

Package ‘aifeducation’

March 15, 2024

Type Package

Title Artificial Intelligence for Education

Version 0.3.2

Description In social and educational settings, the use of Artificial Intelligence (AI) is a challenging task. Relevant data is often only available in handwritten forms, or the use of data is restricted by privacy policies. This often leads to small data sets. Furthermore, in the educational and social sciences, data is often unbalanced in terms of frequencies. To support educators as well as educational and social researchers in using the potentials of AI for their work, this package provides a unified interface for neural nets in 'keras', 'tensorflow', and 'pytorch' to deal with natural language problems. In addition, the package ships with a shiny app, providing a graphical user interface. This allows the usage of AI for people without skills in writing python/R scripts. The tools integrate existing mathematical and statistical methods for dealing with small data sets via pseudo-labeling (e.g. Lee (2013) <https://www.researchgate.net/publication/280581078_Pseudo-Label_The_Simple_and_Efficient_Semi-Supervised_Learning_Method_for_Deep_Neural_Networks>, Cascante-Bonilla et al. (2020) <[doi:10.48550/arXiv.2001.06001](https://doi.org/10.48550/arXiv.2001.06001)>) and imbalanced data via the creation of synthetic cases (e.g. Bunkhumpornpat et al. (2012) <[doi:10.1007/s10489-011-0287-y](https://doi.org/10.1007/s10489-011-0287-y)>). Performance evaluation of AI is connected to measures from content analysis which educational and social researchers are generally more familiar with (e.g. Berding & Pargmann (2022) <[doi:10.30819/5581](https://doi.org/10.30819/5581)>, Gwet (2014) <ISBN:978-0-9708062-8-4>, Krippendorff (2019) <[doi:10.4135/9781071878781](https://doi.org/10.4135/9781071878781)>). Estimation of energy consumption and CO2 emissions during model training is done with the 'python' library 'codecarbon'. Finally, all objects created with this package allow to share trained AI models with other people.

License GPL-3

URL <https://fberding.github.io/aifeducation/>

BugReports <https://github.com/cran/aifeducation/issues>

Depends R (>= 3.5.0)

Imports abind, foreach, doParallel, iotarelr(>= 0.1.5), irr, irrCAC, methods, Rcpp (>= 1.0.10), reshape2, reticulate (>= 1.34.0), smotefamily, stringr, rlang, utils

Suggests text2vec, tidytext, topicmodels, udpipe, quanteda, quanteda.textmodels, knitr, rmarkdown, testthat (>= 3.0.0), ggplot2, shiny, shinyFiles, shinyWidgets, shinydashboard, shinyjs, fs, readtext, readxl

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

NeedsCompilation yes

Author Berding Florian [aut, cre] (<<https://orcid.org/0000-0002-3593-1695>>), Pargmann Julia [ctb], Riebenbauer Elisabeth [ctb], Rebmann Karin [ctb], Slopinski Andreas [ctb]

Maintainer Berding Florian <florian.berding@uni-hamburg.de>

Repository CRAN

Date/Publication 2024-03-15 21:40:02 UTC

R topics documented:

| | |
|---|----|
| AifeducationConfiguration | 3 |
| aifeducation_config | 4 |
| array_to_matrix | 5 |
| bow_pp_create_basic_text_rep | 6 |
| bow_pp_create_vocab_draft | 8 |
| calc_standard_classification_measures | 9 |
| check_aif_py_modules | 10 |
| combine_embeddings | 10 |
| create_bert_model | 11 |
| create_deberta_v2_model | 13 |
| create_funnel_model | 15 |
| create_longformer_model | 18 |
| create_roberta_model | 20 |
| create_synthetic_units | 22 |
| EmbeddedText | 23 |
| get_coder_metrics | 25 |
| get_n_chunks | 27 |
| get_synthetic_cases | 27 |
| install_py_modules | 28 |

