Documentation of MASV

Munich **A**utomatic **S**peaker **V**erification system
Documentation version 1.3.00 (16.02.2004)
Release 1.3 (16.02.2004)

Ulrich Türk

tuerk@phonetik.uni-muenchen.de
Department of Phonetics and Speech Communication
University Munich, Schellingstr. 3, D-80799 München, Germany
http://www.phonetik.uni-muenchen.de

Copyright © 2001-2004 Ulrich Türk. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Contents

Chapter 1

General information

1.1 Overview

MASV stands for "Munich Automatic Speaker Verification". It is an experiment environment for performing speaker verification (SV) that allows easy set up of SV systems using different kinds of models including sub-word-model HMMs and GMMs. Different commonly used types of score normalizations are available: world normalization, simple cohort normalization (mean score of n-best cohort speakers or maximum score of cohort speakers) and handset normalization (h-norm).

It comprises two parts: a Perl script part (based on the HTK tools) which handles training and testing of the system and a Matlab script part, which performs evaluation and performance measurement of the trained system. Information transfer from the Perl part to the Matlab part takes place by simple text files (MLF files of HTK, simple configuration files with lists of speakers used for training, no. of iterations for a model, etc.).

The whole system is developed for the German VeriDat Speaker Verification database and the general file structure of this database must be taken over when using other datases. The adaptation process is now facilitated as all database specific functions and documentation files are packed into an own directory.

Unfortunately the VeriDat database is not freely available. In order to help you getting started there is a dummy database for download: it contains only the label files of one of the speakers and empty speech data files. This should give you some idea about the structure and should help you in adapting the system to your own databases.

Please note that there are many things to improve and I would appreciate any kind of comments and advises for further improvement!

The latest version of MASV-package and also of this document is available from http://www.phonetik.uni-muenchen.de/Bas/SV.

1.2 Notation

In the following descriptions, all references to programs, command line entries, paths, filenames and values of parameters are set in a `monospace style`.

After installation, a set of environment variables is defined which can be accessed in the shell. They are also used for notating paths in this document. These values are referred by `ENV::nameOfTheVariable`, where `nameOfTheVariable` is replaced by a selected name of the environment variables.

Some directories are common for all SV systems designed with MASV. These are referred to with the relative path starting in the so-called base directory `ENV::MASV_BASEDIR`. E.g. the directory `hmm_lists` there is referred to by just notating `hmm_lists/`.

If a part of a command string uses a value of a variable, the value is enclosed by < and >. E.g. if a tool takes the parameter `pool_name` and a filename consists of `C_` followed by the value of the parameter, the notation is `C_<pool_name>`. The value `<sv system path>` has a special meaning which refers to the path of current SV system (most of the tools take the name of the SV system as a required parameter).

1.3 Changes

Changes in 1.3

- cleared up terminology: the previously called "experiment" is now called a "SV system", the "sub experiments" are now called "experiments".
- matlab result files need about half the disc space now (converted the `rec_info` field now from a string cell array to a normal string array).
- introduced concept of "database description", making adapting to other databases more easier.

Changes in 1.2

- included documentation.
- reorganized and cleaned up the code.
- better handling of environment variables; more tuning via these variables possible (verbose level, nice factor, ...)

Changes in 1.1

- first public release

Chapter 2

Requirements

MASV needs Perl of version 5 or higher and Matlab (version 5 or higher). Some parts of the code may run only on UNIX compatible operation systems. Many things are already handled by Perl (path delimiter sign, environment variables, etc.) but there are still many calls to UNIX commands like e.g. ssh, cp, ln. Be aware that you may encounter problems when trying to run the system in a non-UNIX flavoured environment.
The Perl part of MASV uses the package `Parallel::ForkManager` made by Balázs Szabó. You can use the module file provided with the MASV tar ball or download it from CPAN (http://www.cpan.org).
All shell scripts use the Bash shell, so please make sure that it is installed on your system.
Some rarely used tools require the SoX package (available from http://sox.sourceforge.net).
MASV uses HTK (http://htk.eng.cam.ac.uk) for training and testing the modules. It is developed to work with version 3.1 of the HTK tools but newer version should be fine. There are some patches available (see the instructions on the MASV home page) which arose from the development of MASV. You should at least install the patch for the HTK command `HHEd` that provides a new mechanism for splitting mixtures in a HMM model.
There are two Matlab packages for MASV (`voicebox` by Mike Brookes and `matdraw` by Keith Rogers) which are used by some functions of the Matlab part of MASV. The `matdraw` package was slightly extended. You can download the archive from the MASV home page (http://www.phonetik.uni-muenchen.de/Bas/SV). The installation is quite similar to the procedure described in 3.1.2. Please make sure that the directories of both packages are in your Matlab path!

Chapter 3

Installation

The installation consists of two main steps:

- Unpacking and installing the scripts and Matlab files.
- Setting up some environment variables e.g. in your rc files of your home directory when you are on a UNIX compatible system.

3.1 Unpacking and installing

3.1.1 Perl scripts

1. Unpack the archive in a temporary directory:

   ```
   tar -xzf MASV_1.3.tgz
   ```

2. Move the directory `MASV_pl/` to a place suitable for you. This directory contains the Perl scripts of MASV. Make sure, that this new location in your PATH environment variable (modify it e.g. in your `.profile` or `.bashrc` when your are on a UNIX compatible system).

3. Copy the content of the file `env.linux` to your rc files. E.g. on UNIX systems with bash as shell paste the content to the end of your `.profile` or `.bashrc` in your home directory. The settings will be completed in the next steps; to give you some idea, the settings on our machine are included in the comments.

4. Set up the environment variable named `MASV_PERL_ROOT` with the directory of your Perl scripts directory.

5. Make sure that you have the Perl modul `ForkManager` installed. You can either install it using the CPAN shell of Perl or you just copy the directory `Parallel/` from the unpacked archive to your directory with the Perl scripts (`ENV::MASV_PERL_ROOT`). The module is required even if you don't use the parallel computing feature.

6. make sure that your Perl installation has the following modules:

   - `Pod::Usage`
   - `Getopt::Long`

   These are normally installed by default.

7. Download the archive `dummy_HTK_Rel-1.3.tgz` archive and unpack it, where you want to store the data of your SV systems. My directory e.g. is called `HTK/` (instead of `dummy_HTK/`). Set up the environment variable `MASV_BASEDIR` pointing to the recently created directory.

8. Create a directory in <ENV::MASV_BASEDIR>database_desc/) and name it according your database. It contains a Perl module and some Matlab functions which are specific for the database. You can create several directories there, for each of your databases one. Move the Perl module MASV_db_desc.pm from the directory <ENV::MASV_PERL_ROOT> to your database specific directory or directories. We will later see, how to adapt the Perl module.

9. Set up the remaining environment variables:

   - MASV_PERL_HTKBIN: path to your HTK Tools binaries
   - MASV_SOURCE_DATABASE: path to your database directory (or to the dummy Veridat directory, if you want to test it).
   - MASV_PARAMPOOL_DIR: path to your directory containing the parameter pools (create a directory yourself on a fast device, especially if you use the parallel computation feature!).
   - MASV_MATLAB_HOME_DIR: path to your personal Matlab directory (normally matlab/ in your home directory).
   - MASV_MATLAB_HOST: hostname of machine doing Matlab computations (for large experiments a machine with 512MB and more memory is recommended); localhost can also be used.
   - MASV_MATLAB_RESULTSDIR: path to a directory for storing result files generated by Matlab.
   - MASV_HOSTLIST: path to file with list of hosts for computation (see 3.2.1).
   - MASV_HOST_INFOS: path to file with speed information about the hosts.
   - MASV_NICE_FACTOR: nice level for processes during parallel computation.
   - MASV_VERBOSE_LEVEL: verbose level for the Perl tools (0: quiet, 1:some information, 2: print HTK tool calls).
   - MASV_DATABASE_DESCRIPTION: path to a database specific directory in <ENV::MASV_BASEDIR>database_desc/.

10. To test the configuration, reload your rc-files (e.g. in the bash, type "source /.bashrc") and simply call one of the tools without any pathname (e.g. type "run_HCompV.pl"). You should see the help text of the tool.

3.1.2 Matlab scripts

1. Move the directory MASV_m to place where you normally have all your Matlab m-files (usually in /matlab/M_files/). Add this directory (and also, very important, all subdirectories except for MASV_db_dep/) to your Matlab path. On UNIX systems, you may prefer to adapt your startup.m file and add lines like e.g.

