# Package: huggingfaceR (via r-universe)

June 30, 2026

**Type** Package

**Title** Access 'Hugging Face' Models and Datasets

**Version** 2.1.0

**Author** Alex Farach [aut, cre, cph], Sam Terfa [aut, cph], Jack Penzer [aut, cph]

**Maintainer** Alex Farach <alexfarach@gmail.com>

**Description** Access models and datasets hosted on the 'Hugging Face' Hub through its Inference Application Programming Interface (API). Run text classification, embeddings, chat, translation, image, audio, and other tasks from tidy 'R' workflows without installing 'Python' by default. Results are returned as data frames or simple 'R' objects so they can be composed with 'dplyr', 'tidyr', and related tooling. Helpers also support Hub search, file download, provider discovery, and guarded uploads for authenticated workflows.

**License** MIT + file LICENSE

**URL** https://farach.github.io/huggingfaceR/, https://github.com/farach/huggingfaceR

**BugReports** https://github.com/farach/huggingfaceR/issues

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.3

**Imports** cli, dplyr, glue, httr2 (>= 1.0.0), jsonlite, magrittr, purrr, rlang, stringr, tibble, tidyr

**Suggests** arrow, generics, knitr, lsa, recipes, reticulate, rmarkdown, testthat (>= 3.0.0), tidymodels, tidytext, uwot

**Config/testthat/edition** 3

**Depends** R (>= 4.1.0)

**NeedsCompilation** no

**Repository** https://cran.r-universe.dev

**Date/Publication** 2026-06-30 21:10:49 UTC

**RemoteUrl** https://github.com/cran/huggingfaceR

**RemoteRef** HEAD

**RemoteSha** a8547dfa72fc05a3f58ff8a89acb3420242a9d0c

# Contents

---

`chat` *Continue a Conversation*

---

## Description

Add a message to an existing conversation and get a response.

## Usage

```
chat(conversation, message, ...)
```

## Arguments

| | |
|---|---|
| `conversation` | An hf_conversation object from hf_conversation(). |
| `message` | Character string. The user message. |
| `...` | Additional parameters passed to the model. |

## Value

Updated conversation object with new messages in history.

## Examples

```
## Not run:
convo <- hf_conversation()
convo <- chat(convo, "Tell me a joke")

## End(Not run)
```

---

`hf_caption_image` *Caption Images*

---

## Description

Generate short image captions. This uses a vision-capable chat model by default because the public 'hf-inference' provider did not expose a broadly available image-to-text captioning model during verification.

## Usage

```
hf_caption_image(
  image,
  prompt = "Write a short, factual caption for this image.",
  model = hf_default_model("caption_image"),
  max_tokens = 80,
  token = NULL,
  endpoint_url = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| `image` | Image input: a local file path, URL, raw vector, or vector/list of paths/URLs. |
| `prompt` | Prompt used to request the caption. |
| `model` | Character string. Vision-capable chat model ID. Default: "google/gemma-3-4b-it". |
| `max_tokens` | Integer. Maximum tokens to generate. |
| `token` | Character string or NULL. API token for authentication. |
| `endpoint_url` | Character string or NULL. A custom Inference Endpoint URL. |
| `...` | Additional arguments passed to `hf_describe_image()`. |

## Value

A tibble with columns: image, caption.

## See Also

hf_describe_image

## Examples

```
## Not run:
hf_caption_image("cat.png")

## End(Not run)
```

---

hf_chat *LLM Chat Interface*

---

## Description

Have a conversation with an open-source language model via the Inference Providers API. Tool calling support depends on the model/provider. The default chat model is optimized for a low-friction first call; for tool-calling examples, use a tool-capable model such as `"Qwen/Qwen2.5-72B-Instruct"`.

## Usage

```
hf_chat(
  message,
  system = NULL,
  model = hf_default_model("chat"),
  max_tokens = 500,
  temperature = 0.7,
  token = NULL,
  tools = NULL,
  tool_choice = NULL,
  stream = FALSE,
  callback = NULL,
  image = NULL,
  endpoint_url = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| message | Character string. The user message to send to the model. |
| system | Character string or NULL. Optional system prompt to set behavior. |
| model | Character string. Model ID from Hugging Face Hub. Default: "meta-llama/Llama-3.1-8B-Instruct". Use ':provider' suffix to select a specific provider (e.g., "meta-llama/Llama-3.1-8B-Instruct:together"). |
| max_tokens | Integer. Maximum tokens to generate. Default: 500. |
| temperature | Numeric. Sampling temperature (0-2). Default: 0.7. |

| | |
|---|---|
| token | Character string or NULL. API token for authentication. |
| tools | A list of tool definitions created by hf_tool(), or NULL. |
| tool_choice | Character string or list controlling tool use. The public Hugging Face router currently supports "auto" and "none"; a tool name, "required", or full list can be used with compatible custom endpoints. Default: NULL. |
| stream | Logical. If TRUE, stream response deltas and return the reassembled final response. Default: FALSE. |
| callback | Function called with each streamed text delta. When NULL and stream = TRUE, deltas are printed to the console. |
| image | Optional image input for vision-capable chat models. Can be a URL, local file path, raw vector, or list/vector of those. |
| endpoint_url | Character string or NULL. A custom Inference Endpoint URL. When provided, requests are sent to this URL instead of the public Inference API. The endpoint must support the chat completions format. |
| ... | Additional parameters passed to the model. |

## Value

A tibble with columns: role, content, model, tokens_used, and tool_calls.

## Examples

```
## Not run:
# Simple question
hf_chat("What is the capital of France?")

# With system prompt
hf_chat(
  "Explain gradient descent",
  system = "You are a statistics professor. Use simple analogies."
)

# Use a specific provider
hf_chat("Hello!", model = "meta-llama/Llama-3-8B-Instruct:together")

# Stream response deltas
hf_chat("Reply with exactly: OK", stream = TRUE)

## End(Not run)
```

---

# hf_check_inference *Check Inference API Availability*

---

## Description

Check whether a model supports the Hugging Face Serverless Inference API. Not all models on the Hub are served by the Inference API. This function queries model metadata to determine availability before you make inference calls.

### Usage

```
hf_check_inference(model_id, token = NULL, quiet = FALSE)
```

### Arguments

| | |
|---|---|
| model_id | Character string. The model ID (e.g., "BAAI/bge-small-en-v1.5"). |
| token | Character string or NULL. API token for authentication. |
| quiet | Logical. If TRUE, suppress console output and return result invisibly. Default: FALSE. |

### Value

A list (invisibly if quiet = TRUE) with components:

| | |
|---|---|
| model_id | The model ID queried. |
| available | Logical. TRUE if the model is available on the Inference API. |
| pipeline_tag | The model's task type (e.g., "feature-extraction"). |
| inference_provider | The inference provider, if available. |

### Examples

```
## Not run:
# Check if a model supports serverless inference
hf_check_inference("BAAI/bge-small-en-v1.5")

# Use programmatically
result <- hf_check_inference("some-org/some-model", quiet = TRUE)
if (result$available) {
  embeddings <- hf_embed("hello", model = "some-org/some-model")
}

## End(Not run)
```

---

## hf_classify *Text Classification*

---

### Description

Classify text using a Hugging Face model. Commonly used for sentiment analysis, topic classification, etc. Vector inputs are sent in a single batched Inference API request when possible, which is substantially faster than one API request per text.

### Usage

```
hf_classify(
  text,
  model = hf_default_model("classify"),
  token = NULL,
  endpoint_url = NULL,
  ...
)
```

### Arguments

| | |
|---|---|
| text | Character vector of text(s) to classify. |
| model | Character string. Model ID from Hugging Face Hub. Default: "distilbert/distilbert-base-uncased-finetuned-sst-2-english" (sentiment analysis). |
| token | Character string or NULL. API token for authentication. |
| endpoint_url | Character string or NULL. A custom Inference Endpoint URL. When provided, requests are sent to this URL instead of the public Inference API. Use for models deployed on dedicated Inference Endpoints. |
| ... | Additional arguments (currently unused). |

### Value

A tibble with columns: text, label, score

### Examples

```
## Not run:
# Sentiment analysis
hf_classify("I love R programming!")

# Multiple texts
hf_classify(c("This is great!", "This is terrible."))

# Use in a pipeline
library(dplyr)
reviews |>
  mutate(sentiment = hf_classify(review_text)) |>
  unnest(sentiment)

## End(Not run)
```

---

## hf_classify_batch *Batch Text Classification (In-Memory)*

---

### Description

Classify multiple texts in parallel. This function processes all inputs in memory and returns results in a single tibble.

## Usage

```
hf_classify_batch(
  text,
  model = hf_default_model("classify"),
  token = NULL,
  batch_size = 100L,
  max_active = 10L,
  progress = TRUE,
  endpoint_url = NULL
)
```

## Arguments

| | |
|---|---|
| text | Character vector of text(s) to classify. |
| model | Character string. Model ID from Hugging Face Hub. Default: "distilbert/distilbert-base-uncased-finetuned-sst-2-english". |
| token | Character string or NULL. API token for authentication. |
| batch_size | Integer. Number of texts per API request. Default: 100. |
| max_active | Integer. Maximum concurrent requests. Default: 10. |
| progress | Logical. Show progress bar. Default: TRUE. |
| endpoint_url | Character string or NULL. A custom Inference Endpoint URL. |

## Value

A tibble with columns: - ‘text‘: Original input text - ‘label‘: Predicted label - ‘score‘: Confidence score - ‘.input_idx‘: Original position in input vector - ‘.error‘: TRUE if request failed - ‘.error_msg‘: Error message or NA

## Examples

```
## Not run:
# Classify many texts in parallel
texts <- c("I love this!", "This is terrible.", "Meh, it's okay.")
result <- hf_classify_batch(texts, max_active = 5)

# Check for errors
errors <- result[result$.error, ]

## End(Not run)
```

`hf_classify_chunks` *Chunked Text Classification (Disk Checkpoints)*

## Description

Classify large datasets with automatic checkpointing to disk. Supports resuming interrupted processing.

## Usage

```
hf_classify_chunks(
  text,
  output_dir,
  model = hf_default_model("classify"),
  token = NULL,
  chunk_size = 1000L,
  batch_size = 100L,
  max_active = 10L,
  resume = TRUE,
  progress = TRUE,
  endpoint_url = NULL
)
```

## Arguments

| | |
|---|---|
| `text` | Character vector of text(s) to classify. |
| `output_dir` | Character string. Directory to write chunk files. |
| `model` | Character string. Model ID from Hugging Face Hub. Default: "distilbert/distilbert-base-uncased-finetuned-sst-2-english". |
| `token` | Character string or NULL. API token for authentication. |
| `chunk_size` | Integer. Number of texts per disk chunk. Default: 1000. |
| `batch_size` | Integer. Number of texts per API request. Default: 100. |
| `max_active` | Integer. Maximum concurrent requests. Default: 10. |
| `resume` | Logical. Skip already-completed chunks. Default: TRUE. |
| `progress` | Logical. Show progress bar. Default: TRUE. |
| `endpoint_url` | Character string or NULL. A custom Inference Endpoint URL. |

## Value

Invisibly returns the output directory path. Use ‘hf_read_chunks()‘ to read results.

## Examples

```
## Not run:
# Process large dataset with checkpoints
texts <- rep("sample text", 5000)
hf_classify_chunks(texts, output_dir = "classify_output", chunk_size = 1000)

# Read results
results <- hf_read_chunks("classify_output")

## End(Not run)
```

---

`hf_classify_image` *Classify Images*

---

## Description

Classify images using an image-classification model via the Hugging Face Inference Providers API.

## Usage

```
hf_classify_image(
  image,
  top_k = 5L,
  model = hf_default_model("classify_image"),
  token = NULL,
  endpoint_url = NULL,
  content_type = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| `image` | Image input: a local file path, URL, raw vector, or vector/list of paths/URLs. |
| `top_k` | Integer. Maximum labels to return per image. |
| `model` | Character string. Model ID from Hugging Face Hub. Default: "google/vit-base-patch16-224". |
| `token` | Character string or NULL. API token for authentication. |
| `endpoint_url` | Character string or NULL. A custom Inference Endpoint URL. |
| `content_type` | Character string or NULL. MIME type to use for raw image inputs. Paths and URLs are inferred when possible. |
| `...` | Additional arguments (currently unused). |

## Value

A tibble with columns: image, label, score.

## See Also

https://huggingface.co/docs/inference-providers/tasks/image-classification

## Examples

```
## Not run:
hf_classify_image("cat.png", top_k = 3)

## End(Not run)
```

---

`hf_classify_zero_shot` *Zero-Shot Classification*

---

## Description

Classify text into custom categories without training a model. The model determines which labels best describe the input text.

## Usage

```
hf_classify_zero_shot(
  text,
  labels,
  model = hf_default_model("zero_shot"),
  multi_label = FALSE,
  token = NULL,
  endpoint_url = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| `text` | Character vector of text(s) to classify. |
| `labels` | Character vector of candidate labels/categories. |
| `model` | Character string. Model ID from Hugging Face Hub. Default: "facebook/bart-large-mnli" |
| `multi_label` | Logical. If TRUE, allows multiple labels per text. Default: FALSE (single label per text). |
| `token` | Character string or NULL. API token for authentication. |
| `endpoint_url` | Character string or NULL. A custom Inference Endpoint URL. |
| `...` | Additional arguments (currently unused). |

## Value

A tibble with columns: text, label, score (sorted by score descending)

## Examples

```
## Not run:
# Classify into custom categories
hf_classify_zero_shot(
  "I just bought a new laptop",
  labels = c("technology", "sports", "politics", "food")
)

# Multi-label classification
hf_classify_zero_shot(
  "This laptop is great for gaming",
  labels = c("technology", "gaming", "entertainment"),
  multi_label = TRUE
)

## End(Not run)
```

---

`hf_classify_zero_shot_batch` *Batch Zero-Shot Classification (In-Memory)*

---

## Description

Classify multiple texts into custom categories in parallel without training.

## Usage

```
hf_classify_zero_shot_batch(
  text,
  labels,
  model = hf_default_model("zero_shot"),
  multi_label = FALSE,
  token = NULL,
  batch_size = 50L,
  max_active = 10L,
  progress = TRUE,
  endpoint_url = NULL
)
```

## Arguments

| | |
|---|---|
| `text` | Character vector of text(s) to classify. |
| `labels` | Character vector of candidate labels/categories. |
| `model` | Character string. Model ID from Hugging Face Hub. Default: "facebook/bart-large-mnli". |
| `multi_label` | Logical. If TRUE, allows multiple labels per text. Default: FALSE. |
| `token` | Character string or NULL. API token for authentication. |

| | |
|---|---|
| batch_size | Integer. Number of texts per API request. Default: 50. |
| max_active | Integer. Maximum concurrent requests. Default: 10. |
| progress | Logical. Show progress bar. Default: TRUE. |
| endpoint_url | Character string or NULL. A custom Inference Endpoint URL. |

## Value

A tibble with columns: - ‘text‘: Original input text - ‘label‘: Predicted label (or labels if multi_label) - ‘score‘: Confidence score(s) - ‘.input_idx‘: Original position in input vector - ‘.error‘: TRUE if request failed - ‘.error_msg‘: Error message or NA

## Examples

```
## Not run:
texts <- c("I love my new laptop", "The game was exciting", "This recipe is delicious")
labels <- c("technology", "sports", "food")
result <- hf_classify_zero_shot_batch(texts, labels, max_active = 5)

## End(Not run)
```

---

# hf_cluster_texts *Cluster Texts by Semantic Similarity*

---

## Description

Perform k-means clustering on text embeddings.

## Usage

```
hf_cluster_texts(data, k = 3, ...)
```

## Arguments

| | |
|---|---|
| data | A data frame with an 'embedding' column (from hf_embed_text). |
| k | Integer. Number of clusters. Default: 3. |
| ... | Additional arguments passed to stats::kmeans(). |

## Value

The input data frame with an added 'cluster' column.

## Examples

```
## Not run:
library(ggplot2)

# Cluster documents
docs_clustered <- docs_embedded |>
  hf_cluster_texts(k = 3)

# Reduce dimensions and visualize
library(uwot)
emb_matrix <- do.call(rbind, docs_clustered$embedding)
coords <- umap(emb_matrix)

docs_clustered |>
  mutate(umap_1 = coords[, 1], umap_2 = coords[, 2]) |>
  ggplot(aes(umap_1, umap_2, color = factor(cluster))) +
  geom_point(size = 3)

## End(Not run)
```

---

`hf_conversation` *Multi-turn Conversation*

---

## Description

Create and manage a multi-turn conversation with an LLM.

