `noNode` | NO successor node |
| `yesNode` | YES successor node |
| `undefNode` | UNDEF successor node |
| `model` | name of the model |
| `ptree` | name of the ptree |

`cluster <rootNode> [-questionSet questionset] [-minCount mincount] [-minScore minscore] [-maxSplit maxsplit] [-file file] [-bottomUp bottomup] [-lee lee] [-verbose verbose]`

split whole subtree of a given root node

| | |
|---|---|
| `rootNode` | root node |
| `questionset` | question set (`QuestionSet`) |
| `mincount` | minimum count (ptree) |
| `minscore` | minimum score |
| `maxsplit` | maximum number of splits |
| `file` | cluster log file |
| `bottomup` | cluster bottom up (agglomerative) |
| `lee` | Kai-Fu Lee's bottom up cluster extension |
| `verbose` | verbose |

`get <node> <tagged phones> <leftContext> <rightContext> [-node node]`

descend a tree for a given phone sequence

| | |
|---|---|
| `node` | root node |
| `tagged phones` | list of tagged phones |
| `leftContext` | left context |
| `rightContext` | right context |
| `node` | want node name (0/1) |

`index <names*>`

return the index of a node

| | |
|---|---|
| `names*` | list of names |

`list`

list a tree contents in TCL list format

`name <idx*>`

return the name of an indexed node

| | |
|---|---|
| `idx*` | list of indices |

`puts`

displays the contents of a tree object

`question <node> [-questionSet questionset] [-minCount mincount]`

return best splitting question to ask

| | |
|---|---|
| `node` | root node |
| `questionset` | question set (`QuestionSet`) |
| `mincount` | minimum count |

`read <filename>`

read a tree from a file

| | |
|---|---|
| `filename` | name of tree file |

`split <node> <question> <noNode> <yesNode> <undefNode> [-minCount mincount]`

split node according to a question

| | |
|---|---|
| `node` | node |
| `question` | question |
| `noNode` | NO successor node |
| `yesNode` | YES successor node |
| `undefNode` | UNDEF successor node |
| `mincount` | minimum count |

`trace <node> <tagged phones> <leftContext> <rightContext> [-node node]`

trace a tree for a given phone sequence

| | |
|---|---|
| `node` | root node |
| `tagged phones` | list of tagged phones |
| `leftContext` | left context |
| `rightContext` | right context |
| `node` | want node name (0/1) |

`transform <tree> <mainTree> <modTree> <questionSet> [-dummyName dummyname] [-rootIdentifier rootidentifier] [-divide divide]`

transform tree for modalities

| | |
|---|---|
| `tree` | tree with modality questions (`Tree`) |
| `mainTree` | tree to add later the normal nodes (`Tree`) |
| `modTree` | tree to add later the modality nodes (`Tree`) |
| `questionSet` | set of only modality questions (`QuestionSet`) |
| `dummyname` | name for dummy distributions |
| `rootidentifier` | string with rootIdentifiers separated by space |
| `divide` | divide tree into subtrees |

`write <filename>`

write a tree into a file

| | |
|---|---|
| `filename` | name of tree file |

**Subobjects:**

| | |
|---|---|
| `list` | (`List`) |
| `modelSet` | (`CBNewSet`) |
| `ptreeSet` | (`PTreeSet`) |
| `questionSet` | (`QuestionSet`) |

### 8.5.43 TreeNode

This section describes the '*TreeNode*': *TreeNode*

**Creation:** `TreeNode` cannot be created directly.
It is accessible as a sub-object of `Tree`!

**Configuration:** `treenode configure`

| | |
|---|---|
| `-model` | = -1 |
| `-name` | = ROOT-b |
| `-no` | = 1 |
| `-ptree` | = -1 |
| `-question` | = 0 |
| `-undef` | = 1 |
| `-yes` | = 1 |

## 8.6 Semi-tied covariances (src/stc)

### 8.6.1 CBNewParMatrixSet

This section describes the '*CBNewParMatrixSet*': *Set of CBNewParMatrix parameter matrices*

**Creation:** `CBNewParMatrixSet <name> <par1> [-defLearnRate deflearnrate]`

| | |
|---|---|
| `name` | name of the set |
| `par1` | number of dimensions of feature space (<dimN>) or @<fName>: structure file to loa |
| `deflearnrate` | SUPERFLUOUS |

**Configuration:** `cbnewparmatrixset configure`

| | |
|---|---|
| `-blkSize` | = 5000 |
| `-defLearnRate` | = 0.100000 |
| `-dimN` | = 1 |
| `-itemN` | = 0 |
| `-name` | = cbnewparmatrixset |
| `-useN` | = 1 |

**Methods:** `cbnewparmatrixset`

`add <parMatName> <blockN> <sizeVect> [-dimVect dimvect] [-learnRate learnrate]`

add new list element

| | |
|---|---|
| `parMatName` | name of parameter matrix object |
| `blockN` | number of blocks in parameter matrix |
| `sizeVect` | vector holding block sizes (`SVector`) |
| `dimvect` | dimension index vector (`SVector`) |
| `learnrate` | SUPERFLUOUS |

`cleanup`

remove all parameter matrices without no links

`cluster`

cluster Gaussians

`convert <FeatureSet> <name>`

convert feature

| | |
|---|---|
| `FeatureSet` | name of the feature set (`FeatureSet`) |
| `name` | source feature |

`evalKL`

evaluate KL criterion

`index <names*>`

get index of list element

`names*` list of names

`loadWeights <fName>`

load weights

`fName` name of structure file to create

`name <idx*>`

get name of list element

`idx*` list of indices

`save <fName>`

save object structure

`fName` name of structure file to create

`saveWeights <fName>`

save weights

`fName` name of structure file to create

`update`

update parameter matrices

### 8.6.2 CBNewSet

This section describes the '*CBNewSet*': *Set of CBNew codebooks*

**Creation:** `CBNewSet <name> <parmatSet> <featureSet> <par2>`

| | |
|---|---|
| `name` | name of the set |
| `parmatSet` | parameter matrix set (`CBNewParMatrixSet`) |
| `featureSet` | feature set (`FeatureSet`) |
| `par2` | feature space dimensions <dimN> OR @<fname>: name of structure file to load |

**Configuration:** `cbnewset configure`

| | |
|---|---|
| `-blkSize` | = 50000 |
| `-dimN` | = 1 |
| `-featureSet` | = featureSetISLci |
| `-itemN` | = 0 |
| `-name` | = cbnewset |
| `-parmatSet` | = cbnewparmatrixset |
| `-phase` | = cons |
| `-trainParmats` | = 1 |
| `-useN` | = 0 |

**Methods:** `cbnewset`

`accu`

accumulate data

`accuMatrix`

accumulate sample matrix

`add <cbName> <featName> <refN>`

add new codebook

| | |
|---|---|
| cbName | name for codebook |
| featName | name of feature to use |
| refN | number of densities |

`clearAccus`

clear accumulators

`clearTrainParmats`

switch off parmat training

`closeProt <cbIndex> <protNum>`

INTERNAL! Use CBNewSetCloseProt

| | |
|---|---|
| cbIndex | codebook index |
| protNum | protocol number |

`compare <CBNewSet>`

compare two codebooks

| | |
|---|---|
| CBNewSet | CBNewSet to compare (CBNewSet) |

`convert <CodebookSet> <DistribSet>`

convert new codebook style to old style

| | |
|---|---|
| CodebookSet | name of the codebook set (CodebookSet) |
| DistribSet | name of the distrib set (DistribSet) |

`dataPlot`

scatter plot of most relev. dimensions

`evalKL`

evaluate KL criterion

`evalProt`

INTERNAL! Use CBNewSetEvalProt

`genSamples <cbIndex> <sampN> <sampMat> [-seed seed]`

generate samples using codebook model

| | |
|---|---|
| cbIndex | index of codebook |
| sampN | number of samples to create |
| sampMat | sample matrix (FMatrix) |
| seed | seed to use for PRNG |

`index <names*>`

get index of list element

| | |
|---|---|
| names* | list of names |

`link <parmatName> <cbIndex> <refIndex>`

link parameter matrix to gaussian(s)

| | |
|---|---|
| parmatName | name of parameter matrix |
| cbIndex | index of codebook |
| refIndex | reference index (or 'allFree' / 'all') |

`loadAccus`

load accumulators incrementally

`loadAccusDep`

load accumulators incrementally

`loadWeights <fName>`

load codebook weights

| | |
|---|---|
| fName | name of weight file to load |

`name` <idx*>

get name of list element

| | |
|---|---|
| `idx*` | list of indices |

`openProt` <cbIndex> <dataMat> [-critFunc critfunc] [-begin begin] [-end end]

INTERNAL! Use CBNewSetOpenProt

| | |
|---|---|
| `cbIndex` | codebook index |
| `dataMat` | evaluation data matrix (FMatrix) |
| `critfunc` | criterion function |
| `begin` | first row of eval. epoch in data matrix |
| `end` | last row of eval. epoch in data matrix |

`phase` <phaseName>

change object phase

| | |
|---|---|
| `phaseName` | new phase ('work','test') |

`reset`

reset active flags

`save` <fName>

save object structure

| | |
|---|---|
| `fName` | name of structure file to create |

`saveAccus`

save accumulators

`saveAccusDep`

save accumulators

`saveWeights` <fName>

save codebook weights

| | |
|---|---|
| `fName` | name of weight file to create |

`setTrainParmats`

switch on parmat training

`unlink` <cbIndex> [-refIndex refindex]

unlink gaussian(s)

| | |
|---|---|
| `cbIndex` | index of codebook |
| `refindex` | index of gaussian to unlink |

`update`

update parameters based on accus

### 8.6.3 CBNewStream

This section describes the '*CBNewStream*': *Stream based on extended codebooks (CBNew)*

**Creation:** `CBNewStream <name> <cbnewSet> <tree>`

| | |
|---|---|
| `name` | name of the object |
| `cbnewSet` | codebook set (CBNewSet) |
| `tree` | model tree (Tree) |

**Configuration:** `cbnewstream configure`

| | |
|---|---|
| `-cbnewSet` | = cbnewset |
| `-name` | = cbnewstream |
| `-tree` | = cbsdt |
| `-useN` | = 1 |

**Methods:** cbnewstream

accu

accumulate sufficient statistic

get <node> <tagged phones> <leftContext> <rightContext> [-node node]

returns a codebook given a tagged phone sequence

| | |
|---|---|
| node | root node |
| tagged phones | list of tagged phones |
| leftContext | left context |
| rightContext | right context |
| node | want node name (0/1) |

index <names*>

returns indices of named codebooks

| | |
|---|---|
| names* | list of names |

name <idx*>

returns names of indexed codebooks

| | |
|---|---|
| idx* | list of indices |

update

update parameters

**Subobjects:**

cbnewSet (CBNewSet)

**Elements:** are of type CBNew.

## 8.7 Diverse

### 8.7.1 TODO

This section lists the modules that yet have to be documented. This page exists to avoid too many dead links.

| | |
|---|---|
| ??? | Objects of unknown type |
| BBILeaf | look in BBINode |
| BBITree | look in BBINode |
| CBNew | look in CBNewSet |
| CBNewParMatrix | look in CBNewSet |
| DCovMatrix | internal object |
| Duration | look in DurationSet |
| FArray | internal object |
| Feature | look in FeatureSet |
| Filter | internal object |
| IArray | internal object |
| LCM | look in LCMSet |
| LDAClass | look in LDA |
| MLAdaptItem | look in MLAdapt |

`NGramLMSubs` look in `LingKS`

`PHMM` look in `PHMMSet`

`PTreeNode` look in `PTree`

`PhraseLMItem` look in `LingKS`

`Question` look in `QuestionSet`

`RCM` look in `RCMSet`

`Rewrite` look in `RewriteSet`

`SNode` look in `STree`

`SampleSetClass` look in `SampleSet`

`StreamNormItem` look in `DistribStream`

`Tm` look in `TmSet`

`Word` look in `Dictionary`

`XCM` look in `XCMSet`

Also, there is code to work with Neural Networks under src/net.

