Package: audio.whisper (via r-universe)

January 9, 2026

Type Package

Title Transcribe Audio Files using the ``Whisper'' Automatic Speech Recognition Model

Version 0.5.0

Maintainer Jan Wijffels <jwijffels@bnosac.be>

Description The ``Whisper'' models are trained for speech recognition and translation tasks, capable of transcribing speech audio into the text in the language it is spoken (Automatic Speech Recognition) as well as translated into English (speech translation). The package is an ``Rcpp'' wrapper around the standalone C++ implementation provided at <https://github.com/ggerganov/whisper.cpp>. There are 10 pretrained models available of different sizes and language capabilities. ``Whisper'' is explained in the paper: 'Robust Speech Recognition via Large-Scale Weak Supervision' by Radford et al. (2022), available at <doi:10.48550/arXiv.2212.04356>.

License MIT + file LICENSE

URL https://github.com/bnosac/audio.whisper

Encoding UTF-8

Depends R (>= 2.10)

Imports Rcpp (>= 0.11.5), utils

Suggests tinytest, audio, data.table (>= 1.12.4), audio.vadwebrtc (>= 0.2.0)

LinkingTo Rcpp

SystemRequirements GNU make, CMake (>= 3.5)

RoxygenNote 7.3.3

Remotes bnosac/audio.vadwebrtc

Config/pak/sysreqs cmake make

Repository https://test.r-universe.dev

Date/Publication 2026-01-07 22:30:56 UTC

RemoteUrl https://github.com/bnosac/audio.whisper

RemoteRef 0.5.0

RemoteSha 03904914c7dd3b35b15719a6b4901635dedcdbb9

Contents

`predict.whisper` *Transcribe audio files using a Whisper model*

Description

Automatic Speech Recognition using Whisper on 16-bit WAV files

Usage

```
## S3 method for class 'whisper'
predict(
  object,
  newdata,
  type = c("transcribe", "translate"),
  language = "auto",
  sections = data.frame(start = integer(), duration = integer()),
  offset = 0L,
  duration = 0L,
  trim = FALSE,
  trace = TRUE,
  vad = FALSE,
 vad_model = system.file(package = "audio.whisper", "silero", "ggml-silero-v5.1.2.bin"),
  ...
)
```

Arguments

`object`	a whisper object
`newdata`	the path to a 16-bit .wav file

`type`	character string with the type of prediction, can either be 'transcribe' or 'translate', where 'translate' will put the spoken text in English.
`language`	the language of the audio. Defaults to 'auto'. For a list of all languages the model can handle: see `whisper_languages`.
`sections`	a data.frame with columns start and duration (measured in milliseconds) indicating voice segments to transcribe. This will make a new audio file with these sections, do the transcription and make sure the from/to timestamps are aligned to the original audio file. Defaults to transcribing the full audio file.
`offset`	an integer vector of offsets in milliseconds to start the transcription. Defaults to 0 - indicating to transcribe the full audio file.
`duration`	an integer vector of durations in milliseconds indicating how many milliseconds need to be transcribed from the corresponding `offset` onwards. Defaults to 0 - indicating to transcribe the full audio file.
`trim`	logical indicating to trim leading/trailing white space from the transcription using `trimws`. Defaults to `FALSE`.
`trace`	logical indicating to print the trace of the evolution of the transcription. Defaults to `TRUE`
`vad`	logical indicating to perform Voice Activity Detection using a Silero model
`vad_model`	string with the path to a .bin file containing the Silero model. Defaults to the Silero v5.1.2 shipped in this package.
`...`	further arguments, directly passed on to the C++ function, for expert usage only and subject to naming changes. See the details.