## Usage

```
hf_conversation(
  system = NULL,
  model = hf_default_model("chat"),
  endpoint_url = NULL
)
```

## Arguments

| | |
|---|---|
| `system` | Character string or NULL. System prompt for the conversation. |
| `model` | Character string. Model ID from Hugging Face Hub. Default: "meta-llama/Llama-3.1-8B-Instruct". |
| `endpoint_url` | Character string or NULL. A custom Inference Endpoint URL. |

## Value

A conversation object (list) that can be extended with chat().

## Examples

```
## Not run:
# Create conversation
convo <- hf_conversation(system = "You are a helpful R tutor.")

# Add messages (see chat() method)
convo <- chat(convo, "How do I read a CSV?")
convo <- chat(convo, "What about Excel files?")

# View history
convo$history

## End(Not run)
```

---

`hf_create_repo` *Create a Hub Repository*

---

## Description

Create a model, dataset, or Space repository. This is a write operation and requires an API token with write scope plus ‘confirm = TRUE‘.

## Usage

```
hf_create_repo(
  repo_id,
  repo_type = "model",
  private = FALSE,
  exist_ok = FALSE,
  token = NULL,
  confirm = FALSE
)
```

## Arguments

| | |
|---|---|
| `repo_id` | Character string. Repository name or "namespace/name". |
| `repo_type` | Character string. One of "model", "dataset", or "space". |
| `private` | Logical. Whether to create a private repository. |
| `exist_ok` | Logical. If TRUE, do not fail when the repo already exists. |
| `token` | Character string or NULL. API token with write scope. |
| `confirm` | Logical. Must be TRUE to perform the write operation. |

## Value

The parsed Hub API response.

## Examples

```
## Not run:
hf_create_repo("my-dataset", repo_type = "dataset", private = TRUE, confirm = TRUE)

## End(Not run)
```

---

`hf_dataset_info` *Get Dataset Information*

---

## Description

Retrieve metadata about a dataset from Hugging Face Hub.

## Usage

```
hf_dataset_info(dataset, token = NULL)
```

## Arguments

| | |
|---|---|
| `dataset` | Character string. Dataset name. |
| `token` | Character string or NULL. API token for private datasets. |

## Value

A list with dataset information.

## Examples

```
## Not run:
hf_dataset_info("imdb")

## End(Not run)
```

---

`hf_default_model` *Default Models for Hugging Face Tasks*

---

## Description

Central registry of the default model used by each ‘huggingfaceR‘ inference function. Every ‘hf_*‘ function that takes a ‘model‘ argument resolves its default through this single function, so default models live in exactly one place and can be audited or updated without hunting through the codebase.

## Usage

```
hf_default_model(task = NULL)
```

### Arguments

| | |
|---|---|
| task | Character string naming the task, or ‘NULL‘. One of: "chat", "generate", "fill_mask", "classify", "zero_shot", "embed", "summarize", "translate", "ner", "question_answer", "table_question_answer", "vision_chat", "transcribe", "text_to_speech", "text_to_image", "classify_image", "caption_image", "detect_objects". When ‘NULL‘ (the default), the full registry is returned as a tibble. |

### Details

Defaults are chosen to be **beginner-friendly**: broadly known, small and fast, low cost, and usable with no extra arguments — the goal is the quickest path to a working first call (think of ‘mtcars‘ in base R). Power users can always override any default by passing their own ‘model‘.

### Value

When ‘task‘ is supplied, a single model-ID character string. When ‘task‘ is ‘NULL‘, a tibble with columns ‘task‘ and ‘model‘ listing every default.

### Examples

```
# The default model for a given task
hf_default_model("translate")

# The whole registry at a glance
hf_default_model()
```

---

## hf_delete_repo *Delete a Hub Repository*

---

### Description

Delete a model, dataset, or Space repository. This destructive operation is guarded: it requires ‘confirm = TRUE‘ and is refused when ‘CI=true‘.

### Usage

```
hf_delete_repo(repo_id, repo_type = "model", token = NULL, confirm = FALSE)
```

### Arguments

| | |
|---|---|
| repo_id | Character string. Repository ID. |
| repo_type | Character string. One of "model", "dataset", or "space". |
| token | Character string or NULL. API token with write scope. |
| confirm | Logical. Must be TRUE to delete the repository. |

### Value

The parsed Hub API response.

## Examples

```
## Not run:
hf_delete_repo("me/old-test-dataset", repo_type = "dataset", confirm = TRUE)

## End(Not run)
```

---

`hf_describe_image` *Describe an Image*

---

## Description

Ask a vision-capable chat model to describe an image.

## Usage

```
hf_describe_image(
  image,
  prompt = "Describe this image.",
  model = hf_default_model("vision_chat"),
  max_tokens = 200,
  token = NULL,
  endpoint_url = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| `image` | Image URL, local file path, raw vector, or a character vector/list of image URLs or paths. |
| `prompt` | Prompt to send with each image. Default: "Describe this image." |
| `model` | Character string. Vision-capable chat model ID. |
| `max_tokens` | Integer. Maximum tokens to generate. Default: 200. |
| `token` | Character string or NULL. API token for authentication. |
| `endpoint_url` | Character string or NULL. A custom Inference Endpoint URL. |
| `...` | Additional parameters passed to `hf_chat()`. |

## Value

A tibble with columns: image, description.

## Examples

```
## Not run:
image <- paste0(
  "https://huggingface.co/datasets/huggingface/",
  "documentation-images/resolve/main/cat.png"
)
hf_describe_image(image)

## End(Not run)
```

---

`hf_detect_objects` *Detect Objects in Images*

---

## Description

Detect objects and bounding boxes in images using an object-detection model via the Hugging Face Inference Providers API.

## Usage

```
hf_detect_objects(
  image,
  threshold = NULL,
  model = hf_default_model("detect_objects"),
  token = NULL,
  endpoint_url = NULL,
  content_type = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| `image` | Image input: a local file path, URL, raw vector, or vector/list of paths/URLs. |
| `threshold` | Numeric or NULL. Optional minimum confidence score to keep. |
| `model` | Character string. Model ID from Hugging Face Hub. Default: "facebook/detr-resnet-50". |
| `token` | Character string or NULL. API token for authentication. |
| `endpoint_url` | Character string or NULL. A custom Inference Endpoint URL. |
| `content_type` | Character string or NULL. MIME type to use for raw image inputs. Paths and URLs are inferred when possible. |
| `...` | Additional arguments (currently unused). |

## Value

A tibble with columns: image, label, score, xmin, ymin, xmax, ymax.

## See Also

https://huggingface.co/docs/inference-providers/tasks/object-detection

## Examples

```
## Not run:
boxes <- hf_detect_objects("cat.png", threshold = 0.5)
boxes

## End(Not run)
```

---

`hf_embed` *Generate Text Embeddings*

---

## Description

Generate dense vector representations (embeddings) for text using transformer models. Useful for semantic similarity, clustering, and as features for ML models. Vector inputs are sent in a single batched Inference API request when possible, which is substantially faster than one API request per text.

## Usage

```
hf_embed(
  text,
  model = hf_default_model("embed"),
  token = NULL,
  endpoint_url = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| `text` | Character vector of text(s) to embed. |
| `model` | Character string. Model ID from Hugging Face Hub. Default: "BAAI/bge-small-en-v1.5" (384-dim embeddings). |
| `token` | Character string or NULL. API token for authentication. |
| `endpoint_url` | Character string or NULL. A custom Inference Endpoint URL. When provided, requests are sent to this URL instead of the public Inference API. Use for models deployed on dedicated Inference Endpoints. |
| `...` | Additional arguments (currently unused). |

## Value

A tibble with columns: text, embedding (list-column of numeric vectors), n_dims

## Examples

```
## Not run:
# Generate embeddings
embeddings <- hf_embed(c("Hello world", "Goodbye world"))

# Access embedding vectors
embeddings$embedding[[1]]  # First embedding vector

## End(Not run)
```

---

`hf_embed_batch` *Batch Embedding Generation (In-Memory)*

---

## Description

Generate embeddings for multiple texts in parallel. This function processes all inputs in memory and returns results in a single tibble.

## Usage

```
hf_embed_batch(
  text,
  model = hf_default_model("embed"),
  token = NULL,
  batch_size = 100L,
  max_active = 10L,
  progress = TRUE,
  endpoint_url = NULL
)
```

## Arguments

| | |
|---|---|
| `text` | Character vector of text(s) to embed. |
| `model` | Character string. Model ID from Hugging Face Hub. Default: "BAAI/bge-small-en-v1.5" (384-dim embeddings). |
| `token` | Character string or NULL. API token for authentication. |
| `batch_size` | Integer. Number of texts per API request. Default: 100. |
| `max_active` | Integer. Maximum concurrent requests. Default: 10. |
| `progress` | Logical. Show progress bar. Default: TRUE. |
| `endpoint_url` | Character string or NULL. A custom Inference Endpoint URL. |

## Value

A tibble with columns: - ‘text‘: Original input text - ‘embedding‘: List-column of numeric vectors - ‘n_dims‘: Dimension of embedding (or NA on error) - ‘.input_idx‘: Original position in input vector - ‘.error‘: TRUE if request failed - ‘.error_msg‘: Error message or NA

## Examples

```
## Not run:
# Embed many texts in parallel
texts <- c("Hello world", "Goodbye world", "R is great")
result <- hf_embed_batch(texts, max_active = 5)

# Check for errors
errors <- result[result$.error, ]

## End(Not run)
```

---

`hf_embed_chunks` *Chunked Embedding Generation (Disk Checkpoints)*

---

## Description

Generate embeddings for large datasets with automatic checkpointing to disk. Supports resuming interrupted processing.

## Usage

```
hf_embed_chunks(
  text,
  output_dir,
  model = hf_default_model("embed"),
  token = NULL,
  chunk_size = 1000L,
  batch_size = 100L,
  max_active = 10L,
  resume = TRUE,
  progress = TRUE,
  endpoint_url = NULL
)
```

## Arguments

| | |
|---|---|
| `text` | Character vector of text(s) to embed. |
| `output_dir` | Character string. Directory to write chunk files. |
| `model` | Character string. Model ID from Hugging Face Hub. Default: "BAAI/bge-small-en-v1.5". |
| `token` | Character string or NULL. API token for authentication. |
| `chunk_size` | Integer. Number of texts per disk chunk. Default: 1000. |
| `batch_size` | Integer. Number of texts per API request. Default: 100. |
| `max_active` | Integer. Maximum concurrent requests. Default: 10. |
| `resume` | Logical. Skip already-completed chunks. Default: TRUE. |
| `progress` | Logical. Show progress bar. Default: TRUE. |
| `endpoint_url` | Character string or NULL. A custom Inference Endpoint URL. |

## Value

Invisibly returns the output directory path. Use ‘hf_read_chunks()‘ to read results.

## Examples

```
## Not run:
# Process large dataset with checkpoints
texts <- rep("sample text", 5000)
hf_embed_chunks(texts, output_dir = "embeddings_output", chunk_size = 1000)

# Read results
results <- hf_read_chunks("embeddings_output")

# Resume interrupted processing
hf_embed_chunks(more_texts, output_dir = "embeddings_output", resume = TRUE)

## End(Not run)
```

---

# hf_embed_text *Embed Text with Tidytext Compatibility*

---

## Description

Generate embeddings for text in a tidy data frame. Designed to work seamlessly with tidytext workflows.

## Usage

```
hf_embed_text(
  data,
  text_col,
  model = hf_default_model("embed"),
  token = NULL,
  keep_text = TRUE
)
```

## Arguments

| | |
|---|---|
| data | A data frame or tibble. |
| text_col | Unquoted column name containing text to embed. |
| model | Character string. Hugging Face model ID for embeddings. Default: "BAAI/bge-small-en-v1.5". |
| token | Character string or NULL. API token for authentication. |
| keep_text | Logical. Keep original text column? Default: TRUE. |

## Value

The input data frame with added embedding and n_dims columns.

## Examples

```
## Not run:
library(dplyr)
library(tidytext)

# Embed documents
docs <- tibble(
  doc_id = 1:3,
  text = c("I love R", "Python is great", "Julia is fast")
)

docs_embedded <- docs |>
  hf_embed_text(text)

# Find similar documents
docs_embedded |>
  hf_nearest_neighbors("I love R", k = 2)

## End(Not run)
```

---

`hf_embed_umap` *Dimensionality Reduction with UMAP*

---

## Description

Reduce embedding dimensions to 2D using UMAP for visualization. Requires the 'uwot' package to be installed.

## Usage

```
hf_embed_umap(
  text,
  model = hf_default_model("embed"),
  token = NULL,
  endpoint_url = NULL,
  n_neighbors = 15,
  min_dist = 0.1,
  ...
)
```

## Arguments

| | |
|---|---|
| `text` | Character vector of text(s) to embed and reduce. |
| `model` | Character string. Model ID for generating embeddings. Default: "BAAI/bge-small-en-v1.5". |
| `token` | Character string or NULL. API token for authentication. |
| `endpoint_url` | Character string or NULL. A custom Inference Endpoint URL. |

| | |
|---|---|
| n_neighbors | Integer. UMAP n_neighbors parameter. Default: 15. |
| min_dist | Numeric. UMAP min_dist parameter. Default: 0.1. |
| ... | Additional arguments passed to uwot::umap(). |

## Value

A tibble with columns: text, umap_1, umap_2

## Examples

```
## Not run:
# Reduce and visualize
library(ggplot2)
texts <- c("cat", "dog", "kitten", "puppy", "car", "truck")
coords <- hf_embed_umap(texts)

ggplot(coords, aes(umap_1, umap_2, label = text)) +
  geom_text() +
  theme_minimal()

## End(Not run)
```

---

# hf_extract *Extract Structured Data from Text*

---

## Description

Convert unstructured text into tidy columns using a chat model with structured JSON output. The schema argument can be a lightweight named character vector such as c(name = "string", score = "number") or a full JSON Schema list. The function returns one row per input text and one column per schema field.

## Usage

```
hf_extract(
  text,
  schema,
  model = hf_default_model("chat"),
  strict = TRUE,
  system = paste("Extract the requested fields from the user's text.",
    "Return only JSON that matches the schema."),
  token = NULL,
  endpoint_url = NULL,
  ...
)
```

### Arguments

| | |
|---|---|
| text | Character vector of text(s) to extract from. |
| schema | A named character vector of field names and JSON types, or a JSON Schema list with object properties. |
| model | Character string. Model ID from Hugging Face Hub. Default: "meta-llama/Llama-3.1-8B-Instruct". |
| strict | Logical. Whether to request strict JSON Schema adherence. Default: TRUE. |
| system | Character string. System prompt sent with each extraction request. Default: a concise extraction instruction. |
| token | Character string or NULL. API token for authentication. |
| endpoint_url | Character string or NULL. A custom Inference Endpoint URL. The endpoint must support the chat completions format. |
| ... | Additional parameters passed to the chat-completions request. |

### Value

A tibble with one row per input and one column per schema field.

### Examples

```
## Not run:
hf_extract(
  "Amelie is a chef in Paris.",
  c(name = "string", occupation = "string", city = "string")
)

hf_extract(
  c("Great service.", "The delivery was late."),
  c(sentiment = "string", is_complaint = "boolean")
)

## End(Not run)
```

---

## hf_extract_topics *Extract Semantic Topics from Text*

---

### Description

Identify semantic topics by clustering embeddings and extracting representative keywords from each cluster.

### Usage

```
hf_extract_topics(data, text_col = "text", k = 5, top_n = 10)
```

### Arguments

| | |
|---|---|
| data | A data frame with text and embeddings. |
| text_col | Character string. Name of text column. |
| k | Integer. Number of topics/clusters. Default: 5. |
| top_n | Integer. Number of top words per topic. Default: 10. |

### Value

A tibble with topics and their top terms.

### Examples

```
## Not run:
library(tidytext)

# Extract topics
topics <- docs_embedded |>
  hf_extract_topics(text_col = "text", k = 3, top_n = 5)

## End(Not run)
```

---

## hf_ez_conversational *Create a Conversational Agent*

---

### Description

This task corresponds to any chatbot like structure. Models tend to have shorter max_length, so please check with caution when using a given model if you need long range dependency or not.

### Usage

