julius

Name

`Julius` — open source multi-purpose LVCSR engine

Synopsis

julius [-C *`jconffile`*] [*`options`*...]

Description

julius is a high-performance, multi-purpose, open-source speech recognition engine for researchers and developers. It is capable of performing almost real-time recognition of continuous speech with over 60k-word 3-gram language model and triphone HMM model, on most current PCs.

julius can perform recognition on audio files, live microphone input, network input and feature parameter files.

Supported Models

julius needs a language model and an acoustic model to run as a speech recognizer. **julius** supports the following models.

Acoustic model

Sub-word HMM (Hidden Markov Model) in HTK ascii format are supported. Phoneme models (monophone), context dependent phoneme models (triphone), tied-mixture and phonetic tied-mixture models of any unit can be used. When using context dependent models, inter-word context dependency is also handled. You can further use a tool mkbinhmm(1) to convert the ascii HMM file to a compact binary format.

Note that **julius** itself can only extract MFCC features from speech data. If you use acoustic HMM trained for other feature, you should give the input in HTK parameter file of the same feature type.

Language model: word N-gram

Word N-gram language model, up to 10-gram, is supported. Julius uses different N-gram for each pass: left-to-right 2-gram on 1st pass, and right-to-left N-gram on 2nd pass. It is recommended to use both LR

2-gram and RL N-gram for Julius. However, you can use only single LR N-gram or RL N-gram. In such case, approximated LR 2-gram computed from the given N-gram will be applied at the first pass.

The Standard ARPA format is supported. In addition, a binary format is also supported for efficiency. The tool mkbingram(1) can convert ARPA format N-gram to binary format.

Language model: grammar

The grammar format is an original one, and tools to create a recognirion grammar are included in the distribution. A grammar consists of two files: one is a 'grammar' file that describes sentence structures in a BNF style, using word 'category' name as terminate symbols. Another is a 'voca' file that defines words with its pronunciations (i.e. phoneme sequences) for each category. They should be converted by mkdfa.pl(1) to a deterministic finite automaton file (.dfa) and a dictionary file (.dict), respectively. You can also use multiple grammars.

Language model: isolated word

You can perform isolated word recognition using only word dictionary. With this model type, Julius will perform rapid one pass recognition with static context handling. Silence models will be added at both head and tail of each word. You can also use multiple dictionaries in a process.

Search Algorithm

Recognition algorithm of **julius** is based on a two-pass strategy. Word 2-gram and reverse word 3-gram is used on the respective passes. The entire input is processed on the first pass, and again the final searching process is performed again for the input, using the result of the first pass to narrow the search space. Specifically, the recognition algorithm is based on a tree-trellis heuristic search combined with left-to-right frame-synchronous beam search and right-to-left stack decoding search.

When using context dependent phones (triphones), interword contexts are taken into consideration. For tied-mixture and phonetic tied-mixture models, high-speed acoustic likelihood calculation is possible using gaussian pruning.

For more details, see the related documents.

Options

These options specify the models, system behaviors and various search parameters. These option can be set at the command line, but it is recommended that you write them in a text file as a "jconf file", and specify it by "-C" option.

Julius application option

`-outfile`

On file input, this option write the recognition result of each file to a separate file. The output file of an input file will be the same name but the suffix will be changed to ".out". (rev.4.0)

`-separatescore`

Output the language and acoustic scores separately.

`-callbackdebug`

Print the callback names at each call for debug. (rev.4.0)

`-charconv` *`from`* *`to`*

Print with character set conversion. *`from`* is the source character set used in the language model, and *`to`* is the target character set you want to get.

On Linux, the arguments should be a code name. You can obtain the list of available code names by invoking the command "iconv --list". On Windows, the arguments should be a code name or codepage number. Code name should be one of "ansi", "mac", "oem", "utf-7", "utf-8", "sjis", "euc". Or you can specify any codepage number supported at your environment.

`-nocharconv`

Disable character conversion.

`-module [port]`

Run Julius on "Server Module Mode". After startup, Julius waits for tcp/ip connection from client. Once connection is established, Julius start communication with the client to process incoming commands from the client, or to output recognition results, input trigger information and other system status to the client. The default port number is 10500.

`-record` *`dir`*

Auto-save all input speech data into the specified directory. Each segmented inputs are recorded each by one. The file name of the recorded data is generated from system time when the input ends, in a style of `YYYY.MMDD.HHMMSS.wav`. File format is 16bit monoral WAV. Invalid for mfcfile input.