| | |
|---|----|
| load_ai_model | 29 |
| matrix_to_array_c | 30 |
| save_ai_model | 31 |
| set_config_cpu_only | 32 |
| set_config_gpu_low_memory | 32 |
| set_config_os_envirom_logger | 33 |
| set_config_tf_logger | 33 |
| set_transformers_logger | 34 |
| start_aifeducation_studio | 35 |
| TextEmbeddingClassifierNeuralNet | 35 |
| TextEmbeddingModel | 45 |
| to_categorical_c | 53 |
| train_tune_bert_model | 54 |
| train_tune_deberta_v2_model | 56 |
| train_tune_funnel_model | 58 |
| train_tune_longformer_model | 61 |
| train_tune_roberta_model | 63 |
| update_aifeducation_progress_bar | 65 |
| update_aifeducation_progress_bar_epochs | 66 |
| update_aifeducation_progress_bar_steps | 67 |

| | |
|--------------|-----------|
| Index | 68 |
|--------------|-----------|

AifeducationConfiguration

R6 class for setting the global machine learning framework.

Description

R6 class for setting the global machine learning framework.

R6 class for setting the global machine learning framework.

Details

R6 class for setting the global machine learning framework to 'PyTorch' or 'tensorflow'.

Value

The function does nothing return. It is used for its side effects.

Methods

Public methods:

- [AifeducationConfiguration\\$get_framework\(\)](#)
- [AifeducationConfiguration\\$set_global_ml_backend\(\)](#)
- [AifeducationConfiguration\\$global_framework_set\(\)](#)
- [AifeducationConfiguration\\$clone\(\)](#)

Method `get_framework()`: Method for requesting the used machine learning framework.

Usage:

```
AifeducationConfiguration$get_framework()
```

Returns: Returns a string containing the used machine learning framework for [TextEmbeddingModels](#) as well as for [TextEmbeddingClassifierNeuralNet](#).

Method `set_global_ml_backend()`: Method for setting the global machine learning framework.

Usage:

```
AifeducationConfiguration$set_global_ml_backend(backend)
```

Arguments:

`backend` string Framework to use for training and inference. `backend="tensorflow"` for 'tensorflow' and `backend="pytorch"` for 'PyTorch'.

Returns: This method does nothing return. It is used for setting the global configuration of 'aifeducation'.

Method `global_framework_set()`: Method for checking if the global ml framework is set.

Usage:

```
AifeducationConfiguration$global_framework_set()
```

Returns: Return TRUE if the global machine learning framework is set. Otherwise FALSE.

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
AifeducationConfiguration$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other Installation and Configuration: [aifeducation_config](#), [check_aif_py_modules\(\)](#), [install_py_modules\(\)](#), [set_config_cpu_only\(\)](#), [set_config_gpu_low_memory\(\)](#), [set_config_os_envIRON_logger\(\)](#), [set_config_tf_logger\(\)](#), [set_transformers_logger\(\)](#)

aifeducation_config *R6 object of class AifeducationConfiguration*

Description

Object for managing setting the machine learning framework of a session.

Usage

```
aifeducation_config
```

Format

An object of class `aifeducationConfiguration` (inherits from `R6`) of length 5.

See Also

Other Installation and Configuration: [AifeducationConfiguration](#), [check_aif_py_modules\(\)](#), [install_py_modules\(\)](#), [set_config_cpu_only\(\)](#), [set_config_gpu_low_memory\(\)](#), [set_config_os_envIRON_logger](#), [set_config_tf_logger\(\)](#), [set_transformers_logger\(\)](#)

| | |
|-----------------|------------------------|
| array_to_matrix | <i>Array to matrix</i> |
|-----------------|------------------------|

Description

Function transforming an array to a matrix.

Usage

```
array_to_matrix(text_embedding)
```

Arguments

`text_embedding` array containing the text embedding. The array should be created via an object of class [TextEmbeddingModel](#).

Value

Returns a matrix which contains the cases in the rows and the columns represent the features of all sequences. The sequences are concatenated.