```
hf_ez_conversational(model_id = "microsoft/DialoGPT-large", use_api = FALSE)
```

### Arguments

| | |
|---|---|
| model_id | A model_id. Run hf_search_models(...) for model_ids. Defaults to 'microsoft/DialoGPT-large'. |
| use_api | Whether to use the Inference API to run the model (TRUE) or download and run the model locally (FALSE). Defaults to FALSE |

### Value

A conversational object

### See Also

https://huggingface.co/docs/api-inference/detailed_parameters#zero-shot-classification-task

## Examples

```
## Not run:
# Load the default model
ez <- hf_ez_conversational()

# Continue the conversation
ez$infer(
  past_user_inputs = list("Which movie is the best?"),
  generated_responses = list("It's Die Hard for sure."),
  text = "Can you explain why?",
  min_length = 10,
  max_length = 50
)

## End(Not run)
```

---

# hf_ez_conversational_api_inference

*Conversational API Inference*

---

## Description

Conversational API Inference

## Usage

```
hf_ez_conversational_api_inference(
  text,
  generated_responses = NULL,
  past_user_inputs = NULL,
  min_length = NULL,
  max_length = NULL,
  top_k = NULL,
  top_p = NULL,
  temperature = 1,
  max_time = NULL,
  tidy = TRUE,
  use_gpu = FALSE,
  use_cache = FALSE,
  wait_for_model = FALSE,
  use_auth_token = NULL,
  stop_on_error = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| `text` | The last input from the user in the conversation. |
| `generated_responses` | A list of strings corresponding to the earlier replies from the model. |
| `past_user_inputs` | A list of strings corresponding to the earlier replies from the user. Should be of the same length of generated_responses. |
| `min_length` | (Default: None). Integer to define the minimum length in tokens of the output summary. |
| `max_length` | (Default: None). Integer to define the maximum length in tokens of the output summary. |
| `top_k` | (Default: None). Integer to define the top tokens considered within the sample operation to create new text. |
| `top_p` | (Default: None). Float to define the tokens that are within the sample operation of text generation. Add tokens in the sample for more probable to least probable until the sum of the probabilities is greater than top_p. |
| `temperature` | (Default: 1.0). Float (0.0-100.0). The temperature of the sampling operation. 1 means regular sampling, 0 means always take the highest score, 100.0 is getting closer to uniform probability. |
| `max_time` | (Default: None). Float (0-120.0). The amount of time in seconds that the query should take maximum. Network can cause some overhead so it will be a soft limit. |
| `tidy` | Whether to tidy the results into a tibble. Default: TRUE (tidy the results) |
| `use_gpu` | Whether to use GPU for inference. |
| `use_cache` | Whether to use cached inference results for previously seen inputs. |
| `wait_for_model` | Whether to wait for the model to be ready instead of receiving a 503 error after a certain amount of time. |
| `use_auth_token` | The token to use as HTTP bearer authorization for the Inference API. Defaults to HUGGING_FACE_HUB_TOKEN environment variable. |
| `stop_on_error` | Whether to throw an error if an API error is encountered. Defaults to FALSE (do not throw error). |
| `...` | Additional arguments passed internally, including the model object or model ID. |

## Value

The results of the inference

## See Also

https://huggingface.co/docs/api-inference/index

`hf_ez_conversational_local_inference` *Conversational Local Inference*

## Description

Conversational Local Inference

## Usage

```
hf_ez_conversational_local_inference(
  text,
  generated_responses = NULL,
  past_user_inputs = NULL,
  min_length = NULL,
  max_length = NULL,
  top_k = NULL,
  top_p = NULL,
  temperature = 1,
  max_time = NULL,
  tidy = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| `text` | The last input from the user in the conversation. |
| `generated_responses` | A list of strings corresponding to the earlier replies from the model. |
| `past_user_inputs` | A list of strings corresponding to the earlier replies from the user. Should be of the same length of generated_responses. |
| `min_length` | (Default: None). Integer to define the minimum length in tokens of the output summary. |
| `max_length` | (Default: None). Integer to define the maximum length in tokens of the output summary. |
| `top_k` | (Default: None). Integer to define the top tokens considered within the sample operation to create new text. |
| `top_p` | (Default: None). Float to define the tokens that are within the sample operation of text generation. Add tokens in the sample for more probable to least probable until the sum of the probabilities is greater than top_p. |
| `temperature` | (Default: 1.0). Float (0.0-100.0). The temperature of the sampling operation. 1 means regular sampling, 0 means always take the highest score, 100.0 is getting closer to uniform probability. |

| | |
|---|---|
| max_time | (Default: None). Float (0-120.0). The amount of time in seconds that the query should take maximum. Network can cause some overhead so it will be a soft limit. |
| tidy | Whether to tidy the results into a tibble. Default: TRUE (tidy the results) |
| ... | Additional arguments passed internally, including the model object or model ID. |

## Value

The results of the inference

## See Also

https://huggingface.co/docs/transformers/main/en/pipeline_tutorial

---

# hf_ez_fill_mask *Perform Fill-in-the-Blank Tasks*

---

## Description

Tries to fill in a hole with a missing word (token to be precise). That's the base task for BERT models.

## Usage

```
hf_ez_fill_mask(model_id = "google-bert/bert-base-uncased", use_api = FALSE)
```

## Arguments

| | |
|---|---|
| model_id | A model_id. Run hf_search_models(...) for model_ids. Defaults to 'bert-base-uncased'. |
| use_api | Whether to use the Inference API to run the model (TRUE) or download and run the model locally (FALSE). Defaults to FALSE |

## Value

A fill mask object

## See Also

https://huggingface.co/docs/api-inference/detailed_parameters#fill-mask-task

## Examples

```
## Not run:
# Load the default model and use local inference
ez <- hf_ez_fill_mask()
ez$infer(string = "The answer to the universe is [MASK].")

# Load a specific model and use the API for inference.
# Note the mask is different for different models.
ez <- hf_ez_fill_mask(model_id = 'xlm-roberta-base', use_api = TRUE)
ez$infer(string = "The answer to the universe is <MASK>.")

## End(Not run)
```

---

`hf_ez_fill_mask_api_inference` *Fill Mask API Inference*

---

## Description

Fill Mask API Inference

## Usage

```
hf_ez_fill_mask_api_inference(
  string,
  tidy = TRUE,
  use_gpu = FALSE,
  use_cache = FALSE,
  wait_for_model = FALSE,
  use_auth_token = NULL,
  stop_on_error = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| `string` | a string to be filled from, must contain the [MASK] token (check model card for exact name of the mask) |
| `tidy` | Whether to tidy the results into a tibble. Default: TRUE (tidy the results) |
| `use_gpu` | Whether to use GPU for inference. |
| `use_cache` | Whether to use cached inference results for previously seen inputs. |
| `wait_for_model` | Whether to wait for the model to be ready instead of receiving a 503 error after a certain amount of time. |
| `use_auth_token` | The token to use as HTTP bearer authorization for the Inference API. Defaults to HUGGING_FACE_HUB_TOKEN environment variable. |
| `stop_on_error` | Whether to throw an error if an API error is encountered. Defaults to FALSE (do not throw error). |
| `...` | Additional arguments passed internally, including the model object or model ID. |

## Value

The results of the inference

## See Also

https://huggingface.co/docs/api-inference/index

---

`hf_ez_fill_mask_local_inference`
*Fill Mask Local Inference*

---

## Description

Fill Mask Local Inference

## Usage

```
hf_ez_fill_mask_local_inference(string, tidy = TRUE, ...)
```

## Arguments

| | |
|---|---|
| `string` | a string to be filled from, must contain the [MASK] token (check model card for exact name of the mask) |
| `tidy` | Whether to tidy the results into a tibble. Default: TRUE (tidy the results) |
| `...` | Additional arguments passed internally, including the model object or model ID. |

## Value

The results of the inference

## See Also

https://huggingface.co/docs/transformers/main/en/pipeline_tutorial

`hf_ez_question_answering` *Answer Questions about a Text based on Context*

## Description

Want to have a nice know-it-all bot that can answer any question?

## Usage

```
hf_ez_question_answering(
  model_id = "deepset/roberta-base-squad2",
  use_api = FALSE
)
```

## Arguments

| | |
|---|---|
| `model_id` | A model_id. Run hf_search_models(...) for model_ids. Defaults to 'deepset/roberta-base-squad2'. |
| `use_api` | Whether to use the Inference API to run the model (TRUE) or download and run the model locally (FALSE). Defaults to FALSE |

## Value

A question answering object

## See Also

https://huggingface.co/docs/api-inference/detailed_parameters#question-answering-task

## Examples

```
## Not run:
# Load the default model and use local inference
ez <- hf_ez_question_answering()
ez$infer(question = "What's my name?", context = "My name is Clara and I live in Berkeley.")

# Use the api for inference.
ez <- hf_ez_fill_mask(use_api = TRUE)
ez$infer(question = "What's my name?", context = "My name is Clara and I live in Berkeley.")

## End(Not run)
```

`hf_ez_question_answering_api_inference` *Question Answering API Inference*

## Description

Question Answering API Inference

## Usage

```
hf_ez_question_answering_api_inference(
  question,
  context,
  tidy = TRUE,
  use_gpu = FALSE,
  use_cache = FALSE,
  wait_for_model = FALSE,
  use_auth_token = NULL,
  stop_on_error = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| `question` | a question to be answered based on the provided context |
| `context` | the context to consult for answering the question |
| `tidy` | Whether to tidy the results into a tibble. Default: TRUE (tidy the results) |
| `use_gpu` | Whether to use GPU for inference. |
| `use_cache` | Whether to use cached inference results for previously seen inputs. |
| `wait_for_model` | Whether to wait for the model to be ready instead of receiving a 503 error after a certain amount of time. |
| `use_auth_token` | The token to use as HTTP bearer authorization for the Inference API. Defaults to HUGGING_FACE_HUB_TOKEN environment variable. |
| `stop_on_error` | Whether to throw an error if an API error is encountered. Defaults to FALSE (do not throw error). |
| `...` | Additional arguments passed internally, including the model object or model ID. |

## Value

The results of the inference

## See Also

https://huggingface.co/docs/api-inference/index

---

`hf_ez_question_answering_local_inference`
*Question Answering Local Inference*

---

## Description

Question Answering Local Inference

## Usage

```
hf_ez_question_answering_local_inference(question, context, tidy = TRUE, ...)
```

## Arguments

| | |
|---|---|
| `question` | a question to be answered based on the provided context |
| `context` | the context to consult for answering the question |
| `tidy` | Whether to tidy the results into a tibble. Default: TRUE (tidy the results) |
| `...` | Additional arguments passed internally, including the model object or model ID. |

## Value

The results of the inference

## See Also

https://huggingface.co/docs/transformers/main/en/pipeline_tutorial

---

`hf_ez_sentence_similarity`
*Compare Sentence Similarity Semantically*

---

## Description

Calculate the semantic similarity between one text and a list of other sentences by comparing their embeddings.

## Usage

```
hf_ez_sentence_similarity(
  model_id = "sentence-transformers/all-MiniLM-L6-v2",
  use_api = FALSE
)
```

### Arguments

| | |
|---|---|
| model_id | A model_id. Run hf_search_models(...) for model_ids. Defaults to 'sentence-transformers/all-MiniLM-L6-v2'. |
| use_api | Whether to use the Inference API to run the model (TRUE) or download and run the model locally (FALSE). Defaults to FALSE |

### Value

A sentence similarity object

### See Also

https://huggingface.co/docs/api-inference/detailed_parameters#sentence-similarity-task

### Examples

```
## Not run:
# Load the default model and use local inference
ez <- hf_ez_sentence_similarity()
ez$infer(
  source_sentence = "That is a happy person",
  sentences = list(
    "That is a happy dog",
    "That is a very happy person",
    "Today is a sunny day"
  )
)

# Use the API for inference.
ez <- hf_ez_sentence_similarity(use_api = TRUE)
ez$infer(
  source_sentence = "That is a happy person",
  sentences = list(
    "That is a happy dog",
    "That is a very happy person",
    "Today is a sunny day"
  )
)

## End(Not run)
```

---

## hf_ez_sentence_similarity_api_inference

*Sentence Similarity API Inference*

---

### Description

Sentence Similarity API Inference

## Usage

```
hf_ez_sentence_similarity_api_inference(
  source_sentence,
  sentences,
  tidy = TRUE,
  use_gpu = FALSE,
  use_cache = FALSE,
  wait_for_model = FALSE,
  use_auth_token = NULL,
  stop_on_error = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| source_sentence | The string that you wish to compare the other strings with. This can be a phrase, sentence, or longer passage, depending on the model being used. |
| sentences | A list of strings which will be compared against the source_sentence. |
| tidy | Whether to tidy the results into a tibble. Default: TRUE (tidy the results) |
| use_gpu | Whether to use GPU for inference. |
| use_cache | Whether to use cached inference results for previously seen inputs. |
| wait_for_model | Whether to wait for the model to be ready instead of receiving a 503 error after a certain amount of time. |
| use_auth_token | The token to use as HTTP bearer authorization for the Inference API. Defaults to HUGGING_FACE_HUB_TOKEN environment variable. |
| stop_on_error | Whether to throw an error if an API error is encountered. Defaults to FALSE (do not throw error). |
| ... | Additional arguments passed internally, including the model object or model ID. |

## Value

The results of the inference

## See Also

https://huggingface.co/docs/api-inference/index

## hf_ez_sentence_similarity_local_inference *Sentence Similarity Local Inference*

### Description

Sentence Similarity Local Inference

### Usage

```
hf_ez_sentence_similarity_local_inference(
  source_sentence,
  sentences,
  tidy = TRUE,
  ...
)
```

### Arguments

| | |
|---|---|
| source_sentence | The string that you wish to compare the other strings with. This can be a phrase, sentence, or longer passage, depending on the model being used. |
| sentences | A list of strings which will be compared against the source_sentence. |
| tidy | Whether to tidy the results into a tibble. Default: TRUE (tidy the results) |
| ... | Additional arguments passed internally, including the model object or model ID. |

### Value

The results of the inference

### See Also

https://huggingface.co/docs/transformers/main/en/pipeline_tutorial

## hf_ez_summarization *Summarize a Text*

### Description

This task is well known to summarize longer text into shorter text. Be careful, some models have a maximum length of input. That means that the summary cannot handle full books for instance. Be careful when choosing your model.