With input rejection by `-rejectshort`, the rejected input will also be recorded even if they are rejected.

`-logfile` *`file`*

Save all log output to a file instead of standard output. (Rev.4.0)

`-nolog`

Disable all log output. (Rev.4.0)

`-help`

Output help message and exit.

Global options

Misc. options

`-C` *`jconffile`*

Load a jconf file. The options written in the file are expanded at the point. This option can be used within other jconf file.

`-version`

Print version information to standard error, and exit.

`-setting`

Print engine setting information to standard error, and exit.

`-quiet`

Output less log. For result, only the best word sequence will be printed.

`-debug`

(For debug) output enoumous internal message and debug information to log.

`-check {wchmm|trellis|triphone}`

For debug, enter interactive check mode.

Audio input

`-input {mic|rawfile|mfcfile|adinnet|stdin|netaudio}`

Choose speech input source. ’file’ or ’rawfile’ for waveform file, ’htkparam’ or ’mfcfile’ for HTK parameter file. Users will be prompted to enter the file name from stdin, or you can use "-filelist" option to specify list of files to process.

’mic’ is to get audio input from live microphone device, and ’adinnet’ means receiving waveform data via tcpip network from an adinnet client. ’netaudio’ is from DatLink/NetAudio input, and ’stdin’ means data input from standard input.

For waveform file input, only WAV (no compression) and RAW (noheader, 16bit, big endian) are supported by default. Other format can be read when compiled with `libsnd` library. To see what format is actually supported, see the help message using option "-help". For stdin input, only WAV and RAW is supported. (default: mfcfile)

`-filelist` *`filename`*

(With -input rawfile|mfcfile) perform recognition on all files listed in the file. The file should contain an input file per line. Engine ends when all of the files are processed.

`-notypecheck`

By default, Julius checks the input parameter type whether it matches the AM or not. This option will disable the check and use the input vector as is.

`-48`

Record input with 48kHz sampling, and down-sample it to 16kHz on-the-fly. This option is valid for 16kHz model only. The down-sampling routine was ported from **sptk**. (Rev. 4.0)

`-NA` *`devicename`*

Host name for DatLink server input (`-input netaudio`).

`-adport` *`port_number`*

With `-input adinnet`, specify adinnet port number to listen. (default: 5530)

`-nostrip`

Julius by default removes successive zero samples in input speech data. This option inhibits this removal.

`-zmean`
`-nozmean`

This option enables/disables DC offset removal of input waveform. Offset will be estimated from the whole input. For microphone / network input, zero mean of the first 48000 samples (3 seconds in 16kHz sampling) will be used for the estimation. (default: disabled)

This option uses static offset for the channel. See also `-zmeansource` for frame-wise offset removal.

Speech segment detection by level and zero-cross

`-cutsilence`
`-nocutsilence`

Turn on / off the speech detection by level and zero-cross. Default is on for mic / adinnet input, off for files.

`-lv` *thres*

Level threshold for speech input detection. Values should be from 0 to 32767.

`-zc` *thres*

Zero crossing threshold per second. Only waves over the level threshold (`-lv`) will be counted. (default: 60)

`-headmargin` *msec*

Silence margin at the start of speech segment in milliseconds. (default: 300)

`-tailmargin` *msec*

Silence margin at the end of speech segment in milliseconds. (default: 400)

`-rejectshort` *msec*

Reject input shorter than specified milliseconds. Search will be terminated and no result will be output.

Input rejection by average power

This feature will be enabled by `--enable-power-reject` on compilation. Should be used with Decoder VAD or GMM VAD. Valid for real-time input only.

`-powerthres` *thres*

Reject the inputted segment by its average energy. If the average energy of the last recognized input is below the threshold, Julius will reject the input. (Rev.4.0)

This option is valid when `--enable-power-reject` is specified at compilation time.

Gaussian mixture model

GMM will be used for input rejection by accumurated score, or for GMM-based frontend VAD when `--enable-gmm-vad` is specified.

NOTE: You should also set the proper MFCC parameters required for the GMM, specifying the acoustic parameters described in AM section `-AM_GMM`.

`-gmm` *hmmdefs_file*

GMM definition file in HTK format. If specified, GMM-based input verification will be performed concurrently with the 1st pass, and you can reject the input according to the result as specified by `-gmmreject`. The GMM should be defined as one-state HMMs.