   ```
   path(path,'/homes/tuerk/matlab/M_files/MASV_m/');
   ```

2. Please make sure that in your Matlab path, the directory voicebox_mod/ is before the original directory of the voicebox package. This ensures, that the extended version of the function readhtk is used.

3. Move all four Matlab functions in the directory MASV_db_dep/ of MASV_m/ to the database specific directory (or directories) that you created in step 8 in 3.1.1.

4. Make sure that you have created your Matlab directory to which ENV::MASV_MATLAB_HOME_DIR is pointing to (if it is not already there).

5. In case that you don't work with the VeriDat database, you can download a dummy data base (dummy_VD.tgz). Put this directory where you want to store the speech data (in this case: pseudo-speech data). Make sure, that ENV::MASV_SOURCE_DATABASE is pointing to this directory.

3.2 Additional setup

3.2.1 Setting up the hosts list

- create a simple text file with the host names of all your machines available for parallel computing; one host name in each line. Make sure that `ENV::MASV_HOSTLIST` is pointing to this file.
- make a copy of the first file and append the speed of each machine (bogo mips, available under Linux from the file `/proc/cpuinfo`), seperated by white space (tab, or space).

3.2.2 Adapting database specific functions

Handling of the format of the speech data and the label data is done via a collection of Perl subroutines and Matlab functions in the directories in `database_desc`. After installation, these functions are by default adapted to the VeriDat database. Change them to according to your database.

Common file structure

As already mentioned the structure the VeriDat database is used as guideline for the databases used with the MASV environment. You have to adopt your databases to the following characteristics:

- Each speaker performs the same number of sessions.
- The same session number of all speakers corresponds to the same defined recording setup like e.g. environment (quiet, noisy) or handset type.
- The label and audio files use the same file name basis (eight characters) and a distinctive file extension (three characters). Base file name and extension are seperated with a dot (“.”).
- The last two characters of the base file name specify the recording (type of item, repetition counter in the session).

Perl module `MASV_db_desc.pm`

This Perl module contains three subroutines and several constants used in the Perl tools.
Subroutines:

- `get_SessionsListName`
  This function translates the session part of a recording configuration (part after the slash) to a name of a script file in a speaker’s directory. The default functions defines several two-letter codes which are aliases to often used script files. The function must return a string with the name of the script file.
- `read_db_label`
  This function reads the a label file and returns two references to arrays: one containing the prompted words, one the labeling of the utterances. These arrays are further processed later with the following function.
- `convert_prompt_string`
  In case that you are using subword models (not a GMM recognizer), the prompt labels have to be split into subwords when generation the MLF files containing the prompts. This function takes a string (as prompted) and converts it to sub words. An additional function (`writeOutNumbers`) converts digits to expanded number words. The subwords are stored in an array whose reference is returned.
- `convert_label_string`
  This subroutine takes a string (the transcription of a utterance) and converts to a string with all words seperated by newlines. The result is pasted to the MLF files containing the transcription.

Variables:

- `audio_type`
  define a type which will be by `run_HCopy.pl` and the Matlab function `play_sound_file.m`
- `sampling_freq`
  set here the sampling frequency in Hertz
- `label_file_ext`
  set the file extension of your label files here.
- `audio_file_ext`
  set the file extension of your audio files here.
- `sessions_lists`
  define several subsets of your sessions (e.g. specifying the recording environments). These lists can be used to perform a hnorm normalization or evaluate performance on subsets of the recordings. The labels (i.e. the key names) of this hash and the two hashes described in the next paragraph are used by the environment lists of the Perl tool `create_MASV_sv_system.pl`. Define at least the label `all` for the three hashes.
- `training_sessions_lists`, `evaluate_sessions_lists`
  define here various subsets of sessions. There must be a corresponding set of test (evaluation) sessions for a defined training sessions set.
- `all_sessions`
  define in here all sessions of a speaker.

Several comments in the file indicate the part which is also read by the Matlab functions.

Matlab functions

- `get_model_translation.m`
  the model names are stored in the Matlab result files by integer codes (one byte, range 0 to 255). The mapping of the model name, an associated color for drawing the segmentation and the integer code is stored in this file.
- `get_sessions_4_hnorm_type.m`
  this function returns a cell array with two lists of sessions for a given split type for performing hnorm normalization. The hnorm type is defined here, the name of the sesssions subsets are defined in `MASV_db_desc.pm`.
- `play_sound_file.m`
  converts a soundfile according to its type and plays it. Optionally only a part of the file can be played (given by a time struct, which specifies start and end time in seconds).
- `read_label_file.m`
  reads a label file a returns a string of the prompted text and a string of the labeled utterance.

Chapter 4

System description

As mentioned in the first chapter, the MASV system comes in two parts. The first part is a collection of Perl scripts which act as wrapper script for several HTK command line tools. These scripts are used to create a setup for a selected type of speaker verification system (SV system), train the necessary models and test the system. The result at the end of this process is a collection of MLF files and some information files about the training/testing process (called "info files" in this documentation). There are two kinds of Perl tools: the high level tools which allow an easy configuration of a SV system and the low level tools which are called by the shell script generated by the high level tools.
The high level part provides some predefined types of SV systems which can be easily configured. If you like to create your genuine SV design you can either tweak the shell scripts or create your own sequence of low level tool calls.
The low level part consists of several wrapper scripts for the HTK tools. Beside preparing options and script files for the HTK commands, they provide the mechanism for parallel computing on several hosts, if appropriate.
The second part of MASV is a bunch of Matlab functions. Some of the high level functions perform further processing of the data produced in the first step. Here the MLF files and the info files are read and the performance of the system is computed. The final results are stored in a special Matlab data type, a structure, which is stored in a file. In fact, there are three different types of structures for the tests of the designed SV system with varying level of detail.
The interface between the Perl part and the Matlab part is described in more detail in 6.1.3.
The common way for running experiments is (after installation of the system):

1. Prepare the database and create one (or more, if desired) parameter pools. The parameter pools store the features extracted from the speech material which are later used to train models (individual models for each registered speaker but also models for normalization techniques). `create_MASV_paramPool.pl` is the tool designed for generating the directory structure of a parameter pool and `run_HCopy.pl` is used to perform the feature extraction.
2. Set up a so-called "SV system". This is a kind of a complete entity were different setups of a SV system can be tested and evaluated. It is possible to have only one SV system (which means also having only one SV system directory), but if you like to prepare and test a setup while another SV system is currently running, you might want to use more than one SV system. Note that especially if you do not use the feature of parallel computing the SV systems on several machines, you will have quite long processing times (up to some thousand minutes depending on your amount of speech data and the speed of your machine). The tool designed for this step is `create_MASV_sv_system.pl`.

   Each SV system has two attributes: the associated parameter pool and the selected speaker set (describing the partitioning of speakers to different sets (registered speakers, impostor speakers, speakers for world model, cohort speakers, ...). These attributes apply to all scripts running in the context of an exeperiment and can be easily changed with the tool `switch_MASV_speaker_data_set.pl`.
3. Create a smaller unit inside the SV system, a so-called "experiment". Only one experiment can run at the time within a SV system. The configuration of such an

experiment is facilitated by a text based configuration file, the experiment configuration (abbrev. expConf). In an experiment all relevant parameters for setting up a certain SV system, selection of training and testing material are stored within a single configuration file. This expConf file can be used by several tools which allow to check the consistency of the parameters (`check_MASV_experiment.pl`), calculate the size (`calc_size_MASV_experiment.pl`), create a shell script for performing the whole process from training to the evaluation (`run_MASV_experiment.pl`) and clean temporary files after evaluation (`make_clean_MASV_experiment.pl`).

4. If desired, make additional changes to the shell script like adding extra options to the tool calls.
5. Finally the designed SV system is started by firing up the shell script created from an expConf file.
6. After processing the results can be analyzed with the provided Matlab tools (mainly by the function `Plot_SR_parameters`).

The main features of the system are

- all HMM types (including GMM) provided by HTK can be used.
- speaker lists allow easy selection of different speaker sets sets (e.g. registered speakers, impostor speakers, speakers for world model, etc.).
- parameter pool can be changed easily.
- easy selection of the speech material used for training and testing.
- various possibilities of seeding the models before training.
- predefined types of modelling (HMMs based on word/subwords or GMMs)
- predefined ways of score normalization (world model, cohort speakers, handset normalization (h-norm)).
- option to split speech material for training seperate models and testing under matched / mismatched conditions.
- sophisticated evaluation tool with Matlab GUI

The following chapters will give you more details for the way from setting up a SV system to analyzing its performance.