Details

- token_timestamps: logical indicating to get the timepoints of each token
- n_threads: how many threads to use to make the prediction. Defaults to 1
- prompt: the initial prompt to pass on the model. Defaults to ''
- entropy_thold: entropy threshold for decoder fail. Defaults to 2.4
- logprob_thold: log probability threshold for decoder fail. Defaults to -1
- beam_size: beam size for beam search. Defaults to -1
- best_of: number of best candidates to keep. Defaults to 5
- max_context: maximum number of text context tokens to store. Defaults to -1
- diarize: logical indicating to perform speaker diarization for audio with more than 1 channel

If sections are provided If multiple offsets/durations are provided

Value

an object of class `whisper_transcription` which is a list with the following elements:

- n_segments: the number of audio segments
- data: a data.frame with the transcription with columns segment, segment_offset, text, from, to and optionally speaker if diarize=TRUE

- tokens: a data.frame with the transcription tokens with columns segment, token_id, token, token_prob indicating the token probability given the context
- params: a list with parameters used for inference
- timing: a list with elements start, end and duration indicating how long it took to do the transcription

See Also

whisper, whisper_languages

Examples

```
model <- whisper("tiny")
audio <- system.file(package = "audio.whisper", "samples", "jfk.wav")
trans <- predict(model, newdata = audio)
trans <- predict(model, newdata = audio, language = "en")
trans <- predict(model, newdata = audio, language = "en", token_timestamps = TRUE)

audio <- system.file(package = "audio.whisper", "samples", "proficiat.wav")
model <- whisper("tiny")
trans <- predict(model, newdata = audio, language = "nl", type = "transcribe")
model <- whisper("tiny")
trans <- predict(model, newdata = audio, language = "nl", type = "translate")




## Predict using a quantised model
audio <- system.file(package = "audio.whisper", "samples", "jfk.wav")
path  <- system.file(package = "audio.whisper", "repo", "ggml-tiny-q5_1.bin")
model <- whisper(path)
trans <- predict(model, newdata = audio, language = "en", trace = FALSE)
trans <- predict(model, newdata = audio, language = "en", token_timestamps = TRUE)
## Predict using a quantised model with the GPU
model <- whisper(path, use_gpu = TRUE)
trans <- predict(model, newdata = audio, language = "en")
trans <- predict(model, newdata = audio, language = "en", token_timestamps = TRUE)
## Example of providing further arguments to predict.whisper
audio <- system.file(package = "audio.whisper", "samples", "stereo.wav")
trans <- predict(model, newdata = audio, language = "auto", diarize = TRUE)
```

---

predict.whisper_transcription

Predict to which channel a transcription section belongs

Description

Audio files containing 2 channels which were transcribed with `predict.whisper`, you can use the results of a Voice Activity Detection by channel (either with R packages `audio.vadwebrtc` or `audio.vadsilero`) to assign the text segments to each of the channels.
This is done by looking for each text segment how many seconds overlap there is with the voiced sections which are identified by the Voice Activity Detection.

Usage

```
## S3 method for class 'whisper_transcription'
predict(object, vad, type = "channel", threshold = 0, ...)
```

Arguments

`object`	an object of class `whisper_transcription` as returned by `predict.whisper`
`vad`	an object of class `webrtc-gmm-bychannel` as returned by function `VAD_channel` from R package `audio.vadwebrtc` with information of the detected voice in at least channels 1 and 2. ar a list with element vad_segments containing a data.frame with columns channel, start, end and has_voice with information at which second there was a voice in the audio
`type`	character string with currently only possible value: 'channel' which does a 2-speaker channel assignment
`threshold`	numeric in 0-1 range indicating if the difference between the probability that the segment was from the left channel 1 or the right channel 2 is smaller than this amount, the column `channel` will be set to 'both'. Defaults to 0.
`...`	not used

Value

an object of class `whisper_transcription` as documented in `predict.whisper` where element `data` contains the following extra columns indicating which channel the transcription is probably from

- channel: either 'left', 'right' or 'both' indicating the transcription segment was either from the left channel (1), the right channel (2) or probably from both as identified by the Voice Activity Detecion
- channel_probability: a number between 0 and 1 indicating for that specific segment the ratio of the amount of voiced seconds in the most probably channel to the sum of the amount of voiced seconds in the left + the right channel
- duration: how long (in seconds) the from-to segment is
- duration_voiced_left: how many seconds there was a voiced signal on the left channel (channel 1) as identified by `vad`
- duration_voiced_right: how many seconds there was a voiced signal on the right channel (channel 2) as identified by `vad`