See Also

Other Auxiliary Functions: [calc_standard_classification_measures\(\)](#), [check_embedding_models\(\)](#), [clean_pytorch_log_transformers\(\)](#), [create_iota2_mean_object\(\)](#), [create_synthetic_units\(\)](#), [generate_id\(\)](#), [get_coder_metrics\(\)](#), [get_folds\(\)](#), [get_n_chunks\(\)](#), [get_stratified_train_test_split\(\)](#), [get_synthetic_cases\(\)](#), [get_train_test_split\(\)](#), [is.null_or_na\(\)](#), [matrix_to_array_c\(\)](#), [split_labeled_unlabeled\(\)](#), [summarize_tracked_sustainability\(\)](#), [to_categorical_c\(\)](#)

`bow_pp_create_basic_text_rep`*Prepare texts for text embeddings with a bag of word approach.*

Description

This function prepares raw texts for use with [TextEmbeddingModel](#).

Usage

```
bow_pp_create_basic_text_rep(  
  data,  
  vocab_draft,  
  remove_punct = TRUE,  
  remove_symbols = TRUE,  
  remove_numbers = TRUE,  
  remove_url = TRUE,  
  remove_separators = TRUE,  
  split_hyphens = FALSE,  
  split_tags = FALSE,  
  language_stopwords = "de",  
  use_lemmata = FALSE,  
  to_lower = FALSE,  
  min_termfreq = NULL,  
  min_docfreq = NULL,  
  max_docfreq = NULL,  
  window = 5,  
  weights = 1/(1:5),  
  trace = TRUE  
)
```

Arguments

| | |
|---------------------------------|---|
| <code>data</code> | vector containing the raw texts. |
| <code>vocab_draft</code> | Object created with bow_pp_create_vocab_draft . |
| <code>remove_punct</code> | bool TRUE if punctuation should be removed. |
| <code>remove_symbols</code> | bool TRUE if symbols should be removed. |
| <code>remove_numbers</code> | bool TRUE if numbers should be removed. |
| <code>remove_url</code> | bool TRUE if urls should be removed. |
| <code>remove_separators</code> | bool TRUE if separators should be removed. |
| <code>split_hyphens</code> | bool TRUE if hyphens should be split into several tokens. |
| <code>split_tags</code> | bool TRUE if tags should be split. |
| <code>language_stopwords</code> | string Abbreviation for the language for which stopwords should be removed. |

| | |
|--------------|--|
| use_lemmata | bool TRUE lemmas instead of original tokens should be used. |
| to_lower | bool TRUE if tokens or lemmas should be used with lower cases. |
| min_termfreq | int Minimum frequency of a token to be part of the vocabulary. |
| min_docfreq | int Minimum appearance of a token in documents to be part of the vocabulary. |
| max_docfreq | int Maximum appearance of a token in documents to be part of the vocabulary. |
| window | int size of the window for creating the feature-co-occurrence matrix. |
| weights | vector weights for the corresponding window. The vector length must be equal to the window size. |
| trace | bool TRUE if information about the progress should be printed to console. |

Value

Returns a list of class `basic_text_rep` with the following components.

- `dfm`: Document-Feature-Matrix. Rows correspond to the documents. Columns represent the number of tokens in the document.
- `fcf`: Feature-Co-Occurrence-Matrix.
- `information`: list containing information about the used vocabulary. These are:
 - `n_sentence`: Number of sentences
 - `n_document_segments`: Number of document segments/raw texts
 - `n_token_init`: Number of initial tokens
 - `n_token_final`: Number of final tokens
 - `n_lemmata`: Number of lemmas
- `configuration`: list containing information if the vocabulary was created with lower cases and if the vocabulary uses original tokens or lemmas.
- `language_model`: list containing information about the applied language model. These are:
 - `model`: the udpipe language model
 - `label`: the label of the udpipe language model
 - `upos`: the applied universal part-of-speech tags
 - `language`: the language
 - `vocab`: a data.frame with the original vocabulary

See Also

Other Preparation: [bow_pp_create_vocab_draft\(\)](#)

bow_pp_create_vocab_draft

Function for creating a first draft of a vocabulary This function creates a list of tokens which refer to specific universal part-of-speech tags (UPOS) and provides the corresponding lemmas.