`-gmmnum` *`number`*

Number of Gaussian components to be computed per frame on GMM calculation. Only the N-best Gaussians will be computed for rapid calculation. The default is 10 and specifying smaller value will speed up GMM calculation, but too small value (1 or 2) may cause degradation of identification performance.

`-gmmreject` *`string`*

Comma-separated list of GMM names to be rejected as invalid input. When recognition, the log likelihoods of GMMs accumulated for the entire input will be computed concurrently with the 1st pass. If the GMM name of the maximum score is within this string, the 2nd pass will not be executed and the input will be rejected.

`-gmmmargin` *`frames`*

Head margin for GMM-based VAD in frames. (Rev.4.0)

This option will be valid only if compiled with `--enable-gmm-vad`.

Decoding option

Real-time processing means concurrent processing of MFCC computation 1st pass decoding. By default, real-time processing on the pass is on for microphone / adinnet / netaudio input, and for others.

`-realtime`
`-norealtime`

Explicitly switch on / off real-time (pipe-line) processing on the first pass. The default is off for file input, and on for microphone, adinnet and NetAudio input. This option relates to the way CMN and energy normalization is performed: if off, they will be done using average features of whole input. If on, MAP-CMN and energy normalization to do rea-time processing.

Instance declaration for multi decoding

The following arguments will create a new configuration set with default parameters, and switch current set to it. Jconf parameters specified after the option will be set into the current set.

To do multi-model decoding, these argument should be specified at the first of each model / search instances with different names. Any options before the first instance definition will be IGNORED.

When no instance definition is found (as older version of Julius), all the options are assigned to a default instance named "_default".

Please note that decoding with a single LM and multiple AMs is not fully supported. For example, you may want to construct the jconf file as following.

```
-AM am_1 -AM am_2
-LM lm (LM spec..)
-SR search1 am_1 lm
-SR search2 am_2 lm
```

This type of model sharing is not supported yet, since some part of LM processing depends on the assigned AM. Instead, you can get the same result by defining the same LMs for each AM, like this:

```
-AM am_1 -AM am_2
-LM lm_1 (LM spec..)
-LM lm_2 (same LM spec..)
-SR search1 am_1 lm_1
-SR search2 am_2 lm_2
```

`-AM` *`name`*

Create a new AM configuration set, and switch current to the new one. You should give a unique name. (Rev.4.0)

`-LM` *`name`*

Create a new LM configuration set, and switch current to the new one. You should give a unique name. (Rev.4.0)

`-SR` *`name am_name lm_name`*

Create a new search configuration set, and switch current to the new one. The specified AM and LM will be assigned to it. The *`am_name`* and *`lm_name`* can be either name or ID number. You should give a unique name. (Rev.4.0)

`-AM_GMM`

A special command to switch AM configuration set for specifying speech analysis parameters of GMM. The current AM will be switched to the GMM specific one already reserved, so be careful not to confuse with normal AM configurations. (Rev.4.0)

Language model (`-LM`)

Only one type of LM can be specified for a LM configuration. If you want to use multi model, you should define them one by one, each as a new LM.

N-gram

`-d` *`bingram_file`*

Use binary format N-gram. An ARPA N-gram file can be converted to Julius binary format by **mkbingram**.

`-nlr` *`arpa_ngram_file`*

A forward, left-to-right N-gram language model in standard ARPA format. When both a forward N-gram and backward N-gram are specified, Julius uses this forward 2-gram for the 1st pass, and the backward N-gram for the 2nd pass.

Since ARPA file often gets huge and requires a lot of time to load, it may be better to convert the ARPA file to Julius binary format by **mkbingram**. Note that if both forward and backward N-gram is used for recognition, they together should be converted to a single binary.

When only a forward N-gram is specified by this option and no backward N-gram specified by `-nrl`, Julius performs recognition with only the forward N-gram. The 1st pass will use the 2-gram entry in the given N-gram, and The 2nd pass will use the given N-gram, with converting forward probabilities to backward probabilities by Bayes rule. (Rev.4.0)

`-nrl` *`arpa_ngram_file`*

A backward, right-to-left N-gram language model in standard ARPA format. When both a forward N-gram and backward N-gram are specified, Julius uses the forward 2-gram for the 1st pass, and this backward N-gram for the 2nd pass.

Since ARPA file often gets huge and requires a lot of time to load, it may be better to convert the ARPA file to Julius binary format by **mkbingram**. Note that if both forward and backward N-gram is used for recognition, they together should be converted to a single binary.

