Package ‘whisper’

February 6, 2026

Title Native R 'torch' Implementation of 'OpenAI' 'Whisper'

Version 0.1.0

Description Speech-to-text transcription using a native R 'torch' implementation of 'OpenAI' 'Whisper' model <https://github.com/openai/whisper>. Supports multiple model sizes from tiny (39M parameters) to large-v3 (1.5B parameters) with integrated download from 'HuggingFace' <https://huggingface.co/> via the 'hfhub' package. Provides automatic speech recognition with optional language detection and translation to English. Audio preprocessing, mel spectrogram computation, and transformer-based encoder-decoder inference are all implemented in R using the 'torch' package.

License MIT + file LICENSE

Encoding UTF-8

URL https://github.com/cornball-ai/whisper

BugReports https://github.com/cornball-ai/whisper/issues

Imports torch, av, jsonlite, hfhub, safetensors, stats, utils

Suggests tinytest

NeedsCompilation no

Author Troy Hernandez [aut, cre],
cornball.ai [cph],
OpenAI [cph] (Whisper model architecture and mel filterbank data (MIT license))

Maintainer Troy Hernandez <troy@cornball.ai>

Repository CRAN

Date/Publication 2026-02-06 20:00:02 UTC

Contents

`apply_bpe` *Apply BPE Merges*

Description

Apply BPE Merges

Usage

```
apply_bpe(tokens, merge_ranks)
```

Arguments

`tokens`	Character vector of tokens
`merge_ranks`	Named vector of merge rankings

Value

Character vector after BPE merges

`audio_duration` *Get Audio Duration*

Description

Get Audio Duration

Usage

```
audio_duration(file)
```

Arguments

`file`	Path to audio file

Value

Duration in seconds

audio_to_mel *Convert Audio to Mel Spectrogram*

Description

Main preprocessing function that converts audio to the mel spectrogram format expected by Whisper.

Usage

```
audio_to_mel(file, n_mels = 80L, device = "auto", dtype = "auto")
```

Arguments

file	Path to audio file, or numeric vector of audio samples
n_mels	Number of mel bins (80 for most models, 128 for large-v3)
device	torch device for output tensor
dtype	torch dtype for output tensor

Value

torch tensor of shape (1, n_mels, 3000) for 30s audio

Examples

```
# Convert audio file to mel spectrogram
audio_file <- system.file("audio", "jfk.mp3", package = "whisper")
mel <- audio_to_mel(audio_file)
dim(mel)
```

byte_to_token *Convert Byte to BPE Token*

Description

GPT-2/Whisper uses a specific byte-to-unicode mapping.

Usage

```
byte_to_token(byte)
```

Arguments

byte	Integer byte value (0-255)

Value

Character token

`clean_text` *Clean Transcribed Text*

Description

Clean Transcribed Text

Usage

```
clean_text(text)
```

Arguments

`text`	Raw decoded text

Value

Cleaned text

`compute_stft` *Compute STFT Magnitude*

Description

Compute STFT Magnitude

Usage

```
compute_stft(audio, n_fft = WHISPER_N_FFT, hop_length = WHISPER_HOP_LENGTH)
```

Arguments

`audio`	Numeric vector of audio samples
`n_fft`	FFT window size
`hop_length`	Hop length between frames

Value

Complex STFT matrix

copy_if_exists *Copy Weight if Exists*

Description

Copy Weight if Exists

Usage

```
copy_if_exists(param, weights, name)
```

Arguments

param	Target parameter
weights	Weight dictionary
name	Weight name

create_decoder *Create Decoder from Config*

Description

Create Decoder from Config

Usage

```
create_decoder(config)
```

Arguments

config	Model configuration from whisper_config()

Value

WhisperDecoder module

`create_encoder` *Create Encoder from Config*

Description

Create Encoder from Config

Usage

```
create_encoder(config)
```

Arguments

`config`	Model configuration from whisper_config()

Value

WhisperEncoder module

`create_mel_filterbank_fallback` *Create Mel Filterbank (Fallback)*

Description

Create a mel filterbank matrix for converting STFT to mel spectrogram. Used when pre-computed filterbank is not available.