Description

Function for creating a first draft of a vocabulary This function creates a list of tokens which refer to specific universal part-of-speech tags (UPOS) and provides the corresponding lemmas.

Usage

```
bow_pp_create_vocab_draft(
  path_language_model,
  data,
  upos = c("NOUN", "ADJ", "VERB"),
  label_language_model = NULL,
  language = NULL,
  chunk_size = 100,
  trace = TRUE
)
```

Arguments

| | |
|----------------------|---|
| path_language_model | string Path to a udpipe language model that should be used for tagging and lemmatization. |
| data | vector containing the raw texts. |
| upos | vector containing the universal part-of-speech tags which should be used to build the vocabulary. |
| label_language_model | string Label for the udpipe language model used. |
| language | string Name of the language (e.g., English, German) |
| chunk_size | int Number of raw texts which should be processed at once. |
| trace | bool TRUE if information about the progress should be printed to console. |

Value

list with the following components.

- vocab: data.frame containing the tokens, lemmas, tokens in lower case, and lemmas in lower case.
- ud_language_model udpipe language model that is used for tagging.
- label_language_model Label of the udpipe language model.

- language Language of the raw texts.
- upos Used universal part-of-speech tags.
- n_sentence int Estimated number of sentences in the raw texts.
- n_token int Estimated number of tokens in the raw texts.
- n_document_segments int Estimated number of document segments/raw texts.

Note

A list of possible tags can be found here: <https://universaldependencies.org/u/pos/index.html>.

A huge number of models can be found here: <https://ufal.mff.cuni.cz/udpipe/2/models>.

See Also

Other Preparation: [bow_pp_create_basic_text_rep\(\)](#)

calc_standard_classification_measures

Calculate standard classification measures

Description

Function for calculating recall, precision, and f1.

Usage

```
calc_standard_classification_measures(true_values, predicted_values)
```

Arguments

true_values factor containing the true labels/categories.

predicted_values

factor containing the predicted labels/categories.

Value

Returns a matrix which contains the cases categories in the rows and the measures (precision, recall, f1) in the columns.

See Also

Other Auxiliary Functions: [array_to_matrix\(\)](#), [check_embedding_models\(\)](#), [clean_pytorch_log_transformers\(\)](#), [create_iota2_mean_object\(\)](#), [create_synthetic_units\(\)](#), [generate_id\(\)](#), [get_coder_metrics\(\)](#), [get_folds\(\)](#), [get_n_chunks\(\)](#), [get_stratified_train_test_split\(\)](#), [get_synthetic_cases\(\)](#), [get_train_test_split\(\)](#), [is.null_or_na\(\)](#), [matrix_to_array_c\(\)](#), [split_labeled_unlabeled\(\)](#), [summarize_tracked_sustainability\(\)](#), [to_categorical_c\(\)](#)

check_aif_py_modules *Check if all necessary python modules are available*

Description

This function checks if all python modules necessary for the package aifeducation to work are available.

Usage

```
check_aif_py_modules(trace = TRUE, check = "all")
```

Arguments

| | |
|-------|--|
| trace | bool TRUE if a list with all modules and their availability should be printed to the console. |
| check | string determining the machine learning framework to check for. check="pytorch" for 'pytorch', check="tensorflow" for 'tensorflow', and check="all" for both frameworks. |

Value

The function prints a table with all relevant packages and shows which modules are available or unavailable.

If all relevant modules are available, the functions returns TRUE. In all other cases it returns FALSE

See Also

Other Installation and Configuration: [AifeducationConfiguration](#), [aifeducation_config.install_py_modules\(\)](#), [set_config_cpu_only\(\)](#), [set_config_gpu_low_memory\(\)](#), [set_config_os_envirom_logger\(\)](#), [set_config_tf_logger\(\)](#), [set_transformers_logger\(\)](#)

combine_embeddings *Combine embedded texts*

Description

Function for combining embedded texts of the same model

Usage

```
combine_embeddings(embeddings_list)
```

Arguments

embeddings_list
list of objects of class [EmbeddedText](#).