See Also

`predict.whisper`

Examples

```
library(audio.whisper)
model <- whisper("tiny")
audio <- system.file(package = "audio.whisper", "samples", "stereo.wav")
trans <- predict(model, audio, language = "es")
## Not run:
library(audio.vadwebrtc)
vad   <- VAD_channel(audio, channels = "all", mode = "veryaggressive", milliseconds = 30)

## End(Not run)
vad   <- list(vad_segments = rbind(
  data.frame(channel = 1, start = c(0, 5, 15, 22), end = c(5, 9, 18, 23), has_voice = TRUE),
  data.frame(channel = 2, start = c(2, 9.5, 19, 22), end = c(2.5, 13.5, 21, 23), has_voice = TRUE)))
out <- predict(trans, vad, type = "channel", threshold = 0)
out$data
```

vad *Voice Activity Detection using Silero*

Description

Voice Activity Detection using Silero

Usage

```
vad(
  path = system.file(package = "audio.whisper", "samples", "jfk.wav"),
  vad_model = system.file(package = "audio.whisper", "silero", "ggml-silero-v5.1.2.bin"),
  threshold = 0.5,
  min_speech_duration = 250,
  min_silence_duration = 100,
  max_speech_duration = -1,
  pad = 30,
  overlap = 0.1,
  n_threads = 1,
  probabilities = FALSE,
  ...
)
```

Arguments

`path`	the path to the wav file
`vad_model`	the path to the VAD model. Defaults to the ggml-silero-v5.1.2.bin in the silero folder shipped with this package
`threshold`	VAD threshold for speech recognition. Defaults to 0.5.
`min_speech_duration`	VAD minimum speech duration of voiced speech in milliseconds. Defaults to 250 milliseconds

`min_silence_duration`	VAD minimum silence duration in milliseconds in order to split segments. Defaults to 100 milliseconds.
`max_speech_duration`	VAD maximum speech duration - auto-split longer speech segments. Defaults to -1.
`pad`	VAD speech padding in milliseconds to extend segments. Defaults to 30.
`overlap`	VAD samples overlap - seconds between segments - to allow a bit more context. Defaults to 0.1.
`n_threads`	multithreading - number of threads to use. Defaults to 1.
`probabilities`	logical indicating to return probabilities. Defaulst to FALSE.
`...`	passed on to the C++ silero_vad function

Value

an object of class `silero_vad` which is list with the following elements:

- n_segments: the number of voiced segments
- probabilities: the probabilities of voice in the audio
- data: a data.frame with columns: segment, from, to, has_voice where from/to are in seconds. The data contains only voices segments.
- params: a list of hyperparameters passed on to the model scoring function

See Also

`predict.whisper`

Examples

```
model <- system.file(package = "audio.whisper", "silero", "ggml-silero-v6.2.0.bin")
model <- system.file(package = "audio.whisper", "silero", "ggml-silero-v5.1.2.bin")
audio <- system.file(package = "audio.whisper", "samples", "jfk.wav")
voice <- vad(audio, vad_model = model)
voice <- vad(audio, threshold = 0.5, min_speech_duration = 1000, min_silence_duration = 100)
voice <- vad(audio, probabilities = TRUE)
```

`whisper` *Automatic Speech Recognition using Whisper*

Description

Automatic Speech Recognition using Whisper on 16-bit WAV files. Load the speech recognition model.

Usage

```
whisper(
  x,
  use_gpu = FALSE,
  flash_attn = TRUE,
  overwrite = FALSE,
  model_dir = Sys.getenv("WHISPER_MODEL_DIR", unset = getwd()),
  ...
)
```

Arguments

`x`	the path to a model, an object returned by `whisper_download_model` or a character string with the name of the model which can be passed on to `whisper_download_model`
`use_gpu`	logical indicating to use the GPU in case you have Metal or an NVIDIA GPU. Defaults to `FALSE`.
`flash_attn`	logical indicating to use flash attention. Defaults to `TRUE`.
`overwrite`	logical indicating to overwrite the model file if the model file was already downloaded, passed on to `whisper_download_model`. Defaults to `FALSE`.
`model_dir`	a path where the model will be downloaded to, passed on to `whisper_download_model`. Defaults to the environment variable `WHISPER_MODEL_DIR` and if this is not set, the current working directory
`...`	further arguments, passed on to the internal C++ function `whisper_load_model`

