Package 'FMAT'

April 8, 2025

Title The Fill-Mask Association Test

Version 2025.4

Date 2025-04-08

Maintainer Han-Wu-Shuang Bao <baohws@foxmail.com>

Description The Fill-Mask Association Test ('FMAT')
<doi:10.1037/pspa0000396>
is an integrative and probability-based method using
Masked Language Models to measure conceptual associations
(e.g., attitudes, biases, stereotypes, social norms, cultural values)
as propositions in natural language.
Supported language models include 'BERT'
<doi:10.48550/arXiv.1810.04805> and its variants available at 'Hugging Face'
<https://huggingface.co/models?pipeline_tag=fill-mask>.
Methodological references and installation guidance are provided at
<https://psychbruce.github.io/FMAT/>.

License GPL-3

Encoding UTF-8

URL https://psychbruce.github.io/FMAT/

BugReports https://github.com/psychbruce/FMAT/issues

SystemRequirements Python (>= 3.9.0)

Depends R (>= 4.0.0)

Imports reticulate, data.table, stringr, forcats, rvest, psych, irr,
glue, crayon, cli, purrr, plyr, dplyr, tidyr

Suggests bruceR, PsychWordVec, text, sweater, nlme

RoxygenNote 7.3.2

NeedsCompilation no

Author Han-Wu-Shuang Bao [aut, cre] (<https://orcid.org/0000-0003-3043-710X>)

Repository CRAN

Date/Publication 2025-04-08 06:50:09 UTC

Contents

. *A simple function equivalent to* `list`.

Description

A simple function equivalent to `list`.

Usage

```
.(...)
```

Arguments

...	Named objects (usually character vectors for this package).

Value

A list of named objects.

Examples

```
.(Male=c("he", "his"), Female=c("she", "her"))
```

BERT_download *Download and save BERT models to local cache folder.*

Description

Download and save BERT models to local cache folder "%USERPROFILE%/.cache/huggingface".

Usage

```
BERT_download(models = NULL, verbose = FALSE)
```

Arguments

`models` A character vector of model names at HuggingFace.

`verbose` Alert if a model has been downloaded. Defaults to `FALSE`.

Value

Invisibly return a data.table of basic file information of local models.

See Also

`set_cache_folder`

`BERT_info`

`BERT_vocab`

Examples

```
## Not run:
models = c("bert-base-uncased", "bert-base-cased")
BERT_download(models)

BERT_download()  # check downloaded models

BERT_info()  # information of all downloaded models

## End(Not run)
```

BERT_info *Get basic information of BERT models.*

Description

Get basic information of BERT models.

Usage

```
BERT_info(models = NULL)
```

Arguments

`models`	A character vector of model names at HuggingFace.

Value

A data.table:

- model name
- model type
- number of parameters
- vocabulary size (of input token embeddings)
- embedding dimensions (of input token embeddings)
- hidden layers
- attention heads
- [MASK] token

See Also

BERT_download

BERT_vocab

Examples

```
## Not run:
models = c("bert-base-uncased", "bert-base-cased")
BERT_info(models)

BERT_info()  # information of all downloaded models
# speed: ~1.2s/model for first use; <1s afterwards

## End(Not run)
```

BERT_info_date — *Scrape the initial commit date of BERT models.*

Description

Scrape the initial commit date of BERT models.

Usage

```
BERT_info_date(models = NULL)
```

Arguments

`models`	A character vector of model names at HuggingFace.

Value

A data.table:

- model name
- initial commit date (scraped from huggingface commit history)

Examples

```
## Not run:
model.date = BERT_info_date()
# get all models from cache folder

one.model.date = FMAT:::get_model_date("bert-base-uncased")
# call the internal function to scrape a model
# that may not have been saved in cache folder

## End(Not run)
```

BERT_remove — *Remove BERT models from local cache folder.*

Description

Remove BERT models from local cache folder.

Usage

```
BERT_remove(models)
```

Arguments

`models`	Model names.

Value

`NULL`.

BERT_vocab *Check if mask words are in the model vocabulary.*

Description

Check if mask words are in the model vocabulary.

Usage

```
BERT_vocab(
  models,
  mask.words,
  add.tokens = FALSE,
  add.method = c("sum", "mean"),
  add.verbose = TRUE
)
```

Arguments

`models`	A character vector of model names at HuggingFace.
`mask.words`	Option words filling in the mask.
`add.tokens`	Add new tokens (for out-of-vocabulary words or phrases) to model vocabulary? Defaults to `FALSE`. It only temporarily adds tokens for tasks but does not change the raw model file.
`add.method`	Method used to produce the token embeddings of newly added tokens. Can be `"sum"` (default) or `"mean"` of subword token embeddings.
`add.verbose`	Print composition information of new tokens (for out-of-vocabulary words or phrases)? Defaults to `TRUE`.