# Chapter 9

# Tcl-Library

**WARNING:** These entries were generated automatically. This list is non-exhaustive, but it includes all 'user-level' functions.

The argument LSID is the 'local system ID'. It is usually given by the variable SID. For example, 'bbiInit' would usually be called as `bbiInit $SID` or `bbiInit $SID -desc bbi.desc -param bbi.param.gz`.

## 9.1 align.tcl

This is a collection of error rate measuring tools. There are four functions for aligning correct and hypothesised data:

```
     rawAlign    will return the alignment path for one sentence
        align    will return the error summary for one sentence
 rawAlignFile    will return the error summary for an entire file
    alignFile    will not return anything but print to stdout like NIST's
```

The purpose of implementing this in Tcl is to have a tool that will allow us to build a 1-button-Janus which will be able to tune itself on a given development (or crossvalidation) test set, using the recognition error rate as the driving objective function.

Procedures defined in `tcl-lib/align.tcl`:

### 9.1.1 align

This function will return the error summary for one sentence.

```
align      co hy [-sub subP] [-ins insP] [-del delP]

   co    = string containing correct sentence (rawAlign, align) or
           file of many correct sentences (alignFiles, rawAlignFiles)
   hy    = string containing the hypothesis (rawAlign, align) or
           file of many hypotheses (alignFiles, rawAlignFiles)
   subP  = two-dimensional array such that subP(w1,w2) is the
           substitution penalty for substituting word w1 by word w2
   insP  = one-dimensional array such that insP(w1) is the
           insertion penalty for inserting the word w1
   delP  = one-dimensional array such that delP(w1) is the
           deletion penalty for deleting the word w1
```

There is also an external scoring program, which runs significantly faster.

**Arguments:**

| | |
|---|---|
| `<corr>` | correct sentence (reference) |
| `<hypo>` | recognizer output (hypothesis) |
| `-sub` | 2D array of substitution penalties |
| `-ins` | 1D array of insertion penalties |
| `-del` | 1D array of deletion penalties |

## 9.2 `bbi.tcl`

BBI (Bucket-Box-Intersection) is a Gaussian selection algorithm, used for speed-up during decoding. Usually, the use of BBI during decodings results in a speed-up of factor 2, with marginal loss in word accuracy. The routines here set up the BBI infrastructure.

Procedures defined in `tcl-lib/bbi.tcl`:

### 9.2.1 `bbiSetInit`

Initializes a BBI tree (loads the description file) and loads the parameters into the corresponding codebook. The codebook's scoring function then uses the BBI tree for future score computations. You can also use this function during creation of a BBI.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `-codebookSet` | codebookSet object |
| `-desc` | description file |
| `-param` | parameter file |
| `-log` | name of log channel |

## 9.3 `cbnew.tcl`

This is the Extended Codebook Set. Use it in conjunction with STCs (semi-tied co-variances), to find the OFS (optimal feature space).

Procedures defined in `tcl-lib/cbnew.tcl`:

### 9.3.1 `cbnewSetInit`

Initializes the CBNew set. Load the 'CBNewParMatrixSet', requires 'ParmatSet'.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `-cbnewSet` | codebook set |
| `-featureSet` | feature set (`FeatureSet`) |
| `-parmatSet` | parameter matrix set (`CBNewParMatrixSet`) |
| `-desc` | description file |
| `-param` | parameter file |
| `-log` | name of log channel |

## 9.4 cfg.tcl

This file contains procedures for using Context Free Grammars together with the Ibis decoder. The grammars can be initialized by calling 'cfgInit $SID' as usual. Other procedures are provided as methods of various objects.

Procedures defined in `tcl-lib/cfg.tcl`:

### 9.4.1 cfgActivate

Sets the status of all grammars to active, which match the given tag. The tags 'SHARED' and 'all' are reserved

**Arguments:**

| | |
|---|---|
| `cfgSet` | ... |
| `<tag>` | tag of the grammar |

### 9.4.2 cfgDeactivate

Sets the status of all grammars to inactive, which match the given tag. The tags 'SHARED' and 'all' are reserved

**Arguments:**

| | |
|---|---|
| `cfgSet` | ... |
| `<tag>` | tag of the grammar |

### 9.4.3 cfgGenerate

Generates terminal sequences for a CFG

**Arguments:**

| | |
|---|---|
| `cfg` | ... |
| `<seqN>` | number of terminal sequences |
| `-mode` | generation mode (random—fixed) |
| `-recurse` | follow recursions |
| `-file` | file name to write output |

### 9.4.4 cfgGetParseTree

Returns the parse tree of a given text string. This method is case sensitive!

**Arguments:**

| | |
|---|---|
| `cfgSet` | ... |
| `<text>` | text string to parse |
| `-svmap` | use SVMap to map SVX<->LVX (`SVMap`) |
| `-format` | output format (soup—jsgf) |
| `-auxNT` | print also auxilliary NTs |

### 9.4.5 cfgInit

Initializes the CFGs. By using the option '-makeDict' and defininig a base dictionary, it is also possible to build a new dictionary limited to the words given by the CFGs.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `-grammars` | list of grammars and tags |
| `-baseDict` | base dict for lookup |
| `-dict` | resulting new dict |
| `-classes` | mapping of classes |
| `-fillers` | list of filler words |
| `-startover` | allow starting over |
| `-makeDict` | make dict out of cfg |

### 9.4.6 `cfgSetGenerate`

Generates terminal sequences for a CFGSet

**Arguments:**

| | |
|---|---|
| `cfgSet` | ... |
| `<seqN>` | number of terminal sequences |
| `-mode` | generation mode (random—fixed) |
| `-recurse` | follow recursions |
| `-file` | file name to write output |

### 9.4.7 `cfgSetWeightRules`

???

**Arguments:**

| | |
|---|---|
| `cfgSet` | ... |
| `<rules>` | list of rules |
| `-weight` | weight apllied to all rules |

### 9.4.8 `cfgSetWriteFSM`

Writes a rule in a CFG or CFGSet in AT&T FSM format

**Arguments:**

| | |
|---|---|
| `cfgSet` | ... |
| `<rule>` | rule to print as FSM |
| `<fsm>` | fsm file |

## 9.5 `cli.tcl`

Procedures to provide backward compatibility for commands included to reduce the need for forks. Usage is not exactly the same as the standard Unix commands.

Procedures defined in `tcl-lib/cli.tcl`:

### 9.5.1 `cp`

Copies files

**Arguments:**

| | |
|---|---|
| `<from>` | file name(s) (glob expression) |
| `<to>` | target (directory) |
| `-f` | 0 return on error, 1 continue |

### 9.5.2 mkdir

Creates directories

**Arguments:**

| | |
|---|---|
| `<dir>` | directory(ies) |
| `-f` | 0 return on error, 1 continue |

### 9.5.3 mv

Moves files

**Arguments:**

| | |
|---|---|
| `<from>` | file name(s) (glob expression) |
| `<to>` | target (directory) |
| `-f` | 0 return on error, 1 continue |

### 9.5.4 rm

Removes files

**Arguments:**

| | |
|---|---|
| `<file>` | file name(s) (glob expression) |
| `-f` | 0 return on error, 1 continue |

### 9.5.5 rmdir

Removes directories

**Arguments:**

| | |
|---|---|
| `<dir>` | directory(ies) (glob expression) |
| `-f` | 0 return on error, 1 continue |

### 9.5.6 sleep

Sleeps.

**Arguments:**

| | |
|---|---|
| `<sec>` | sleep <sec> seconds |

### 9.5.7 touch

Touches files

**Arguments:**

| | |
|---|---|
| `<files>` | file name(s) (no glob) |

### 9.5.8 wait

Waits a while.

**Arguments:**

| | |
|---|---|
| `<file>` | name of file to wait for |
| `-intervall` | poll every n seconds |
| `-maxtime` | wait no longer than (sec) |

## 9.6 codebook.tcl

A CodebookSet contains a number of Codebooks, the standard JRTk object for Gaussian functions. The mixture weights are held in DistribSets.

Procedures defined in `tcl-lib/codebook.tcl`:

### 9.6.1 codebookSetInit

Creates a CodebookSet (reads the description file) and can also load the parameters.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `-codebookSet` | codebookSet object |
| `-featureSet` | feature set (`FeatureSet`) |
| `-desc` | description file |
| `-param` | parameter file |
| `-bmem` | use block memory management |
| `-log` | name of log channel |

## 9.7 dbase.tcl

These functions deal with the Janus database. Most scripts rely on the database to find information related to the current speaker or the current utterance. In most cases, the DBase is organized as two different databases: one holding the information for all speakers (including which utterances they spoke) and one containing the information specific for one utterance (ADC, FROM, TO, speaker, ...).

Procedures defined in `tcl-lib/dbase.tcl`:

### 9.7.1 dbaseInit

Initializes the DBase.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `<name>` | data base name |
| `-dbase` | data base object |
| `-path` | dbase path |
| `-log` | name of log channel |

### 9.7.2 dbaseUttFilter

Can be re-defined to leave out utterances during the training according to certain criteria.

**Arguments:**

| | |
|---|---|
| `<dbase>` | database name (not object) |
| `<uttID>` | utterance ID |

### 9.7.3 dbaseUttInfo

Returns all the information available in the DBase for one given utterance. It combines the information in the speaker and utterance database.

**Arguments:**

| | |
|---|---|
| `<dbase>` | database name (not object) |
| `<spkID>` | speaker ID |
| `<uttID>` | utterance ID |

### 9.7.4 foreachSegment

Can be used to loop over all utterances for a given speaker. Calls dbaseUttFilter to determine, if some segments should be left out.

**Arguments:**

| | |
|---|---|
| `<utt>` | return value: variable to contain the utterance |
| `<dbase>` | the database you use |
| `<spk>` | the speaker |
| `<body>` | the script to execute |

## 9.8 dictionary.tcl

These functions deals with the dictionary.

Procedures defined in `tcl-lib/dictionary.tcl`:

### 9.8.1 dictInit

Creates a dictionary object and possibly loads a dictionary file into it.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `-phones` | phones |
| `-tags` | tags |
| `-dict` | dictionary |
| `-desc` | description file |

## 9.9 displayLabels.tcl

These functions allow you to view the contents of a label object. They are used by featshow.

Procedures defined in `gui-tcl/displayLabels.tcl`:

### 9.9.1 displayLabels

Usage: `displayLabels <path> <hmm>`

**THE UTTERANCE WINDOW** The first window will show a rectangle for each of the words that were aligned in the utterance. (Optional words that were not aligned are not displayed.) Each rectangle's width is proportional to the number of frames that are consumed by the word, and its height is proportional to the number of states (in terms of AModel-states). Every rectangle is labeled with the orthographic spelling of the word, and with the frame range (first frame .. last frame). However, you can decide yourself what is displayed by choosing the appropriate radio-buttons in the 'full-view' menu. Within the word-rectangles you can see smaller rectangles, representing phonemes. These rectangles can be labelled with the phones names, if you choose so. You can choose the size of the display by clicking on the appropriate radio-button in the 'full zoom' window.