Chapter 5

Setting up and running SV systems

5.1 Creating parameter pools

A parameter pool is basically a directory containing parameterized speech files (parameter files). These files with the filename extension `.param` store the feature vectors in the standard HTK parameter format. The parameter pool contains a directory for each speaker and these directories themselves contain a directory for each session of the speaker. At the lowest level, the session level, the parameter files are stored. The files themselves use eight characters for the name; the last two characters specify the type of the recorded item. E.g. a sample file name would be `C1F001P1.param`; the identifier `P1` stands for the first item of the triple numbers of the VeriDat database. We recommend using a similar structure as several tools including the Matlab scripts require the two character identifier at that position.
Different parameter pools can be used later in a SV system; the tool `switch_MASV_spk_data_set.pl` allows to set another parameter pool for a SV system. Several SV systems can refer to the same parameter pool. There are three steps to create a parameter pool:

1. First use the tool `create_MASV_paramPool.pl` to create the basic structure of the pool. Here you can specify the name of the parameter pool, the list of speakers used in the pool and the type of sessions used in the pool. A new directory with the name of the parameter pool is created in the parameter pools directory (as specified in `ENV::MASV_PARAMPOOL_DIR`).

2. In the second step, create a HTK config file for the HTK tool `HCopy` in the subdirectory `configs/` of your new parameter pool. As default name for this file you should use `C_<name_of_parameter_pool>`. By this way, other tools can find the config file automatically (otherwise you have to specify the name when invoking `run_HCopy.pl`). For the syntax of the configuration file, see the HTK documentation. An example is provided in the `templates/` directory.

3. The final step involves the generation of the parameter files in this new directory structure. The HTK tool `HCopy` is wrapped in a more comfortable script named `run_HCopy.pl`. Here you normally specify only the name of the parameter pool. There are more parameters when you chose to use different script files or different config files. You have also an option for parallel processing of the parameter files.

5.2 Creating SV systems

A SV system is defined as a sub directory in the `sv_systems/` directory in `ENV::MASV_BASEDIR`. It is a kind of sandbox, where all data for training and testing of the SV system is stored. You can have several SV systems in parallel and you can even run SV systems in parallel; they do not influence each other. However, please note that the so-called experiments of a given SV system share some data within your SV system directory, namely the speaker set used, the parameter pool used and some temporary files. Therefore, do not run experiments of the same SV system in parallel!!
The SV system contains sub directories for each speaker used in the system. These directories store the models (in seperate folders) and the MLF files when performing tests with the models. Other directories are:

- `info/`: contains `.info` files about the completed tests done with `run_HVite.pl`. They provide information which models from which speaker were tested with what kind of material.
- `links/`: contains two links, one pointing the current parameter pool, the other pointing to the current speaker set.
- `log/`: log files for the processed experiments
- protoconfs/: contains config files for creating HMM prototypes with the tool MakeProtoHMMSet.
- `speakersets/`: contains different speaker sets including a standard set (a copy of <ENV::MASV_BASEDIR>/templates/speakersets..
- `speaker_set_lists/`: lists of speaker sets which can be used to evaluate performance on several speaker sets. These lists are e.g. used by `loop_thru_speakersets.pl`.
- `scripts/`: this directory can be used to store shell scripts which perform a complete run (including training, testing and evaluation with Matlab).
- `tmp/`: temporary files, don't modify them during a running experiment!
- `world/`: world models and template alignments with the world model are stored here; the names of the directories are specified in the experiment configuration. Also test results of the world model (e.g. when finding the best version of a world model) go here.
- `expConfigs/`: config files for generating the shell scripts are stored here.

The tool create_MASV_sv_system.pl needs only two parameters: the name of the new SV system and the poolname, which should be initially used. Several options allow to restrict the kind of recordings used; all restrictions can only be done from the contents tha the parameter pool provides (you can't e.g. include more speakers in the SV system than there are in your parameter pool). The restriction can be done in the domain of speakers, the type of recordings and the type of environments (a configuration given in source file `SR_lib.pm` which declares a combination of recording sessions and defines the training and test material). All three options can take a comma seperated list of values.
The recordings can even be filtered on base of the transcription: the option `-i` allows to skip all recordings with labels for transient or static noise. The option `-m` generates MLF files in the SV system directory. These files can be transformed later to match the requirements for training/testing.

5.3 Creating HMM prototyes

The tool `MakeProtoHMMSet` is mainly based on the homonymous script from the HTKDemo. It uses two parameters: the name of the prototype config file (given with relative path, starting in <`sv system path`>`/protoconfs`) and the name of the SV system. The directory with the final prototypes is created in the SV system directory with the name given in the prototype config file. Note that the original format used by `MakeProtoHMMSet` has been slighly changed: the HMM list is searched now in `hmm_lists/`. A sample prototype config file can be found in the base directory in the `templates/` folder.

5.4 Creating experiments

There are three tools which support you to generate an experiment script. All of them need an experiment configuration file, which is expected to reside in the directory <`sv system path`>`/expConfigs/` (you only specify the relative path to the config file). `check_MASV_experiment.pl` checks the configuration file and makes tests if all necessary list files and data for the experiment can be read and accessed. `calc_size_MASV_experiment.pl` checks if the required disc space is available for the experiment. Please note, that still this tool needs the name of the experiment config file via the option `-expConfig` (making this option not really optional ;-)).
`run_MASV_experiment.pl` creates a shell script from the config file. Some slight modification to the generated script can be made via options: you can modify the name of the generated script, skip the training phase and switch the performance calculation with Matlab on and off. Normally you will use

the configuration file for these options. But you also have the option to activate or deactivate some parts of the processing with switches defined in the experiment script.
As a starting point for your own experiments configurations have a look in the `templates/` directory.

5.5 Description of the experiment config files

The config files are simple Perl code snippets and are used to define all parameters of a speaker verification experiment. Make sure that you don't have any errors in the code! Every code line needs a semicolon at the end. You can use the tool `check_MASV_experiment.pl` or `"perl -c <expConfigFile>"` to check the syntax. Comments (lines starting with a `#`) can be inserted everywhere. The line order does not matter, but the example file in `templates/` describes the parameters in a logical sequence of steps from training to testing. The given name of SV system directory is written to the experiment script when `run_MASV_experiment.pl` actually creates the shell script. Thus, the experiment config files are very flexible and can be reused to generate experiment scripts for other experiments.

- `$createdScript`
  name of the generated batch script. If there is no path given, the file will be generated in the current directory.
- `$hmm_list`
  name of the list file in the directory `hmm_lists/`. All the HMM models listed in this file will be used during training/testing.
- `$dict`
  name of dictionary file (in the directory `syntax/`), used e.g. during world model test.
- `$protoDir`
  the name of the prototype directory (relative path from the SV system directory). The prototype directory must contain a HMM file for each entry in the corresponding HMM list given with the parameter above!.
- `$useGMM`
  set this to "1" when using a GMM (only one HMM model with one state for all speech frames). This needs a slightly different setup when testing.
- `$mixtures`
  no of mixtures per state. The mixtures of the prototype models are split right after the initial estimation (`HCompV/HInit`).
- `$parallelComputation`
  set this to "1" to use several computers in parallel (useful especially during training of the models and testing).
- `$minVar`
  set minimum variance; if empty: default is 0.001. This option corresponds with the option `-v` of the HTK tools.
- `$mlf_training`
  used MLF file during training. This option corresponds with the option `-I` of the HTK tools.
- `$seed_world`
  path to directory (starting in the SV system directory) containing the base models for training of the world model. If set to `""`, the intial models are generated with `run_HCompV.pl`.
- `$num_iterations_world`
  number of training iterations for the world model.
- `@world_list`
  array of training configurations for the world model (see description 7.2). If you specify more than one configuration, you have to give the same amount of values to the parameters `@hmmBaseDirWorld` and `@world_test_list`. This method allows you to train more than one world model with different sets of speech files (e.g. noisy vs. quiet environment).

- `@hmmBaseDirWorld`
  name(s) for the world models. A directory with this name is generated in the path <`sv system path`>`/world/`.

- `@world_test_list`
  test configuration for testing the world model. Normally you will specify your development set. The best world model is selected by performing a speech recognition test (using `HVite` running with the defined language model) with all world model version on the material of the given testing configuration.

- `$makeCohortNorm`
  use cohort normalization instead of world normalization. Experimental state!