When only a backward N-gram is specified by this option and no forward N-gram specified by `-nlr`, Julius performs recognition with only the backward N-gram. The 1st pass will use the forward 2-gram probability computed from the backward 2-gram using Bayes rule. The 2nd pass fully use the given backward N-gram. (Rev.4.0)

`-v` *`dict_file`*

Word dictionary file.

`-silhead` *`word_string`* `-siltail` *`word_string`*

Silence word defined in the dictionary, for silences at the beginning of sentence and end of sentence. (default: "<s>", "</s>")

`-iwspword`

Add a word entry to the dictionary that should correspond to inter-word pauses. This may improve recognition accuracy in some language model that has no explicit inter-word pause modeling. The word entry to be added can be changed by `-iwspentry`.

`-iwspentry` *`word_entry_string`*

Specify the word entry that will be added by `-iwspword`. (default: "<UNK> [sp] sp sp")

`-sepnum` *`number`*

Number of high frequency words to be isolated from the lexicon tree, to ease approximation error that may be caused by the one-best approximation on 1st pass. (default: 150)

Grammar

Multiple grammars can be specified by using `-gram` and `-gramlist`. When you specify grammars using these options multiple times, all of them will be read at startup. Note that this is unusual behavior from other options (in normal Julius option, last one override previous ones). You can use `-nogram` to reset the already specified grammars at that point.

`-gram  gramprefix1[,gramprefix2[,gramprefix3,...]]`

Comma-separated list of grammars to be used. the argument should be prefix of a grammar, i.e. if you have `foo.dfa` and `foo.dict`, you can specify them by single argument `foo`. Multiple grammars can be specified at a time as a comma-separated list.

`-gramlist` *`list_file`*

Specify a grammar list file that contains list of grammars to be used. The list file should contain the prefixes of grammars, each per line. A relative path in the list file will be treated as relative to the list file, not the current path or configuration file.

`-dfa` *`dfa_file`* `-v` *`dict_file`*

An old way of specifying grammar files separately.

`-nogram`

Remove the current list of grammars already specified by `-gram`, `-gramlist`, `-dfa` and `-v`.

Isolated word

Multiple dictionary can be specified by using `-w` and `-wlist`. When you specify multiple times, all of them will be read at startup. You can use `-nogram` to reset the already specified dictionaries at that point.

`-w` *`dict_file`*

Word dictionary for isolated word recognition. File format is the same as other LM. (Rev.4.0)

`-wlist` *`list_file`*

Specify a dictionary list file that contains list of dictionaries to be used. The list file should contain the file name of dictionaries, each per line. A relative path in the list file will be treated as relative to the list file, not the current path or configuration file. (Rev.4.0)

`-nogram`

Remove the current list of dictionaries already specified by `-w` and `-wlist`.

`-wsil` *`head_sil_model_name tail_sil_model_name sil_context_name`*

On isolated word recognition, silence models will be appended to the head and tail of each word at recognition. This option specifies the silence models to be appended. *`sil_context_name`* is the name of the head sil model and tail sil model as a context of word head phone and tail phone. For example, if you specify `-wsil silB silE sp`, a word with phone sequence `b eh t` will be translated as `silB sp-b+eh b-eh+t eh-t+sp silE`. (Rev.4.0)

User-defined LM

`-userlm`

Declare to use user LM defined in program. This option should be specified if you use user-defined LM function. (Rev.4.0)

Misc LM options

`-forcedict`

Ignore dictionary errors and force running. Words with errors will be skipped at startup.

Acoustic model and speech analysis (`-AM`) (`-AM_GMM`)

Acoustic analysis parameters are included in this section, since the AM defines the required parameter. You can use different MFCC type for each AM. For GMM, the same parameter should be specified after `-AM_GMM`

When using multiple AM, the values of `-smpPeriod`, `-smpFreq`, `-fsize` and `-fshift` should have the same value among all AMs.

acoustic HMM and parameters

`-h` *`hmmdef_file`*

Acoustic HMM definition file. File should be in HTK ascii format, or Julius binary format. You can convert HTK ascii hmmdefs to Julius binary format by **mkbinhmm**.

`-hlist` *`hmmlist_file`*

HMMList file for phone mapping. This options is required when using a triphone model. This file provides a mapping between logical triphone names genertated from the dictionary and defined HMM names in hmmdefs.