**THE DETAILED VIEW WINDOW** Clicking on a word's rectangle (not on one of the phones) with the 1st mouse button will open a new window with a detailed display of the selected word. You can also select an area by dragging the mouse while holding the 3rd button. Or select an area by clicking on a phone's rectangle. After you realease the mouse button, you'll get a window with a detailed view of the selected area. In this window you'll find a grid displaying the frames and states of the utterance. The frames are labelled in a synchonuously scrolling canvas below the main display canvas, the states are labelled (with the senone names) in a synchronuously scrolling canvas to the left of the main display canvas. Above the main display canvas is another synchonuously scrolling canvas, whose contents are defined by the procedure 'displayLabelsScore'. The default is to display the local acoustic score for every frame, however you can redifine this function to display whatever you wish from what is available in a CELL (see displayLabelsLaprep for details). Every visited state is represented by a circle. Below the circle are one, up to three, or up to six) lines which show more information about the state. What is displayed below the circles is defined in the procedure 'displayLabelsBelowCircle'. Have a look at the code of the procedure if you'd like to redefine it. Clicking on a circle will display all available information about that state in an extra window. You can choose how detailed your 'detailed view window' is by clicking on the appropriate radio-button in the 'detailed zoom' menu.

Note that this procedure defines several global identifiers.

**Arguments:**

| | |
|---|---|
| <`path`> | underlying Path object (`Path`) |
| <`hmm`> | underlying HMM object (`HMM`) |

## 9.10 `displayTree.tcl`

The functions defined in this file allow you to view a Tree object in a Tk widget.

Procedures defined in `gui-tcl/displayTree.tcl`:

### 9.10.1 `treeDisplay`

Displays a tree if you give it the name of the tree object and the name of the root node to start from.

**Arguments:**

| | |
|---|---|
| <`tree`> | tree object (`Tree`) |
| <`root`> | name of root node |

## 9.11 `distrib.tcl`

This file provides an easy way to set up the Gaussian mixture weights.

Procedures defined in `tcl-lib/distrib.tcl`:

### 9.11.1 `distribSetInit`

Initializes a set of distributions. It reads the descriptions and can then load the parameters. by default, it assumes that the underlying codebook is called 'codebookSet$SID', which is very easy to achieve if you use 'codebookSetInit'.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `-distribSet` | distribSet object |
| `-codebookSet` | codebook set (`CodebookSet`) |
| `-desc` | description file |
| `-param` | parameter file |
| `-bmem` | bmem option |

## 9.12 `distribTree.tcl`

This file provides a wrapper for the tree of distributions, which is needed to find the distribution for each context.

Procedures defined in `tcl-lib/distribTree.tcl`:

### 9.12.1 `distribTreeInit`

Initializes 'distribTree$SID'. Needs a 'distribSet', a description file and creates a 'distribStream', which the 'senoneSet' takes to compute scores.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `-distribTree` | distribTree object |
| `-distribStrea` | m distribStream object |
| `-distribSet` | distribution set (`DistribSet`) |
| `-phones` | phones set (`Phones`) |
| `-phonesSet` | phonesSet set (`PhonesSet`) |
| `-tags` | tags set (`Tags`) |
| `-ptree` | polyphonic tree |
| `-desc` | description file |
| `-padPhone` | padding phone |
| `-log` | name of log channel |

## 9.13 `featshow.tcl`

These functions allow you to display features.

Procedures defined in `gui-tcl/featshow.tcl`:

### 9.13.1 `featshow`

Shows a feature, USAGE: featshow <featureset> <feature> [<width> [<height>].

**Arguments:**

| | |
|---|---|
| `<FeatureSet>` | FeatureSet to use (`FeatureSet`) |
| `<Feature>` | name of feature to display |
| `-width` | width of window |
| `-height` | height of window |

## 9.14 `feature.tcl`

This file covers the initialization of the FeatureSet. See 'featshow.tcl' to find out more about the visualization of these features.

Procedures defined in `tcl-lib/feature.tcl`:

### 9.14.1 `featureSetInit`

Initializes a FeatureSet.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `-featureSet` | feature set name |
| `-desc` | description procedure |
| `-access` | access function |
| `-lda` | ptr to LDA matrix |
| `-ldaFeat` | feat for LDA matrix |
| `-log` | name of log channel |

## 9.15 `featview.tcl`

These functions allow you to display features in a FeatureSet.

Procedures defined in `gui-tcl/featview.tcl`:

### 9.15.1 `featview`

Displays different aspects of a FeatureSet, i.e. view its contents, load alternate files ...

**Arguments:**

| | |
|---|---|
| <`FeatureSet`> | FeatureSet to use (`FeatureSet`) |

## 9.16 `ibis.tcl`

This file contains procedures for the Ibis decoder. It can be initialized by calling 'ibisInit $SID' as usual. The other rotines are available as methods of various objects.

Procedures defined in `tcl-lib/ibis.tcl`:

### 9.16.1 `ibisInit`

Initializes the Ibis decoder object hierarchy. It is possible to integrate existing objects (e.g. language models) into the decoder, although this procedure can build objects and load the appropriate descriptions, data- or dumpfiles, too.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `-dict` | Search Dictionary (`Dictionary`) |
| `-ttree` | Topology Tree (`Tree`) |
| `-phmmSet` | Phonetic HMM Set (`PHMMSet`) |
| `-lcmSet` | Left Context Model Set (`LCMSet`) |
| `-rcmSet` | Right Context Model Set (`RCMSet`) |
| `-xcmSet` | X-Word Context Model Set (`XCMSet`) |
| `-vocab` | Search Vocabulary (`SVocab`) |
| `-svmap` | Mapper SVX->LVX (`SVMap`) |
| `-stree` | Search Tree (Phonetic) (`STree`) |
| `-ltree` | Search Tree (Linguistic) (`LTree`) |
| `-spass` | Search Object (`Tree`) |
| `-streeDump` | search tree dump file |
| `-vocabDesc` | search vocabulary |
| `-mapDesc` | traditional LM-map file or pron. variants |
| `-maplaDesc` | traditional LM-map file or pron. variants |
| `-readSubs` | read map-table from 'NGramLM' into SVMap |
| `-phraseLMDesc` | multi-word LM file |
| `-baseLMDesc` | base lmodel |
| `-ipolLMDesc` | interpolation lmodel |
| `-lmDesc` | language model |
| `-lmlaDesc` | language model lookahead |
| `-lalz` | LM lookahead weight |
| `-lz` | language model weight |
| `-lp` | language model penalty |
| `-fp` | filler word penalty |
| `-masterBeam` | master beam setting |
| `-lmType` | Language Model Type |
| `-lks` | Language Model (`LingKS`) |
| `-lm` | Language Model (discouraged) (`LingKS`) |
| `-lksla` | LookAhead Language Model (`LingKS`) |
| `-cacheN` | cache lines in ltree |
| `-depth` | depth of ltree |
| `-xcm` | use XCMSet |
| `-useCtx` | use context |
| `-smemFree` | free memory |
| `-ignoreRCM` | ignore RCMs in XCM |
| `-fastMatch` | Fast Match SID |
| `-verbose` | verbose |

### 9.16.2 `lksInit`

Can be used to create a language model.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `<LKSType>` | type of LingKS |
| `-dict` | dictionary |
| `-vocab` | vocabulary |
| `-svmap` | mapper SVX->LVX |
| `-lks` | name of LingKS object |
| `-lksBase` | name of LingKS base object |
| `-lksIPol` | name of LingKS ipol object |
| `-segSize` | segSize parameter for NGramLM |
| `-lksWeights` | weights-file to load into main LingKS |
| `-lksDesc` | file to load into main LingKS |
| `-ipolDesc` | file to load into ipol LingKS |
| `-baseDesc` | file to load into base LingKS |
| `-phraseDesc` | file to load into phrase LingKS |
| `-vocabDesc` | vocabulary file to load |
| `-mapDesc` | mapping file to load |
| `-readSubs` | LM to read substitutions from |
| `-verbose` | verbosity |

## 9.17 `kmeans.tcl`

This file makes it easier to start EM training by initializing the codebooks with the K-Means algorithm. Before you can do that, you need to extract samples.

Procedures defined in `tcl-lib/kmeans.tcl`:

### 9.17.1 `doKMeans`

Performs K-Means in parallel, creating a CodebookSet ( a DistribSet is produced, too, but the weights are equally distributed). This procedure can combine and cluster data from different sample extractions.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `<cbListFile>` | file of codebook names |
| `-codebookSet` | codebook set (`CodebookSet`) |
| `-distribSet` | distribution set (`DistribSet`) |
| `-paramFile` | base name of parameters |
| `-dataPath` | path of sample files |
| `-kmeansPath` | path of kmeans files |
| `-distribUpdat` | e update distributions |
| `-tempF` | final temperature |
| `-maxIter` | number of iterations |
| `-maxCount` | max no of samples |
| `-semFile` | semaphore file |
| `-doCombine` | combine samples on demand |

## 9.18 `label.tcl`

???

Procedures defined in `gui-tcl/label.tcl`:

## 9.19 `labels.tcl`

Look here if you need to write labels (time-alignments).

Procedures defined in `tcl-lib/labels.tcl`:

### 9.19.1 `labelsMLAdaptWrite`

Equivalent to 'labelsWrite', except that it performs speaker-specific MLLR adaptation on the reference before computing the labels, which often results in better alignments. Takes more time, though.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `<spkIDfile>` | file of speaker IDs |
| `<MLAdapt>` | ML adaptation object (`MLAdapt`) |
| `-path` | name of path |
| `-lbox` | name of lbox |
| `-labelPath` | path of label files |
| `-update` | only try nonexisting paths |
| `-beam` | viterbi beam |
| `-topN` | topN beam |
| `-optWord` | optional word |
| `-variants` | variants 0/1 |
| `-minCount` | adaptation minCount |
| `-putPath` | write path into log |
| `-tryMax` | increasing beam |

### 9.19.2 `labelsWrite`

Writes labels, i.e. computes and stores a viterbi path for every utterance of every speaker found in the speaker list. You can store the labels in separate files or in a 'label-box', which contains all alignments for one speaker in one singel file.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `<spkIDfile>` | file of speaker IDs |
| `-path` | name of path |
| `-lbox` | name of lbox |
| `-labelPath` | path of label files |
| `-update` | only try nonexisting paths |
| `-beam` | viterbi beam |
| `-topN` | topN beam |
| `-optWord` | optional word |
| `-variants` | variants 0/1 |
| `-putPath` | write path into log |
| `-tryMax` | increasing beam |

## 9.20 `latview.tcl`

A viewer for GLat objects.

Procedures defined in `tcl-lib/latview.tcl`:

### 9.20.1 showlat

Display the contents of a GLat lattice in a Tk window. Be careful with large objects.

**Arguments:**

| | |
|---|---|
| obj | ... |
| ARGS | ??? |

## 9.21 lda.tcl

LDA (Linear Discriminant Analysis) is part of the standard preprocessing in the JRTk toolkit.

Procedures defined in tcl-lib/lda.tcl:

### 9.21.1 doLDA

Computes the LDA matrix. Also extracts the counts (i.e. frames) for every codebook, which is useful information and is used to determine the module during sample extraction.