Usage

```
create_mel_filterbank_fallback(
  n_fft = WHISPER_N_FFT,
  n_mels = 80L,
  sample_rate = WHISPER_SAMPLE_RATE
)
```

Arguments

`n_fft`	FFT size
`n_mels`	Number of mel bins
`sample_rate`	Audio sample rate

Value

Mel filterbank matrix (n_mels x n_freqs) decode_bpe_bytes *Decode BPE Bytes Back to Text*

Description

Decode BPE Bytes Back to Text

Usage

```
decode_bpe_bytes(text)
```

Arguments

text	Text with BPE byte tokens

Value

Decoded text

decode_timestamp *Decode Timestamp Token*

Description

Decode Timestamp Token

Usage

```
decode_timestamp(token_id, model = "tiny")
```

Arguments

token_id	Token ID
model	Model name for correct token IDs

Value

Time in seconds

`download_tokenizer_files`
Download Tokenizer Files from HuggingFace

Description

Download Tokenizer Files from HuggingFace

Usage

```
download_tokenizer_files(model)
```

Arguments

`model`	Model name

`download_whisper_model`
Download Model from HuggingFace

Description

Download Whisper model weights and tokenizer files from HuggingFace. In interactive sessions, asks for user consent before downloading.

Usage

```
download_whisper_model(model = "tiny", force = FALSE)
```

Arguments

`model`	Model name: "tiny", "base", "small", "medium", "large-v3"
`force`	Re-download even if exists

Value

Path to model directory (invisibly)

Examples

```
if (interactive()) {
  # Download tiny model (smallest, ~150MB)
  download_whisper_model("tiny")

  # Download larger model for better accuracy
  download_whisper_model("small")
}
```

`ensure_tokenizer_files` *Ensure Tokenizer Files are Downloaded*

Description

Ensure Tokenizer Files are Downloaded

Usage

```
ensure_tokenizer_files(model)
```

Arguments

`model`	Model name

Value

Path to vocab directory (directory containing vocab.json)

`extract_segments` *Extract Segments with Timestamps*

Description

Extract Segments with Timestamps

Usage

```
extract_segments(tokens, tokenizer, time_offset = 0)
```

Arguments

`tokens`	Token IDs
`tokenizer`	Tokenizer
`time_offset`	Offset in seconds for chunk processing

Value

Data frame with start, end, text

get_initial_tokens *Get Initial Decoder Tokens*

Description

Build the initial token sequence for decoder input.

Usage

```
get_initial_tokens(
  language = "en",
  task = "transcribe",
  model = "tiny",
  timestamps = FALSE
)
```

Arguments

language	Two-letter language code or NULL for auto
task	"transcribe" or "translate"
model	Model name for correct special token IDs
timestamps	Whether to include timestamps (internal use)

Value

Integer vector of initial token IDs

get_model_path *Get Model Cache Path*

Description

Get Model Cache Path

Usage

```
get_model_path(model)
```

Arguments

model	Model name

Value

Path to model directory in hfhub cache

get_weights_path — *Get Path to Model Weights*

Description

Get Path to Model Weights

Usage

```
get_weights_path(model)
```

Arguments

`model`	Model name

Value

Path to safetensors file

greedy_decode — *Greedy Decoding*

Description

Greedy Decoding

Usage

```
greedy_decode(
  model,
  encoder_output,
  initial_tokens,
  tokenizer,
  max_length = 448L,
  device
)
```

Arguments

`model`	WhisperModel
`encoder_output`	Encoder hidden states
`initial_tokens`	Initial token tensor
`tokenizer`	Tokenizer
`max_length`	Maximum output length
`device`	Device