- `$doHNorm` use additional information about used handset (from labeling) or recognized handset (from handset detector) and perform h-norm.

- `$seed_models`
  base models for starting training of speaker models. Leave empty (`""` for complete initialization with HCompV; use `"bestWorld"` for seeding with best world model, otherwise use a directory (starting at the SV system directory level) containing the seed models.

- `$num_iterations_model`
  num of training iterations; use "" for same amount of training iterations as used in world model training.

- `@trainingModelConf`
  array of training configuration(s) for the speaker models. Similar to the world model, you can build up several different models per speaker. The number of model names in the parameter `@hmmBaseDirModel` has to match with the number of training configurations.

- `@hmmBaseDirModel`
  name(s) of the speaker models. A directory for each of these names is generated in the speaker directory.

- `$mlf_test`
  MLF for testing the models (used in `HVite` as MLF given with option `-I` when doing force alignment with option `-a`).

- `@HViteTrainedModelConf`
  test configurations for the genuine tests (speaker of speech material and speaker model are matching). You can use more than one entry when you have trained several models.

- `@HViteWorldConf`
  test configuration for the test of speech material with the world model. Normally you will specify all the material that you use for genuine tests and imposter tests (the world normalization is needed for both kinds of tests).

- `@HViteCrossTestConf`
  test configuration for the imposter tests (speaker of speech material and speaker model do not match).

- `$hmm_version_world_test`
  specify the version of the world model to be used for world tests. Leave empty(`""`) in case you want the best world model determined before.

- `$hmm_version_model_test`
  specify the version of the speaker model to be used for genuine tests. Leave empty (`""`) in case you want to use the same version as the best world model.

- `$FA_id`
  this string is used as a part of the file name for all MLF files generated during the force aligned recognition with `HVite`. Use different names here for your experiments when you want to keep the files from several different experiments. Otherwise, the files are overwritten.

- `$startMatlab`
  set this to "1" if you want to do the performance evaluation right after training and testing. When executing `run_MASV_experiment.pl`, a Matlab script will be incorporated in the shell script and extracted at runtime.

- `$pathMatlabResults`
  use a different path for storing the results of the Matlab run. Default is `<ENV::MASV_BASEDIR>/sv_systems/results/`.

- `$autoClean_FA_Files`
  if set to `"1"`, clean all MLF files after the evaluation part with Matlab. The model directories of the speakers and the world models are kept. Keep the MLF files when debugging and testing of system.

5.6 Running experiments

After a run of `run_MASV_experiment.pl` a shell script is written to the current directory. It can be moved to any place in the file system but a good place will be the directory `<sv system path>/scripts`. At the beginning of the shell script there are some switches defined starting `"do_"`. Here you can quickly enable or disable a part in the training or testing sequence. By default, all switches are enabled except for the clean up part; here the setting of the expConf file is used.
The shell script generated with `run_MASV_experiment.pl` contains also a template for a Matlab script which does the score normalization and further calculations of performance figures. During runtime of the shell script the Matlab script is written to a file `startup.m` in the directory specified in the variable `MATLAB_CALC_PATH`. If you chose to do the Matlab processing, the shell script will change to `MATLAB_CALC_PATH` after all testing procedures and will start Matlab there. By default, Matlab executes then all commands in the `startup.m` batch file. For a more detailed description of the commands used in the Matlab script, see the next chapter.
To start the shell script, simply change to its directory and start it from the command line. You may need to add `"./"` to the name in case that it resides in a directory which is not contained in your `PATH` variable. E.g.:

```
./batch_script_gmm_16_mixtures.sh
```

Chapter 6

Analyzing results

The Matlab part of MASV is mainly used to collect and process the data generated with the training/testing shell script, to calculate various performance figures and to offer a GUI for exploring the results. In the first section we will present the Matlab functions used for the data handling and we will give detailed information about the information exchange between the Perl part and the Matlab functions via files. The second section will describe the graphical interface of the data exploration tool `Plot_SR_parameters` which gives easy access to the data calculated before.

6.1 Matlab functions for data processing and evaluation

There are two data types for storing information about an experiment defined in Matlab: a "result struct" and a "score result struct". The result struct contains all post-processed llh scores which were originally in the MLF files of the tests. Post processing of the scores means here different kinds of normalization applied to the scores. Also, the scores for each speaker and test type are sorted by their values. The score result struct contains one or more "parts" which describe various sets of performance figures. For each part there is a specification, which kinds of recordings from what speakers were included in computing the perfomance figures. More details are given in 6.1.2.
As already mentioned in 5.6 there is a template for a Matlab script which performs all necessary post-processing. An example script could look like this:

```
1   current_path = pwd; % store current path
2   cd ../..            % change to normal Matlab directory
3   startup             % executed standard startup.m file
4   cd(current_path);   % change back
5   [startScript]=read_experiment_Details; % read content of shell script
6   rs=calc_FR_FA('LPCC_SD','GMM_128_altTraining1',0,0,1,'simple␣world','
        without␣pauses','per␣frame','force_aligned_GMM_128_altTraining1_');
         % create result struct
7   rs.expDetails.script = startScript;  % add content of shell script for
        documentation purposes
8   save_result(rs,'/raid/tera6/VERIDAT/sv_systems/results/LPCC_SD/GMM/');
         % save the result struct
9   sr=calc_score_result(rs);                % create a score result struct
10  sr.expDetails.script = startScript;   % add content of shell script for
        documentation purposes
11  save_score_result(sr,'/raid/tera6/VERIDAT/sv_systems/results/LPCC_SD/GMM
        /'); % save
```

The first four lines make sure that all common settings are done; the fifth line reads the contents of the shell script to a string cell. Line 6 creates a result struct for the given experiment; it is described in more detail in 6.1.1. The shell script is stored in the field `expDetails.script` of the result struct (line 7). Other data can be stored here as well. Note that there is also a field `info` which can store any kind of comments in a string cell. Line 8 saves the new result struct, line 9 creates a score result struct which includes one standard part (speaker dependent EERs with same gender impostors, training material

from genuine speakers excluded from evaluation). Finally, the information about the shell script is also included similar to line 7 and the resulting score result struct is saved (line 11).

6.1.1 Creating and saving results

The function `calc_FR_FA` takes at minimum 8 parameters; there are two more, optional parameters:

```
function result_struct = calc_FR_FA(sv_system_name, ...
                                    result_name, ...
                                    full_calc_flag, ...
                                    full_info_flag, ...
                                    incl_world_flag, ...
                                    norm_type, ...
                                    llhsum_type, ...
                                    score_type, ...
                                    [mlf_file_base = 'force_aligned_', ...
                                    cohort_size = 3] )
```

`sv_system_name`
: Name of the SV system from which the data will be read. The resulting filename of the result struct will be build up using this name.

`result_name`
: An identifier for the result; useful to distinguish between several result structs within one SV system.

`full_calc_flag`
: Flag for more additional computations of mean and standard deviation of llh scores (per file and per speaker). Values: 0 or 1.

`full_info_flag`
: Flag for including also segment information (start and end time plus model name of each segment) in the result struct. Warning: gives increased file size by factor 4! Values: 0 or 1.

`incl_world_flag`
: Flag for including also raw llh scores of the world model. Useful for debugging purposes. Values: 0 or 1.

`norm_type`
: Type of normalization of llh scores. Possible values:

  'none' keep raw llh values.

  'simple world' normalize with world model score: $\text{llh}_{\text{normed}} = \text{llh}_{\text{raw}} - \text{llh}_{\text{world}}$

  'cohort mean' normalize with mean score of n-best cohort speakers (see also parameter `cohort_size`).

  'cohort max' normalize with best score of all cohort speakers.

`llhsum_type`
: Type of segments included in the overall llh sum per recording. Values:

  'with pauses' include all segments

  'without pauses' ignore segments 'sp', 'silEnd', 'silBegin', 'sil'

`score_type`
: Type of final llh score. Values:

  'simple sum' sum over selected segments of recording.

  'per frame' like above, but normalized by length (number of frames).

`mlf_file_base`
: Prefix of the test MLF files used. This string is used to generate the file names of the test MLF files, which are read:

  world mlf <`mlf_file_base`>`client_x_world.mlf`

genuine spk mlf <mlf_file_base>client_x_model.mlf

impostor spk mlf <mlf_file_base><imp_speaker>_x_model.mlf

impostor on cohort spk mlf <mlf_file_base>cohort_<imp_speaker>_x_self.mlf

The name of the info files are generated with the same scheme:

world test <mlf_file_base>client_x_world.info

genuine test <mlf_file_base>client_x_model.info

impostor test <mlf_file_base>imp_x_model.info

cohort test <mlf_file_base>cohort_x_model.info

`cohort_size`
Number of cohort speakers when computing mean cohort score.