**Arguments:**

| | |
|---|---|
| LSID | The system id, usually $SID. |
| <LDA> | LDA object (LDA) |
| <spkIDfile> | file of speaker IDs |
| -countsFile | file to save counts |
| -labelPath | path of label files |
| -stream | stream index |
| -optWord | optional word |
| -variants | variants 0/1 |
| -featureSet | feature set |
| -hmm | hidden markov model |
| -senoneSet | senone set |
| -path | path object |
| -lbox | lbox object |
| -semFile | semaphore file |
| -log | name of log channel |

## 9.22 misc.tcl

This file contains various procedures.

Procedures defined in tcl-lib/misc.tcl:

### 9.22.1 printDo

Performs an action (its argument) and prints the command line to stderr. Don't try to set variables within printDo, though.

**Arguments:**

| | |
|---|---|
| args | The commands to execute |

## 9.23 parmat.tcl

Library to initialize semi-tied full covariances.

Procedures defined in `tcl-lib/parmat.tcl`:

### 9.23.1 parmatSetInit

Initializes semi-tied full covariances.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `-parmatSet` | parameter matrix set |
| `-desc` | description file |
| `-dimN` | number of feature space dim. (if no desc. file is used) |
| `-param` | parameter file |
| `-log` | name of log channel |

## 9.24 phones.tcl

Deals with the PhonesSet.

Procedures defined in `tcl-lib/phones.tcl`:

### 9.24.1 phonesSetInit

Initializes a PhonesSet.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `-phonesSet` | phones set |
| `-desc` | description file |
| `-log` | name of log channel |

## 9.25 samples.tcl

Allows to extract samples, i.e. store the pre-processed data for every frame given labels and use it directly at a later stage, for example for KMeans.

Procedures defined in `tcl-lib/samples.tcl`:

### 9.25.1 doExtract

Extract the data in separate files for each codebook according to a given alignment. This is very heavy on file I/O, so plan your setup accordingly. If you specify a counts file, you can also specify the 'maxCount'; the system will then automatically compute a modulo, which prevents more than 'maxCount' samples to be extracted for every codebook.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `<SampleSet>` | SampleSet object (`SampleSet`) |
| `<spkIDfile>` | file of speaker IDs |
| `-path` | name of path |
| `-lbox` | name of lbox |
| `-labelPath` | path of label files |
| `-dataPath` | path of data files |
| `-combPath` | path for combining files |
| `-countsFile` | file to save counts |
| `-maxCount` | max count in file |
| `-modulus` | modulus |
| `-stream` | stream index |
| `-optWord` | optional word |
| `-variants` | variants 0/1 |
| `-doCombine` | doCombine 0/1 |
| `-semFile` | semaphore file |
| `-log` | name of log channel |

## 9.26 senone.tcl

This file contains various procedures.

Procedures defined in `tcl-lib/senone.tcl`:

### 9.26.1 senoneSetInit

Initializes the SenoneSet.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `<streams>` | stream array |
| `-phones` | phones set (`Phones`) |
| `-tags` | tags set (`Tags`) |
| `-desc` | description file |

## 9.27 showSTree.tcl

These functions allow you to view trees, too.

Procedures defined in `gui-tcl/showSTree.tcl`:

### 9.27.1 showSTree

Displays a tree object! This procedure does not display an STree object, though!

**Arguments:**

| | |
|---|---|
| `<tree>` | tree object to display (`Tree`) |
| `<startNode>` | name of start node (of tree to display) |
| `<depth>` | depth of displayed tree |

## 9.28 speech.tcl

Sil/Speech Detector based on Gaussian mixture.

Procedures defined in `tcl-lib/speech.tcl`:

### 9.28.1 speechInit

Creation and initialization of a speech detector using a codebookSet and a distribSet based on a description file and a parameter file.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `-featureSet` | feature set (`FeatureSet`) |
| `-cbsdesc` | description file |
| `-cbsparam` | parameter file |
| `-dssdesc` | description file |
| `-dssparam` | parameter file |
| `-apriori` | speech a priori prob |
| `-log` | name of log channel |

## 9.29 tags.tcl

This file initializes the tags.

Procedures defined in `tcl-lib/tags.tcl`:

### 9.29.1 tagsInit

Creates a 'Tags' object, usually called tags$SID.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `-tags` | tags object name |
| `-desc` | description file |
| `-log` | name of log channel |

## 9.30 topo.tcl

This file initializes the TopoSet.

Procedures defined in `tcl-lib/topo.tcl`:

### 9.30.1 topoSetInit

Creates a 'TopoSet'.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `-tm` | transistion description |
| `-senoneSet` | senoneSet set (`SenoneSet`) |
| `-tmSet` | tmSet set (`TmSet`) |
| `-desc` | topology description |

### 9.30.2 ttreeInit

Creates a 'TopoTree'.

**Arguments:**

| | |
|---|---|
| LSID | The system id, usually $SID. |
| -phones | phones set (Phones) |
| -phonesSet | phonesSet set (PhonesSet) |
| -tags | tags set (Tags) |
| -topoSet | topoSet set (TopoSet) |
| -ptree | polyphonic tree |
| -desc | description file |
| -padPhone | padding phone |

## 9.31 train.tcl

This file contains various procedures helpful during recognizer development. Once initialized with 'trainInit $SID', the training environment provides path, hmm and other objects along with a number of Tcl-defined methods.

Procedures defined in tcl-lib/train.tcl:

### 9.31.1 fwdBwdUtterance

Performs forward-backward alignment of an utterance. The necessary information can be read from the database.

**Arguments:**

| | |
|---|---|
| LSID | The system id, usually $SID. |
| <speaker> | speaker ID |
| <uttID> | utterance ID |
| -text | text to align |
| -hmm | hmm |
| -path | path |
| -lbox | name of lbox |
| -topN | topN beam |
| -width | width of path |
| -optWord | optional word |
| -variants | variants 0/1 |

### 9.31.2 labelUtterance

Reads a binary dumped path into the path$SID structure and translates the senone indices by referring to the utterance HMM and using the path state indices to find the new senone indices therein.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `<speaker>` | speaker ID |
| `<uttID>` | utterance ID |
| `<file>` | filename |
| `-text` | text to align |
| `-optWord` | optional word |
| `-variants` | variants 0/1 |
| `-eval` | eval string extension |
| `-evalFES` | eval feature set 0/1 |
| `-featureSet` | feature set |
| `-hmm` | hmm |
| `-path` | path |
| `-lbox` | name of lbox |
| `-evalScore` | compute path score |
| `-log` | name of log channel |

### 9.31.3 `pathWriteCTM`

Writes a CTM-format hypothesis file from a path object.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `<speaker>` | speaker ID |
| `<uttID>` | utterance ID |
| `<from>` | from frame |
| `-file` | filename |

### 9.31.4 `trainInit`

Initializes the standard JRTk training environment.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `-amodelSet` | acoustic models |
| `-hmm` | hidden markov model |
| `-path` | path object |
| `-lbox` | Labelbox object |
| `-topoTree` | topology tree (`Tree`) |
| `-topoTreeRoot` | root of topoTree |
| `-durTree` | duration tree (`Tree`) |
| `-durTreeRoot` | root of duration tree |
| `-rcmSdp` | use right context for context-dependent single phone words |
| `-dict` | dictionary (`Dictionary`) |

### 9.31.5 `viterbiUtterance`

Performs viterbi alignment of an utterance. The necessary information can be read from the database.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `<speaker>` | speaker ID |
| `<uttID>` | utterance ID |
| `-text` | text to align |
| `-hmm` | name of hmm |
| `-path` | name of path |
| `-lbox` | name of Labelbox |
| `-beam` | viterbi beam |
| `-topN` | topN beam |
| `-bpMod` | after every X frames clean up bpTable (<0 never) |
| `-bpMul` | go Y * X frames back during cleanup (<1 start at first frame) |
| `-optWord` | optional word |
| `-variants` | variants 0/1 |

## 9.32 tree.tcl

Various handy procedures for Tree objects.

Procedures defined in `tcl-lib/tree.tcl`:

### 9.32.1 treeCluster

Clusters tree given a set of questions, the minimum number of counts expected to be in the ModelArray for each answer node, the minimum count and the maximum number of splits for each node.

**Arguments:**

| | |
|---|---|
| `<tree>` | tree (`Tree`) |
| `<questionSet>` | question set (`QuestionSet`) |
| `-file` | cluster log file |
| `-nodeList` | list of nodes |
| `-minCount` | minimum count |
| `-maxSplit` | max.number of split |

### 9.32.2 treeQuestion

Find a question for a given node in the tree (if there is a polyphonic tree attached to the node).

**Arguments:**

| | |
|---|---|
| `<tree>` | tree (`Tree`) |
| `<node>` | node name |
| `<questionSet>` | question set (`QuestionSet`) |
| `<parent>` | parent name |
| `<nodes>` | nodes array |
| `<count>` | count array |

### 9.32.3 treeReadSplits

Reads cluster log file into an array indexed by the gain of each split. This array is used to split a decision tree.

**Arguments:**

| | |
|---|---|
| `<files>` | cluster log files |
| `-list` | initial split list |

## 9.33 vtln.tcl

Procedure to handle VTLN (Vocal Tract Length Normalization) estimation and use.

Procedures defined in `tcl-lib/vtln.tcl`:

### 9.33.1 findLabelWarp

Given a path (in a label file) rescore all utterances of the given speaker within a window of different warp scales. Utterances of the speaker are taken until a maximum number of frames is reached. Return warp factor with best score and frames used.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `<speaker>` | speaker ID |
| `-labelPath` | path of label files |
| `-warp` | center warp |
| `-window` | window width/2 |
| `-delta` | delta steps |
| `-maxFrame` | maximal number of frames to use |
| `-v` | verbosity |
| `-optWord` | optional HMM word |
| `-variants` | use pronunciation variants |
| `-trl` | Text field in database entry |
| `-phoneLst` | list of phones |

### 9.33.2 findViterbiWarp

Find the best warp factor within a given window around an initial value. Use a first hypothesis given in $HYPO($utt) and do viterbi. Rescore for different warp scales and all utterances of the speaker. Return warp factor with best score and frames used.

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `<speaker>` | speaker ID |
| `-warp` | center warp |
| `-window` | window width/2 |
| `-delta` | delta steps |
| `-maxFrame` | maximal number of frames to use |
| `-phoneLst` | list of phones |

### 9.33.3 vtlnInit

Reads in a file containing warp factors (there is no procedure to write them, though; the file format is '<spk> <scale>' in every line).