Value

A data.table of model name, mask word, real token (replaced if out of vocabulary), and token id (0~N).

See Also

`BERT_download`

`BERT_info`

`FMAT_run`

Examples

```
## Not run:
models = c("bert-base-uncased", "bert-base-cased")
BERT_info(models)

BERT_vocab(models, c("bruce", "Bruce"))

BERT_vocab(models, 2020:2025)  # some are out-of-vocabulary
BERT_vocab(models, 2020:2025, add.tokens=TRUE)  # add vocab

BERT_vocab(models,
           c("individualism", "artificial intelligence"),
           add.tokens=TRUE)

## End(Not run)
```

`fill_mask` *Run the fill-mask pipeline and check the raw results.*

Description

Normal users should use `FMAT_run()`. This function is only for technical check.

Usage

```
fill_mask(query, model, targets = NULL, topn = 5, gpu)

fill_mask_check(query, models, targets = NULL, topn = 5, gpu)
```

Arguments

`query` Query sentence with mask token.

`model, models` Model name(s).

`targets` Target words to fill in the mask. Defaults to `NULL` (return the top 5 most likely words).

`topn` Number of the most likely predictions to return. Defaults to 5.

`gpu` Use GPU (3x faster than CPU) to run the fill-mask pipeline? Defaults to missing value that will *automatically* use available GPU (if not available, then use CPU). An NVIDIA GPU device (e.g., GeForce RTX Series) is required to use GPU. See Guidance for GPU Acceleration.

Options passing to the `device` parameter in Python:

- `FALSE`: CPU (`device = -1`).
- `TRUE`: GPU (`device = 0`).
- Any other value: passing to transformers.pipeline(device=...) which defines the device (e.g., `"cpu"`, `"cuda:0"`, or a GPU device id like `1`) on which the pipeline will be allocated.

Value

A data.table of raw results.

Functions

- fill_mask(): Check performance of one model.
- fill_mask_check(): Check performance of multiple models.

Examples

```
## Not run:
query = "Paris is the [MASK] of France."
models = c("bert-base-uncased", "bert-base-cased")

d.check = fill_mask_check(query, models, topn=2)

## End(Not run)
```

FMAT_query *Prepare a data.table of queries and variables for the FMAT.*

Description

Prepare a data.table of queries and variables for the FMAT.

Usage

```
FMAT_query(
  query = "Text with [MASK], optionally with {TARGET} and/or {ATTRIB}.",
  MASK = .(),
  TARGET = .(),
  ATTRIB = .()
)
```

Arguments

query Query text (should be a character string/vector with at least one [MASK] token). Multiple queries share the same set of MASK, TARGET, and ATTRIB. For multiple queries with different MASK, TARGET, and/or ATTRIB, please use FMAT_query_bind to combine them.

MASK A named list of [MASK] target words. Must be single words in the vocabulary of a certain masked language model.

For model vocabulary, see, e.g., https://huggingface.co/bert-base-uncased/raw/main/vocab.txt

Infrequent words may be not included in a model's vocabulary, and in this case you may insert the words into the context by specifying either TARGET or ATTRIB.

`TARGET, ATTRIB` A named list of Target/Attribute words or phrases. If specified, then `query` must contain `{TARGET}` and/or `{ATTRIB}` (in all uppercase and in braces) to be replaced by the words/phrases.

Value

A data.table of queries and variables.

See Also

`FMAT_query_bind`

`FMAT_run`

Examples

```
FMAT_query("[MASK] is a nurse.", MASK = .(Male="He", Female="She"))

FMAT_query(
  c("[MASK] is {TARGET}.", "[MASK] works as {TARGET}."),
  MASK = .(Male="He", Female="She"),
  TARGET = .(Occupation=c("a doctor", "a nurse", "an artist"))
)

FMAT_query(
  "The [MASK] {ATTRIB}.",
  MASK = .(Male=c("man", "boy"),
           Female=c("woman", "girl")),
  ATTRIB = .(Masc=c("is masculine", "has a masculine personality"),
             Femi=c("is feminine", "has a feminine personality"))
)
```

`FMAT_query_bind` *Combine multiple query data.tables and renumber query ids.*

Description

Combine multiple query data.tables and renumber query ids.