6.1.2 Creating score results

The function `calc_score_results` computes a score result struct with a standard part.

```
function result_struct = calc_score_result(result_struct)
```

The first part is a summary of speaker dependent EERs with same gender impostors. Training material from genuine speaker tests is excluded from evaluation; all material from impostor tests is taken.
More different parts with different performance figures and with different selection of speech material can be created with the function `add_part_to_score_result`:

```
score_result_struct = add_part_to_score_result(result_struct, ...
                                               score_result_struct, ...
                                               result_type, ...
                                               description, ...
                                               menu_descr, ...
                                               threshold, ...
                                               include_models, ...
                                               imp_spks_list, ..
                                               fr_sel_sessions, ..
                                               fa_sel_sessions)
```

The functions takes a score result struct, adds a part and returns an updated version.

`result_struct`
Matching result struct, needed for computation of the performance figures.

`result_type`
Type of part to add:

'fr_fa' FR rates and FA rates (a priori) for given threshold(s).

'det' DET curve based on all selected recordings plus individual FR/FA rates for determined EER threshold.

'eer' speaker dependent EER (a posteriori, per genuine speaker).

`description`
A description string with arbitrary length.

`menu_descr`
A short description string, will be used in the menus of the data exploration GUI.

`threshold`
Threshold(s) for computing a 'fr_fa' part. A single value will be used for all genuine speakers, a vector with the same number of entries as genuine speakers allows individual thresholds when computing FR/FA rates.

`include_models`
A string cell for selecting the genuine speakers (given by four digit speaker ids). Usually you will use the list `model_info.trained_models` stored in the result struct.

`imp_spks_list`
A string cell for selecting impostor speakers. Contains a list of speaker ids or one of the following strings:

'all' all impostors (as defined in the list `model_info.imp_test_spks` in the result struct.

'same gender' select for each genuine speaker all impostor speakers with matching gender.

'cross-gender' like above, but select other gender.

'all female'

'all male'

`fr_sel_sessions`
A string cell for selecting the genuine speaker recordings. Contains one of the following values:

'all' all genuine recordings.

'training' all genuine recordings which were used for training the speaker models.

'without training' exclude training recordings.

In addition, the names of the session lists defined in the file `SR_lib.pm` of the Perl scripts can also be used here.

`fa_sel_sessions`
A string cell for selecting the impostor speaker recordings. Contains only values from the predefined session lists.

6.1.3 Interface between MASV's Perl part and Matlab part

The information about a completed test of a SV system is transfered from the Perl part to the Matlab part via two types of files:

MLF files (defined by HTK): they contain the sequence of models defined for forced alignment and the corresponding log likelihood values (llh).

info files: These files reside in the SV system directory `info/` and describe for each test type (defined later), for which speakers which models were used with the selected type of material..

There are four possible test types:

genuine test the material of a speaker is tested with its own model.

impostor test material from other speakers is tested with a speaker model (vulnerated speaker).

world test material from one or more speakers is tested against a world model (used for normalization later).

cohort test like world test, but here, the set of cohort models is used instead of a single world model (used for cohort normalization).

MLF files

There are two possible formats for the entries in the MLF files:

- `begin end model llh word`
- `begin end model llh`

For each speaker and test type there is one MLF file in the speaker's directory. It contains all recordings that were used for the particular test.
The filenames are build up with the following structure: The base name of the files (called here `mlf_file_base`) starts with `force_aligned_`, followed by an optional string (as defined in the expConf in the variable `$FA_id`). For each test type, the complete file name is:

world mlf <mlf_file_base>client_x_world.mlf

genuine spk mlf <mlf_file_base>client_x_model.mlf

impostor spk mlf <mlf_file_base>0000_x_model.mlf

impostor on cohort spk mlf <mlf_file_base>cohort_0000_x_self.mlf

Info files

The format of these files is plain text with at least two or three lines:

line 1 name of speaker list, as defined in `SR_lib.pm`. From these speakers the speech material is used for the test (in case of genuine test, world test and cohort test). In case of a impostor test, these speakers provide the models for the test (vulnerated speakers).

line 2 (optional, only for impostor tests) another name of speaker list. It defines the imposter speakers.

line 3 and following Each line contains an entry for each sub model used in testing. Normally the data is not split to train and test several models, so there is only one entry here. An entry consists of three fields, seperated by commas. The first field gives the script file (selected speech material for the test), the second field gives the version of the (sub) model; the name of the model directory is given in the last field. Note, that in case of a world test, the model directory is located in `world/`.

The name of the info files are generated with a similar scheme used for the MLF files:

world test <mlf_file_base>client_x_world.info

genuine test <mlf_file_base>client_x_model.info

impostor test <mlf_file_base>imp_x_model.info

cohort test <mlf_file_base>cohort_x_model.info

6.2 GUI for data exploration

The top directory of the Matlab scripts contains the Matlab function `Plot_SR_parameters`. Enter this command at the Matlab prompt. If you have not loaded already result structs or score result structs in your current Matlab session, you are prompted to select a struct from your standard directory for results (as defined in the `ENV::MASV_MATLAB_RESULTSDIR`). After loading the GUI depicted in figure 6.1 shows up.

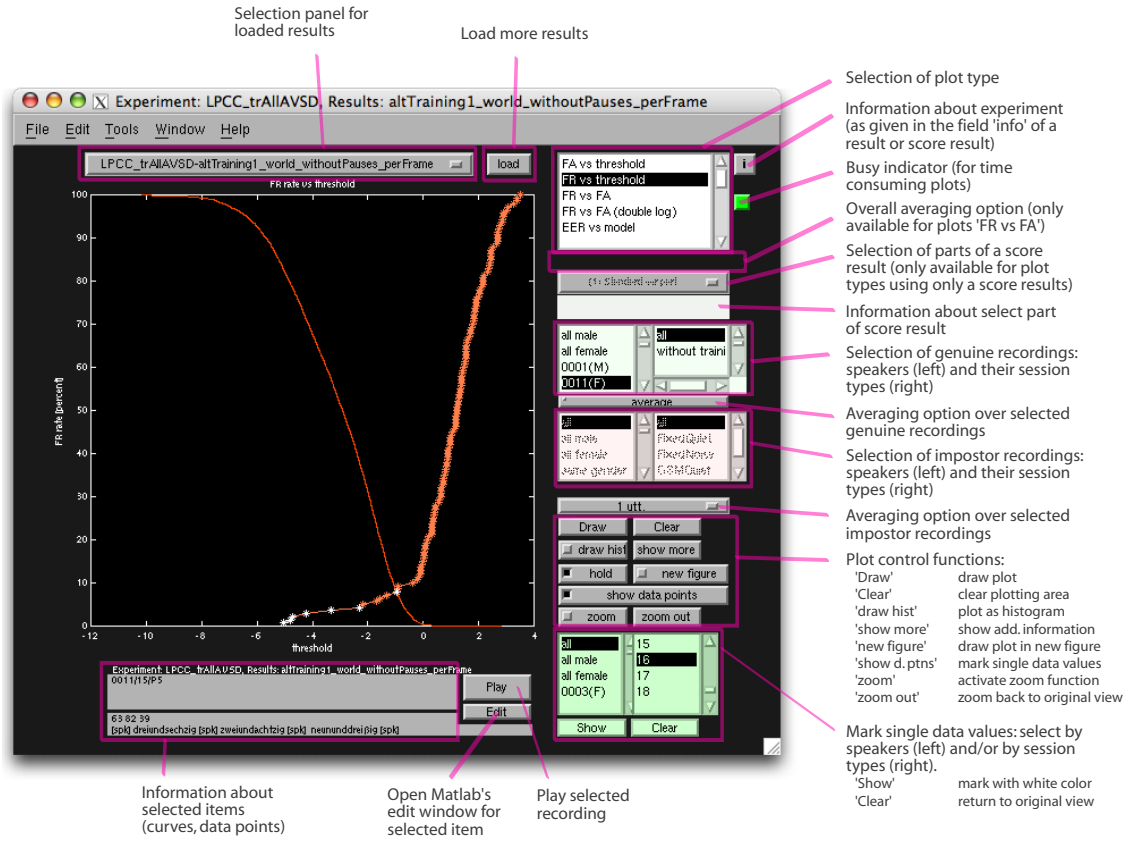

Figure 6.1: Graphical interface for exploration of test datasets of a SV system.