Value

Returns an object of class [EmbeddedText](#) which contains all unique cases of the input objects.

See Also

Other Text Embedding: [EmbeddedText](#), [TextEmbeddingModel](#)

| | |
|-------------------|--|
| create_bert_model | <i>Function for creating a new transformer based on BERT</i> |
|-------------------|--|

Description

This function creates a transformer configuration based on the BERT base architecture and a vocabulary based on WordPiece by using the python libraries 'transformers' and 'tokenizers'.

Usage

```
create_bert_model(  
    ml_framework = aifeducation_config$get_framework(),  
    model_dir,  
    vocab_raw_texts = NULL,  
    vocab_size = 30522,  
    vocab_do_lower_case = FALSE,  
    max_position_embeddings = 512,  
    hidden_size = 768,  
    num_hidden_layer = 12,  
    num_attention_heads = 12,  
    intermediate_size = 3072,  
    hidden_act = "gelu",  
    hidden_dropout_prob = 0.1,  
    attention_probs_dropout_prob = 0.1,  
    sustain_track = TRUE,  
    sustain_iso_code = NULL,  
    sustain_region = NULL,  
    sustain_interval = 15,  
    trace = TRUE,  
    pytorch_safetensors = TRUE  
)
```

Arguments

| | |
|------------------------------|--|
| ml_framework | string Framework to use for training and inference. ml_framework="tensorflow" for 'tensorflow' and ml_framework="pytorch" for 'pytorch'. |
| model_dir | string Path to the directory where the model should be saved. |
| vocab_raw_texts | vector containing the raw texts for creating the vocabulary. |
| vocab_size | int Size of the vocabulary. |
| vocab_do_lower_case | bool TRUE if all words/tokens should be lower case. |
| max_position_embeddings | int Number of maximal position embeddings. This parameter also determines the maximum length of a sequence which can be processed with the model. |
| hidden_size | int Number of neurons in each layer. This parameter determines the dimensionality of the resulting text embedding. |
| num_hidden_layer | int Number of hidden layers. |
| num_attention_heads | int Number of attention heads. |
| intermediate_size | int Number of neurons in the intermediate layer of the attention mechanism. |
| hidden_act | string name of the activation function. |
| hidden_dropout_prob | double Ratio of dropout. |
| attention_probs_dropout_prob | double Ratio of dropout for attention probabilities. |
| sustain_track | bool If TRUE energy consumption is tracked during training via the python library codecarbon. |
| sustain_iso_code | string ISO code (Alpha-3-Code) for the country. This variable must be set if sustainability should be tracked. A list can be found on Wikipedia: https://en.wikipedia.org/wiki/List_of_ISO_3166_country_codes . |
| sustain_region | Region within a country. Only available for USA and Canada See the documentation of codecarbon for more information. https://mlco2.github.io/codecarbon/parameters.html |
| sustain_interval | integer Interval in seconds for measuring power usage. |
| trace | bool TRUE if information about the progress should be printed to the console. |
| pytorch_safetensors | bool If TRUE a 'pytorch' model is saved in safetensors format. If FALSE or 'safetensors' not available it is saved in the standard pytorch format (.bin). Only relevant for pytorch models. |

Value

This function does not return an object. Instead the configuration and the vocabulary of the new model are saved on disk.

Note

To train the model, pass the directory of the model to the function [train_tune_bert_model](#).

This models uses a WordPiece Tokenizer like BERT and can be trained with whole word masking. Transformer library may show a warning which can be ignored.

References

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In J. Burstein, C. Doran, & T. Solorio (Eds.), Proceedings of the 2019 Conference of the North (pp. 4171–4186). Association for Computational Linguistics. doi:10.18653/v1/N191423

Hugging Face documentation https://huggingface.co/docs/transformers/model_doc/bert#transformers.TFBertForMaskedLM

See Also

Other Transformer: [create_deberta_v2_model\(\)](#), [create_funnel_model\(\)](#), [create_longformer_model\(\)](#), [create_roberta_model\(\)](#), [train_tune_bert_model\(\)](#), [train_tune_deberta_v2_model\(\)](#), [train_tune_funnel_model\(\)](#), [train_tune_longformer_model\(\)](#), [train_tune_roberta_model\(\)](#)

create_deberta_v2_model

Function for creating a new transformer based on DeBERTa-V2

Description

This function creates a transformer configuration based on the DeBERTa-V2 base architecture and a vocabulary based on SentencePiece tokenizer by using the python libraries 'transformers' and 'tokenizers'.