**Arguments:**

| | |
|---|---|
| `LSID` | The system id, usually $SID. |
| `-param` | file with warp scales |

# Chapter 10

# Files

## 10.1 ContextFreeGrammars

This section describes our internal context free grammar format, called SOUP-Format. We are usually using semantic instead of syntactic context free grammars. They are read by CFGSet. A not completely specified example looks like:

```
# ======================================================================
# example grammar
# ======================================================================

# ----------------------------------------
# request path description
#       how do i
#       i want to find the way
#       can you take me
# ----------------------------------------
s[request-path-description]
        ( *PLEASE [_NT_how-to-go]    [obj_desc]     *PLEASE )
        ( *PLEASE [_NT_find-the-way] [obj_desc]     *PLEASE )
        ( *PLEASE [_NT_take-me]      [obj_desc]     *PLEASE )
        ( *PLEASE [_NT_how-about]    [obj_desc]     *PLEASE )
        ( *PLEASE [_NT_how-to-find]  [_NT_obj_desc] *PLEASE )

[_NT_how-to-go]
        ( how do i                           GO )
        ( [_NT_can-you-show|tell] *me how to GO )
        ( i WANT                          to GO )
        ( i NEED                          to GO )

[_NT_can-you-show|tell]
        ( *CAN_YOU SHOW )
        ( *CAN_YOU TELL )


[obj_desc]
        (                     to [_NT_obj_desc] )
        ( from [_NT_obj_desc] to [_NT_obj_desc] )
```

```
[_NT_obj_desc]
        ( *the          [objnm] )
        (  the *NEAREST [objcl] )
        (  A            [objcl] *NEARBY )

[objcl]
        ( [objcl_bakery] )
        ( [objcl_bank] )

[objcl_bakery]
        ( bakery )

CAN
        ( can )
        ( could )

CAN_YOU
        ( CAN you )

SHOW
        ( show )
        ( display )

TELL
        ( tell )
        ( explain *to )

GO
        ( get )
        ( go )

# ----------------------------------------
# greeting / farewell
#       hello
#       good bye
#       bye bye
# ----------------------------------------
s[greeting]
        ( [_NT_greeting] )

[_NT_greeting]
        ( hello )
        ( hi )

s[farewell]
        ( [_NT_farewell] )

[_NT_farewell]
        ( *good +bye )
```

Non terminal symbols could either be surrounded by [] or could be started with a capital letter. Terminal symbols have to be started with a lower case letter. If you start a non terminal with a capital our with the modifier _NT_, it is classified as an auxilliary non terminal and will per default not occur in the parse tree. To express optionality of a terminal or non terminal you have to use * and to express repeatability you have to use + in front of a symbol. It is also posible to combine optionality and repeatability by using *+.

Rules consist of a left hand side (LHS, the head of the rule) and a right hand side (RHS, the body of the rule). If you want to use a rule also as top level rule, i.e. a rule where you can start to parse from, you have to put the modifier s in front of the rule. As you can see above there are three top level rules: [request-path-description], [greeting] and [farewell]. The lines in a RHS of a rule are interpreted as a disjunction, the terminals and non terminals in one line as a conjunction. It is not neccessary to define the rules in a special order.

## 10.2 codebookSet

The description file read by CodebookSet. An example looks like:

```
; -------------------------------------------------------
;  Name            : codebookSetISLci
;  Type            : CodebookSet
;  Number of Items : 199
;  Date            : Thu Jul 11 20:21:13 2002
; -------------------------------------------------------
+QK-b           LDA                 48     32 DIAGONAL
+QK-m           LDA                 48     32 DIAGONAL
+QK-e           LDA                 48     32 DIAGONAL
SCH-b           LDA                 48     32 DIAGONAL
SCH-m           LDA                 48     32 DIAGONAL
SCH-e           LDA                 48     32 DIAGONAL
SIL-m           LDA                 48     32 DIAGONAL
T-b             LDA                 48     32 DIAGONAL
T-m             LDA                 48     32 DIAGONAL
T-e             LDA                 48     32 DIAGONAL
```

The columns mean the codebook, the feature, the number of gaussians, the number of dimensions and the covariance type.

## 10.3 desc.tcl

A description file for a system. A typical file looks like:

```
# ========================================================================
#  JanusRTK   Janus Speech Recognition Toolkit
#             ------------------------------------------------------------
#             Object: System description
#             ------------------------------------------------------------
#
#  Author  :  Florian Metze
#  Module  :  desc.tcl
#  Date    :  $Id: desc.tcl 2390 2003-08-14 11:20:32Z fuegen $
#
#  Remarks :  This is the description file for the ISLci system
#
# ========================================================================
#
#  $Log$
#  Revision 1.2  2003/08/14 11:19:43  fuegen
#  Merged changes on branch jtk-01-01-15-fms (jaguar -> ibis-013)
#
#  Revision 1.1.2.7  2003/08/13 14:27:19  fuegen
#  formattings
#
#  Revision 1.1.2.6  2003/08/13 14:13:46  fuegen
#  readded definitions for CFGs
#
#  Revision 1.1.2.5  2003/08/11 12:41:08  soltau
#  windows support
#
# ========================================================================

# to make some scripts happy
set host [info hostname]
set pid  [pid]

# ------------------------------------------------------------------------
#  System and Path Definitions
# ------------------------------------------------------------------------

set SID                     ISLci

set projectHome             /home/njd/IslData
set ${SID}(path)            /home/njd/IslSystem/${SID}
set ${SID}(descPath)        [file join [set ${SID}(path)] desc]
set ${SID}(dictPath)        $projectHome
set ${SID}(lmPath)          $projectHome
set ${SID}(cfgPath)         $projectHome


# ------------------------------------------------------------------------
#  Welcome
# ------------------------------------------------------------------------

writeLog stderr "      ------ System $SID -----"
writeLog stderr "${argv0} reads desc.tcl: on $env(HOST).[pid], [exec date]"
writeLog stderr "using lib: $auto_path"


# ------------------------------------------------------------------------
#  Database
# ------------------------------------------------------------------------

set ${SID}(dbaseName)       db
set ${SID}(dbasePath)       $projectHome


# ------------------------------------------------------------------------
#  Phones & Tags
# ------------------------------------------------------------------------

set ${SID}(phonesSetDesc)    [set ${SID}(descPath)]/phonesSet
set ${SID}(tagsDesc)         [set ${SID}(descPath)]/tags


# ------------------------------------------------------------------------
#  Feature Set
# ------------------------------------------------------------------------

set ${SID}(testFeatureSetDesc)   @[file join [set ${SID}(descPath)] featDesc.test]
set ${SID}(meanFeatureSetDesc)   @[file join [set ${SID}(descPath)] featDesc.mean]
set ${SID}(featureSetDesc)       @[file join [set ${SID}(descPath)] featDesc]
set ${SID}(featureSetAccess)     @[file join [set ${SID}(descPath)] featAccess]
set ${SID}(featureSetLDAMatrix)   [file join [set ${SID}(path)] train lda${SID}.bmat]
set ${SID}(warpFile)             ""
set ${SID}(warpPhones)           "STIMMHAFT"
set ${SID}(meanPath)              [file join [set ${SID}(path)] train means]


# ------------------------------------------------------------------------
#  Stream: Codebook, Distribution, Tree
# ------------------------------------------------------------------------

set ${SID}(codebookSetDesc)  [file join [set ${SID}(descPath)] codebookSet]
set ${SID}(codebookSetParam) [set ${SID}(path)]/train/Weights/4.cbs.gz
set ${SID}(distribSetDesc)   [file join [set ${SID}(descPath)] distribSet]
set ${SID}(distribSetParam)  [set ${SID}(path)]/train/Weights/4.dss.gz
set ${SID}(padPhone)         @
set ${SID}(ptreeSetDesc)     ""
set ${SID}(distribTreeDesc)  [file join [set ${SID}(descPath)] distribTree]


# ------------------------------------------------------------------------
#  Transition models, topology and duration modelling
# ------------------------------------------------------------------------

set ${SID}(durSetDesc)       ""
set ${SID}(durPTreeDesc)     ""
set ${SID}(durTreeDesc)      ""

set ${SID}(tmDesc)           [set ${SID}(descPath)]/tmSet
set ${SID}(topoSetDesc)      [set ${SID}(descPath)]/topoSet
set ${SID}(ttreeDesc)        [set ${SID}(descPath)]/topoTree


# ------------------------------------------------------------------------
#  LM, Dictionary and Vocabulary
# ------------------------------------------------------------------------

set ${SID}(dictDesc)         [set ${SID}(dictPath)]/dict.50phones
set ${SID}(useXwt)           1
set ${SID}(optWord)          \$
set ${SID}(variants)         1

# -------------------------------------------------------
# Context Free Grammars
# -------------------------------------------------------

set cfgPath                  [set ${SID}(cfgPath)]
set ${SID}(cfg,grammars)     [list [list NAV \
                                        $cfgPath/cfg.ka.nav \
                                        $cfgPath/cfg.base.nav] \
                                   [list SHARED \
                                        $cfgPath/cfg.shared]]

# ------------------------------------------------------------------------
#  Testing
# ------------------------------------------------------------------------

set ${SID}(testDictDesc)     [set ${SID}(dictDesc)]
set ${SID}(vocabDesc)        [set ${SID}(lmPath)]/vocab.germNews
set ${SID}(lmDesc)           [set ${SID}(lmPath)]/sz.ibis.gz
set ${SID}(ngramLMsegSize)   6
set ${SID}(lmWeight)         32
set ${SID}(lmPenalty)        3
set ${SID}(bbiSetDesc)       ""
set ${SID}(bbiSetParam)      ""


# ------------------------------------------------------------------------
#  Label Path
# ------------------------------------------------------------------------

set ${SID}(labelPath)     {/home/njd/IslSystem/ISLinit/labels/$spk/$utt.lbl}


set ${SID}(SPK)                  SPK     ; # speaker key
set ${SID}(UTT)                  UTTS    ; # utt     key
set ${SID}(TRL)                  TEXT    ; # trl     key
```

desc.tcl also is a good place to re-define other common functions such as **dbaseUttFilter** or **hmmMakeUtterance**. In principle, you are free to re-configure everything in this script, it is however common practice, to set the Tcl-variable SID to the name of the directory, in which this incarnation of desc.tcl resides.

## 10.4 dictionary

A Dictionary description file. It contains phones and tags.

An examples looks like this:

```
; ----------------------------
;   Example Dictionary
; ----------------------------
{$I} {{AY WB}}
{$} {{SIL WB}}
{(} {{SIL WB}}
{)} {{SIL WB}}
{Anne} {{AE WB} N {EH WB}}
{Anne(2)} {{AA WB} {N WB}}
```

## 10.5 distribSet

The description file used in a DistribSet. An example looks like this:

```
; -------------------------------------------------------
;  Name            : distribSetISLci
;  Type            : DistribSet
;  Number of Items : 199
;  Date            : Thu Jul 11 20:21:13 2002
; -------------------------------------------------------
+QK-b              +QK-b
+QK-m              +QK-m
+QK-e              +QK-e
SCH-b              SCH-b
SCH-m              SCH-m
SCH-e              SCH-e
SIL-m              SIL-m
T-b                T-b
T-m                T-m
T-e                T-e
```

The second column tells you which codebook to use.