`-tmix` *`number`*

Specify the number of top Gaussians to be calculted in a mixture codebook. Small number will speed up the acoustic computation namely in a tied-mixture model, but AM accuracy may get worse on too small value. (default: 2)

`-spmodel` *`name`*

Specify an HMM name that corresponds to short-pause model in HMM. This option will affect various aspects in recognition: short-pause skipping process on grammar recognition, word-end short-pause model insertion with `-iwsp` on N-gram recognition, or short-pause segmentation (`-spsegment`). (default: "sp")

`-multipath`

Enable multi-path mode. Multi-path mode expand state transition availability to allow model-skipping, or multiple output/input transitions in HMMs. However, since defining additional word begin / end node and perform extra transition check on decoding, the beam width may be required to set larger and recognition becomes a bit slower.

By default (without this option), Julius automatically check the transition type of specified HMMs, and enable the multi-path mode if required. You can force Julius to enable multi-path mode with this option. (rev.4.0)

`-gprune {safe|heuristic|beam|none|default}`

Set Gaussian pruning algotrihm to use. The default setting will be set according to the model type and engine setting. "default" will force accepting the default setting. Set this to "none" to disable pruning and perform full computation. "safe" gualantees the top N Gaussians to be computed. "heuristic" and "beam" do more aggressive computational cosst reduction, but may result in small loss of accuracy model (default: 'safe' (standard), 'beam' (fast) for tied mixture model, 'none' for non tied-mixture model).

`-iwcd1 {max|avg|best number}`

Select method to approximate inter-word triphone on the head and tail of a word in the first pass.

"max" will apply the maximum likelihood of the same context triphones. "avg" will apply the average likelihood of the same context triphones. "best number" will apply the average of top N-best likelihoods of the same context triphone.

Default is "best 3" for use with N-gram, and "avg" for grammar and word. When this AM is shared by LMs of both type, latter one will be chosen.

`-iwsppenalty` *`float`*

Short pause insertion penalty for appended short pauses by `-iwsp`.

`-gshmm` *`hmmdef_file`*

If this option is specified, Julius performs Gaussian Mixture Selection for efficient decoding. The hmmdefs should be a monophone model generated from an ordinary monophone HMM model, using **mkgshmm**.

`-gsnum` *`number`*

On GMS, specify number of monophone state from top to compute the detailed corresponding triphones. (default: 24)

Speech analysis parameters

`-smpPeriod` *`period`*

Set sampling frequency of input speech by its sampling period, in unit of 100 nanoseconds. Sampling rate can also be specified by `-smpFreq`. Please note that the input frequency should be the same as trained conditions of acoustic model you use. (default: 625 = 16000Hz)

This option corresponds to the HTK Option "SOURCERATE". The same value can be given to this option.

When using multiple AM, this value should be the same among all AMs.

`-smpFreq` *`Hz`*

Set sampling frequency of input speech in Hz. Sampling rate can also be specified using "-smpPeriod". Please note that this frequency should be the same as the trained conditions of acoustic model you use. (default: 16000)

When using multiple AM, this value should be the same among all AMs.

`-fsize` *`sample_num`*

Window size in number of samples. (default: 400)

This option corresponds to the HTK Option "WINDOWSIZE", but value should be in samples (HTK value / smpPeriod).

When using multiple AM, this value should be the same among all AMs.

`-fshift` *`sample_num`*

Frame shift in number of samples. (default: 160)

This option corresponds to the HTK Option "TARGETRATE", but value should be in samples (HTK value / smpPeriod).

When using multiple AM, this value should be the same among all AMs.

`-preemph` *`float`*

Pre-emphasis coefficient. (default: 0.97)

This option corresponds to the HTK Option "PREEMCOEF". The same value can be given to this option.

`-fbank` *`num`*

Number of filterbank channels. (default: 24)

This option corresponds to the HTK Option "NUMCHANS". The same value can be given to this option. Be aware that the default value differs from HTK (22).

`-ceplif` *`num`*

Cepstral liftering coefficient. (default: 22)

This option corresponds to the HTK Option "CEPLIFTER". The same value can be given to this option.

`-rawe`
`-norawe`

Enable/disable using raw energy before pre-emphasis (default: disabled)

This option corresponds to the HTK Option "RAWENERGY". Be aware that the default value differs from HTK (enabled at HTK, disabled at Julius).

`-enormal`
`-noenormal`

Enable/disable normalizing log energy. On live input, this normalization will be approximated from the average of last input. (default: disabled)

This option corresponds to the HTK Option "ENORMALISE". Be aware that the default value differs from HTK (enabled at HTK, disabled at Julius).