### Usage

```
hf_ez_summarization(model_id = "facebook/bart-large-cnn", use_api = FALSE)
```

### Arguments

| | |
|---|---|
| model_id | A model_id. Run hf_search_models(...) for model_ids. Defaults to 'facebook/bart-large-cnn'. |
| use_api | Whether to use the Inference API to run the model (TRUE) or download and run the model locally (FALSE). Defaults to FALSE |

### Value

A summarization object

### See Also

https://huggingface.co/docs/api-inference/detailed_parameters#summarization-task

### Examples

```
## Not run:
# Load the default model and use local inference
ez <- hf_ez_summarization()
text <- paste(
  "The Eiffel Tower is 324 metres tall and located in Paris.",
  "It was the world's tallest man-made structure for 41 years."
)
ez$infer(string = text, min_length = 10, max_length = 40)

# Load a specific model and use the API for inference.
ez <- hf_ez_summarization(model_id = 'xlm-roberta-base', use_api = TRUE)
ez$infer(string = "huggingface is the <mask>!")

## End(Not run)
```

---

## hf_ez_summarization_api_inference

*Summarization API Inference*

---

### Description

Summarization API Inference

### Usage

```
hf_ez_summarization_api_inference(
  string,
  min_length = NULL,
  max_length = NULL,
  top_k = NULL,
  top_p = NULL,
  temperature = 1,
```

```
  repetition_penalty = NULL,
  max_time = NULL,
  tidy = TRUE,
  use_gpu = FALSE,
  use_cache = FALSE,
  wait_for_model = FALSE,
  use_auth_token = NULL,
  stop_on_error = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| string | a string to be summarized |
| min_length | Integer to define the minimum length in tokens of the output summary. Default: NULL |
| max_length | Integer to define the maximum length in tokens of the output summary. Default: NULL |
| top_k | Integer to define the top tokens considered within the sample operation to create new text. Default: NULL |
| top_p | Float to define the tokens that are within the sample operation of text generation. Add tokens in the sample for more probable to least probable until the sum of the probabilities is greater than top_p. Default: NULL |
| temperature | Float (0.0-100.0). The temperature of the sampling operation. 1 means regular sampling, 0 means always take the highest score, 100.0 is getting closer to uniform probability. Default: 1.0 |
| repetition_penalty | Float (0.0-100.0). The more a token is used within generation the more it is penalized to not be picked in successive generation passes. Default: NULL |
| max_time | Float (0-120.0). The amount of time in seconds that the query should take maximum. Network can cause some overhead so it will be a soft limit. Default: NULL |
| tidy | Whether to tidy the results into a tibble. Default: TRUE (tidy the results) |
| use_gpu | Whether to use GPU for inference. |
| use_cache | Whether to use cached inference results for previously seen inputs. |
| wait_for_model | Whether to wait for the model to be ready instead of receiving a 503 error after a certain amount of time. |
| use_auth_token | The token to use as HTTP bearer authorization for the Inference API. Defaults to HUGGING_FACE_HUB_TOKEN environment variable. |
| stop_on_error | Whether to throw an error if an API error is encountered. Defaults to FALSE (do not throw error). |
| ... | Additional arguments passed internally, including the model object or model ID. |

## Value

The results of the inference

## See Also

https://huggingface.co/docs/api-inference/index

---

`hf_ez_summarization_local_inference` *Summarization Local Inference*

---

## Description

Summarization Local Inference

## Usage

```
hf_ez_summarization_local_inference(
  string,
  min_length = NULL,
  max_length = NULL,
  top_k = NULL,
  top_p = NULL,
  temperature = 1,
  repetition_penalty = NULL,
  max_time = NULL,
  tidy = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| `string` | a string to be summarized |
| `min_length` | Integer to define the minimum length in tokens of the output summary. Default: NULL |
| `max_length` | Integer to define the maximum length in tokens of the output summary. Default: NULL |
| `top_k` | Integer to define the top tokens considered within the sample operation to create new text. Default: NULL |
| `top_p` | Float to define the tokens that are within the sample operation of text generation. Add tokens in the sample for more probable to least probable until the sum of the probabilities is greater than top_p. Default: NULL |
| `temperature` | Float (0.0-100.0). The temperature of the sampling operation. 1 means regular sampling, 0 means always take the highest score, 100.0 is getting closer to uniform probability. Default: 1.0 |
| `repetition_penalty` | Float (0.0-100.0). The more a token is used within generation the more it is penalized to not be picked in successive generation passes. Default: NULL |

| | |
|---|---|
| max_time | Float (0-120.0). The amount of time in seconds that the query should take maximum. Network can cause some overhead so it will be a soft limit. Default: NULL |
| tidy | Whether to tidy the results into a tibble. Default: TRUE (tidy the results) |
| ... | Additional arguments passed internally, including the model object or model ID. |

## Value

The results of the inference

## See Also

https://huggingface.co/docs/transformers/main/en/pipeline_tutorial

---

# hf_ez_table_question_answering

*Answer Questions about a Data Table*

---

## Description

Don't know SQL? Don't want to dive into a large spreadsheet? Ask questions in plain english!

## Usage

```
hf_ez_table_question_answering(
  model_id = "google/tapas-base-finetuned-wtq",
  use_api = FALSE
)
```

## Arguments

| | |
|---|---|
| model_id | A model_id. Run hf_search_models(...) for model_ids. Defaults to 'google/tapas-base-finetuned-wtq'. |
| use_api | Whether to use the Inference API to run the model (TRUE) or download and run the model locally (FALSE). Defaults to FALSE |

## Value

A table question answering object

## See Also

https://huggingface.co/docs/api-inference/detailed_parameters#table-question-answering-task

## Examples

```
## Not run:
# Create a table to query
qa_table <-
  tibble::tibble(Repository = c('Transformers', 'Datasets', 'Tokenizers'),
                 Stars = c('36542', '4512', '3934'),
                 Contributors = c('651', '77', '34'),
                Programming.language = c('Python', 'Python', 'Rust, Python and NodeJS'))

# Load the default model and use local inference
ez <- hf_ez_table_question_answering()
ez$infer(query = "How many stars does the transformers repository have?", table = qa_table)

# Use the api for inference.
ez <- hf_ez_fill_mask(use_api = TRUE)
ez$infer(query = "How many stars does the transformers repository have?", table = qa_table)

## End(Not run)
```

---

# hf_ez_table_question_answering_api_inference

*Table Question Answering API Inference*

---

## Description

Table Question Answering API Inference

## Usage

```
hf_ez_table_question_answering_api_inference(
  query,
  table,
  tidy = TRUE,
  use_gpu = FALSE,
  use_cache = FALSE,
  wait_for_model = FALSE,
  use_auth_token = NULL,
  stop_on_error = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| query | The query in plain text that you want to ask the table |
| table | A dataframe with all text columns. |
| tidy | Whether to tidy the results into a tibble. Default: TRUE (tidy the results) |
| use_gpu | Whether to use GPU for inference. |

| | |
|---|---|
| use_cache | Whether to use cached inference results for previously seen inputs. |
| wait_for_model | Whether to wait for the model to be ready instead of receiving a 503 error after a certain amount of time. |
| use_auth_token | The token to use as HTTP bearer authorization for the Inference API. Defaults to HUGGING_FACE_HUB_TOKEN environment variable. |
| stop_on_error | Whether to throw an error if an API error is encountered. Defaults to FALSE (do not throw error). |
| ... | Additional arguments passed internally, including the model object or model ID. |

## Value

The results of the inference

## See Also

https://huggingface.co/docs/api-inference/index

---

`hf_ez_table_question_answering_local_inference`
*Table Question Answering Local Inference*

---

## Description

Table Question Answering Local Inference

## Usage

```
hf_ez_table_question_answering_local_inference(query, table, tidy = TRUE, ...)
```

## Arguments

| | |
|---|---|
| query | The query in plain text that you want to ask the table |
| table | A dataframe with all text columns. |
| tidy | Whether to tidy the results into a tibble. Default: TRUE (tidy the results) |
| ... | Additional arguments passed internally, including the model object or model ID. |

## Value

The results of the inference

## See Also

https://huggingface.co/docs/transformers/main/en/pipeline_tutorial

`hf_ez_text_classification` *Classify Texts into Pre-trained Categories*

## Description

Usually used for sentiment-analysis this will output the likelihood of classes of an input.

## Usage

```
hf_ez_text_classification(
  model_id = "distilbert/distilbert-base-uncased-finetuned-sst-2-english",
  use_api = FALSE
)
```

## Arguments

| | |
|---|---|
| `model_id` | A model_id. Run hf_search_models(...) for model_ids. Defaults to 'distilbert-base-uncased-finetuned-sst-2-english'. |
| `use_api` | Whether to use the Inference API to run the model (TRUE) or download and run the model locally (FALSE). Defaults to FALSE |

## Value

A text classification object

## See Also

https://huggingface.co/docs/api-inference/detailed_parameters#text-classification-task

## Examples

```
## Not run:
# Load the default model and use local inference
ez <- hf_ez_text_classification()
ez$infer(string = c('I like you. I love you'), flatten = FALSE)

# Use the API for inference.
ez <- hf_ez_text_classification(use_api = TRUE)
ez$infer(string = c('I like you. I love you'), flatten = FALSE)

## End(Not run)
```

hf_ez_text_classification_api_inference
*Text Classification API Inference*

## Description

Text Classification API Inference

## Usage

```
hf_ez_text_classification_api_inference(
  string,
  tidy = TRUE,
  use_gpu = FALSE,
  use_cache = FALSE,
  wait_for_model = FALSE,
  use_auth_token = NULL,
  stop_on_error = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| string | a string to be classified |
| tidy | Whether to tidy the results into a tibble. Default: TRUE (tidy the results) |
| use_gpu | Whether to use GPU for inference. |
| use_cache | Whether to use cached inference results for previously seen inputs. |
| wait_for_model | Whether to wait for the model to be ready instead of receiving a 503 error after a certain amount of time. |
| use_auth_token | The token to use as HTTP bearer authorization for the Inference API. Defaults to HUGGING_FACE_HUB_TOKEN environment variable. |
| stop_on_error | Whether to throw an error if an API error is encountered. Defaults to FALSE (do not throw error). |
| ... | Additional arguments passed internally, including the model object or model ID. |

## Value

The results of the inference

## See Also

https://huggingface.co/docs/api-inference/index

## hf_ez_text_classification_local_inference *Text Classification Local Inference*

### Description

Text Classification Local Inference

### Usage

```
hf_ez_text_classification_local_inference(string, tidy = TRUE, ...)
```

### Arguments

| | |
|---|---|
| string | a string to be classified |
| tidy | Whether to tidy the results into a tibble. Default: TRUE (tidy the results) |
| ... | Additional arguments passed internally, including the model object or model ID. |

### Value

The results of the inference

### See Also

https://huggingface.co/docs/transformers/main/en/pipeline_tutorial

## hf_ez_text_generation *Generate Text from a Prompt*

### Description

Use to continue text from a prompt. This is a very generic task.

### Usage

```
hf_ez_text_generation(model_id = "openai-community/gpt2", use_api = FALSE)
```

### Arguments

| | |
|---|---|
| model_id | A model_id. Run hf_search_models(...) for model_ids. Defaults to 'gpt2'. |
| use_api | Whether to use the Inference API to run the model (TRUE) or download and run the model locally (FALSE). Defaults to FALSE |

### Value

A text generation object

## See Also

https://huggingface.co/docs/api-inference/detailed_parameters#text-generation-task

## Examples

```
## Not run:
# Load the default model and use local inference
ez <- hf_ez_text_generation()
ez$infer(string = 'The answer to the universe is')

# Use the api for inference.
ez <- hf_ez_text_generation(use_api = TRUE)
ez$infer(string = 'The answer to the universe is')

## End(Not run)
```

---

# hf_ez_text_generation_api_inference

*Text Generation API Inference*

---

## Description

Text Generation API Inference

## Usage

```
hf_ez_text_generation_api_inference(
  string,
  top_k = NULL,
  top_p = NULL,
  temperature = 1,
  repetition_penalty = NULL,
  max_new_tokens = NULL,
  max_time = NULL,
  return_full_text = TRUE,
  num_return_sequences = 1L,
  do_sample = TRUE,
  tidy = TRUE,
  use_gpu = FALSE,
  use_cache = FALSE,
  wait_for_model = FALSE,
  use_auth_token = NULL,
  stop_on_error = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| string | a string to be generated from |
| top_k | (Default: None). Integer to define the top tokens considered within the sample operation to create new text. |
| top_p | (Default: None). Float to define the tokens that are within the sample operation of text generation. Add tokens in the sample for more probable to least probable until the sum of the probabilities is greater than top_p |
| temperature | Float (0.0-100.0). The temperature of the sampling operation. 1 means regular sampling, 0 means always take the highest score, 100.0 is getting closer to uniform probability. Default: 1.0 |
| repetition_penalty | (Default: None). Float (0.0-100.0). The more a token is used within generation the more it is penalized to not be picked in successive generation passes. |
| max_new_tokens | (Default: None). Int (0-250). The amount of new tokens to be generated, this does not include the input length it is a estimate of the size of generated text you want. Each new tokens slows down the request, so look for balance between response times and length of text generated. |
| max_time | (Default: None). Float (0-120.0). The amount of time in seconds that the query should take maximum. Network can cause some overhead so it will be a soft limit. Use that in combination with max_new_tokens for best results. |
| return_full_text | (Default: True). Bool. If set to False, the return results will not contain the original query making it easier for prompting. |
| num_return_sequences | (Default: 1). Integer. The number of proposition you want to be returned. |
| do_sample | (Optional: True). Bool. Whether or not to use sampling, use greedy decoding otherwise.#' |
| tidy | Whether to tidy the results into a tibble. Default: TRUE (tidy the results) |
| use_gpu | Whether to use GPU for inference. |
| use_cache | Whether to use cached inference results for previously seen inputs. |
| wait_for_model | Whether to wait for the model to be ready instead of receiving a 503 error after a certain amount of time. |
| use_auth_token | The token to use as HTTP bearer authorization for the Inference API. Defaults to HUGGING_FACE_HUB_TOKEN environment variable. |
| stop_on_error | Whether to throw an error if an API error is encountered. Defaults to FALSE (do not throw error). |
| ... | Additional arguments passed internally, including the model object or model ID. |

## Value

The results of the inference

## See Also

https://huggingface.co/docs/api-inference/index

`hf_ez_text_generation_local_inference` *Text Generation Local Inference*

## Description

Text Generation Local Inference

## Usage

```
hf_ez_text_generation_local_inference(
  string,
  top_k = NULL,
  top_p = NULL,
  temperature = 1,
  repetition_penalty = NULL,
  max_new_tokens = NULL,
  max_time = NULL,
  return_full_text = TRUE,
  num_return_sequences = 1L,
  do_sample = TRUE,
  tidy = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| `string` | a string to be generated from |
| `top_k` | (Default: None). Integer to define the top tokens considered within the sample operation to create new text. |
| `top_p` | (Default: None). Float to define the tokens that are within the sample operation of text generation. Add tokens in the sample for more probable to least probable until the sum of the probabilities is greater than top_p |
| `temperature` | Float (0.0-100.0). The temperature of the sampling operation. 1 means regular sampling, 0 means always take the highest score, 100.0 is getting closer to uniform probability. Default: 1.0 |
| `repetition_penalty` | (Default: None). Float (0.0-100.0). The more a token is used within generation the more it is penalized to not be picked in successive generation passes. |
| `max_new_tokens` | (Default: None). Int (0-250). The amount of new tokens to be generated, this does not include the input length it is a estimate of the size of generated text you want. Each new tokens slows down the request, so look for balance between response times and length of text generated. |
| `max_time` | (Default: None). Float (0-120.0). The amount of time in seconds that the query should take maximum. Network can cause some overhead so it will be a soft limit. Use that in combination with max_new_tokens for best results. |

`return_full_text`
: (Default: True). Bool. If set to False, the return results will not contain the original query making it easier for prompting.

`num_return_sequences`
: (Default: 1). Integer. The number of proposition you want to be returned.

`do_sample`
: (Optional: True). Bool. Whether or not to use sampling, use greedy decoding otherwise.#'

`tidy`
: Whether to tidy the results into a tibble. Default: TRUE (tidy the results)

`...`
: Additional arguments passed internally, including the model object or model ID.

## Value

The results of the inference

## See Also

https://huggingface.co/docs/transformers/main/en/pipeline_tutorial

---

# hf_ez_text2text_generation

*Answer General Questions*

---

## Description

Essentially Text-generation task. But uses Encoder-Decoder architecture, so might change in the future for more options.

## Usage

```
hf_ez_text2text_generation(model_id = "google/flan-t5-large", use_api = FALSE)
```

## Arguments

`model_id`
: A model_id. Run hf_search_models(...) for model_ids. Defaults to 'google/flan-t5-small'.