Usage

```
FMAT_query_bind(...)
```

Arguments

`...` Query data.tables returned from `FMAT_query`.

Value

A data.table of queries and variables.

See Also

FMAT_query

FMAT_run

Examples

```
FMAT_query_bind(
  FMAT_query(
    "[MASK] is {TARGET}.",
    MASK = .(Male="He", Female="She"),
    TARGET = .(Occupation=c("a doctor", "a nurse", "an artist"))
  ),
  FMAT_query(
    "[MASK] occupation is {TARGET}.",
    MASK = .(Male="His", Female="Her"),
    TARGET = .(Occupation=c("doctor", "nurse", "artist"))
  )
)
```

FMAT_run *Run the fill-mask pipeline on multiple models (CPU / GPU).*

Description

Run the fill-mask pipeline on multiple models with CPU or GPU (faster but requiring an NVIDIA GPU device).

Usage

```
FMAT_run(
  models,
  data,
  gpu,
  add.tokens = FALSE,
  add.method = c("sum", "mean"),
  add.verbose = TRUE,
 pattern.special = list(uncased = "uncased|albert|electra|muhtasham", prefix.u2581 =
    "albert|xlm-roberta|xlnet", prefix.u2581.excl = "chinese", prefix.u0120 =
   "roberta|bart|deberta|bertweet-large", prefix.u0120.excl = "chinese|xlm-|kornosk/"),
  file = NULL,
  progress = TRUE,
  warning = TRUE,
  na.out = TRUE
)
```

Arguments

`models` A character vector of model names at HuggingFace.

`data` A data.table returned from `FMAT_query` or `FMAT_query_bind`.

`gpu` Use GPU (3x faster than CPU) to run the fill-mask pipeline? Defaults to missing value that will *automatically* use available GPU (if not available, then use CPU). An NVIDIA GPU device (e.g., GeForce RTX Series) is required to use GPU. See Guidance for GPU Acceleration.

Options passing to the `device` parameter in Python:

- `FALSE`: CPU (`device = -1`).
- `TRUE`: GPU (`device = 0`).
- Any other value: passing to transformers.pipeline(device=...) which defines the device (e.g., `"cpu"`, `"cuda:0"`, or a GPU device id like `1`) on which the pipeline will be allocated.

`add.tokens` Add new tokens (for out-of-vocabulary words or phrases) to model vocabulary? Defaults to `FALSE`. It only temporarily adds tokens for tasks but does not change the raw model file.

`add.method` Method used to produce the token embeddings of newly added tokens. Can be `"sum"` (default) or `"mean"` of subword token embeddings.

`add.verbose` Print composition information of new tokens (for out-of-vocabulary words or phrases)? Defaults to `TRUE`.

`pattern.special`

Regular expression patterns (matching model names) for special model cases that are uncased or require a special prefix character in certain situations.

WARNING: As the developer is not able to check all models, users are responsible for checking the models they would use and for modifying this argument if necessary.

- `prefix.u2581`: adding prefix `\u2581` for all mask words
- `prefix.u0120`: adding prefix `\u0120` for only non-starting mask words

`file` File name of `.RData` to save the returned data.

`progress` Show a progress bar? Defaults to `TRUE`.

`warning` Alert warning of out-of-vocabulary word(s)? Defaults to `TRUE`.

`na.out` Replace probabilities of out-of-vocabulary word(s) with `NA`? Defaults to `TRUE`.

Details

The function automatically adjusts for the compatibility of tokens used in certain models: (1) for uncased models (e.g., ALBERT), it turns tokens to lowercase; (2) for models that use `<mask>` rather than `[MASK]`, it automatically uses the corrected mask token; (3) for models that require a prefix to estimate whole words than subwords (e.g., ALBERT, RoBERTa), it adds a certain prefix (usually a white space; \u2581 for ALBERT and XLM-RoBERTa, \u0120 for RoBERTa and DistilRoBERTa).

Note that these changes only affect the `token` variable in the returned data, but will not affect the `M_word` variable. Thus, users may analyze data based on the unchanged `M_word` rather than the `token`.

Note also that there may be extremely trivial differences (after 5~6 significant digits) in the raw probability estimates between using CPU and GPU, but these differences would have little impact on main results.