`-escale` *`float_scale`*

Scaling factor of log energy when normalizing log energy. (default: 1.0)

This option corresponds to the HTK Option "ESCALE". Be aware that the default value differs from HTK (0.1).

`-silfloor` *`float`*

Energy silence floor in dB when normalizing log energy. (default: 50.0)

This option corresponds to the HTK Option "SILFLOOR".

`-delwin` *`frame`*

Delta window size in number of frames. (default: 2)

This option corresponds to the HTK Option "DELTAWINDOW". The same value can be given to this option.

`-accwin` *`frame`*

Acceleration window size in number of frames. (default: 2)

This option corresponds to the HTK Option "ACCWINDOW". The same value can be given to this option.

`-hifreq` *`Hz`*

Enable band-limiting for MFCC filterbank computation: set upper frequency cut-off. Value of -1 will disable it. (default: -1)

This option corresponds to the HTK Option "HIFREQ". The same value can be given to this option.

`-lofreq` *`Hz`*

Enable band-limiting for MFCC filterbank computation: set lower frequency cut-off. Value of -1 will disable it. (default: -1)

This option corresponds to the HTK Option "LOFREQ". The same value can be given to this option.

`-zmeanframe`
`-nozmeanframe`

With speech input, this option enables/disables frame-wise DC offset removal. This corresponds to HTK configuration ZMEANSOURCE. This cannot be used with "-zmean". (default: disabled)

Real-time cepstral mean normalization

`-cmnload` *`file`*

Load initial cepstral mean vector from file on startup. The file shoudld be one saved by `-cmnsave`. Loading an initial cepstral mean enables Julius to better recognize the first utterance on a microphone / network input.

`-cmnsave` *`file`*

Save cepstral mean vector at each input. The parameters will be saved to the file at each input end, so the output file always keeps the last cepstral mean. If output file already exist, it will be overridden.

`-cmnupdate` `-cmnnoupdate`

Control whether to update the cepstral mean at each input on microphone / network input. Disabling this and specifying `-cmnload` will make engine to use the initial cepstral mean parmanently.

`-cmnmapweight` *`float`*

Specify weight of initial cepstral mean for MAP-CMN. Specify larger value to retain the initial cepstral mean for a longer period, and smaller value to rely more on the current input. (default: 100.0)

Spectral subtraction

`-sscalc`

Perform spectral subtraction using head part of each file. Valid only for raw speech file input. Conflict with `-ssload`.

`-sscalclen` *`msec`*

With `-sscalc`, specify the length of head part silence in milliseconds. (default: 300)

`-ssload` *`file`*

Perform spectral subtraction for speech input using pre-estimated noise spectrum from file. The noise spectrum should be computed beforehand by **mkss**. Valid for all speech input. Conflict with `-sscalc`.

`-ssalpha` *float*

Alpha coefficient of spectral subtraction for **-sscalc** and **-ssload**. Noise will be subtracted stronger as this value gets larger, but distortion of the resulting signal also becomes remarkable. (default: 2.0)

`-ssfloor` *float*

Flooring coefficient of spectral subtraction. The spectral power that goes below zero after subtraction will be substituted by the source signal with this coefficient multiplied. (default: 0.5)

Misc AM options

`-htkconf` *file*

Parse the given HTK Config file, and set corresponding parameters to Julius. When using this option, the default parameter values are switched from Julius defaults to HTK defaults.

Recognizer and search (`-SR`)

Default values for beam width and LM weights will change according to compile-time setup of JuliusLib and model specification. Please see the startup log for the actual values.

General parameters

`-inactive`

Start this recognition process instance with inactive state. (Rev.4.0)

`-1pass`

Perform only the first pass. This mode is automatically set at isolated word recognition.

`-no_ccd`
`-force_ccd`

Normally Julius determines whether the specified acoustic model is a context-dependent model from the model names, i.e., whether the model names contain character + and -. You can explicitly specify by these options to avoid mis-detection. These option will override automatic detection.

`-cmalpha` *float*

Smoothing patemeter for confidence scoring. (default: 0.05)

`-iwsp`

(Multi-path mode only) Enable inter-word context-free short pause handling. This option appends a skippable short pause model for every word end. The added model will be skipped on inter-word context handling. The HMM model to be appended can be specified by `-spmodel`.

`-transp` *`float`*

Additional insertion penalty for transparent words. (default: 0.0)

`-demo`

Equivalent to `-progout -quiet`.

1st pass parameters

`-lmp` *`weight penalty`*

(N-gram) Language model weights and word insertion penalties for the first pass.