## 10.6 distribTree

A Tree description file, used for the distribution tree. An example looks like this:

```
; -------------------------------------------------------
;  Name            : distribTreeISLci
;  Type            : Tree
;  Number of Items : 401
;  Date            : Thu Jul 11 20:21:13 2002
; -------------------------------------------------------
ROOT-b          {} ROOT-+QK-b ROOT-+QK-b ROOT-+QK-b -
ROOT-+QK-b      {0=+QK} ROOT-+hBR-b +QK-b - -
+QK-b           {} - - - +QK-b
ROOT-m          {} ROOT-+QK-m ROOT-+QK-m ROOT-+QK-m -
ROOT-+QK-m      {0=+QK} ROOT-+hBR-m +QK-m - -
+QK-m           {} - - - +QK-m
ROOT-e          {} ROOT-+QK-e ROOT-+QK-e ROOT-+QK-e -
ROOT-+QK-e      {0=+QK} ROOT-+hBR-e +QK-e - -
+QK-e           {} - - - +QK-e
ROOT-+hBR-b     {0=+hBR} ROOT-+hEH-b +hBR-b - -
+hBR-b          {} - - - +hBR-b
ROOT-+hBR-m     {0=+hBR} ROOT-+hEH-m +hBR-m - -
+hBR-m          {} - - - +hBR-m
ROOT-+hBR-e     {0=+hBR} ROOT-+hEH-e +hBR-e - -
+hBR-e          {} - - - +hBR-e
...
```

## 10.7 featAccess

This tells the featDesc where to find the data. An example looks like this:

```
set      adcfile      [file join /project/florian/isldata/adcs $arg(ADC)]
set      accessList   $sampleList
lappend  accessList  "ADCFILE $adcfile"
```

## 10.8 featDesc

The feature description file, read by the FeatureSet. An example looks like this:

```
# ========================================================
#  JanusRTk     Janus Recognition Toolkit
#               ------------------------------------------
#               Object: Feature Description
#               ------------------------------------------
#
#  Author    :  Hagen Soltau
#  Module    :  featDesc
#  Remarks   :  based on Hua's new frontend, 40 dimensions
#
#  $Log$
#  Revision 1.2  2003/08/14 11:18:59  fuegen
#  Merged changes on branch jtk-01-01-15-fms (jaguar -> ibis-013)
#
#  Revision 1.1.2.4  2002/11/19 13:23:30  metze
#  Beautification
#
#  Revision 1.1.2.3  2002/11/19 09:17:44  fuegen
#  minor changes for overfull hboxes
#
#  Revision 1.1.2.2  2002/07/31 13:10:12  metze
#  *** empty log message ***
#
#  Revision 1.1.2.1  2002/07/30 13:57:39  metze
#  *** empty log message ***
#
#  Revision 1.1  2002/03/04 16:10:49  soltau
#  Initial revision
#
# ========================================================

global WARPSCALE warpScales meanPath
global WAVFILE OLDSPK sas pms


# --------------------------------------------------------
#  Load Mean Vectors
# --------------------------------------------------------

if {![info exist OLDSPK] || $OLDSPK != $arg(spk) } {

  if {[llength [info command ${fes}Mean]]} {
    ${fes}Mean  destroy
    ${fes}SMean destroy
  }
  if {[file exist $meanPath/$arg(spk).mean]} {
    FVector ${fes}Mean  13
    FVector ${fes}SMean 13
    writeLog stderr "$fes Loading $meanPath/$arg(spk).mean"
    ${fes}Mean  bload $meanPath/$arg(spk).mean
    ${fes}SMean bload $meanPath/$arg(spk).smean
    set OLDSPK $arg(spk)
  } else {
    writeLog stderr "$fes Loading $meanPath/$arg(spk).mean FAILED"
  }
}


# -------------------------------------------------------
#  Load ADC segment...
# -------------------------------------------------------

if {![info exist WAVFILE] || $WAVFILE != $arg(ADCFILE)} {

  set WAVFILE $arg(ADCFILE)

  if {[file exist $arg(ADCFILE).shn]} {
    $fes    readADC         ADC             $arg(ADCFILE).shn \
      -h 0 -v 0 -offset mean -bm shorten
  } else {
    $fes    readADC         ADC             $arg(ADCFILE)     \
      -h 0 -v 0 -offset mean -bm auto
  }

  $fes spectrum FFT     ADC 20ms
}


# -------------------------------------------------------
#  Get warp
# -------------------------------------------------------

if {![info exist WARPSCALE]} {
  if [info exist warpScales($arg(spk))] {
     set WARP $warpScales($arg(spk))
  }  else {
     set WARP 1.00
  }
} else { set WARP $WARPSCALE }

writeLog stderr "$fes ADCfile $arg(utt) WARP $WARP"


# -------------------------------------------------------
#  Vocal Tract Length Normalization + MCEP
# -------------------------------------------------------

$fes VTLN     WFFT      FFT  $WARP -mod lin -edge 0.8
```

```
if { [llength [objects FBMatrix matrixMEL]] != 1} {
   set melN 30
   set points [$fes:FFT configure -coeffN]
   set rate   [expr 1000 * [$fes:FFT configure -samplingRate]]
   [FBMatrix matrixMEL] mel -N $melN -p $points -rate $rate
}

$fes   filterbank       MEL             WFFT           matrixMEL
$fes   log              lMEL            MEL            1.0 1.0

set cepN 13

if { [llength [objects FMatrix matrixCOS]] != 1} {
   set n [$fes:lMEL configure -coeffN]
   [FMatrix matrixCOS] cosine $cepN $n -type 1
}

$fes   matmul           MCEP            lMEL            matrixCOS


# -------------------------------------------------------
#  Mean Subtraction, Delta, Delta-Delta and LDA
# -------------------------------------------------------

$fes meansub  FEAT    MCEP  -a 2 -mean ${fes}Mean -smean ${fes}SMean
$fes adjacent FEAT+   FEAT  -delta 5

if { [$fes index LDAMatrix] > -1} {
    $fes matmul   LDA     FEAT+ $fes:LDAMatrix.data -cut 32
}

if [info exists pms] {
    foreach p [$pms] {
        $fes matmul OFS-$p LDA $pms:$p.item(0)
        if [info exists sas] {
            $sas adapt $fes:OFS-$p.data $fes:OFS-$p.data 0
        }
    }
} else {
    if [info exists sas] {
        $sas adapt $fes:LDA.data $fes:LDA.data 0
    }
}
```

Errors in the featDesc are not always easy to track. A much-used strategy to debug errors in the featDesc is to plaster it with `puts ''I am here''` commands, to find out where exactly in the code the offending operation occurs.

## 10.9 .janusrc

This describes the file `.janusrc`, which is the main configuration file for Janus. A copy of this file can be found in `~/janus/scripts/janusrc`. It is usable for both OSs, Unix and Windows.

```
# ========================================================
#  JanusRTK   Janus Speech Recognition Toolkit
#             ------------------------------------------
#             Object: .janusrc - Resources file
#             ------------------------------------------
#
#  Author  :  Florian Metze and Christian Fuegen
#  Module  :  ~/.janusrc
#  Date    :  2000-08-07
#
#  Remarks :  This file is read in by janus on startup
#
#             It contains a few settings and redefines some
#             functions for compatibility with Linux and Windows
#
#             Original by Martin Westphal,
#             Dec. 4, 1996 for Janus3.2
#
# ========================================================
#  RCS info: $Id: janusrc 2895 2009-07-27 17:38:13Z metze $
#
#  $Log$
#  Revision 1.5  2007/02/23 10:15:13  fuegen
#  JANUSHOME can now be defined externally
#
#  Revision 1.4  2003/08/25 16:09:55  soltau
#  Added library path to auto_path
#
#  Revision 1.3  2003/08/18 13:03:36  soltau
#  removed some windows specific proc's (supported now by cli)
#
#  Revision 1.2  2003/08/14 11:19:43  fuegen
#  Merged changes on branch jtk-01-01-15-fms (jaguar -> ibis-013)
#
#  Revision 1.1.2.12  2003/08/13 09:41:01  soltau
#  final fixes
#
#  Revision 1.1.2.11  2003/08/12 16:12:37  metze
#  Cleanup for P013
#
#  Revision 1.1.2.10  2003/08/11 15:09:26  soltau
#  made GLOBALFP global
#
#  Revision 1.1.2.9  2003/08/11 14:29:32  soltau
#  exec windows support
#
#  Revision 1.1.2.8  2003/08/11 12:24:08  soltau
#  Windows fix for writing log-files:
#    set LOGFILE "janus.log" to pipe stdout from 'puts' to file
#
#  Revision 1.1.2.6  2003/06/26 15:09:20  metze
#  Changes for V5.0 P013
#
#  Revision 1.1.2.5  2003/04/30 15:42:00  metze
#  Final team
#
#  Revision 1.1.2.4  2003/04/09 14:42:05  metze
#  Typo fixed
#
#  Revision 1.1.2.3  2003/04/09 14:41:51  metze
#  Switched ngets off by default
#
#  Revision 1.1.2.2  2003/04/09 13:22:45  metze
#  Cleaned up ngets stuff
#
#  Revision 1.2  2003/01/17 15:42:24  fuegen
#  Merged changes on branch jtk-01-01-15-fms
#
#  Revision 1.1.2.1  2002/11/15 14:33:13  fuegen
#  Initial version
#
#
# =======================================================

# -------------------------------------------------------
#  check host and home
# -------------------------------------------------------

# for Condor & SLURM this is unreliable, therefore always set env(HOST)
set env(HOST) [lindex [split [info hostname] .] 0]

if {![info exists env(HOME)]} {
    set env(HOME) "HOME"
    #puts "set home directory : $env(HOME)"
}

# -------------------------------------------------------
#  Set the auto path so that tcl libraries can be found.
# -------------------------------------------------------

if {[info exists env(JANUSHOME)]} {
    set JANUSHOME $env(JANUSHOME)
} else {
    # E.g. for Windows:
    # set JANUSHOME "e:/ISL/hagen"

    # For Unix:
    set JANUSHOME  [file join $env(HOME) janus]
}

set JANUSLIB                        [file join $JANUSHOME gui-tcl]
set auto_path [linsert $auto_path 0 [file join $JANUSHOME tcl-lib]]
set auto_path [linsert $auto_path 0 [file join $JANUSHOME library]]
set auto_path [linsert $auto_path 0 $JANUSLIB]
regsub -all {\\} $auto_path {/} auto_path

# ----------------------------------------------------------------
#  WINDOWS dependent settings
#  1. define global variable LOGFILE to pipe stdout/stderr to file
#  2. manual sourcing of tcl-lib and gui-tcl
#  3. function redefinitions
#     exit  - for logging
#     puts  - output teeing into logfiles
#     exec  - to support some unix commands also under windows
# ----------------------------------------------------------------

if {[regexp {indows} $tcl_platform(os)]} {
    # uncomment this to pipe stdout/stderr to file
    # set LOGFILE "janus.log"

    # auto-sourcing
    set flist [concat [glob $JANUSHOME/gui-tcl/*.tcl] [glob $JANUSHOME/tcl-lib/*.tcl]]
    foreach f $flist {
if [string match "*makeIndex*" $f] continue
if [string match "*JRTk*" $f]      continue
if [string match "*test*" $f]      continue
catch {source $f}
    }
    catch { rename exit exit-org }
    proc exit { args } {
global GLOBALFP
if [info exists GLOBALFP] { close $GLOBALFP }
exit-org
    }

    catch { rename puts puts-org }
    proc puts { args } {
global LOGFILE GLOBALFP
set argc [llength $args]
if {! [info exists LOGFILE] } {
    return [eval "puts-org $args"]
}
if {! [info exists GLOBALFP]} { set GLOBALFP [open $LOGFILE w] }
set fp $GLOBALFP
if {"-nonewline" == [lindex $args 0]} {
    if {$argc == 3 } { set fp [lindex $args 1] }
    if {$fp == "stdout" || $fp == "stderr"} { set fp $GLOBALFP}
    puts-org -nonewline $fp [lindex $args end]
} else {
    if {$argc == 2} { set fp [lindex $args 0] }
    if {$fp == "stdout" || $fp == "stderr"} { set fp $GLOBALFP}
    puts-org $fp [lindex $args end]
}
return
    }

    catch { rename exec exec-org }
    proc exec { args } {
global LOGFILE
        set cmd  [lindex $args 0]
        set opts [lrange $args 1 end]
set cmdX [lsearch [list touch rm mkdir touch date] $cmd]
```

```
if { $cmdX >= 0} { return [eval "$args"] }

if { [catch {set res [eval exec-org $args]} msg] } {
    # write error message to log file
    if [info exists LOGFILE] {
puts "ERROR pseudo-exec: \n called '$args' \n and got \n '$msg'\n"
    }
    error "ERROR pseudo-exec: \n called '$args' \n and got \n '$msg'\n"
} else {
    return $res
}
    }
}

# ----------------------------------------------------------------
# Unix dependent settings
#  - socket based redefinitions of fgets and ngets
#  - define socket host and port number
#  - start NGETS server via tcl-lib/ngetGUI.tcl
#
# ----------------------------------------------------------------

if {0 && ![regexp {indows} $tcl_platform(os)]} {
    if {![info exists NGETS(HOST)]} {
set NGETS(HOST)         islpc13
set NGETS(PORT)         63060
set NGETS(VERBOSE)      1
set NGETS(MGETS)        0

catch {
    regexp {uid=(\d+)} [exec id] dummy NGETS(PORT)
    set NGETS(PORT) [expr $NGETS(PORT) + 52000]
    unset dummy
}
    }

    if {[regexp "^isl" $env(HOST)] && [string length $NGETS(HOST)] &&
[string compare $env(HOST) $NGETS(HOST)]} {

set NGETS(STARTUP) "using ngets: $NGETS(HOST):$NGETS(PORT)"

# -------------------------------------
#   FGETS from server
# -------------------------------------
catch {rename fgets fgets-org}
proc fgets {file line_ } {
    upvar  $line_ line
    global NGETS

    if {[file pathtype $file] == "relative"} {
set file "[pwd]/$file"
    }
    regsub -all "^/net"    $file ""         file
    regsub -all "^/export" $file "/project" file

    return [ngets $file line]
}

# ------------------------------------
#   GLOB from server, too
# ------------------------------------
catch {rename glob glob-org}
proc glob { args } {
    global NGETS

    set line ""
    set nc   [regsub -- "-nocomplain " $args "" args]
    regsub -- "--" $args "" args

    foreach f $args {
set rel 0
if {[file pathtype $f] == "relative"} {
    set f [file join [pwd] $f]
    set rel 1
}

# Strip '/net' from filenames
regsub -all "^/net" $f "" f

# Local filesystems don't need nglob
if {[regexp "^/export" $f] || [regexp "^/tmp" $f]} {
    set tmp [glob-org -nocomplain -- $f]
} else {
    set tmp [nglob $f]
}
if {$rel} {regsub -all " [pwd]/" " $tmp" " " tmp}
append line [string trim $tmp]
    }

    return $line
}
    }
}

# ----------------------------------------------------------------------
#  Set the audio device for featshow.
# ----------------------------------------------------------------------

switch {tcl_platform(os)} {
    SunOS {
set DEVICE SUN
#set USERDEVICE {exec aplay -file $filename -g $gaindb -e int}
    }
    Linux {
set DEVICEPLAY(User) {exec sox -q -t raw -r $rate -s -w $filename -t ossdsp -s -w /dev/dsp}
    }
}
```

```
# -------------------------------------------------------
#  General stuff
# -------------------------------------------------------

proc general_info {} {
   global tcl_platform tcl_version tk_version tcl_precision
   catch {puts "machine: $tcl_platform(machine) \
                         $tcl_platform(os) \
                         $tcl_platform(osVersion)"}
   catch {puts "tcl $tcl_version"}
   catch {puts "tk $tk_version"}
   catch {puts "tcl_precision: $tcl_precision"}
}

proc writeJanusLog msg {
    global env
    puts stdout $msg
    flush stdout
}

catch { randomInit [pid] }

if {!$tcl_interactive} {
    set clicksatstart [clock clicks -milliseconds]
    catch { rename exit exit-org }
    proc exit { args } {
global clicksatstart env
set wt [expr .001*([clock clicks -milliseconds]-$clicksatstart)]
puts stderr "ended [info nameofexecutable]: $env(HOST).[pid], [clock format [clock seconds]], s
exit-org
    }
}

# -------------------------------------------------------
#  print start-up message
# -------------------------------------------------------

if {!$tcl_interactive} {
    puts stderr "started [info nameofexecutable]: $env(HOST).[pid], [clock format [clock secon
    #puts stderr "script: [info script] directory [pwd]"
    puts stderr "library: $auto_path"
}

if {[info exists NGETS(STARTUP)]} {
    puts $NGETS(STARTUP)
}
```