`use_api`
: Whether to use the Inference API to run the model (TRUE) or download and run the model locally (FALSE). Defaults to FALSE

## Value

A text2text generation object

## See Also

https://huggingface.co/docs/api-inference/detailed_parameters#text2text-generation-task

## Examples

```
## Not run:
# Load the default model and use local inference
ez <- hf_ez_text2text_generation()
ez$infer("Please answer the following question. What is the boiling point of Nitrogen?")

# Use the api for inference.
ez <- hf_ez_text2text_generation(use_api = TRUE)
ez$infer("Please answer the following question. What is the boiling point of Nitrogen?")

## End(Not run)
```

---

# hf_ez_text2text_generation_api_inference

*Text2Text Generation API Inference*

---

## Description

Text2Text Generation API Inference

## Usage

```
hf_ez_text2text_generation_api_inference(
  string,
  tidy = TRUE,
  use_gpu = FALSE,
  use_cache = FALSE,
  wait_for_model = FALSE,
  use_auth_token = NULL,
  stop_on_error = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| string | a general request for the model to perform or answer |
| tidy | Whether to tidy the results into a tibble. Default: TRUE (tidy the results) |
| use_gpu | Whether to use GPU for inference. |
| use_cache | Whether to use cached inference results for previously seen inputs. |
| wait_for_model | Whether to wait for the model to be ready instead of receiving a 503 error after a certain amount of time. |
| use_auth_token | The token to use as HTTP bearer authorization for the Inference API. Defaults to HUGGING_FACE_HUB_TOKEN environment variable. |
| stop_on_error | Whether to throw an error if an API error is encountered. Defaults to FALSE (do not throw error). |
| ... | Additional arguments passed internally, including the model object or model ID. |

## Value

The results of the inference

## See Also

https://huggingface.co/docs/api-inference/index

---

`hf_ez_text2text_generation_local_inference`
*Text2Text Generation Local Inference*

---

## Description

Text2Text Generation Local Inference

## Usage

```
hf_ez_text2text_generation_local_inference(string, tidy = TRUE, ...)
```

## Arguments

| | |
|---|---|
| `string` | a general request for the model to perform or answer |
| `tidy` | Whether to tidy the results into a tibble. Default: TRUE (tidy the results) |
| `...` | Additional arguments passed internally, including the model object or model ID. |

## Value

The results of the inference

## See Also

https://huggingface.co/docs/transformers/main/en/pipeline_tutorial

`hf_ez_token_classification` *Classify parts of a Text*

## Description

Usually used for sentence parsing, either grammatical, or Named Entity Recognition (NER) to understand keywords contained within text.

## Usage

```
hf_ez_token_classification(
  model_id = "dbmdz/bert-large-cased-finetuned-conll03-english",
  use_api = FALSE
)
```

## Arguments

| | |
|---|---|
| `model_id` | A model_id. Run hf_search_models(...) for model_ids. Defaults to 'dbmdz/bert-large-cased-finetuned-conll03-english'. |
| `use_api` | Whether to use the Inference API to run the model (TRUE) or download and run the model locally (FALSE). Defaults to FALSE |

## Value

A text2text generation object

## See Also

https://huggingface.co/docs/api-inference/detailed_parameters#token-classification-task

## Examples

```
## Not run:
# Load the default named entity recognition model
ez <- hf_ez_token_classification()

# Run NER. Note how the full name is aggregated into one named entity.
ez$infer(
  string = "My name is Sarah Jessica Parker but you can call me Jessica",
  aggregation_strategy = "simple"
)

# Run NER without aggregation.
# Note how the full name is separated into distinct named entities.
ez$infer(
  string = "My name is Sarah Jessica Parker but you can call me Jessica",
  aggregation_strategy = "none"
```

```
)

## End(Not run)
```

---

`hf_ez_token_classification_api_inference` *Token Classification API Inference*

---

## Description

Token Classification API Inference

## Usage

```
hf_ez_token_classification_api_inference(
  string,
  aggregation_strategy = "simple",
  tidy = TRUE,
  use_gpu = FALSE,
  use_cache = FALSE,
  wait_for_model = FALSE,
  use_auth_token = NULL,
  stop_on_error = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| `string` | a string to be classified |
| `aggregation_strategy` | (Default: simple). There are several aggregation strategies.<br>none: Every token gets classified without further aggregation.<br>simple: Entities are grouped according to the default schema (B-, I- tags get merged when the tag is similar).<br>first: Same as the simple strategy except words cannot end up with different tags. Words will use the tag of the first token when there is ambiguity.<br>average: Same as the simple strategy except words cannot end up with different tags. Scores are averaged across tokens and then the maximum label is applied.<br>max: Same as the simple strategy except words cannot end up with different tags. Word entity will be the token with the maximum score. |
| `tidy` | Whether to tidy the results into a tibble. Default: TRUE (tidy the results) |
| `use_gpu` | Whether to use GPU for inference. |
| `use_cache` | Whether to use cached inference results for previously seen inputs. |
| `wait_for_model` | Whether to wait for the model to be ready instead of receiving a 503 error after a certain amount of time. |

| | |
|---|---|
| use_auth_token | The token to use as HTTP bearer authorization for the Inference API. Defaults to HUGGING_FACE_HUB_TOKEN environment variable. |
| stop_on_error | Whether to throw an error if an API error is encountered. Defaults to FALSE (do not throw error). |
| ... | Additional arguments passed internally, including the model object or model ID. |

## Value

The results of the inference

## See Also

https://huggingface.co/docs/api-inference/index

---

`hf_ez_token_classification_local_inference`
*Token Classification Local Inference*

---

## Description

Token Classification Local Inference

## Usage

```
hf_ez_token_classification_local_inference(
  string,
  aggregation_strategy = "simple",
  tidy = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| string | a string to be classified |
| aggregation_strategy | (Default: simple). There are several aggregation strategies.<br>none: Every token gets classified without further aggregation.<br>simple: Entities are grouped according to the default schema (B-, I- tags get merged when the tag is similar).<br>first: Same as the simple strategy except words cannot end up with different tags. Words will use the tag of the first token when there is ambiguity.<br>average: Same as the simple strategy except words cannot end up with different tags. Scores are averaged across tokens and then the maximum label is applied.<br>max: Same as the simple strategy except words cannot end up with different tags. Word entity will be the token with the maximum score. |
| tidy | Whether to tidy the results into a tibble. Default: TRUE (tidy the results) |
| ... | Additional arguments passed internally, including the model object or model ID. |

## Value

The results of the inference

## See Also

https://huggingface.co/docs/transformers/main/en/pipeline_tutorial

---

`hf_ez_translation` *Translate between Languages*

---

## Description

This task is well known to translate text from one language to another

## Usage

```
hf_ez_translation(model_id = "Helsinki-NLP/opus-mt-en-es", use_api = FALSE)
```

## Arguments

| | |
|---|---|
| `model_id` | A model_id. Run hf_search_models(...) for model_ids. Defaults to 'Helsinki-NLP/opus-mt-en-es'. |
| `use_api` | Whether to use the Inference API to run the model (TRUE) or download and run the model locally (FALSE). Defaults to FALSE |

## Value

A translation object

## See Also

https://huggingface.co/docs/api-inference/detailed_parameters#translation-task

## Examples

```
## Not run:
# Load the default translation model
ez <- hf_ez_translation()

# Translate from English to Spanish.
ez$infer(string = "My name is Sarah and I live in London")

## End(Not run)
```

hf_ez_translation_api_inference
*Translation API Inference*

## Description

Translation API Inference

## Usage

```
hf_ez_translation_api_inference(
  string,
  tidy = TRUE,
  use_gpu = FALSE,
  use_cache = FALSE,
  wait_for_model = FALSE,
  use_auth_token = NULL,
  stop_on_error = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| string | a string to be translated |
| tidy | Whether to tidy the results into a tibble. Default: TRUE (tidy the results) |
| use_gpu | Whether to use GPU for inference. |
| use_cache | Whether to use cached inference results for previously seen inputs. |
| wait_for_model | Whether to wait for the model to be ready instead of receiving a 503 error after a certain amount of time. |
| use_auth_token | The token to use as HTTP bearer authorization for the Inference API. Defaults to HUGGING_FACE_HUB_TOKEN environment variable. |
| stop_on_error | Whether to throw an error if an API error is encountered. Defaults to FALSE (do not throw error). |
| ... | Additional arguments passed internally, including the model object or model ID. |

## Value

The results of the inference

## See Also

https://huggingface.co/docs/api-inference/index

## hf_ez_translation_local_inference

*Translation Local Inference*

### Description

Translation Local Inference

### Usage

```
hf_ez_translation_local_inference(string, tidy = TRUE, ...)
```

### Arguments

| | |
|---|---|
| string | a string to be translated |
| tidy | Whether to tidy the results into a tibble. Default: TRUE (tidy the results) |
| ... | Additional arguments passed internally, including the model object or model ID. |

### Value

The results of the inference

### See Also

https://huggingface.co/docs/transformers/main/en/pipeline_tutorial

## hf_ez_zero_shot_classification

*Perform Text Classification with No Context Required*

### Description

This task is super useful to try out classification with zero code, you simply pass a sentence/paragraph and the possible labels for that sentence, and you get a result.

### Usage

```
hf_ez_zero_shot_classification(
  model_id = "facebook/bart-large-mnli",
  use_api = FALSE
)
```

### Arguments

| | |
|---|---|
| model_id | A model_id. Run hf_search_models(...) for model_ids. Defaults to 'facebook/bart-large-mnli'. |
| use_api | Whether to use the Inference API to run the model (TRUE) or download and run the model locally (FALSE). Defaults to FALSE |

### Value

A zero shot classification object

### See Also

https://huggingface.co/docs/api-inference/detailed_parameters#zero-shot-classification-task

### Examples

```
## Not run:
# Load the default model
ez <- hf_ez_zero_shot_classification()

# Classify the string
ez$infer(
  string = paste(
    "Hi, I recently bought a device from your company but it is not working",
    "as advertised and I would like to get reimbursed!"
  ),
  candidate_labels = c("refund", "legal", "faq")
)

## End(Not run)
```

---

## hf_ez_zero_shot_classification_api_inference

*Zero Shot Classification API Inference*

---

### Description

Zero Shot Classification API Inference

### Usage

```
hf_ez_zero_shot_classification_api_inference(
  string,
  candidate_labels,
  multi_label = FALSE,
  tidy = TRUE,
  use_gpu = FALSE,
  use_cache = FALSE,
```

```
  wait_for_model = FALSE,
  use_auth_token = NULL,
  stop_on_error = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| string | a string or list of strings |
| candidate_labels | a list of strings that are potential classes for inputs. (max 10 candidate_labels, for more, simply run multiple requests, results are going to be misleading if using too many candidate_labels anyway. If you want to keep the exact same, you can simply run multi_label=True and do the scaling on your end. ) |
| multi_label | (Default: false) Boolean that is set to True if classes can overlap |
| tidy | Whether to tidy the results into a tibble. Default: TRUE (tidy the results) |
| use_gpu | Whether to use GPU for inference. |
| use_cache | Whether to use cached inference results for previously seen inputs. |
| wait_for_model | Whether to wait for the model to be ready instead of receiving a 503 error after a certain amount of time. |
| use_auth_token | The token to use as HTTP bearer authorization for the Inference API. Defaults to HUGGING_FACE_HUB_TOKEN environment variable. |
| stop_on_error | Whether to throw an error if an API error is encountered. Defaults to FALSE (do not throw error). |
| ... | Additional arguments passed internally, including the model object or model ID. |

## Value

The results of the inference

## See Also

https://huggingface.co/docs/api-inference/index

---

# hf_ez_zero_shot_classification_local_inference

*Zero Shot Classification Local Inference*

---

## Description

Zero Shot Classification Local Inference

### Usage

```
hf_ez_zero_shot_classification_local_inference(
  string,
  candidate_labels,
  multi_label = FALSE,
  tidy = TRUE,
  ...
)
```

### Arguments

| | |
|---|---|
| string | a string or list of strings |
| candidate_labels | a list of strings that are potential classes for inputs. (max 10 candidate_labels, for more, simply run multiple requests, results are going to be misleading if using too many candidate_labels anyway. If you want to keep the exact same, you can simply run multi_label=True and do the scaling on your end. ) |
| multi_label | (Default: false) Boolean that is set to True if classes can overlap |
| tidy | Whether to tidy the results into a tibble. Default: TRUE (tidy the results) |
| ... | Additional arguments passed internally, including the model object or model ID. |

### Value

The results of the inference

### See Also

https://huggingface.co/docs/transformers/main/en/pipeline_tutorial

---

## hf_fill_mask *Fill Mask*

---

### Description

Fill in a [MASK] token in text with predicted words. Commonly used with BERT-style models.

### Usage

```
hf_fill_mask(
  text,
  model = hf_default_model("fill_mask"),
  mask_token = "[MASK]",
  top_k = 5,
  token = NULL,
  endpoint_url = NULL,
  ...
)
```

### Arguments

| | |
|---|---|
| text | Character vector of text(s) containing [MASK] token. |
| model | Character string. Model ID from Hugging Face Hub. Default: "google-bert/bert-base-uncased". |
| mask_token | Character string. The mask token to use. Default: "[MASK]". Some models use different tokens like "<mask>". |
| top_k | Integer. Number of top predictions to return. Default: 5. |
| token | Character string or NULL. API token for authentication. |
| endpoint_url | Character string or NULL. A custom Inference Endpoint URL. |
| ... | Additional arguments (currently unused). |

### Value

A tibble with columns: text, token, score, filled (the complete text)

### Examples

```
## Not run:
# Fill in the blank
hf_fill_mask("The capital of France is [MASK].")

# Get top predictions
hf_fill_mask("Paris is the [MASK] of France.", top_k = 3)

# Use with different mask token
hf_fill_mask("The capital of France is <mask>.", mask_token = "<mask>")

## End(Not run)
```

---

## hf_fill_mask_payload *Fill Mask Task Payload*

---

### Description

Tries to fill in a hole with a missing word (token to be precise). That's the base task for BERT models.

### Usage

```
hf_fill_mask_payload(string)
```

### Arguments

| | |
|---|---|
| string | a string to be filled from, must contain the [MASK] token (check model card for exact name of the mask) |

### Value

An inference payload

### See Also

https://huggingface.co/docs/api-inference/detailed_parameters#fill-mask-task

### Examples

```
hf_fill_mask_payload("The capital of France is [MASK].")
```

---

## hf_generate *Text Generation*

---

### Description

Generate text from a prompt using a language model via the Inference Providers API.

### Usage

```
hf_generate(
  prompt,
  model = hf_default_model("generate"),
  max_new_tokens = 50,
  temperature = 1,
  top_p = NULL,
  token = NULL,
  endpoint_url = NULL,
  ...
)
```

### Arguments

| | |
|---|---|
| prompt | Character vector of text prompt(s) to generate from. |
| model | Character string. Model ID from Hugging Face Hub. Default: "meta-llama/Llama-3.1-8B-Instruct". |
| max_new_tokens | Integer. Maximum number of tokens to generate. Default: 50. |
| temperature | Numeric. Sampling temperature (0-2). Default: 1.0. |
| top_p | Numeric. Nucleus sampling parameter. Default: NULL. |
| token | Character string or NULL. API token for authentication. |
| endpoint_url | Character string or NULL. A custom Inference Endpoint URL. The endpoint must support the chat completions format. |
| ... | Additional parameters passed to the model. |

## Value

A tibble with columns: prompt, generated_text

## Examples

```
## Not run:
# Simple text generation
hf_generate("Once upon a time in a land far away,")

# With different model
hf_generate("The future of AI is", model = "meta-llama/Llama-3-8B-Instruct:together")

## End(Not run)
```

---

# hf_hub_download *Download a File from the Hub*

---

## Description

Download a single file from a model, dataset, or Space repository.

## Usage

```
hf_hub_download(
  repo_id,
  filename,
  repo_type = "model",
  revision = "main",
  dest = NULL,
  token = NULL,
  overwrite = FALSE
)
```

## Arguments

| | |
|---|---|
| repo_id | Character string. Repository ID. |
| filename | Character string. Path to the file inside the repository. |
| repo_type | Character string. One of "model", "dataset", or "space". |
| revision | Character string. Git revision, branch, or tag. |
| dest | Character path or NULL. If NULL, writes to a temporary file. If an existing directory, the repository filename basename is used inside it. |
| token | Character string or NULL. API token for private repositories. |
| overwrite | Logical. If TRUE, overwrite an existing destination file. |

## Value

The downloaded file path.

## Examples

```
## Not run:
hf_hub_download("BAAI/bge-small-en-v1.5", "README.md")

## End(Not run)
```

---

`hf_inference` *Inference using a downloaded Hugging Face model or pipeline, or using the Inference API*

---

## Description

If a model_id is provided, the Inference API will be used to make the prediction. If you wish to download a model or pipeline rather than running your predictions through the Inference API, download the model with one of the hf_load_*_model() or hf_load_pipeline() functions.

## Usage

```
hf_inference(
  model,
  payload,
  flatten = TRUE,
  use_gpu = FALSE,
  use_cache = FALSE,
  wait_for_model = FALSE,
  use_auth_token = NULL,
  stop_on_error = FALSE
)
```

## Arguments

| | |
|---|---|
| `model` | Either a downloaded model or pipeline from the Hugging Face Hub (using hf_load_pipeline()), or a model_id. Run hf_search_models(...) for model_ids. |
| `payload` | The data to predict on. Use one of the hf_*_payload() functions to create. |
| `flatten` | Whether to flatten the results into a data frame. Default: TRUE (flatten the results) |
| `use_gpu` | API Only - Whether to use GPU for inference. |
| `use_cache` | API Only - Whether to use cached inference results for previously seen inputs. |
| `wait_for_model` | API Only - Whether to wait for the model to be ready instead of receiving a 503 error after a certain amount of time. |
| `use_auth_token` | API Only - The token to use as HTTP bearer authorization for the Inference API. Defaults to HUGGING_FACE_HUB_TOKEN environment variable. |
| `stop_on_error` | API Only - Whether to throw an error if an API error is encountered. Defaults to FALSE (do not throw error). |

## Value

The results of the inference

## See Also

https://huggingface.co/docs/api-inference/index

## Examples