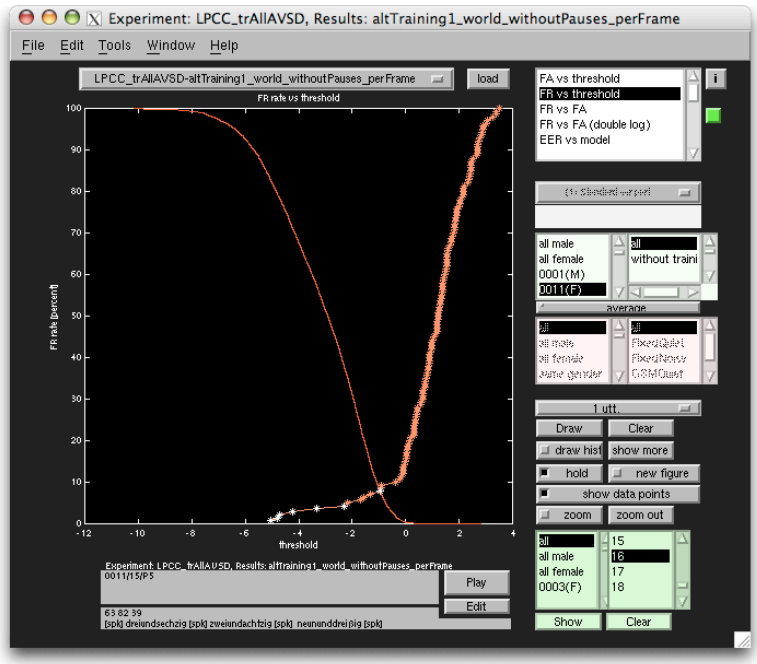

Figure 6.2:

Figure 6.3:

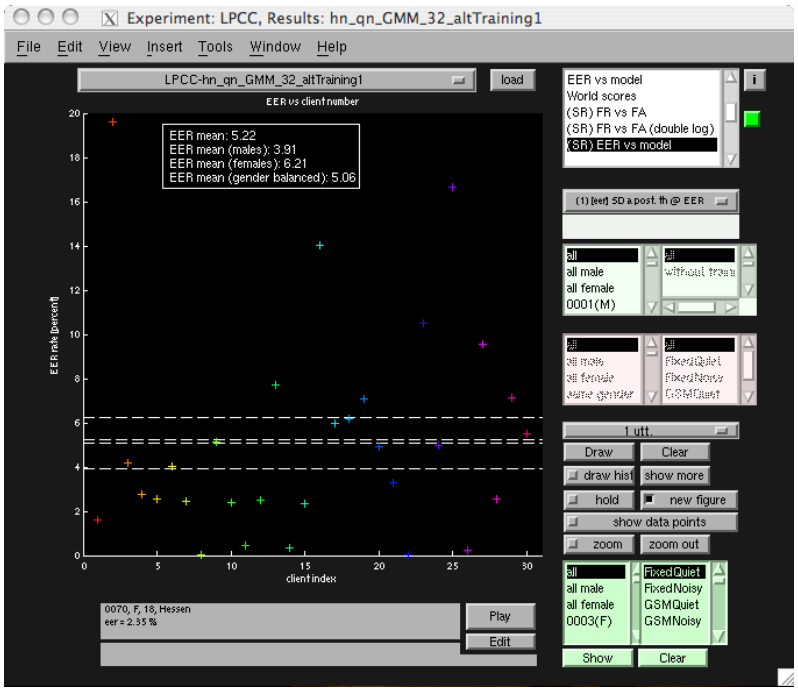

Figure 6.4:

Chapter 7

Description of the perl tools

Overview - not available yet

7.1 Common options

Common to all tools are the following two options:

`-version`
: print version and release information

`-? | help`
: print a help message.

The help message is also printed when no option and parameter is given to a tool.

7.2 Recording configuration

Several tools and also the expConf files use parameters which are referred to as "training configurations" or "test configurations". As a generic label we use the term "recording configuration" here. It is a compact description of a speaker set (defined in the speaker list) and the script files used by these speakers. An example for a recording configuration would be the string `dev_test/S_list_all`. The part before the slash describes the speakers selected. These definition are done in the speaker list file (residing in <sv system path>`/speaker_lists/`). The label `all_set` refers to all speakers in the SV system. The part after the slash describes the script files (or S-list file named after the option `-S` of the HTK tools).It is either simply the name of the S-list file or, in case of only two letters, an abbreviation of an S-list file name as defined in the Perl source file `SR_lib.pm`. An example would be the string `aa` which is an alias to the S-list file name `S_list_all`. This mechanism makes the assumption that for each speaker directory in a SV system there is the same set of S-list files. By this way the selection of recording files works with arbitrary speaker sets.

7.3 High-Level-Tools

7.3.1 create_MASV_paramPool.pl

Creates a new parameter pool and configuration scripts for further processing with `run_HCopy.pl`.

```
create_MASV_paramPool.pl [options] poolname
```

Creates script files for `HCopy` with name `S_pool_`<poolname>. Speech files are read from <ENV::MASV_SOURCE_DATABASE>; output path for script files is the generated directory <ENV::MASV_PARAMPOOL_DIR>`/`<poolname>`/scripts/`. Script files are generated for

- the selected speakers from the `-s` option

- for all speakers of the corpus

The scripts contain all recordings of the given type (`-t`) of all available sessions.
Options:

`-t | typesession = s`
specify type of recordings. default: `'P[1-7]'` (all triple numbers)

`-s | speakers = s`
specify set of speakers. default: speaker set `@all` as defined in the template file `<ENV::MASV_BASEDIR>/templates/speaker_sets/standard`. Include additional sets here.

7.3.2 run_HCopy.pl

Runs `HCopy` with the script files for given pool.

```
run_HCopy.pl [options] poolname [script_file [config_file]]
```

Run `HCopy` for the parameter pool `<poolname>` with the given script file and the given `HCopy` config file. If one or both are omitted, the file `S_pool_<poolname>` is assumed to be the script file and `C_<poolname>` is used as `HCopy` config file. Script files are searched for in the `scripts/`-directory of the pool; config files are searched for in the `config/`-directory.
Options:

`-xo | extraOptions`
pass extra options to `HCopy`

`-p | parallelComputation`
use several hosts to run the `HCopy` command.

7.3.3 create_MASV_sv_system.pl

Creates an SV system directory for running experiments

```
create_MASV_sv_system.pl [options] sv_system_name poolname
```

Creates a directory <sv_system_name> in the `sv_systems/` directory; signal files are used from pool <poolname>.
Options:

`-t | typesession = s`
specify type of recordings. Default: `'P[1-7]'` (all VeriDat triple numbers)

`-s | speakers = s`
specify set of speakers. Default: speakers of set `@all` from the template file `<ENV::MASV_BASEDIR>/templates/speaker_sets/standard`. Add additional sets here.

`-e | environment = s`
specify environment set. Default: all. Examples for possible values (from VeriDat definition):

- FixedQuiet
  training: sessions 01,09,13,12;
  evaluate: sessions 03,05,17
- all
  (training: sessions 01,02,03,04;
  evaluate: sessions 05,06,07,08,09,10,11,12,13,14,15,16,17,18,19,20

Make your own definitions in `MASV_db_desc.pm` in the hashes `$training_sessions_lists` and `$evaluate_sessions_lists`.

`-m | makeMLFs`
create two MLF files, one with prompted text, one with transcription of speaker's utterance. default: not set

`-f | filterNoises = s`
: only used, when MLF files are generated; filters noise labels from the labels. Default: not set

`-c | createStatisticFile`
: create main statistic file, showing distribution of labels in test and training set. Does not consider skipped recordings from individual subjects (see option `-i`).

`-i | skipInterruptedSessions`
: skip Sessions containing noise labels (defined in `MASV_db_desc.pm`).

`onlyScripts = s`
: do not create new SV system, only create script files in existing SV system. Existing script files are not overwritten. Two MLF files (for prompted text and the transcriptions) are generated with the tag `for_<onlyScripts>` in the file name.

7.3.4 check_MASV_experiment.pl

Check files for an experiment

```
check_MASV_experiment.pl [options] expConfig sv_system_name
```

Check the given experiment configuration `expConfig` of the SV system <`sv_system_name`> for completeness of the list files and of the configuration files. The path to the file `expConfig` is given relative to the directory `expConfig/` of the SV system.