Value

an object of class `whisper` which is list with the following elements:

- file: path to the model
- model: an Rcpp pointer to the loaded Whisper model

See Also

`predict.whisper`

Examples

```
## Not run:
## Provide shorthands 'tiny', 'base', 'small', 'medium', ...
model <- whisper("tiny")
trans <- predict(model, newdata = system.file(package = "audio.whisper", "samples", "jfk.wav"))
trans
model <- whisper("base")
trans <- predict(model, newdata = system.file(package = "audio.whisper", "samples", "jfk.wav"))
trans
model <- whisper("small")
trans <- predict(model, newdata = system.file(package = "audio.whisper", "samples", "jfk.wav"))
trans
```

```
model <- whisper("medium")
trans <- predict(model, newdata = system.file(package = "audio.whisper", "samples", "jfk.wav"))
trans
model <- whisper("large-v1")
trans <- predict(model, newdata = system.file(package = "audio.whisper", "samples", "jfk.wav"))
trans
model <- whisper("large-v3-turbo-q8_0")
trans <- predict(model, newdata = system.file(package = "audio.whisper", "samples", "jfk.wav"))
trans

## Or download the model explicitely
path  <- whisper_download_model("tiny")
model <- whisper(path)
trans <- predict(model, newdata = system.file(package = "audio.whisper", "samples", "jfk.wav"))

## End(Not run)

## Or provide the path to the model you have downloaded previously
path  <- system.file(package = "audio.whisper", "repo", "ggml-tiny-q5_1.bin")
path
model <- whisper(path)
trans <- predict(model, newdata = system.file(package = "audio.whisper", "samples", "jfk.wav"),
                 language = "en")

## Add diarization
trans <- predict(model, newdata = system.file(package = "audio.whisper", "samples", "stereo.wav"),
                 language = "es", diarize = TRUE)
## Provide multiple offsets and durations to get the segments in there
trans <- predict(model, newdata = system.file(package = "audio.whisper", "samples", "stereo.wav"),
                 language = "es", diarize = TRUE,
                 offset = c( 650, 6060, 10230), duration = c(4990, 3830, 11650))
## Provide sections - this will make a new audio file and next do the transcription
if(require(data.table) && require(audio)){
trans <- predict(model, newdata = system.file(package = "audio.whisper", "samples", "stereo.wav"),
                 language = "es", diarize = TRUE,
                 sections = data.frame(start    = c( 650, 6060, 10230),
                                       duration = c(4990, 3830, 11650)))
}
```

`whisper_benchmark` *Benchmark a Whisper model*

Description

Benchmark a Whisper model to see how good it runs on your architecture by printing it's performance on fake data. https://github.com/ggerganov/whisper.cpp/issues/89

Usage

```
whisper_benchmark(
  object = whisper(system.file(package = "audio.whisper", "models",
    "for-tests-ggml-tiny.bin")),
  threads = 1
)
```

Arguments

`object`	a whisper object
`threads`	the number of threads to use, defaults to 1

Value

invisible()

See Also

`whisper`

Examples

```
## Not run:
model <- whisper("tiny", overwrite = FALSE)
whisper_benchmark(model)

## End(Not run)
```

`whisper_download_model` *Download a pretrained Whisper model*

Description

Download a pretrained Whisper model. The list of available models are

- tiny & tiny.en: 75 MB, RAM required: ~390 MB. Multilingual and English only version.
- base & base.en: 142 MB, RAM required: ~500 MB. Multilingual and English only version.
- small & small.en: 466 MB, RAM required: ~1.0 GB. Multilingual and English only version.
- medium & medium.en: 1.5 GB, RAM required: ~2.6 GB. Multilingual and English only version.
- large-v1, large-v2, large-v3: 2.9 GB, RAM required: ~4.7 GB. Multilingual
- quantised models: tiny-q5_1, tiny.en-q5_1, base-q5_1, base.en-q5_1, small-q5_1, small.en-q5_1, medium-q5_0, medium.en-q5_0, large-v2-q5_0 and large-v3-q5_0 (only - from version 1.5.4 onwards)