```
## Not run:
payload <- hf_text_classification_payload("I love R.")
hf_inference("distilbert-base-uncased-finetuned-sst-2-english", payload)

## End(Not run)
```

---

# hf_list_authors *List Authors*

---

## Description

List Model Authors

## Usage

```
hf_list_authors(pattern = NULL)
```

## Arguments

| | |
|---|---|
| pattern | A search term or regular expression. Defaults to NULL (return all results). |

## Value

A character vector of matching model authors.

## See Also

https://huggingface.co/docs/hub/searching-the-hub

## Examples

```
## Not run:
hf_list_authors(pattern = "^sam")

## End(Not run)
```

## hf_list_datasets *List Datasets*

### Description

List Model Datasets

### Usage

```
hf_list_datasets(pattern = NULL)
```

### Arguments

| | |
|---|---|
| pattern | A search term or regular expression. Defaults to NULL (return all results). |

### Value

A character vector of matching dataset names.

### See Also

https://huggingface.co/docs/hub/searching-the-hub

### Examples

```
## Not run:
hf_list_datasets("imdb")

## End(Not run)
```

## hf_list_languages *List Languages*

### Description

List Model Languages

### Usage

```
hf_list_languages(pattern = NULL)
```

### Arguments

| | |
|---|---|
| pattern | A search term or regular expression. Defaults to NULL (return all results). |

## Value

A character vector of matching language tags.

## See Also

https://huggingface.co/docs/hub/searching-the-hub

## Examples

```
## Not run:
hf_list_languages("es")

## End(Not run)
```

---

`hf_list_libraries` *List Libraries*

---

## Description

List Model Libraries

## Usage

```
hf_list_libraries(pattern = NULL)
```

## Arguments

| | |
|---|---|
| `pattern` | A search term or regular expression. Defaults to NULL (return all results). |

## Value

A character vector of matching library names.

## See Also

https://huggingface.co/docs/hub/searching-the-hub

## Examples

```
## Not run:
hf_list_libraries("pytorch|tensorflow")

## End(Not run)
```

## hf_list_licenses *List Licenses*

### Description

List Model Licenses

### Usage

```
hf_list_licenses(pattern = NULL)
```

### Arguments

| | |
|---|---|
| pattern | A search term or regular expression. Defaults to NULL (return all results). |

### Value

A character vector of matching license identifiers.

### See Also

https://huggingface.co/docs/hub/searching-the-hub

### Examples

```
## Not run:
hf_list_licenses("mit")

## End(Not run)
```

## hf_list_models *List Models*

### Description

List Model Names

### Usage

```
hf_list_models(pattern = NULL)
```

### Arguments

| | |
|---|---|
| pattern | A search term or regular expression. Defaults to NULL (return all results). |

### Value

A tibble with a ‘model‘ column containing matching model IDs.

## See Also

https://huggingface.co/docs/hub/searching-the-hub

## Examples

```
## Not run:
hf_list_models("bert-base-cased")

## End(Not run)
```

---

`hf_list_providers` *List Inference Providers for a Model*

---

## Description

Query Hugging Face router metadata for provider availability, pricing, latency, and capabilities. Router provider metadata is available for OpenAI-compatible models; non-router task models return an empty tibble.

## Usage

```
hf_list_providers(model_id, token = NULL)
```

## Arguments

| | |
|---|---|
| `model_id` | Character string. Model ID. |
| `token` | Character string or NULL. API token for authentication. |

## Value

A tibble with one row per provider.

## Examples

```
## Not run:
hf_list_providers("Qwen/Qwen2.5-72B-Instruct")

## End(Not run)
```

`hf_list_repo_files` *List Files in a Hub Repository*

## Description

List files and directories in a model, dataset, or Space repository.

## Usage

```
hf_list_repo_files(
  repo_id,
  repo_type = "model",
  revision = "main",
  recursive = TRUE,
  token = NULL
)
```

## Arguments

| | |
|---|---|
| `repo_id` | Character string. Repository ID, e.g. "BAAI/bge-small-en-v1.5". |
| `repo_type` | Character string. One of "model", "dataset", or "space". |
| `revision` | Character string. Git revision, branch, or tag. Default: "main". |
| `recursive` | Logical. If TRUE, list files recursively. |
| `token` | Character string or NULL. API token for private repositories. |

## Value

A tibble with columns: path, type, size, oid.

## Examples

```
## Not run:
hf_list_repo_files("BAAI/bge-small-en-v1.5")

## End(Not run)
```

## hf_list_tasks *List Available Tasks*

### Description

List all available task types on Hugging Face.

### Usage

```
hf_list_tasks(pattern = NULL)
```

### Arguments

| | |
|---|---|
| pattern | Character string or NULL. Optional regex pattern to filter tasks. |

### Value

A character vector of task names.

### Examples

```
## Not run:
# List all tasks
hf_list_tasks()

# Filter tasks
hf_list_tasks(pattern = "classification")

## End(Not run)
```

## hf_load_AutoModel_for_task *Import a pre-trained AutoModel object for a specific task.*

### Description

Use this function when you need to load an AutoModel for a specific task separate to a pipeline. The model's config should come ready-equipped for a specific task according to what you input as model_type.

### Usage

```
hf_load_AutoModel_for_task(
  model_type = "AutoModelForSequenceClassification",
  model_id,
  use_auth_token = NULL
)
```

### Arguments

| | |
|---|---|
| model_type | The AutoModel's type passed as a string e.g. c("AutoModelForQuestionAnswering", "AutoModelForTokenClassification", "AutoModelForSequenceClassification") |
| model_id | The model's name or id on the Hugging Face hub |
| use_auth_token | For private models, copy and paste your auth token in as a string. |

### Value

an AutoModel object for a specific task

### See Also

https://huggingface.co/transformers/v3.0.2/model_doc/auto.html

### Examples

```
## Not run:
model <- hf_load_AutoModel_for_task(
  "AutoModelForSequenceClassification",
  "distilbert-base-uncased-finetuned-sst-2-english"
)

## End(Not run)
```

---

## hf_load_dataset *Load Dataset via Hugging Face Datasets Server API*

---

### Description

Load a dataset from Hugging Face Hub using the Datasets Server API. This is an API-first approach that doesn't require Python. For local dataset loading with Python, see the legacy function or advanced vignette.

### Usage

```
hf_load_dataset(
  dataset,
  split = "train",
  config = NULL,
  limit = 1000,
  offset = 0,
  token = NULL
)
```

### Arguments

| | |
|---|---|
| dataset | Character string. Dataset name (e.g., "imdb", "squad"). |
| split | Character string. Dataset split: "train", "test", "validation", etc. Supports Hugging Face slice syntax such as ‘"train[100:200]"‘. Percentage slices like ‘"train[:10\ "train". |
| config | Character string or NULL. Dataset configuration/subset name. If NULL (default), auto-detected from the dataset's available configs. |
| limit | Integer. Maximum number of rows to fetch. Default: 1000. Set to Inf to fetch all rows (may be slow for large datasets). |
| offset | Integer. Row offset for pagination. Default: 0. |
| token | Character string or NULL. API token for private datasets. |

### Value

A tibble with dataset rows, plus .dataset and .split columns.

### Examples

```
## Not run:
# Load first 1000 rows of IMDB train set
imdb <- hf_load_dataset("imdb", split = "train", limit = 1000)

# Load test set
imdb_test <- hf_load_dataset("imdb", split = "test", limit = 500)

# Load a slice of a split
imdb_sample <- hf_load_dataset("imdb", split = "train[100:200]", limit = Inf)

## End(Not run)
```

---

## hf_load_model *Load a pre-trained AutoModel object from Hugging Face*

---

### Description

Load a pre-trained AutoModel object from Hugging Face

### Usage

```
hf_load_model(model_id, ...)
```

### Arguments

| | |
|---|---|
| model_id | model_id The id of the model given in the url by https://huggingface.co/model_name. |
| ... | sent to AutoModel.from_pretrained() |

## Value

a pre-trained model object

## Examples

```
## Not run:
model <- hf_load_model("distilbert-base-uncased")

## End(Not run)
```

---

`hf_load_pipeline` *Load a pipeline object from Hugging Face - pipelines usually include a model, tokenizer and task.*

---

## Description

Load Model from Hugging Face

## Usage

```
hf_load_pipeline(model_id, tokenizer = NULL, task = NULL, ...)
```

## Arguments

| | |
|---|---|
| `model_id` | The id of the model given in the url by https://huggingface.co/model_name. |
| `tokenizer` | The tokenizer function used to tokenize inputs. Defaults to NULL (one will be automatically loaded). |
| `task` | The task the model will accomplish. Run hf_list_tasks() for options. |
| `...` | Fed to the hf_pipeline function |

## Value

A Hugging Face model ready for prediction.

## See Also

https://huggingface.co/docs/transformers/main/en/pipeline_tutorial

## Examples

```
## Not run:
pipe <- hf_load_pipeline("distilbert-base-uncased-finetuned-sst-2-english",
                         task = "sentiment-analysis")
pipe("I love R.")

## End(Not run)
```

`hf_load_sentence_model` *Load a Sentence Transformers model to extract sentence/document embeddings*

## Description

Load a Sentence Transformers model to extract sentence/document embeddings

## Usage

```
hf_load_sentence_model(model_id, ...)
```

## Arguments

| | |
|---|---|
| `model_id` | The id of a sentence-transformers model. Use hf_search_models(author = 'sentence-transformers') to find suitable models. |
| `...` | Sent to model call, could include arguments such as use_auth_token = auth_token, device = device etc. |

## Value

A Huggingface model ready for prediction.

## See Also

https://huggingface.co/sentence-transformers

## Examples

```
## Not run:
# Compute sentence embeddings
sentences <- c("Baby turtles are so cute!", "He walks as slowly as a turtle.")
sentences_two <- c("The lake is cold today.", "I enjoy swimming in the lake.")
sentences <- c(sentences, sentences_two)
model <- hf_load_sentence_model('paraphrase-MiniLM-L6-v2')
embeddings <- model$encode(sentences)
embeddings %>% dist() %>% as.matrix() %>% as.data.frame() %>% setNames(sentences)
embddings <- embeddings %>% dplyr::mutate(`sentence 1` = sentences) %>%
tidyr::pivot_longer(cols = -`sentence 1`, names_to = 'sentence 2', values_to = 'distance')
embeddings <- embeddings %>% filter(distance > 0)
# Cluster sentences
embeddings <- embeddings %>%
t() %>% prcomp() %>%
purrr::pluck('rotation') %>%
as.data.frame() %>%
dplyr::mutate(sentence = sentences)
plot <- embedidings %>% ggplot2::ggplot(aes(PC1, PC2)) +
ggplot2::geom_label(ggplot2::aes(PC1, PC2, label = sentence, vjust="inward", hjust="inward")) +
```

```
ggplot2::theme_minimal()

## End(Not run)
```

---

`hf_load_tokenizer` *Load an AutoTokenizer from a pre-tained model*

---

## Description

Load Tokenizer for Hugging Face Model

## Usage

```
hf_load_tokenizer(model_id, ...)
```

## Arguments

| | |
|---|---|
| `model_id` | The id of the model given in the url by https://huggingface.co/model_name. |
| `...` | sent to the 'AutoTokenizer.from_pretained()', accepts named arguments e.g. use_fast https://huggingface.co/docs/transformers/main_classes/tokenizer |

## Value

A Hugging Face model's tokenizer.

## See Also

https://huggingface.co/docs/transformers/main/en/pipeline_tutorial

## Examples

```
## Not run:
tokenizer <- hf_load_tokenizer("distilbert-base-uncased")
tokenizer$tokenize("Hello from R")

## End(Not run)
```

## hf_model_info *Get Model Information*

### Description

Retrieve detailed information about a specific model.

### Usage

```
hf_model_info(model_id, token = NULL)
```

### Arguments

| | |
|---|---|
| model_id | Character string. The model ID (e.g., "bert-base-uncased"). |
| token | Character string or NULL. API token for authentication. |

### Value

A list with detailed model information.

### Examples

```
## Not run:
# Get model details
hf_model_info("sentence-transformers/all-MiniLM-L6-v2")

## End(Not run)
```

## hf_nearest_neighbors *Find Nearest Neighbors by Semantic Similarity*

### Description

Find the k most similar texts to a query text based on embedding similarity.

### Usage

```
hf_nearest_neighbors(
  data,
  query,
  k = 5,
  text_col = "text",
  model = hf_default_model("embed"),
  token = NULL
)
```

### Arguments

| | |
|---|---|
| data | A data frame with an 'embedding' column (from hf_embed_text). |
| query | Character string. The query text to compare against. |
| k | Integer. Number of nearest neighbors to return. Default: 5. |
| text_col | Character string. Name of text column. Default: "text". |
| model | Character string. Model to use for query embedding. Should match the model used for data embeddings. |
| token | Character string or NULL. API token for authentication. |

### Value

A tibble with the k nearest neighbors, sorted by similarity (descending).

### Examples

```
## Not run:
docs_embedded |>
  hf_nearest_neighbors("machine learning", k = 5)

## End(Not run)
```

---

## hf_ner *Named Entity Recognition (Token Classification)*

---

### Description

Extract named entities (people, organizations, locations, etc.) from text using a token-classification model via the Hugging Face Inference Providers API. Returns one row per detected entity, with character offsets that let you highlight or join back to the source text. Inputs that produce no entities (and ‘NA‘ inputs) yield a single row with ‘NA‘ entity fields so every input is represented.

### Usage

```
hf_ner(
  text,
  model = hf_default_model("ner"),
  aggregation_strategy = "simple",
  token = NULL,
  endpoint_url = NULL,
  ...
)
```

### Arguments

| | |
|---|---|
| text | Character vector of text(s) to analyze. |
| model | Character string. Model ID from the Hugging Face Hub. Append ‘":provider"‘ to select an inference provider. Default: "dslim/bert-base-NER". |
| aggregation_strategy | Character string. How sub-word tokens are grouped into entities: one of "none", "simple", "first", "average", "max". Default: "simple". |
| token | Character string or NULL. API token for authentication. |
| endpoint_url | Character string or NULL. A custom Inference Endpoint URL. |
| ... | Additional arguments (currently unused). |

### Value

A tibble with columns: text, word, entity_group, score, start, end.

### See Also

https://huggingface.co/docs/inference-providers/tasks/token-classification

### Examples

```
## Not run:
hf_ner("Barack Obama was born in Hawaii.")