It is read by JANUS at start-up. You'll then have to set your environment variables correctly. Just for reference, my `.tcshrc` contains the following Janus-related entries:

```
# For Janus:
setenv JANUS_LIBRARY $HOME/janus/library
setenv TCL_LIBRARY   /usr/lib/tcl8.3
setenv TK_LIBRARY    /usr/lib/tk8.3
```

```
# Compiling:
setenv IA32ROOT            /home/njd/intel/compiler60/ia32
setenv LD_LIBRARY_PATH     ${IA32ROOT}/lib
```

For Windows, you should set the following environment variables, if not already specified:

```
# take care of the '/' and '\'
HOME          C:\user\fuegen
JANUS_LIBRARY C:/user/fuegen/janus/library
```

## 10.10 phonesSet

The phones that can be used. An example looks like:

```
PHONES     @ A AR AEH AEHR AH AHR AI AU B CH X D E E2 EH EHR ER ER2 EU F G
           I IR IE IHR J K L M N NG O OR OE OEH ANG OH OHR P R S SCH T TS
           TSCH U UR UE UEH UEHR UH UHR V Z SIL +QK +hBR +hEH +hEM +hGH
           +hHM +hLG +hSM +nGN +nKL +nMK
SILENCES   SIL
NOISES     +QK +hBR +hEH +hEM +hGH +hHM +hLG +hSM +nGN +nKL +nMK
AFFRIKATE TS TSCH
VOICED     M N NG L R A AEH AH E E2 EH ER2 I IE O OE OEH ANG OH U UE UEH UH
```

The first item in each line is the name of a "group" of phones in the set, while the remaining items are phones. "PHONES" should contain all phones. Here, "VOICED" is used for VTLN. "AFFRIKATE" and "VOICED", "NOISES" and "SILENCES" can be used as questions during context clustering. "@" is the pad-phone, which is used whenever there is no context available.

## 10.11 ptreeSet

Used to define polyphone trees. An example looks like this:

```
; -------------------------------------------------------
;  Name            : distribTreeISLci
;  Type            : PTreeSet
;  Date            : Thu Jul 11 23:18:06 CEST 2002
; -------------------------------------------------------
+QK-b {+QK} 0 0 -count 1.000000 -model +QK-b
+QK-m {+QK} 0 0 -count 1.000000 -model +QK-m
+QK-e {+QK} 0 0 -count 1.000000 -model +QK-e
...
```

## 10.12 svocab

A SVocab description file. It contains a list of words which should also be contained in the dictionary.

An example looks like this:

```
$ 1
(
)
Anne
Anne(2)
```

The “1” in the first line declares “$” to be a filler-word, i.e. a word which is not handled by the language model. Instead, the -filPen is added for every transition into this word. “(“ and “)” are the begin-of-sentence and end-of-sentence words.

## 10.13 tags

A Tags description file. It contains the modifiers for phones that can be used in the Dictionary.

An examples looks like:

```
WB
```

## 10.14 tmSet

The transition set description file. An example looks like:

```
SIL  { { 0  0.01 } { 1  0.0 } }
1    { { 0  0.01 } { 1  0.0 } }
3    { { 0  0.01 } { 1  0.0 } { 2  0.015 } }
```

The Tcl-list contains the distance to transition (so “0” is a self-loop) and the score for this transition.

## 10.15 topoSet

The description file for a TopoSet:

An example looks like this:

```
6state  { ROOT-b ROOT-b ROOT-m ROOT-m ROOT-e ROOT-e } { 3 3 3 3 3 3 }
3state  { ROOT-b ROOT-m ROOT-e } { 1 1 1 }
SIL     { ROOT-m ROOT-m ROOT-m ROOT-m } { 1 1 1 1 }
```

The second colum defines the root-node for the model tree, while the second column defines the transition to use from the TmSet.

## 10.16 topoTree

The description file for the topology tree, which can be read in a in the Tree object.

An example looks like this:

```
ROOT    { 0=SIL } 6state SIL     -      -
6state  {       }    -    -      -      3state
SIL     {       }    -    -      -      SIL
```

It defines the topologies to use for different phones, defined by the question in the second column (standard tree answer format: “no, yes, don’t-know, leaf” for colums 3-6).

## 10.17 `db-spk, db-utt`

Janus contains a database object which stores all the information needed for a particular system. An example script to generate such a dbase is available in `~/janus/scripts/genDBase.tcl`.

The database consists of two parts, each of which is store in a data-file (*.dat) and an index file (*.idx):

**db-spk** The "speaker database"

Every entry in this database (corresponding to a line in the file) contains information for one "speaker". It should contain a field "UTTS", which lists all the utterances (segments) which belong to this speaker. Also, paths to ADC files, speaker information or warp factors can be stored here.

**db-utt** The "utterance database"

Every entry in this database (corresponding to a line in the file) contains information for one "utterance". It should contain a field "SPK", which links to the corresponding entry in the speaker database, a field "UTT", which repeats the utterance id and further information (transliteration: "TEXT", ADC segment, ...)

Look at `~/janus/scripts/genDBase.tcl` to see how these files can be generated from free-format data.

# Chapter 11

# Maintainers

This is a list of people to contact with questions about the JANUS project (V5.x):

**Sebastian Stüker** Mail: sebastian.stueker@kit.edu
Telephone: +49 (721) 608-6284

**Florian Metze** Mail: fmetze@cs.cmu.edu
Telephone: +1 (412) 268-8984

The standard procedure for asking questions and reporting problems is sending e-mail to jrtk@ira.uka.de.

The *Ibis* decoder and this documentation was mainly written by Christian Fügen, Florian Metze, and Hagen Soltau. *Janus* was mainly developed by Michael Finke, Jürgen Fritsch, Christian Fügen, Hermann Hild, Thomas Kemp, Florian Metze, Klaus Ries, Ivica Rogina, Thomas Schaaf, Hagen Soltau, Martin Westphal, Matthias Wölfel, Monika Woszczyna, Hua Yu, and Torsten Zeppenfeld.

# Bibliography

[1] Michael Finke, Jürgen Fritsch, Petra Geutner, Klaus Ries, Thorsten Zeppenfeld, and Alex Waibel. The JanusRTk Switchboard/ Callhome 1997 Evaluation System. In *Proceedings of LVCSR Hub-5E workshop.*, Baltimore, MD, 5 1997.

[2] Michael Finke, Petra Geutner, Herrmann Hild, Thomas Kemp, Klaus Ries, and Martin Westphal. The Karlsruhe Verbmobil Speech Recognition Engine. In *Proc. ICASSP 97*, 1997.

[3] Michael Finke and Alex Waibel. Flexible Transcription Alignment. In *Proc. Automatic Speech Recognition and Understanding Workshop (ASRU)*, 1997.

[4] Jürgen Fritsch and Ivica Rogina. The bucket box intersection (BBI) algorithm for fast approximative evaluation of diagonal mixture Gaussians. In *Proc. ICASSP 1996*, Atlanta; USA, 1996.

[5] Christian Fügen, Hartwig Holzapfel, and Alex Waibel. Tight Coupling of Speech Recognition and Dialog Management – Dialog-Context Dependent Grammar Weighting for Speech Recognition. In *Proc. of ICSLP 2004*, Jeju Island, South Korea, October 2004. ISCA.