Value

A data.table (of new class `fmat`) appending `data` with these new variables:

- `model`: model name.
- `output`: complete sentence output with unmasked token.
- `token`: actual token to be filled in the blank mask (a note "out-of-vocabulary" will be added if the original word is not found in the model vocabulary).
- `prob`: (raw) conditional probability of the unmasked token given the provided context, estimated by the masked language model.
  - It is NOT SUGGESTED to directly interpret the raw probabilities because the *contrast* between a pair of probabilities is more interpretable. See `summary.fmat`.

See Also

`set_cache_folder`

`BERT_download`

`BERT_vocab`

`FMAT_query`

`FMAT_query_bind`

`summary.fmat`

Examples

```
## Running the examples requires the models downloaded

## Not run:
models = c("bert-base-uncased", "bert-base-cased")

query1 = FMAT_query(
  c("[MASK] is {TARGET}.", "[MASK] works as {TARGET}."),
  MASK = .(Male="He", Female="She"),
  TARGET = .(Occupation=c("a doctor", "a nurse", "an artist"))
)
data1 = FMAT_run(models, query1)
summary(data1, target.pair=FALSE)

query2 = FMAT_query(
  "The [MASK] {ATTRIB}.",
  MASK = .(Male=c("man", "boy"),
           Female=c("woman", "girl")),
  ATTRIB = .(Masc=c("is masculine", "has a masculine personality"),
             Femi=c("is feminine", "has a feminine personality"))
)
```

```
data2 = FMAT_run(models, query2)
summary(data2, mask.pair=FALSE)
summary(data2)

## End(Not run)
```

`ICC_models` *Intraclass correlation coefficient (ICC) of BERT models.*

Description

Interrater agreement of log probabilities (treated as "ratings"/rows) among BERT language models (treated as "raters"/columns), with both row and column as ("two-way") random effects.

Usage

```
ICC_models(data, type = "agreement", unit = "average")
```

Arguments

`data` Raw data returned from `FMAT_run`.

`type` Interrater `"agreement"` (default) or `"consistency"`.

`unit` Reliability of `"average"` scores (default) or `"single"` scores.

Value

A data.table of ICC.

`LPR_reliability` *Reliability analysis (Cronbach's α) of LPR.*

Description

Reliability analysis (Cronbach's α) of LPR.

Usage

```
LPR_reliability(fmat, item = c("query", "T_word", "A_word"), by = NULL)
```

Arguments

`fmat` A data.table returned from `summary.fmat`.

`item` Reliability of multiple `"query"` (default), `"T_word"`, or `"A_word"`.

`by` Variable(s) to split data by. Options can be `"model"`, `"TARGET"`, `"ATTRIB"`, or any combination of them.

Value

A data.table of Cronbach's α.

set_cache_folder *Set (change) HuggingFace cache folder temporarily.*

Description

This function allows you to change the default cache directory (when it lacks disk capacity) to another path (e.g., your portable SSD) temporarily.

Keep in mind: This function takes effect only for the current R session temporarily, so you should run this each time BEFORE you use other FMAT functions in an R session.

Usage

```
set_cache_folder(path)
```

Arguments

`path`	Folder path to store HuggingFace models.

Examples

```
## Not run:
library(FMAT)
set_cache_folder("D:/huggingface_cache/")
# -> models would be saved to "D:/huggingface_cache/hub/"
# run this function each time before using FMAT functions

BERT_download()
BERT_info()

## End(Not run)
```

summary.fmat *[S3 method] Summarize the results for the FMAT.*

Description

Summarize the results of *Log Probability Ratio* (LPR), which indicates the *relative* (vs. *absolute*) association between concepts.

The LPR of just one contrast (e.g., only between a pair of attributes) may *not* be sufficient for a proper interpretation of the results, and may further require a second contrast (e.g., between a pair of targets).

Users are suggested to use linear mixed models (with the R packages `nlme` or `lme4`/`lmerTest`) to perform the formal analyses and hypothesis tests based on the LPR.

Usage

```
## S3 method for class 'fmat'
summary(
  object,
  mask.pair = TRUE,
  target.pair = TRUE,
  attrib.pair = TRUE,
  warning = TRUE,
  ...
)
```

Arguments

`object` A data.table (of new class `fmat`) returned from `FMAT_run`.

`mask.pair`, `target.pair`, `attrib.pair`
Pairwise contrast of `[MASK]`, `TARGET`, `ATTRIB`? Defaults to `TRUE`.

`warning` Alert warning of out-of-vocabulary word(s)? Defaults to `TRUE`.

`...` Other arguments (currently not used).

Value

A data.table of the summarized results with Log Probability Ratio (LPR).

See Also

`FMAT_run`

Examples

```
# see examples in `FMAT_run`
```

Index