# One row per entity, ready to count or join
library(dplyr)
hf_ner(headlines) |>
  filter(!is.na(word)) |>
  count(entity_group, sort = TRUE)

## End(Not run)
```

---

## hf_push_dataset *Push a Data Frame as a Dataset File*

---

### Description

Write a data frame to CSV or Parquet and upload it to a Hub dataset repository.

### Usage

```
hf_push_dataset(
  data,
  repo_id,
  path_in_repo = NULL,
  format = c("csv", "parquet"),
```

```
  create_repo = FALSE,
  private = FALSE,
  commit_message = NULL,
  token = NULL,
  overwrite = FALSE,
  confirm = FALSE
)
```

## Arguments

| | |
|---|---|
| data | A data frame. |
| repo_id | Character string. Dataset repository ID. |
| path_in_repo | Character string or NULL. Destination path. Defaults to "data.csv" or "data.parquet". |
| format | Character string. One of "csv" or "parquet". |
| create_repo | Logical. If TRUE, create the dataset repo first with ‘exist_ok = TRUE‘. |
| private | Logical. Used when ‘create_repo = TRUE‘. |
| commit_message | Character string or NULL. |
| token | Character string or NULL. API token with write scope. |
| overwrite | Logical. If FALSE, error when the destination already exists. |
| confirm | Logical. Must be TRUE to perform write operations. |

## Value

The result from ‘hf_upload_file()‘.

## Examples

```
## Not run:
hf_push_dataset(mtcars, "me/mtcars-small", create_repo = TRUE, confirm = TRUE)

## End(Not run)
```

---

# hf_python_depends *Install Python Dependencies*

---

## Description

Installs python packages needed to run huggingfaceR functions

## Usage

```
hf_python_depends(
  packages = c("transformers", "sentencepiece", "huggingface_hub", "datasets",
    "sentence-transformers")
)
```

### Arguments

| | |
|---|---|
| packages | Python libraries needed for local model usage.<br>Defaults to transformers, sentencepiece, huggingface_hub, datasets, and sentence-transformers. |

### Value

The value returned by ‘reticulate::conda_install()‘, called for its side effect.

### Examples

```
## Not run:
hf_python_depends()

## End(Not run)
```

---

## hf_question_answer *Extractive Question Answering*

---

### Description

Answer a question from a supplied context passage using an extractive question-answering model via the Hugging Face Inference Providers API. The answer is a span extracted verbatim from the context. ‘question‘ and ‘context‘ are recycled to a common length (each may be length 1).

### Usage

```
hf_question_answer(
  question,
  context,
  model = hf_default_model("question_answer"),
  token = NULL,
  endpoint_url = NULL,
  ...
)
```

### Arguments

| | |
|---|---|
| question | Character vector of question(s). |
| context | Character vector of context passage(s) to answer from. |
| model | Character string. Model ID from the Hugging Face Hub. Append ‘":provider"‘ to select an inference provider. Default: "deepset/roberta-base-squad2". |
| token | Character string or NULL. API token for authentication. |
| endpoint_url | Character string or NULL. A custom Inference Endpoint URL. |
| ... | Additional arguments (currently unused). |

## Value

A tibble with columns: question, answer, score, start, end.

## See Also

https://huggingface.co/docs/inference-providers/tasks/question-answering

## Examples

```
## Not run:
hf_question_answer(
  question = "Where was Obama born?",
  context = "Barack Obama was born in Honolulu, Hawaii."
)

# One context, several questions
hf_question_answer(
  question = c("Who?", "Where?"),
  context = "Ada Lovelace worked in London."
)

## End(Not run)
```

---

# hf_question_answering_payload

*Question Answering Payload*

---

## Description

Want to have a nice know-it-all bot that can answer any question?

## Usage

```
hf_question_answering_payload(question, context)
```

## Arguments

| | |
|---|---|
| question | a question to be answered based on the provided context |
| context | the context to consult for answering the question |

## Value

An inference payload

## See Also

https://huggingface.co/docs/api-inference/detailed_parameters#question-answering-task

### Examples

```
hf_question_answering_payload(
  question = "What is R?",
  context = "R is a language for statistical computing."
)
```

---

## hf_read_chunks *Read All Chunks from Directory*

---

### Description

Read and combine all parquet chunk files from a directory.

### Usage

```
hf_read_chunks(output_dir, pattern = "*.parquet")
```

### Arguments

| | |
|---|---|
| output_dir | Character string. Directory containing chunk files. |
| pattern | Character string. Glob pattern to match files. Default: "*.parquet". |

### Value

A tibble combining all chunks, sorted by '.input_idx' if present.

### Examples

```
## Not run:
# After running hf_embed_chunks()
results <- hf_read_chunks("my_output_dir")

## End(Not run)
```

---

## hf_run_tools *Run Tool Calls in a Conversation*

---

### Description

Execute tool calls requested by the last assistant message in an hf_conversation, append tool-result messages, and ask the model for the next response. The loop repeats until the model returns a response without tool calls.

### Usage

```
hf_run_tools(conversation, tools, max_turns = 5L, token = NULL, ...)
```

### Arguments

| | |
|---|---|
| conversation | An hf_conversation object. |
| tools | Named list of R functions, keyed by tool name. |
| max_turns | Integer. Maximum tool-execution/model-response iterations. |
| token | Character string or NULL. API token for authentication. |
| ... | Additional parameters passed to the chat-completions request. |

### Value

Updated hf_conversation object.

### Examples

```
## Not run:
tool <- hf_tool("add", "Add two numbers.", c(x = "number", y = "number"))
convo <- hf_conversation(model = "Qwen/Qwen2.5-72B-Instruct")
convo <- chat(convo, "What is 2 + 3?", tools = list(tool))
convo <- hf_run_tools(convo, list(add = function(x, y) x + y))

## End(Not run)
```

---

## hf_search_datasets *Search Datasets on Hugging Face Hub*

---

### Description

Search for datasets using various filters.

### Usage

```
hf_search_datasets(
  search = NULL,
  task = NULL,
  language = NULL,
  size = NULL,
  sort = "downloads",
  limit = 30,
  token = NULL
)
```

### Arguments

| | |
|---|---|
| search | Character string. Search query to filter datasets. |
| task | Character string. Filter by task. |
| language | Character string. Filter by language. |

| | |
|---|---|
| size | Character string. Filter by size: "small", "medium", "large". |
| sort | Character string. Sort by: "downloads", "likes", "created", "updated". Default: "downloads". |
| limit | Integer. Maximum number of datasets to return. Default: 30. |
| token | Character string or NULL. API token for authentication. |

## Value

A tibble with dataset information.

## Examples

```
## Not run:
# Search datasets
hf_search_datasets(search = "sentiment", limit = 10)

## End(Not run)
```

---

# hf_search_models *Search Models on Hugging Face Hub*

---

## Description

Search for models using various filters. Returns a tibble of matching models.

## Usage

```
hf_search_models(
  search = NULL,
  task = NULL,
  author = NULL,
  language = NULL,
  library = NULL,
  tags = NULL,
  sort = "downloads",
  direction = "desc",
  limit = 30,
  token = NULL
)
```

## Arguments

| | |
|---|---|
| search | Character string. Search query to filter models. |
| task | Character string. Filter by task (e.g., "text-classification"). |
| author | Character string. Filter by model author/organization. |
| language | Character string. Filter by language (e.g., "en"). |

| | |
|---|---|
| library | Character string. Filter by library (e.g., "pytorch", "transformers"). |
| tags | Character vector. Filter by tags. |
| sort | Character string. Sort by field: "downloads", "likes", "created", "updated". Default: "downloads". |
| direction | Character string. Sort direction: "asc" or "desc". Default: "desc". |
| limit | Integer. Maximum number of models to return. Default: 30. |
| token | Character string or NULL. API token for authentication. |

## Value

A tibble with model information.

## Examples

```
## Not run:
# Search by task
hf_search_models(task = "text-classification", limit = 10)

# Search by author
hf_search_models(author = "facebook", sort = "downloads")

# Search with query
hf_search_models(search = "sentiment", task = "text-classification")

## End(Not run)
```

---

# hf_search_papers *Search Papers on Hugging Face*

---

## Description

Search papers indexed by Hugging Face.

## Usage

```
hf_search_papers(search = NULL, limit = 30, token = NULL)
```

## Arguments

| | |
|---|---|
| search | Character string or NULL. Search query. |
| limit | Integer. Maximum number of papers to return. |
| token | Character string or NULL. API token for authentication. |

## Value

A tibble with paper metadata.

## Examples

```
## Not run:
hf_search_papers("transformers", limit = 10)

## End(Not run)
```

---

`hf_search_spaces` *Search Spaces on Hugging Face Hub*

---

## Description

Search hosted Spaces and return one row per result.

## Usage

```
hf_search_spaces(
  search = NULL,
  author = NULL,
  sort = "likes",
  direction = "desc",
  limit = 30,
  token = NULL
)
```

## Arguments

| | |
|---|---|
| `search` | Character string or NULL. Search query. |
| `author` | Character string or NULL. Filter by owner. |
| `sort` | Character string. Sort field passed to the Hub API. |
| `direction` | Character string. Sort direction: "asc" or "desc". |
| `limit` | Integer. Maximum number of Spaces to return. |
| `token` | Character string or NULL. API token for authentication. |

## Value

A tibble with Space metadata.

## Examples

```
## Not run:
hf_search_spaces(search = "chat", limit = 10)

## End(Not run)
```

`hf_sentence_encode` *Use a Sentence Transformers pipeline to extract document(s)/sentence(s) embedding(s)*

## Description

Use a Sentence Transformers pipeline to extract document(s)/sentence(s) embedding(s)

## Usage

```
hf_sentence_encode(
  model,
  text,
  batch_size = 64L,
  show_progress_bar = TRUE,
  tidy = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| `model` | Model object you loaded with 'hf_load_sentence_model()' |
| `text` | The text, or texts, you wish to embed/encode. |
| `batch_size` | How many texts to embed at once. |
| `show_progress_bar` | Whether to print a progress bar in the console or not. |
| `tidy` | Whether to tidy the output into a tibble or not. |
| `...` | other args sent to the model's encode method, e.g. device = device |

## Value

n-dimensional embeddings for every input 'text'

## Examples

```
## Not run:
text <- c("There are things we do know, things we don't know, and then there is quantum mechanics.")
sentence_mod <- hf_load_sentence_model("paraphrase-MiniLM-L6-v2")
embeddings <- hf_sentence_encode(model = sentence_mod, text, show_progress_bar = TRUE)

## End(Not run)
```

## hf_sentence_similarity_payload *Sentence Similarity Payload*

### Description

Calculate the semantic similarity between one text and a list of other sentences by comparing their embeddings.

### Usage

```
hf_sentence_similarity_payload(source_sentence, sentences)
```

### Arguments

| | |
|---|---|
| source_sentence | The string that you wish to compare the other strings with. This can be a phrase, sentence, or longer passage, depending on the model being used. |
| sentences | A list of strings which will be compared against the source_sentence. |

### Value

An inference payload

### See Also

https://huggingface.co/docs/api-inference/detailed_parameters#sentence-similarity-task

### Examples

```
hf_sentence_similarity_payload(
  source_sentence = "A happy person",
  sentences = list("A joyful person", "A rainy day")
)
```

## hf_set_device *Try to set device to GPU for accelerated computation*

### Description

This function currently depends on having a working installation of torch for your GPU in this environment. If running an Apple silicon GPU, you'll need the native mac M+ build (ARM binary). You will also need rust and other transformers dependencies. As you need to make sure that everything that needs to be on the GPU (tensors, model, pipeline etc.), is on the GPU, we currently recommend this for advanced users only. We will be working on integrating this fully with the installation and build of the huggingfaceR environment.

## Usage

```
hf_set_device()
```

## Value

a device that models, pipelines, and tensors can be sent to.

## Examples

```
## Not run:
device <- hf_set_device()

## End(Not run)
```

---

`hf_set_token` *Set Hugging Face API Token*

---

## Description

Set or update your Hugging Face API token for authentication. See https://huggingface.co/docs/hub/security-tokens for token setup.

## Usage

```
hf_set_token(token = NULL, store = FALSE)
```

## Arguments

| | |
|---|---|
| `token` | Character string containing your HF token, or NULL to set interactively. If NULL, will prompt for token input (not echoed to console). |
| `store` | Logical. If TRUE, stores the token in .Renviron for future sessions. Default: FALSE (token only available for current session). |

## Value

Invisibly returns TRUE if token was set successfully.

## Examples

```
## Not run:
# Set token for current session only
hf_set_token("hf_xxxxxxxxxxxx")

# Set token interactively and store permanently
hf_set_token(store = TRUE)

## End(Not run)
```

## hf_similarity *Compute Pairwise Similarity*

### Description

Compute cosine similarity between all pairs of embeddings. Numeric embeddings with consistent dimensions use vectorized matrix operations for better performance on larger result sets; invalid, zero-length, zero-norm, or dimension-mismatched pairs return 'NA_real_'.

### Usage

```
hf_similarity(embeddings, text_col = "text")
```

### Arguments

| | |
|---|---|
| embeddings | A tibble with an 'embedding' column (from hf_embed). |
| text_col | Character string. Name of the text column. Default: "text". |

### Value

A tibble with columns: text_1, text_2, similarity.

### Examples

```
## Not run:
sentences <- c("I love cats", "I adore felines", "Dogs are great")
embeddings <- hf_embed(sentences)
similarities <- hf_similarity(embeddings)

## End(Not run)
```

## hf_summarization_payload *Summarization Task Payload*

### Description

This task is well known to summarize longer text into shorter text. Be careful, some models have a maximum length of input. That means that the summary cannot handle full books for instance.

## Usage

```
hf_summarization_payload(
  string,
  min_length = NULL,
  max_length = NULL,
  top_k = NULL,
  top_p = NULL,
  temperature = 1,
  repetition_penalty = NULL,
  max_time = NULL
)
```

## Arguments

| | |
|---|---|
| string | a string to be summarized |
| min_length | Integer to define the minimum length in tokens of the output summary. Default: NULL |
| max_length | Integer to define the maximum length in tokens of the output summary. Default: NULL |
| top_k | Integer to define the top tokens considered within the sample operation to create new text. Default: NULL |
| top_p | Float to define the tokens that are within the sample operation of text generation. Add tokens in the sample for more probable to least probable until the sum of the probabilities is greater than top_p. Default: NULL |
| temperature | Float (0.0-100.0). The temperature of the sampling operation. 1 means regular sampling, 0 means always take the highest score, 100.0 is getting closer to uniform probability. Default: 1.0 |
| repetition_penalty | Float (0.0-100.0). The more a token is used within generation the more it is penalized to not be picked in successive generation passes. Default: NULL |
| max_time | Float (0-120.0). The amount of time in seconds that the query should take maximum. Network can cause some overhead so it will be a soft limit. Default: NULL |

## Value

An inference payload

## See Also

https://huggingface.co/docs/api-inference/detailed_parameters#summarization-task

## Examples

```
hf_summarization_payload("R is a language for statistical computing.",
                         max_length = 20)
```

`hf_summarize` *Summarize Text*

## Description

Condense longer text into a shorter summary using a summarization model via the Hugging Face Inference Providers API. Accepts a character vector and returns one row per input, composing naturally with dplyr pipelines.

## Usage

```
hf_summarize(
  text,
  model = hf_default_model("summarize"),
  min_length = NULL,
  max_length = NULL,
  token = NULL,
  endpoint_url = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| `text` | Character vector of text(s) to summarize. |
| `model` | Character string. Model ID from the Hugging Face Hub. Append ‘":provider"‘ to select an inference provider. Default: "facebook/bart-large-cnn". |
| `min_length` | Integer or NULL. Minimum length of the summary in tokens. Default: NULL (model default). |
| `max_length` | Integer or NULL. Maximum length of the summary in tokens. Default: NULL (model default). |
| `token` | Character string or NULL. API token for authentication. |
| `endpoint_url` | Character string or NULL. A custom Inference Endpoint URL. When provided, requests are sent to this URL instead of the public Inference API. |
| `...` | Additional arguments (currently unused). |

## Value

A tibble with columns: text, summary.

## See Also

https://huggingface.co/docs/inference-providers/tasks/summarization

## Examples

```
## Not run:
hf_summarize("Long article text goes here ...", max_length = 60)

library(dplyr)
articles |>
  mutate(tldr = hf_summarize(body)$summary)

## End(Not run)
```

---

`hf_table_question_answer` *Table Question Answering*

---

## Description

Ask a question in plain language about a data frame, using a table question-answering model (such as TAPAS) via the Hugging Face Inference Providers API. The data frame is converted to the string-cell format the API expects; all values are coerced to character.

## Usage

```
hf_table_question_answer(
  query,
  table,
  model = hf_default_model("table_question_answer"),
  token = NULL,
  endpoint_url = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| `query` | Character vector of question(s) to ask about the table. |
| `table` | A data frame to query. |
| `model` | Character string. Model ID from the Hugging Face Hub. Append ‘":provider"‘ to select an inference provider. Default: "google/tapas-base-finetuned-wtq". |
| `token` | Character string or NULL. API token for authentication. |
| `endpoint_url` | Character string or NULL. A custom Inference Endpoint URL. |
| `...` | Additional arguments (currently unused). |

## Value

A tibble with columns: query, answer, aggregator, cells (a list-column of the source cells the answer was drawn from).

## See Also

https://huggingface.co/docs/inference-providers/tasks/table-question-answering

## Examples

```
## Not run:
sales <- data.frame(
  product = c("Widgets", "Gadgets", "Gizmos"),
  revenue = c(120, 80, 50)
)
hf_table_question_answer("Which product had the highest revenue?", sales)
hf_table_question_answer("What is the total revenue?", sales)

## End(Not run)
```

---

# hf_table_question_answering_payload

*Table Question Answering Payload*

---

## Description

Don't know SQL? Don't want to dive into a large spreadsheet? Ask questions in plain english!