7.3.5 calc_size_MASV_experiment.pl

Prints the required data amount for an experiment.

```
calc_size_MASV_experiment.pl [-expConfig | other options]
                        sv_system_name
```

Calculate the required data amount for an experiment. Parameters can be given either seperately or with the help of an experiment config file The path to the file `expConfig` is given relative to the directory `expConfig/` of the SV system.
Options:

`-expConfig=s`
: load config file from SV system directory `expConfig/`. The settings can be overwritten from the command line options.

`-wt | world_test_list=s`
: configuration for testing the world model with `run_worldModelTest.pl`. Default is `'dev_set/aa'`

`-iw|num_iterations_world=s`
: training iterations for world model.

`-im|num_iterations_model=s`
: training iterations for speaker models.

`-hvtc|HViteTrainedModelConf=s`
: configuration when testing speaker models; default is `'all_training/aa'`.

`-hvxc|HViteCrossConf=s`
: configuration when testing with impostor speakers; default is `'all_test/aa'`.

7.3.6 run_MASV_experiment.pl

Creates a complete training and test sequence for an experiment.

```
run_MASV_experiment.pl [options] expConfig  sv_system_name
```

Creates a complete shell script for training and test of the experiment of <sv_system_name>. The file expConfig from the SV system directory expConfig/ contains the settings. Creates necessary shell (and Matlab) scripts for execution later on.
Options:

-cs | createScript=s
: create shell script with given name;

-doTestsOnly
: perform only HVite tests for spk models with world, self and impostor test. Default: setting from expConfig

-sm|startMatlab
: include Matlab script for performance calculation. Default: setting from expConfig.

7.3.7 make_clean_MASV_experiment.pl

deletes temporary files of an experiment.

```
make_clean_MASV_experiment.pl [-expConfig | options]
                        sv_system_name
```

Delete files created during a run of an experiment of the SV system <sv_system_name>.
Options:

-expConfig=s
: load config file from SV system directory expConfig/. The settings can be overwritten from the command line options.

-cleanAll
: delete also force alignment files from world test and templates MLFs.

-cleanModels
: delete also world models and speaker models.

-thoroughly
: clean over all speaker directories, not regarding current speaker set.

-hmmBaseDirWorld=s
: use models in given directory in world directory; Default is hmm/. used here to find the world models for -cleanModels.

-hvwc|HViteWorldConf=s
: configuration when testing with world model; default is 'all/aa'.Used here to specify all spk dirs containing MLF files.

-hmmBaseDirModel=s
: use models in given directory in speaker directory. default is hmm/. used here to find all speaker model directories.

-hvtc|HViteTrainedModelConf=s
: configuration when testing speaker models; default is 'all_training/aa'. used here to find all speakers with models.

-id|FA_id=s
: use given string to identify MLFs of this SV system; default is: "" i.e. files with default name forced_aligned_client_x_model.mlf, forced_aligned_client_x_world.mlf and "forced_aligned_0*_x_model.mlf are selected.

7.3.8 loop_thru_speakersets.pl

Repeat a script for different speaker sets.

```
loop_thru_speakersets.pl [options] script_name  speakerset_list  sv_system_name
```

Repeat a shell script for the given speaker sets. The speaker sets are listed in a file (speakerset_list), one entry per line. The speaker sets themselves are defined in the SV system directory speaker_sets/.

7.4 Low-Level-Tools

7.4.1 run_HTKCommand.pl

Runs an arbitrary (not necessarily HTK) command for a SV system

```
run_HTKCommand.pl [options] sv_system_name commandstring
```

Run an arbitrary command (given by <commandstring>) for the SV system <sv_system_name>. Useful to log the commands in the SV system log file. Also used for applying commands to a bunch of speaker directories. The command string can contain '%spk_dir%' which will expand to the current speaker directory when using the -i option.
Options:

-i | iterateOnVPs
: perform the command in each speaker's directory

-tc | trainingModelConf
: configuration to specify the speaker directories; default is 'all_training/aa'; see description 7.2. The script file part of the configuration (part after the slash) is ignored here.

7.4.2 run_HCompV.pl

Runs HCompV for a SV system

```
run_HCompV.pl [options] sv_system_name
```

Run HCompV for SV system <sv_system_name>. Flat start HMMs are written to the directory hmm.0/ located in each speaker's directory.
Options:

-w | world
: do a flat start for the world model. Default: not activated

-wl | world_list
: configuration for world flat start; default is 'world_set/aa'; ignored, when -w option not set.

-tc | trainingModelConf
: configuration for normal flat start; default is 'training_set/at'; see description 7.2; ignored, when option -w set.

-protoDir=s
: directory with prototype models.

-v | minVar
: define minimum value for the elements of the covariance matrix of the model. Default value is 0.001

-hmmBaseDir
: build models in given directory in speaker resp. world directory; default is hmm/.

-useGMM
: adapt behaviour for using a single state model for all speech data (GMM).

-xo | extraOptions
: pass extra options to HCompV

-p | parallelComputation
: use several hosts for running HCompV (useful only for normal training, option -tc)

7.4.3 run_HInit.pl

Runs `HInit` for a SV system (useful for GMM models)

```
run_HInit.pl [options] sv_system_name
```

Run `HInit` for SV system <`sv_system_name`>. Flat start HMMs are written to the directory `hmm.0/` located in each speaker's directory.
Options:

`-w | world`
: do a flat start for the world model. Default: not activated

`-wl | world_list`
: configuration for world flat start; default is `'world_set/aa'`; ignored, when `-w` option not set.

`-tc | trainingModelConf`
: configuration for normal flat start; default is `'training_set/at'`; see description 7.2; ignored, when `-w` option set.

`-v | minVar`
: minimum value for the elements of the covariance matrix of the model. Default value is 0.001

`-m | mlf`
: specify MLF file, default is `mlf_prompted.mlf`.

`-protoDir`
: use models in directory <`protoDir`> as base.

`-hmmBaseDir`
: build models in given directory in speaker resp. world directory; default is `hmm/`

`-useGMM`
: adapt behaviour for using a single state model with multiple mixtures (GMM).

`-xo | extraOptions`
: pass extra options to `HInit`

`-p | parallelComputation`
: use several hosts for running `HInit` (useful only for normal training, option `-tc`)

7.4.4 run_HERest.pl

Run `HERest` for a SV system

```
run_HERest.pl [options] sv_system_name num_iterations [first_iteration_no]
```

Run `HERest` for the SV system <`sv_system_name`>. Performs <`num_iterations`> iterations, starting by default with model version 0. Optional parameter <`first_iteration_no`> gives number of the first model version to be built when starting the iteration loop. Default value is therefore 1.
Options:

`-w | world`
: make a single (world) model

`-wl | world_list`
: configuration for world flat start; default is `'world_set/aa'`; see description 7.2

`-tc | trainingModelConf`
: use configuration for normal estimation step; default is `'training_set/at'`; see description 7.2.

`-m | mlf`
: specify MLF file, default is `mlf_prompted.mlf`

`-h | hmm_list`
specify HMM list, default is `HMM_number.list`

`-v | minVar`
minimum value for the elements of the covariance matrix of the model. Default value is 0.001

`-hmmBaseDir`
build models in given directory in speaker resp. world directory; default is `hmm/`

`-macros`
use given file for HMM macros; it must reside in the directory of the HMM models

`-useGMM`
adapt behaviour for using a single state model with multiple mixtures (GMM).

`-useHEAdapt`
use adapting scheme of HEAdapt with MAP procedure instead of Baum-Welch reestimation. MAP adaption factor is fixed to 15.0. Works here only with patched version of HEAdapt.

`-xo | extraOptions`
pass extra options to `HERest`

`-p | parallelComputation`
use several hosts for running `HERest`; (useful for both normal and world training).

7.4.5 run_HVite.pl

Runs `HVite` for a SV system

```
run_HVite.pl [options] sv_system_name hmm_version
```

Run `HVite` for the SV system <`sv_system_name`>. Uses HMM models of the version <`hmm_version`>. Options:

`-o | outputMLF=s`
use given name for output MLF file; default is

- `force_aligned_client_x_model.mlf` for normal operation
- `force_aligned_0000_x_model.mlf` for impostor test with 0000 replaced by the speaker ID
- `force_aligned_client_x_world.mlf` for world test
- `force_aligned_cohort_0000_x_model.mlf` for cohort test with 0000 replaced by the speaker ID

`-wm | outputWordMLF`
use word level for output MLF file. Useful when using the MLF file later on as template for force alignment and single tee models can occur. Note that "words" have to defined by a normal model and a tee model (e.g. "izwanzig", using "zwanzig" and "sp")

`-a | appendToMLF`
append `HVite` output MLF to existing MLF with same name. If the MLF file does not exist, it will be created. Default value: not set

`-t | templateMLF=s`
use the given MLF file as a template for force alignment with contraints both on label sequence and time sequence. Note: normal forced alignment enforces only the label sequence. The `-m` option is ignored, but the training configuration of the `-tc` option is used instead: all recordings that also occur in the training data are used.