Value

Integer vector of generated tokens

`hz_to_mel` *Convert Hz to Mel Scale*

Description

Convert Hz to Mel Scale

Usage

```
hz_to_mel(hz)
```

Arguments

`hz`	Frequency in Hz

Value

Frequency in mel scale

`is_timestamp_token` *Check if Token is Timestamp*

Description

Check if Token is Timestamp

Usage

```
is_timestamp_token(token_id, model = "tiny")
```

Arguments

`token_id`	Token ID
`model`	Model name for correct token IDs

Value

TRUE if timestamp token

list_downloaded_models *List Downloaded Models*

Description

List Downloaded Models

Usage

```
list_downloaded_models()
```

Value

Character vector of downloaded model names

Examples

```
list_downloaded_models()
```

list_whisper_models *List Available Models*

Description

List Available Models

Usage

```
list_whisper_models()
```

Value

Character vector of model names

Examples

```
list_whisper_models()
```

`load_added_tokens` *Load Added Tokens from HuggingFace*

Description

Load Added Tokens from HuggingFace

Usage

```
load_added_tokens(repo)
```

Arguments

`repo`	HuggingFace repo ID

Value

Named list of token -> ID mappings, or NULL if not found

`load_audio` *Load and Preprocess Audio*

Description

Load audio from file, convert to mono, resample to 16kHz.

Usage

```
load_audio(file)
```

Arguments

`file`	Path to audio file (WAV, MP3, etc.)

Value

Numeric vector of audio samples normalized to -1 to 1 range

Examples

```
# Load included sample audio
audio_file <- system.file("audio", "jfk.mp3", package = "whisper")
samples <- load_audio(audio_file)
length(samples)
range(samples)
```

load_decoder_weights *Load Decoder Weights*

Description

Load Decoder Weights

Usage

```
load_decoder_weights(decoder, weights)
```

Arguments

decoder	WhisperDecoder module
weights	Named list of tensors

load_encoder_weights *Load Encoder Weights*

Description

Load Encoder Weights

Usage

```
load_encoder_weights(encoder, weights)
```

Arguments

encoder	WhisperEncoder module
weights	Named list of tensors

load_mel_filterbank — *Load Pre-computed Mel Filterbank*

Description

Load the official Whisper mel filterbank from bundled CSV file.

Usage

```
load_mel_filterbank(n_mels = 80L)
```

Arguments

`n_mels`	Number of mel bins (80 or 128)

Value

Mel filterbank matrix (n_mels x n_freqs)

load_whisper_model — *Load Whisper Model*

Description

Load a Whisper model with weights from HuggingFace.

Usage

```
load_whisper_model(
  model = "tiny",
  device = "auto",
  dtype = "auto",
  download = FALSE,
  verbose = TRUE
)
```

Arguments

`model`	Model name: "tiny", "base", "small", "medium", "large-v3"
`device`	Device to load model on ("auto", "cpu", "cuda")
`dtype`	Data type ("auto", "float16", "float32")
`download`	If TRUE and model not present, prompt to download
`verbose`	Print loading messages

Value

WhisperModel module

Examples

```
# Load tiny model (requires prior download)
if (model_exists("tiny")) {
  model <- load_whisper_model("tiny")
}
```

---

load_whisper_weights *Load Weights from Safetensors*

Description

Load Weights from Safetensors

Usage

```
load_whisper_weights(model, weights_path, verbose = TRUE)
```

Arguments

model	WhisperModel module
weights_path	Path to safetensors file
verbose	Print loading messages

---

mel_to_hz *Convert Mel Scale to Hz*

Description

Convert Mel Scale to Hz

Usage

```
mel_to_hz(mel)
```

Arguments

mel	Frequency in mel scale

Value

Frequency in Hz

`model_exists` *Check if Model is Downloaded*

Description

Check if Model is Downloaded

Usage

```
model_exists(model)
```

Arguments

`model`	Model name

Value

TRUE if model weights exist locally

Examples

```
model_exists("tiny")
model_exists("large-v3")
```

`pad_or_trim` *Pad or Trim Audio to Fixed Length*

Description

Pad or Trim Audio to Fixed Length

Usage

```
pad_or_trim(audio, length = WHISPER_N_SAMPLES)
```

Arguments

`audio`	Numeric vector of audio samples
`length`	Target length in samples (default: 30s at 16kHz)

Value

Numeric vector of specified length

parse_device *Parse Device Argument*

Description

Parse Device Argument

Usage

```
parse_device(device = "auto")
```

Arguments

device	Character or torch device. "auto" uses GPU if available.