`-penalty1` *`penalty`*

(Grammar) word insertion penalty for the first pass. (default: 0.0)

`-b` *`width`*

Beam width for rank beam in number of HMM nodes on the first pass. This value defines search width on the 1st pass, and has great effect on the total processing time. Smaller width will speed up the decoding, but too small value will result in a substantial increase of recognition errors due to search failure. Larger value will make the search stable and will lead to failure-free search, but processing time and memory usage will grow in proportion to the width.

The default value is dependent on acoustic model type: 400 (monophone), 800 (triphone), or 1000 (triphone, setup=v2.1)

`-nlimit` *`num`*

Upper limit of token per node. This option is valid when `--enable-wpair` and `--enable-wpair-nlimit` are enabled at compilation time.

`-progout`

Enable progressive output of the partial results on the first pass.

`-proginterval` *`msec`*

Set the output time interval of `-progout` in milliseconds.

2nd pass parameters

`-lmp2` *`weight penalty`*

(N-gram) Language model weights and word insertion penalties for the second pass.

`-penalty2` *`penalty`*

(Grammar) word insertion penalty for the second pass. (default: 0.0)

`-b2` *`width`*

Envelope beam width (number of hypothesis) in second pass. If the count of word expantion at a certain length of hypothesis reaches this limit while search, shorter hypotheses are not expanded further. This prevents search to fall in breadth-first-like status stacking on the same position, and improve search failure. (default: 30)

`-sb` *`float`*

Score envelope width for enveloped scoring. When calculating hypothesis score for each generated hypothesis, its trellis expansion and viterbi operation will be pruned in the middle of the speech if score on a frame goes under the width. Giving small value makes the second pass faster, but computation error may occur. (default: 80.0)

`-s` *`num`*

Stack size, i.e. the maximum number of hypothesis that can be stored on the stack during the search. A larger value may give more stable results, but increases the amount of memory required. (default: 500)

`-m` *`count`*

Number of expanded hypotheses required to discontinue the search. If the number of expanded hypotheses is greater then this threshold then, the search is discontinued at that point. The larger this value is, The longer Julius gets to give up search. (default: 2000)

`-n` *`num`*

The number of candidates Julius tries to find. The search continues till this number of sentence hypotheses have been found. The obtained sentence hypotheses are sorted by score, and final result is displayed in the order (see also the `-output`). The possibility that the optimum hypothesis is correctly found increases as this value gets increased, but the processing time also becomes longer. The default value depends on the engine setup on compilation time: 10 (standard) or 1 (fast or v2.1)

`-output` *`num`*

The top N sentence hypothesis to be output at the end of search. Use with `-n` (default: 1)

`-lookuprange` *`frame`*

When performing word expansion on the second pass, this option sets the number of frames before and after to look up next word hypotheses in the word trellis. This prevents the omission of short words, but with a large value, the number of expanded hypotheses increases and system becomes slow. (default: 5)

`-looktrellis`

(Grammar) Expand only the words survived on the first pass instead of expanding all the words predicted by grammar. This option makes second pass decoding slightly faster especially for large vocabulary condition, but may increase deletion error of short words. (default: disabled)

Short-pause segmentation

When compiled with `--enable-decoder-vad`, the short-pause segmentation will be extended to support decoder-based VAD.

`-spsegment`

Enable short-pause segmentation mode. Input will be segmented when a short pause word (word with only silence model in pronunciation) gets the highest likelihood at certain successive frames on the first pass. When detected segment end, Julius stop the 1st pass at the point, perform 2nd pass, and continue with next segment. The word context will be considered among segments. (Rev.4.0)

When compiled with `--enable-decoder-vad`, this option enables decoder-based VAD, to skip long silence.

`-spdur` *`frame`*

Short pause duration length to detect end of input segment, in number of frames. (default: 10)

`-pausemodels` *`string`*

A comma-separated list of pause model names to be used at short-pause segmentation. The word with only the pause models will be treated as "pause word" for pause detectionin. If not specified, name of `-spmodel`, `-silhead` and `-siltail` will be used. (Rev.4.0)

`-spmargin` *`frame`*

Backstep margin at trigger up for decoder-based VAD. (Rev.4.0)

This option will be valid only if compiled with `--enable-decoder-vad`.

`-spdelay` *`frame`*

Trigger decision delay frame at trigger up for decoder-based VAD. (Rev.4.0)

This option will be valid only if compiled with `--enable-decoder-vad`.