`-d | dict=s`
use given dictionary from the directory `syntax/`. Default value: `generaldict`

`-s | singleTestFiles=s`
perform only test with given test files (comma seperated list).

`-m | mlf=s`
specify MLF file, default is `mlf_prompted.mlf`

`-h | hmm_list=s`
specify HMM list, default is `HMM_number.list`

`-x | crossTest`
perform test of all test sentences against all models; impostor test needs the options `-xc` and `-tc` !

`-xc | crossTestConf=s`
configuration for impostor test; default is `'test_set/aa'`.

`-w | worldTest`
perform test of all test sentences against world model.

`-wc | worldListConf=s`
configuration for world test; default is `'all_tt/aa'` (all speakers from training and test set).

`-wmt | worldModelTest`
perform test of world model, write results in a single MLF; this option is normally used by `run_worldModelTest.pl`.

`-wmtc | worldModelTestConf=s`
configuration to specify speakers for world model test; default is `'dev_set/aa'`; see description 7.2; this option is normally used by `run_worldModelTest.pl`.

`-tc | trainingModelConf=s`
configuration to specify trained models; default is `'training_set/at'`; see description 7.2

`-c | cohortTest`
perform tests of cohort speakers on given models; use `-tc` to specify models to be tested (`'all_tt/aa'` e.g.), use `-xc` to specify test speakers (`'world_set/aa'` e.g.); session specification given by `-xc` is used (similar to option `-x`)

`-hmmBaseDir=s`
use models in given directory in speaker resp. world directory; default is `hmm/`

`-useGMM`
adapt behaviour for using a single state model with multiple mixtures (GMM).

`-xo | extraOptions=s`
pass extra options to `HVite`

`-p|parallelComputation`
use several hosts for running `HVite`

7.4.6 run_worldModelTest.pl

Runs tests on world model with given speaker configuration.

`run_worldModelTest.pl [options] sv_system_name hmm_version [start hmm_version, [iterations]]`

Runs `HVite` for the SV system <`sv_system_name`> with given world models in normal speech recognition mode. It uses HMM models of the version <`hmm_version`>. Test samples are taken by default from the recording configuration `'dev_set/aa'`.
Options:

`-l | list=s`
recording configuration. Default is `'dev_set/aa'`.

`-o | outputMLF=s`
use given name for output MLF file; default is `test_worldModel_with_`<`list_name`>`.mlf`

`-d | dict=s`
use given dictionary from the directory `syntax/`; default is `generaldict`.

`-m | mlf=s`
specify MLF file; default is `mlf_prompted.mlf`.

`-h | hmm_list=s`
specify HMM list; default is `HMM_number.list`

`-hmmBaseDir=s`
use models in given directory; default is `hmm/`

`-macros=s`
use given file for HMM macros; it must reside in the directory of the HMM models.

`-n | wdnet`
specify word network from the directory `syntax/`; default is `numbernet`.

`-xo | extraOptions`
pass extra options to `HVite`

`-p | parallelComputation`
use several hosts for running `HVite`

7.5 Other tools

7.5.1 calc_GMM_world_llh.pl

Calculates the log-likelihood per frame over all entries in a MLF file. The result (a single float) is printed to STDOUT.

```
calc_GMM_world_llh.pl  mlf_file
```

This tool is useful to determine how good a GMM model fits to speech data. It is usually only used by `run_worldModelTest.pl`.

`- fd|frame_duration = i`
duration of a frame, given in HTK's 100ns units; default is 100000 (= 10ms).

7.5.2 get_bestWorldModel_GMM.pl

Reads given result files for GMM world test (made with `calc_GMM_world_llh.pl`) and finds the version which gives the minimum of the overall llh. The minimum difference in the score between suceeding versions must be 0.1, otherwise the current version is declared as the best model.

```
get_bestWorldModel_GMM.pl  sv_system_name  hmmBaseDir
```

Note. the result files must be in the `world/` directory of the SV system <sv_system_name> and their name must start with the string

```
calc_GMM_world_llh_test_<hmmBaseDir>_<version_of_model>
```

and end with the extension `.txt` .

7.5.3 get_bestWorldModel.pl

Reads given `HResults` files and finds the version which gives the first maximum of the word accuracy.

```
get_bestWorldModel.pl  sv_system_name  hmmBaseDir
```

Note: the files generated with `HResults` must reside in the `world/` directory of the SV system <sv_system_name> and their name must start with the string

```
hresults_test_<hmmBaseDir>_<version_of_model>
```

and end with the extension `.txt` .

7.5.4 change_entry_transitions_to_tee.pl

Corrects tee models after `HCompV` estimation: it reactivates transitions from the non-emitting start state to the non-emitting end state.
The model <`hmm_model_name`> is searched in the directory <`hmm_base_dir`>.

```
change_entry_transitions_to_tee.pl sv_system_name hmm_base_dir hmm_model_name
```

7.5.5 loop_thru_speakersets.pl

Repeats a script for different speaker sets.

```
loop_thru_speakersets.pl [options] script_name  speakerset_list  sv_system_name
```

Repeats a shell script for the given speaker sets. The speakersets are listed in a file (`speakerset_list`), one entry per line. The speakersets themselves are defined in the SV system directory `speaker_sets/`. This tool is useful to evaluate the statistical dependency of the performance of a SV system on different speaker sets (a kind of bootstrap analysis).

7.5.6 message2log.pl

Writes message to log file of a SV system.

```
message2log.pl [options] message sv_system_name
```

Writes the message to the log file of a SV system. The default log file name is `log.txt`. All log files are kept in the `log/` directory inside the SV system directory.
Options:

`-visible | v`
: display message also on STDOUT.

`-logfile`
: set logfile; default is `log.txt`.

7.5.7 raw2wav.pl

Convert raw A-law files with 8kHz sampling frequency, one channel and byte coding to a sound file in WAV format with the same encoding.

```
raw2wav.pl <infile> <outfile>
```

This script is used in `run_HCopy.pl` as a filter for the raw sound files.

7.5.8 split_file_lists.pl

```
split_file_lists.pl [options]   inputlist   outputlist   output_no_of_entries
```

Reduce a script file to <`output_no_of_entries`> list entries. If more than one speaker is contained in the inputlist (e.g. world lists), the reduction is done for each speaker seperately. The selection is done evenly spaced over all entries of a speaker. In contrast, the option `-h` takes the first <`output_no_of_entries`> entries.
Options:

`-h | head`
: return the first <`output_no_of_entries`> entries in a contiguous block.

Use the shell script `create_difference_list.sh` to create the complementary part.

7.5.9 create_difference_list.sh

Create a difference file for a script file.

```
create_difference_list.sh  complete_list  subset  difference_subset
```

Create a difference file for a script file which contains all entries of `complete_list` that are not in the file `subset`.

7.5.10 switch_MASV_spk_data_set.pl

Changes speaker set configuration and parameter pool.

```
switch_MASV_spk_data_set.pl [options]  sv_system_name  speaker_set_name  [poolname]
```

Creates a new link structure in the SV system directory `links/` which points to the given speaker set and (optionally) to the given parameterpool.

Chapter 8

Description of the Matlab tools

not available yet

Appendix A

GNU Free Documentation License

Version 1.2, November 2002
Copyright ©2000,2001,2002 Free Software Foundation, Inc.

59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.
This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.
We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The **"Document"**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as **"you"**. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.
A **"Modified Version"** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.
A **"Secondary Section"** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.
The **"Invariant Sections"** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The

Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.
The **"Cover Texts"** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.
A **"Transparent"** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called **"Opaque"**.
Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.
The **"Title Page"** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.
A section **"Entitled XYZ"** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **"Acknowledgements"**, **"Dedications"**, **"Endorsements"**, or **"History"**.) To **"Preserve the Title"** of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.
The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.
You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.
If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.
If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each

Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.
It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.
You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.
You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.
The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.
The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.
In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.
You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.
If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.
If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.
Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright ©YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.
If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.