Value

torch device object

parse_dtype *Parse Dtype Argument*

Description

Parse Dtype Argument

Usage

```
parse_dtype(dtype = "auto", device = whisper_device())
```

Arguments

dtype	Character or torch dtype. "auto" uses float16 on GPU, float32 on CPU.
device	torch device (used for auto selection)

Value

torch dtype

split_audio — *Split Long Audio into Chunks*

Description

Split audio longer than 30 seconds into overlapping chunks.

Usage

```
split_audio(file, chunk_length = 30, overlap = 1)
```

Arguments

`file`	Path to audio file
`chunk_length`	Chunk length in seconds
`overlap`	Overlap between chunks in seconds

Value

List of audio chunks (numeric vectors)

tokenizer_decode — *Decode Token IDs to Text*

Description

Decode Token IDs to Text

Usage

```
tokenizer_decode(ids, id_to_token, special_tokens)
```

Arguments

`ids`	Integer vector of token IDs
`id_to_token`	Mapping from ID to token
`special_tokens`	Special token info

Value

Character string

tokenizer_encode — *Encode Text to Token IDs*

Description

Encode Text to Token IDs

Usage

```
tokenizer_encode(text, vocab, merge_ranks)
```

Arguments

`text`	Character string to encode
`vocab`	Vocabulary mapping (token -> id)
`merge_ranks`	Merge ranking for BPE

Value

Integer vector of token IDs

transcribe — *Whisper Transcription*

Description

Main transcription API for Whisper. Transcribe Audio Transcribe speech from an audio file using Whisper.

Usage

```
transcribe(
  file,
  model = "tiny",
  language = "en",
  task = "transcribe",
  device = "auto",
  dtype = "auto",
  verbose = TRUE
)
```

Arguments

`file`	Path to audio file (WAV, MP3, etc.)
`model`	Model name: "tiny", "base", "small", "medium", "large-v3"
`language`	Language code (e.g., "en", "es"). NULL for auto-detection.
`task`	"transcribe" or "translate" (translate to English)
`device`	Device: "auto", "cpu", "cuda"
`dtype`	Data type: "auto", "float16", "float32"
`verbose`	Print progress messages

Value

List with text, language, and metadata

Examples

```
# Transcribe included sample (JFK "ask not" speech)
if (model_exists("tiny")) {
  audio_file <- system.file("audio", "jfk.mp3", package = "whisper")
  result <- transcribe(audio_file, model = "tiny")
  result$text

  # Translate Spanish audio to English
  spanish_file <- system.file("audio", "allende.mp3", package = "whisper")
  result <- transcribe(spanish_file, model = "tiny",
                       language = "es", task = "translate")
  result$text
}
```

`transcribe_chunk` *Transcribe Single Chunk*

Description

Transcribe Single Chunk

Usage

```
transcribe_chunk(
  file,
  model,
  tokenizer,
  config,
  language = "en",
  task = "transcribe",
  device,
```

```
  dtype,
  verbose = TRUE
)
```

Arguments

file	Audio file or mel spectrogram
model	WhisperModel
tokenizer	Tokenizer
config	Model config
language	Language code
task	Task type
device	Device
dtype	Dtype
verbose	Verbose output

Value

Transcription result

---

transcribe_long   *Transcribe Long Audio*

Description

Process audio longer than 30 seconds in chunks.

Usage

```
transcribe_long(
  file,
  model,
  tokenizer,
  config,
  language,
  task,
  device,
  dtype,
  verbose
)
```

Arguments

`file`	Audio file
`model`	WhisperModel
`tokenizer`	Tokenizer
`config`	Model config
`language`	Language
`task`	Task
`device`	Device
`dtype`	Dtype
`verbose`	Verbose

Value

Combined transcription result

`whisper_attention` *Whisper Encoder*

Description

Transformer encoder for processing mel spectrograms. Multi-Head Self-Attention

Usage

```
whisper_attention(n_state, n_head)
```

Arguments

`n_state`	Hidden dimension
`n_head`	Number of attention heads

whisper_config *Whisper Model Configurations*

Description

Get configuration for a Whisper model variant.