## Usage

```
hf_table_question_answering_payload(query, table)
```

## Arguments

| | |
|---|---|
| query | The query in plain text that you want to ask the table |
| table | A table of data represented as a dict of list where entries are headers and the lists are all the values, all lists must have the same size. |

## Value

An inference payload

## See Also

https://huggingface.co/docs/api-inference/detailed_parameters#table-question-answering-task

## Examples

```
hf_table_question_answering_payload(
  query = "How many rows are shown?",
  table = list(name = c("Alice", "Bob"), rows = c("1", "2"))
)
```

## hf_text_classification_payload

*Text Classification Payload*

### Description

Usually used for sentiment-analysis this will output the likelihood of classes of an input.

### Usage

```
hf_text_classification_payload(string)
```

### Arguments

| | |
|---|---|
| string | a string to be classified |

### Value

An inference payload

### See Also

https://huggingface.co/docs/api-inference/detailed_parameters#text-classification-task

### Examples

```
hf_text_classification_payload("I love using R.")
```

## hf_text_generation_payload

*Text Generation Payload*

### Description

Use to continue text from a prompt. This is a very generic task.

### Usage

```
hf_text_generation_payload(
  string,
  top_k = NULL,
  top_p = NULL,
  temperature = 1,
  repetition_penalty = NULL,
  max_new_tokens = NULL,
  max_time = NULL,
```

```
  return_full_text = TRUE,
  num_return_sequences = 1L,
  do_sample = TRUE
)
```

## Arguments

| | |
|---|---|
| string | a string to be generated from |
| top_k | (Default: None). Integer to define the top tokens considered within the sample operation to create new text. |
| top_p | (Default: None). Float to define the tokens that are within the sample operation of text generation. Add tokens in the sample for more probable to least probable until the sum of the probabilities is greater than top_p |
| temperature | Float (0.0-100.0). The temperature of the sampling operation. 1 means regular sampling, 0 means always take the highest score, 100.0 is getting closer to uniform probability. Default: 1.0 |
| repetition_penalty | (Default: None). Float (0.0-100.0). The more a token is used within generation the more it is penalized to not be picked in successive generation passes. |
| max_new_tokens | (Default: None). Int (0-250). The amount of new tokens to be generated, this does not include the input length it is a estimate of the size of generated text you want. Each new tokens slows down the request, so look for balance between response times and length of text generated. |
| max_time | (Default: None). Float (0-120.0). The amount of time in seconds that the query should take maximum. Network can cause some overhead so it will be a soft limit. Use that in combination with max_new_tokens for best results. |
| return_full_text | (Default: True). Bool. If set to False, the return results will not contain the original query making it easier for prompting. |
| num_return_sequences | (Default: 1). Integer. The number of proposition you want to be returned. |
| do_sample | (Optional: True). Bool. Whether or not to use sampling, use greedy decoding otherwise. |

## Value

An inference payload

## See Also

https://huggingface.co/docs/api-inference/detailed_parameters#text-generation-task

## Examples

```
hf_text_generation_payload("Once upon a time", max_new_tokens = 10)
```

`hf_text_to_image` *Generate an Image from Text*

## Description

Generate an image from a prompt and write it to disk using a text-to-image model via the Hugging Face Inference Providers API.

## Usage

```
hf_text_to_image(
  prompt,
  output = NULL,
  seed = NULL,
  model = hf_default_model("text_to_image"),
  token = NULL,
  endpoint_url = NULL,
  overwrite = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| `prompt` | Character vector of prompts. |
| `output` | Character path(s) or NULL. When NULL, files are written to temporary paths with an extension inferred from the response content type. |
| `seed` | Integer or NULL. Optional random seed for reproducibility when the provider/model supports it. |
| `model` | Character string. Model ID from Hugging Face Hub. Default: "black-forest-labs/FLUX.1-schnell". |
| `token` | Character string or NULL. API token for authentication. |
| `endpoint_url` | Character string or NULL. A custom Inference Endpoint URL. |
| `overwrite` | Logical. If TRUE, overwrite existing output files. |
| `...` | Additional generation parameters passed to the model. |

## Value

A tibble with columns: prompt, path, content_type, image.

## See Also

https://huggingface.co/docs/inference-providers/tasks/text-to-image

## Examples

```
## Not run:
img <- hf_text_to_image("a small red cube on a white background", seed = 42)
img$path

## End(Not run)
```

---

`hf_text_to_speech` *Convert Text to Speech*

---

## Description

Generate speech audio from text and write it to disk. The public 'hf-inference' provider did not expose a broadly available TTS model during verification; use this with a compatible model/provider or dedicated Inference Endpoint.

## Usage

```
hf_text_to_speech(
  text,
  output = NULL,
  model = hf_default_model("text_to_speech"),
  token = NULL,
  endpoint_url = NULL,
  overwrite = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| `text` | Character vector of text to synthesize. |
| `output` | Character path(s) or NULL. When NULL, files are written to temporary paths with an extension inferred from the response content type. |
| `model` | Character string. Model ID from Hugging Face Hub. Default: "facebook/mms-tts-eng". |
| `token` | Character string or NULL. API token for authentication. |
| `endpoint_url` | Character string or NULL. A custom Inference Endpoint URL. |
| `overwrite` | Logical. If TRUE, overwrite existing output files. |
| `...` | Additional generation parameters passed to the model. |

## Value

A tibble with columns: text, path, content_type, audio.

## See Also

https://huggingface.co/tasks/text-to-speech

## Examples

```
## Not run:
hf_text_to_speech("Hello from R.")

## End(Not run)
```

---

`hf_text2text_generation_payload` *Text2Text Generation Payload*

---

## Description

takes an input containing the sentence including the task and returns the output of the accomplished task.

## Usage

```
hf_text2text_generation_payload(string)
```

## Arguments

| | |
|---|---|
| `string` | a string containing a question or task and a sentence from which the answer is derived |

## Value

An inference payload

## See Also

https://huggingface.co/docs/api-inference/detailed_parameters#text2text-generation-task

## Examples

```
hf_text2text_generation_payload("translate English to French: Hello")
```

`hf_token_classification_payload` *Token Classification Payload*

## Description

Usually used for sentence parsing, either grammatical, or Named Entity Recognition (NER) to understand keywords contained within text.

## Usage

```
hf_token_classification_payload(string, aggregation_strategy = "simple")
```

## Arguments

| | |
|---|---|
| `string` | a string to be classified |
| `aggregation_strategy` | (Default: simple). There are several aggregation strategies.<br>none: Every token gets classified without further aggregation.<br>simple: Entities are grouped according to the default schema (B-, I- tags get merged when the tag is similar).<br>first: Same as the simple strategy except words cannot end up with different tags. Words will use the tag of the first token when there is ambiguity.<br>average: Same as the simple strategy except words cannot end up with different tags. Scores are averaged across tokens and then the maximum label is applied.<br>max: Same as the simple strategy except words cannot end up with different tags. Word entity will be the token with the maximum score. |

## Value

An inference payload

## See Also

https://huggingface.co/docs/api-inference/detailed_parameters#token-classification-task

## Examples

```
hf_token_classification_payload("My name is Sarah Jessica Parker.")
```

## hf_tool *Define a Chat Tool*

### Description

Build an OpenAI-compatible function-calling tool definition for `hf_chat()`. The `parameters` argument can be a lightweight named character vector, for example `c(city = "string")`, or a full JSON Schema object.

### Usage

```
hf_tool(
  name,
  description,
 parameters = list(type = "object", properties = list(), additionalProperties = FALSE)
)
```

### Arguments

| | |
|---|---|
| `name` | Tool/function name. Must be a non-empty character scalar. |
| `description` | Human-readable description of what the tool does. |
| `parameters` | A named character vector or JSON Schema list describing function arguments. |

### Value

A list suitable for the `tools` argument of `hf_chat()`.

### Examples

```
weather_tool <- hf_tool(
  "get_weather",
  "Get current weather for a city.",
  c(city = "string")
)
```

## hf_transcribe *Transcribe Audio*

### Description

Transcribe speech from an audio file, URL, or raw vector using automatic speech recognition via the Hugging Face Inference Providers API.

## Usage

```
hf_transcribe(
  audio,
  return_timestamps = FALSE,
  model = hf_default_model("transcribe"),
  token = NULL,
  endpoint_url = NULL,
  content_type = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| audio | Audio input: a local file path, URL, raw vector, or vector/list of paths/URLs. |
| return_timestamps | Logical or character. Use 'FALSE' for text only, 'TRUE' for chunk timestamps, or a model-supported value such as '"word"'. |
| model | Character string. Model ID from Hugging Face Hub. Default: "openai/whisper-large-v3". |
| token | Character string or NULL. API token for authentication. |
| endpoint_url | Character string or NULL. A custom Inference Endpoint URL. |
| content_type | Character string or NULL. MIME type to use for raw audio inputs. Paths and URLs are inferred when possible. |
| ... | Additional arguments (currently unused). |

## Value

A tibble with columns: audio, text, chunks.

## See Also

https://huggingface.co/docs/inference-providers/tasks/automatic-speech-recognition

## Examples

```
## Not run:
hf_transcribe("interview.flac")
hf_transcribe("interview.flac", return_timestamps = "word")

## End(Not run)
```

`hf_translate` *Translate Text*

## Description

Translate text from one language to another using a translation model via the Hugging Face Inference Providers API.

## Usage

```
hf_translate(
  text,
  model = hf_default_model("translate"),
  source = NULL,
  target = NULL,
  token = NULL,
  endpoint_url = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| `text` | Character vector of text(s) to translate. |
| `model` | Character string. Model ID from the Hugging Face Hub. Append ‘":provider"‘ to select an inference provider. Default: "Helsinki-NLP/opus-mt-en-fr" (English to French). |
| `source` | Character string or NULL. Source language code (model-specific; ignored by ‘opus-mt-*‘ language-pair models). |
| `target` | Character string or NULL. Target language code (model-specific; ignored by ‘opus-mt-*‘ language-pair models). |
| `token` | Character string or NULL. API token for authentication. |
| `endpoint_url` | Character string or NULL. A custom Inference Endpoint URL. |
| `...` | Additional arguments (currently unused). |

## Details

The default model, ‘Helsinki-NLP/opus-mt-en-fr‘, translates English to French and is chosen for easy onboarding: it is small, fast, broadly known, and encodes the translation direction in the model ID, so ‘hf_translate("Hello")‘ works with no extra arguments. To translate a different language pair, swap in another Helsinki-NLP ‘opus-mt-*‘ model (for example ‘"Helsinki-NLP/opus-mt-en-es"‘ for English to Spanish).

Translation models vary in how they expect languages to be specified. Language-pair models such as the Helsinki-NLP ‘opus-mt-*‘ family encode the direction in the model ID and ignore ‘source‘/‘target‘. Multilingual models such as NLLB (‘facebook/nllb-200-distilled-600M‘) instead require ‘source‘ and ‘target‘ to be set to FLORES-200 codes (for example "eng_Latn", "fra_Latn").

## Value

A tibble with columns: text, translation.

## See Also

https://huggingface.co/docs/inference-providers/tasks/translation

## Examples

```
## Not run:
# Simplest call: English to French with the default model
hf_translate("Hello, how are you?")

# A different language pair (English to Spanish)
hf_translate("Hello, how are you?", model = "Helsinki-NLP/opus-mt-en-es")

# Multilingual model (FLORES-200 codes)
hf_translate(
  "Hello, how are you?",
  model = "facebook/nllb-200-distilled-600M",
  source = "eng_Latn",
  target = "fra_Latn"
)

## End(Not run)
```

---

# hf_translation_payload

*Translation Payload*

---

## Description

This task is well known to translate text from one language to another

## Usage

```
hf_translation_payload(string)
```

## Arguments

| | |
|---|---|
| string | a string to be translated in the original languages |

## Value

An inference payload

## See Also

https://huggingface.co/docs/api-inference/detailed_parameters#translation-task

### Examples

```
hf_translation_payload("Hello, world.")
```

---

`hf_upload_file` *Upload a File to a Hub Repository*

---

### Description

Upload a local file into a model, dataset, or Space repository. This is a write operation and requires an API token with write scope plus ‘confirm = TRUE‘.

### Usage

```
hf_upload_file(
  path,
  repo_id,
  path_in_repo = NULL,
  repo_type = "model",
  commit_message = NULL,
  token = NULL,
  overwrite = FALSE,
  confirm = FALSE
)
```

### Arguments

| | |
|---|---|
| `path` | Local file path to upload. |
| `repo_id` | Character string. Repository ID. |
| `path_in_repo` | Character string or NULL. Destination path in the repo. Defaults to ‘basename(path)‘. |
| `repo_type` | Character string. One of "model", "dataset", or "space". |
| `commit_message` | Character string or NULL. Commit message when supported by the Hub upload endpoint. |
| `token` | Character string or NULL. API token with write scope. |
| `overwrite` | Logical. If FALSE, error when ‘path_in_repo‘ already exists. |
| `confirm` | Logical. Must be TRUE to perform the write operation. |

### Value

The parsed Hub API response, or the response path when the endpoint returns no JSON body.

### Examples

```
## Not run:
hf_upload_file("results.csv", "me/my-dataset", repo_type = "dataset", confirm = TRUE)

## End(Not run)
```

## hf_whoami *Get Current Hugging Face User Information*

### Description

Retrieve information about the currently authenticated user. Requires a valid Hugging Face token to be set.

### Usage

```
hf_whoami(token = NULL)
```

### Arguments

| | |
|---|---|
| token | Character string containing your HF token. If NULL, uses the HUGGING_FACE_HUB_TOKEN environment variable. |

### Value

A tibble with user, billing, organization, and token-scope metadata.

### Examples

```
## Not run:
# Check current user
hf_whoami()

## End(Not run)
```

## hf_zero_shot_classification_payload *Zero Shot Classification Payload*

### Description

This task is super useful to try out classification with zero code, you simply pass a sentence/paragraph and the possible labels for that sentence, and you get a result.

### Usage

```
hf_zero_shot_classification_payload(
  string,
  candidate_labels,
  multi_label = FALSE
)
```

## Arguments

| | |
|---|---|
| `string` | a string or list of strings |
| `candidate_labels` | a list of strings that are potential classes for inputs. (max 10 candidate_labels, for more, simply run multiple requests, results are going to be misleading if using too many candidate_labels anyway. If you want to keep the exact same, you can simply run multi_label=True and do the scaling on your end. ) |
| `multi_label` | (Default: false) Boolean that is set to True if classes can overlap |

## Value

An inference payload

## See Also

https://huggingface.co/docs/api-inference/detailed_parameters#zeroshot-classification-task

## Examples

```
hf_zero_shot_classification_payload(
  "I need help with my laptop.",
  candidate_labels = c("technology", "sports", "food")
)
```

---

`models_with_downloads` *Dataset of model names, tasks & download info*

---

## Description

#'

name of model #'

**mdldownloads** number of downloads #'

**task** task model performsl #'

**sha** model's secure hash algorithm #'

**private** whether model is public or private

## Usage

```
models_with_downloads
```

## Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 55524 rows and 5 columns.

step_hf_embed *Embedding Recipe Step*

## Description

Create text embeddings using a Hugging Face model as part of a tidymodels recipe. This step converts text columns into embedding features for downstream modeling.

## Usage

```
step_hf_embed(
  recipe,
  ...,
  role = "predictor",
  trained = FALSE,
  model = hf_default_model("embed"),
  token = NULL,
  embeddings = NULL,
  skip = FALSE,
  id = recipes::rand_id("hf_embed")
)

## S3 method for class 'step_hf_embed'
tidy(x, ...)

## S3 method for class 'step_hf_embed'
tunable(x, ...)
```

## Arguments

| | |
|---|---|
| recipe | A recipe object. |
| ... | One or more text column selectors (see recipes::selections()). |
| role | Character string. Role for the new embedding variables. Default: "predictor". |
| trained | Logical. Internal use only. |
| model | Character string. Hugging Face model ID for embeddings. Default: "BAAI/bge-small-en-v1.5". |
| token | Character string or NULL. API token for authentication. |
| embeddings | List. Internal use only (stores embeddings during training). |
| skip | Logical. Should step be skipped when baking? Default: FALSE. |
| id | Character string. Unique ID for this step. |
| x | A step_hf_embed object |

## Value

An updated recipe object.

## Examples

```
## Not run:
library(tidymodels)
library(dplyr)

# Create a recipe with embeddings
rec <- recipe(sentiment ~ text, data = train_data) |>
  step_hf_embed(text, model = "BAAI/bge-small-en-v1.5")

# Use in a workflow
wf <- workflow() |>
  add_recipe(rec) |>
  add_model(logistic_reg()) |>
  fit(data = train_data)

## End(Not run)
```

# Index