[6] Marsal Gavaldà. SOUP: A Parser for Real-world Spontaneous Speech. In *Proc. of the 6th International Workshop on Parsing Technologies (IWPT-2000)*, Trento, Italy, February 2000.

[7] Thomas Kemp and Thomas Schaaf. Estimating confidence using word lattices. In *Proc. EuroSpeech 97*, Rhodes; Greece, 1997.

[8] Florian Metze, Christian Fügen, Yue Pan, Tanja Schultz, and Hua Yu. The ISL RT-04S Meeting Transcription System. In *Proceedings ICASSP 2004 Meeting Recognition Workshop*. NIST, 5 2004.

[9] Mehryar Mohri, Fernando C.N. Pereira, and Michael Riley. The Design Priniciples of a Weighted Finite-State Transducer Library. *Theoretical Computer Science*, 231:17–32, January 2000.

[10] Ivica Rogina. *Parameterraumoptimierung für Diktiersysteme mit unbeschränktem Vokabular*. PhD thesis, Fakultät für Informatik der Universität Karlsruhe (TH), Karlsruhe, Germany, 1997.

[11] Hagen Soltau, Florian Metze, Christian Fügen, and Alex Waibel. A one-pass decoder based on polymorphic linguistic context assignment. In *Proc. ASRU 2001*, Madonna di Campiglio, Italy, 12 2001. IEEE.

[12] Hagen Soltau, Thomas Schaaf, Florian Metze, and Alex Waibel. The ISL Evaluation System for Verbmobil - II. In *Proc. ICASSP 2001*, Salt Lake City, USA, 5 2001.

[13] Hagen Soltau, Hua Yu, Florian Metze, Christian Fügen, Qin Jin, and Szu-Chen Jou. The 2003 ISL Rich Transcription System for Conversational Telephony Speech. In *Proc. ICASSP 2004*, Montreal; Canada, 2004. IEEE.

[14] Andreas Stolcke. SRILM – An Extensible Language Modeling Toolkit. In *Proc. ICSLP 2002*, volume 2, pages 901–904, Denver, Colorado, USA, September 2002. ISCA.

[15] Sun Microsystems, http://java.sun.com/products/java-media/speech/forDevelopers/JSGF. *Java Speech Grammar Format Specification*, version 1.0 edition, October 1998.

[16] Wayne Ward. Understanding Spontaneous Speech: the Phoenix System. In *Proc. of ICASSP 1991*, Toronto, Canada, May 1991. IEEE.

[17] M. Wölfel. Warped-twice minimum variance distortionless response spectral estimation. In *Proc. of EUSIPCO*, 2006.

[18] M. Wölfel. Enhanced speech features by single channel joint compensation of noise and reverberation. *accepted for publication Trans. on ASLP*, 2008.

[19] M. Wölfel and J. McDonough. Minimum variance distortionless response spectral estimation, review and refinements. *IEEE Signal Processing Magazine: Special Issue on Speech Technology and Systems in Human-Machine Communication*, 22(5):117–126, Sept. 2005.

[20] M. Wölfel and J. McDonough. *Distant Speech Recognition*. John Wiley & Sons, 2009.

# Glossary

**AModelSet** set of acoustic models

**AModel** acoustic model

**BBILeaf** leaf in a BBI search tree

**BBINode** node in a BBI search tree

**BBITree** BBI search tree

**BMem** Block Memory

**Cbcfg** configuration of a codebook

**CBNewParMatrixSet** Set of CBNewParMatrix parameter matrices

**CBNewParMatrix** Parameter matrix used by CBNew codebooks

**CBNewSet** Set of CBNew codebooks

**CBNewStream** Stream based on extended codebooks (CBNew)

**CBNew** Codebook using additional parameter matrices

**CFGArc** A 'CFGArc' object is an arc between two nodes of a context free grammar.

**CFGLexiconItem** A 'CFGLexiconItem' object is a item of a CFG lexicon.

**CFGLexicon** A 'CFGLexicon' object is a lexicon of a Context Free Grammar.

**CFGNode** A 'CFGNode' object is a node in a context free grammar.

**CFGParseTree** A 'CFGParseTree' object is a parse tree.

**CFGPTItem** A 'CFGPTItem' object is a item in a parse tree node.

**CFGPTNode** A 'CFGPTNode' object is a node of a parse tree.

**CFGRSItem** A 'CFGRSItem' object is an item in the stack of CFG rules.

**CFGRuleStack** A 'CFGRuleStack' object is a stack of CFG rules.

**CFGRule** A 'CFGRule' object is a rule of a context free grammar.

**CFGSet** A 'CFGSet' object is a set of context free grammar.

**CFG** A 'CFG' object is a context free grammar.

**CMatrix** Matrix of char values

**CMLLR** Constrained MLLR

**CMU** Carnegie Mellon University

**CodebookAccu** a single codebook's accumulator

**CodebookMapItem** CodebookMapItem

**CodebookMap** CodebookMap

**CodebookSet** Set of codebooks

**Codebook** Codebook

**DBaseIdx** DBase Index Object

**DBase** DBase

**DCovMatrix** Covariance matrix type (double)

**Dictionary** Set of words

**DictWord** Word with tagged phone transcription

**DistribAccu** a single distribution's accumulator

**DistribSet** Set of distributions

**DistribStream** Distribution based stream

**Distrib** A single distribution

**DMatrix** Matrix of double values

**Dscfg** configuration of a distribution

**DurationSet** A 'DurationSet' object is an array of explicite duration models.

**Duration** explicite duration model

**DVector** Vector of double values

**FArray** Array of floats

**FBMatrix** Band matrix of float values

**FCovMatrix** Covariance matrix type (float)

**FeatureSet** set of features

**Feature** Feature

**FFLayer** Single Layer in a FFNet

**FFLink** Single link between FFLayer's in a FFNet

**FFNet** General Feed Forward Multilayer Neural Network

**Filter** LTI filter

**FlatFwd** Search: Flat Forward Module

**FMatrix** Matrix of float values

**FMLLR** Feature-Space Maximum Likelihood Linear Regression

**Forced** Search: EXPERIMENTAL Beam Optimization Pass

**FVector** Vector of float values

**GLat** Generic Lattice (pronounced 'Gillette, everything a man ...')

**GSClusterSet** A 'GSClusterSet' object is a cluster set on the Gaussians of a CodebookSet.

**HMM3gramState** HMM3gram State

**HMM3gram** HMM3gram

**HMM** An 'HMM' object contains states, transitions and acoustic references

**HypoList** The object HypoList contains a list of hypotheses.

**Hypo** Hypo is a subtype of HypoList only.

**IArray** Array of integers

**Ibis** The standard one-pass decoder in Janus 5.x.

**IMatrix** Matrix of integer values

**ISL** The Interactive Systems Labs at UKA and CMU

**JANUS** Equivalent to JRTK, or only the janus binary

**Janus** Equivalent to JRTK, sometimes used for pre-Ibis janus binaries

**janus** The 'janus' binary

**JRTk** The Janus Recognition Toolkit

**KeyspotterLP** A keyspotter-lp object

**Keyspotter** A keyspotter object

**Labelbox** Labelbox

**LatNode** Lattice Node

**Lattice** Lattice

**Lat** Lat

**LCMSet** set of left context models

**LCM** left context model

**LDAClass** LDA class

**LDA** LDA

**Lh** a codebook-likelihoods accumulator

**LingKS** Generic Linguistic Knowledge Source:

**List** List of indexed items

**LModelBackoffItem** Language Model Selection

**LModelBackoff** Language Model Selection

**LModelIntItem** Language Model interpolation

**LModelInt** Language Model Intper

**LModelItem** 3G Language Model Item

**LModelLongItem** Mgram Language Model Item

**LModelLong** Mgram Language Model

**LModelMapItem** Language Model mapping

**LModelMap** Language Model Mapper

**LModel** 3G Language Model

**Lm** The object Lm contains a language model.

**LookAhead** LookAhead (fast match) part for tree and flat search

**LTree** Language-Model Look-Ahead object (Lexical tree)

**MAM** Model-Based Acoustic Mapping

**MAP** Maximum A-Posteriori Estimation

**MetaLMElem** Meta language model element (sub-LM).

**MetaLMItem** Meta language model item.

**MetaLM** Meta language model: flexible LM using sub-LMs.

**MicvSet** Set of Mixture of Inverse CoVariances Codebooks

**MicvStream** Stream based on MIC codebooks (Micv)

**MLAdaptItem** MLAdaptItem

**MLAdapt** Maximum Likelihood Adaptation

**MLE** Maximum Likelihood Estimation

**MLLR** Maximum Likelihood Linear Regression

**MMIE** Maximum Mutual Information Estimation

**ModalitySet** A 'ModalitySet' object is a set of modalities.

**Modality** A 'Modality' object answers a question about the modality of a recording.

**ModelArray** Array of models.

**NGramLMItem** N-gram Language Model Item

**NGramLMSubs** N-gram Language Model Substitution Item

**NGramLM** N-gram Language Model

**PathItemList** PathItemList

**PathItem** PathItem

**Path** A 'Path' object is filled by a forced alignment function and is used by training functions

**PHMMSet** set of phone hidden markov models

**PHMM** phone hidden markov model

**PhoneGraph** PhoneGraph

**PhonesSet** A 'PhonesSet' object is a set of 'Phones' objects.

**Phones** A 'Phones' object is an array of strings, each of which is a phoneme.

**Phone** Phone

**PhraseLMItem** Phrase language model item.

**PhraseLM** This module takes a LM and adds phrases (aka. multi-words) to it.

**PTreeNode** PTreeNode

**PTreeSet** A 'PTreeSet' object is a set of polyphone context trees.

**PTree** Polyphonic Tree

**QuestionSet** A 'QuestionSet' object is a set of characteristic function definitions and a set of questionSet.

**Question** A 'Question' object is a definition of a single question.

**RCMSet** set of right context models

**RCM** right context model

**RewriteSet** Set of rewrite rules

**Rewrite** Rewrite Rule

**SampleSetClass** a class in a SampleSet

**SampleSet** containers for samples

**Search** Search Module

**SenoneSet** Set of senones

**SenoneTag** SenoneTag

**Senone** Senone

**SignalAdapt** Signal Adaption

**SMem** Search Memory Manager

**SNode** Search Root

**SPass** Single Pass Decoder

**SRoot** Search Root

**STab** Backpointer table

**StateGraph** StateGraph

**StateTable** A 'StateTable' object is a matrix for looking up distribution indices.

**StreamNormItem** A stream normalizer

**STree** Search Tree

**SVector** Vector of short values

**SVMap** Search Vocabulary Mapper

**SVocab** Search Vocabulary

**SWord** Search Vocabulary Word

**Tags** A 'Tags' object is an array of strings.

**Tag** Tag

**TextGraph** Text Graph

**TmSet** A TmSet is a set of state transition model objects (Tm)

**TopoSet** A 'TopoSet' object is a set of different topologies.

**Topo** A 'Topo' object is a definition of a single topology description.

**TreeFwd** Search: Tree Forward Module

**TreeNode** TreeNode

**Tree** A 'Tree' object is an allophone clustering tree.

**UKA** Universität Karlsruhe (TH)

**Vocab** A Vocab is the list of words the recognizer can recognize

**WordGraph** WordGraph

**WordList** WordList to communicate between tree and flat pass of searches

**Word** Word with tagged phone transcription

**XCMSet** set of left/right context models

**XCM** left and right context model

**XWModelSet** set of blah

**XWModel** blah

# Index