Usage

```
whisper_config(model = "tiny")
```

Arguments

model	Character. Model name: "tiny", "base", "small", "medium", "large-v3"

Value

List with model configuration parameters

Examples

```
# Get tiny model configuration
cfg <- whisper_config("tiny")
cfg$n_mels
cfg$n_audio_layer

# Compare model sizes
whisper_config("tiny")$n_text_layer
whisper_config("large-v3")$n_text_layer
```

whisper_decoder *Text Decoder*

Description

Full Whisper decoder: token embedding + positional embedding + transformer layers.

Usage

```
whisper_decoder(n_vocab, n_ctx, n_state, n_head, n_layer)
```

Arguments

n_vocab	Vocabulary size
n_ctx	Maximum context length
n_state	Hidden dimension
n_head	Number of attention heads
n_layer	Number of transformer layers

whisper_decoder_layer *Whisper Decoder*

Description

Transformer decoder with cross-attention to encoder outputs. Decoder Layer Pre-norm transformer decoder layer with self-attention and cross-attention.

Usage

```
whisper_decoder_layer(n_state, n_head)
```

Arguments

n_state	Hidden dimension
n_head	Number of attention heads

whisper_device *Device and Dtype Management*

Description

Utilities for managing torch devices and data types. Get Default Device Returns CUDA device if available, otherwise CPU.

Usage

```
whisper_device()
```

Value

torch device object

Examples

```
if (torch::torch_is_installed()) {
  device <- whisper_device()
  device$type
}
```

whisper_dtype *Get Default Dtype*

Description

Returns float16 on CUDA, float32 on CPU.

Usage

```
whisper_dtype(device = whisper_device())
```

Arguments

device	torch device

Value

torch dtype

Examples

```
if (torch::torch_is_installed()) {
  dtype <- whisper_dtype()
  dtype
}
```

whisper_encoder *Audio Encoder*

Description

Full Whisper encoder: Conv stem + positional encoding + transformer layers.

Usage

```
whisper_encoder(n_mels, n_ctx, n_state, n_head, n_layer)
```

Arguments

n_mels	Number of mel spectrogram bins
n_ctx	Maximum context length (1500 for 30s audio)
n_state	Hidden dimension
n_head	Number of attention heads
n_layer	Number of transformer layers

whisper_encoder_layer *Encoder Layer*

Description

Pre-norm transformer encoder layer.

Usage

```
whisper_encoder_layer(n_state, n_head)
```

Arguments

n_state	Hidden dimension
n_head	Number of attention heads

whisper_lang_token *Get Language Token ID*

Description

Get Language Token ID

Usage

```
whisper_lang_token(lang = "en", model = "tiny")
```

Arguments

lang	Two-letter language code (e.g., "en", "es", "fr")
model	Model name for correct token IDs

Value

Token ID for the language

whisper_model *Whisper Model*

Description

Full Whisper model combining encoder and decoder. Whisper Model Module

Usage

```
whisper_model(config)
```

Arguments

config	Model configuration

WHISPER_SAMPLE_RATE *Audio Preprocessing for Whisper*

Description

Convert audio files to mel spectrograms for Whisper input. Whisper Audio Constants

Usage

```
WHISPER_SAMPLE_RATE
```

Format

An object of class `integer` of length 1.

whisper_special_tokens *Special Token IDs*

Description

Get special token IDs for a Whisper model. Token IDs differ between model variants (e.g., large-v3 has extra language tokens).

Usage

```
whisper_special_tokens(model = "tiny")
```

Arguments

`model`	Model name (default: "tiny")

Value

Named list of special token IDs

`whisper_tokenizer` *Whisper BPE Tokenizer*

Description

Byte-pair encoding tokenizer for Whisper models. Create Whisper Tokenizer Load or create a Whisper tokenizer from HuggingFace vocab files.

Usage

```
whisper_tokenizer(model = "tiny")
```

Arguments

`model`	Model name for vocab lookup

Value

Tokenizer object (list with encode/decode functions)

Examples

```
# Load tokenizer (requires prior model download)
if (model_exists("tiny")) {
  tok <- whisper_tokenizer("tiny")
  tok$encode("Hello world")
  tok$decode(c(50258, 50259, 50359, 50363))
}
```

Index