Lattice / confusion network output

`-lattice`
`-nolattice`

Enable / disable generation of word graph. Search algorithm also has changed to optimize for better word graph generation, so the sentence result may not be the same as normal N-best recognition. (Rev.4.0)

`-confnet`
`-noconfnet`

Enable / disable generation of confusion network. Enabling this will also activates `-lattice` internally. (Rev.4.0)

`-graphrange` *`frame`*

Merge same words at neighbor position at graph generation. If the position of same words differs smaller than this value, they will be merged. The default is 0 (allow merging on exactly the same location) and specifying larger value will result in smaller graph output. Setting to -1 will disable merging, in that case same words on the same location of different scores will be left as they are. (default: 0)

`-graphcut` *`depth`*

Cut the resulting graph by its word depth at post-processing stage. The depth value is the number of words to be allowed at a frame. Setting to -1 disables this feature. (default: 80)

`-graphboundloop` *`count`*

Limit the number of boundary adjustment loop at post-processing stage. This parameter prevents Julius from blocking by infinite adjustment loop by short word oscillation. (default: 20)

`-graphsearchdelay`
`-nographsearchdelay`

When "-graphsearchdelay" option is set, Julius modifies its graph generation alogrithm on the 2nd pass not to terminate search by graph merging, until the first sentence candidate is found. This option may improve graph accuracy, especially when you are going to generate a huge word graph by setting broad search. Namely, it may result in better graph accuracy when you set wide beams on both 1st pass `-b` and 2nd pass `-b2`, and large number for `-n`. (default: disabled)

Multi-gram / multi-dic output

`-multigramout`
`-nomultigramout`

On grammar recognition using multiple grammars, Julius will output only the best result among all grammars. Enabling this option will make Julius to output result for each grammar. (default: disabled)

Forced alignment

`-walign`

Do viterbi alignment per word units for the recognition result. The word boundary frames and the average acoustic scores per frame will be calculated.

`-palign`

Do viterbi alignment per phone units for the recognition result. The phone boundary frames and the average acoustic scores per frame will be calculated.

`-salign`

Do viterbi alignment per state for the recognition result. The state boundary frames and the average acoustic scores per frame will be calculated.

EXAMPLES

For examples of system usage, refer to the tutorial section in the Julius documents.

NOTICE

Note about jconf files: relative paths in a jconf file are interpreted as relative to the jconf file itself, not to the current directory.

SEE ALSO

julian(1), jcontrol(1), adinrec(1), adintool(1), mkbingram(1), mkbinhmm(1), mkgsmm(1), wav2mfcc(1), mkss(1)

http://julius.sourceforge.jp/en/

DIAGNOSTICS

Julius normally will return the exit status 0. If an error occurs, Julius exits abnormally with exit status 1. If an input file cannot be found or cannot be loaded for some reason then Julius will skip processing for that file.

BUGS

There are some restrictions to the type and size of the models Julius can use. For a detailed explanation refer to the Julius documentation. For bug-reports, inquires and comments please contact julius-info at lists.sourceforge.jp.

COPYRIGHT

AUTHORS

Rev.1.0 (1998/02/20)

Designed by Tatsuya KAWAHARA and Akinobu LEE (Kyoto University)

Development by Akinobu LEE (Kyoto University)

Rev.1.1 (1998/04/14)
Rev.1.2 (1998/10/31)
Rev.2.0 (1999/02/20)
Rev.2.1 (1999/04/20)
Rev.2.2 (1999/10/04)
Rev.3.0 (2000/02/14)
Rev.3.1 (2000/05/11)

Development of above versions by Akinobu LEE (Kyoto University)

Rev.3.2 (2001/08/15)
Rev.3.3 (2002/09/11)
Rev.3.4 (2003/10/01)
Rev.3.4.1 (2004/02/25)
Rev.3.4.2 (2004/04/30)

Development of above versions by Akinobu LEE (Nara Institute of Science and Technology)

Rev.3.5 (2005/11/11)
Rev.3.5.1 (2006/03/31)
Rev.3.5.2 (2006/07/31)
Rev.3.5.3 (2006/12/29)
Rev.4.0 (2007/12/19)

Development of above versions by Akinobu LEE (Nagoya Institute of Technology)

THANKS TO

From rev.3.2, Julius is released by the "Information Processing Society, Continuous Speech Consortium".

The Windows DLL version was developed and released by Hideki BANNO (Nagoya University).

The Windows Microsoft Speech API compatible version was developed by Takashi SUMIYOSHI (Kyoto University).