Note that the larger models may take longer than 60 seconds to download, so consider increasing the timeout option in R via `options(timeout = 120)`

Usage

```
whisper_download_model(
  x = c("tiny", "tiny.en", "base", "base.en", "small", "small.en", "medium", "medium.en",
     "large-v1", "large-v2", "large-v3", "large-v3-turbo", "large", "tiny-q5_1",
     "tiny-q8_0", "tiny.en-q5_1", "tiny.en-q8_0", "base-q5_1", "base-q8_0",
    "base.en-q5_1", "base.en-q8_0", "small-q5_1", "small-q8_0", "small.en-q5_1",
   "small.en-q8_0", "medium-q5_0", "medium-q8_0", "medium.en-q5_0", "medium.en-q8_0",
     "large-v2-q5_0", "large-v2-q8_0", "large-v3-q5_0", "large-v3-turbo-q5_0",
     "large-v3-turbo-q8_0"),
  model_dir = Sys.getenv("WHISPER_MODEL_DIR", unset = getwd()),
  repos = c("huggingface", "ggerganov"),
  version = c("1.8.2", "1.5.4", "1.2.1"),
  overwrite = TRUE,
  ...
)
```

Arguments

`x`	the name of the model
`model_dir`	a path where the model will be downloaded to. Defaults to the environment variable `WHISPER_MODEL_DIR` and if this is not set, the current working directory
`repos`	character string with the repository to download the model from. Either • 'huggingface': https://huggingface.co/ggerganov/whisper.cpp - the default • 'ggerganov': https://ggml.ggerganov.com/ - no longer supported as the resource by ggerganov can become unavailable
`version`	character string with the version of the model. Defaults to "1.8.2".
`overwrite`	logical indicating to overwrite the file if the file was already downloaded. Defaults to `TRUE` indicating it will download the model and overwrite the file if the file already existed. If set to `FALSE`, the model will only be downloaded if it does not exist on disk yet in the `model_dir` folder.
`...`	currently not used

Value

A data.frame with 1 row and the following columns:

- model: The model as provided by the input parameter `x`
- file_model: The path to the file on disk where the model was downloaded to
- url: The URL where the model was downloaded from
- download_success: A logical indicating if the download has succeeded or not due to internet connectivity issues
- download_message: A character string with the error message in case the downloading of the model failed

See Also

`whisper`, `predict.whisper`, `whisper_languages`

Examples

```
path <- whisper_download_model("tiny")
path <- whisper_download_model("tiny", overwrite = FALSE)
## Not run:
whisper_download_model("tiny.en")
whisper_download_model("base")
whisper_download_model("base.en")
whisper_download_model("small")
whisper_download_model("small.en")
whisper_download_model("medium")
whisper_download_model("medium.en")
whisper_download_model("large-v1")
whisper_download_model("large-v2")
whisper_download_model("large-v3")
whisper_download_model("large-v3-turbo")
whisper_download_model("tiny-q5_1")
whisper_download_model("tiny-q8_0")
whisper_download_model("base-q5_1")
whisper_download_model("base-q8_0")
whisper_download_model("small-q5_1")
whisper_download_model("small-q8_0")
whisper_download_model("medium-q5_0")
whisper_download_model("medium-q8_0")
whisper_download_model("large-v2-q5_0")
whisper_download_model("large-v2-q8_0")
whisper_download_model("large-v3-q5_0")
whisper_download_model("large-v3-turbo-q5_0")
whisper_download_model("large-v3-turbo-q8_0")

## End(Not run)
```

---

whisper_languages *Get the language capabilities of Whisper*

Description

Extract the list of languages a multilingual whisper model is able to handle

Usage

```
whisper_languages()
```

Value

a data.frame with columns id, language and language_label showing the languages

Examples

```
x <- whisper_languages()
x
```

Index