Usage

```
create_deberta_v2_model(
    ml_framework = aifeducation_config$get_framework(),
    model_dir,
    vocab_raw_texts = NULL,
    vocab_size = 128100,
    do_lower_case = FALSE,
    max_position_embeddings = 512,
    hidden_size = 1536,
    num_hidden_layer = 24,
    num_attention_heads = 24,
    intermediate_size = 6144,
    hidden_act = "gelu",
    hidden_dropout_prob = 0.1,
    attention_probs_dropout_prob = 0.1,
```

```

    sustain_track = TRUE,
    sustain_iso_code = NULL,
    sustain_region = NULL,
    sustain_interval = 15,
    trace = TRUE,
    pytorch_safetensors = TRUE
)

```

Arguments

`ml_framework` string Framework to use for training and inference. `ml_framework="tensorflow"` for 'tensorflow' and `ml_framework="pytorch"` for 'pytorch'.

`model_dir` string Path to the directory where the model should be saved.

`vocab_raw_texts` vector containing the raw texts for creating the vocabulary.

`vocab_size` int Size of the vocabulary.

`do_lower_case` bool If TRUE all characters are transformed to lower case.

`max_position_embeddings` int Number of maximal position embeddings. This parameter also determines the maximum length of a sequence which can be processed with the model.

`hidden_size` int Number of neurons in each layer. This parameter determines the dimensionality of the resulting text embedding.

`num_hidden_layer` int Number of hidden layers.

`num_attention_heads` int Number of attention heads.

`intermediate_size` int Number of neurons in the intermediate layer of the attention mechanism.

`hidden_act` string name of the activation function.

`hidden_dropout_prob` double Ratio of dropout.

`attention_probs_dropout_prob` double Ratio of dropout for attention probabilities.

`sustain_track` bool If TRUE energy consumption is tracked during training via the python library codecarbon.

`sustain_iso_code` string ISO code (Alpha-3-Code) for the country. This variable must be set if sustainability should be tracked. A list can be found on Wikipedia: https://en.wikipedia.org/wiki/List_of_ISO_3166_country_codes.

`sustain_region` Region within a country. Only available for USA and Canada See the documentation of codecarbon for more information. <https://mlco2.github.io/codecarbon/parameters.html>

`sustain_interval` integer Interval in seconds for measuring power usage.

`trace` bool TRUE if information about the progress should be printed to the console.

pytorch_safetensors

bool If TRUE a 'pytorch' model is saved in safetensors format. If FALSE or 'safetensors' not available it is saved in the standard pytorch format (.bin). Only relevant for pytorch models.

Value

This function does not return an object. Instead the configuration and the vocabulary of the new model are saved on disk.

Note

To train the model, pass the directory of the model to the function [train_tune_deberta_v2_model](#).

For this model a WordPiece tokenizer is created. The standard implementation of DeBERTa version 2 from HuggingFace uses a SentencePiece tokenizer. Thus, please use AutoTokenizer from the 'transformers' library to use this model.

References

He, P., Liu, X., Gao, J. & Chen, W. (2020). DeBERTa: Decoding-enhanced BERT with Disentangled Attention. [doi:10.48550/arXiv.2006.03654](https://arxiv.org/abs/2006.03654)

Hugging Face Documentation https://huggingface.co/docs/transformers/model_doc/deberta-v2#debertav2

See Also

Other Transformer: [create_bert_model\(\)](#), [create_funnel_model\(\)](#), [create_longformer_model\(\)](#), [create_roberta_model\(\)](#), [train_tune_bert_model\(\)](#), [train_tune_deberta_v2_model\(\)](#), [train_tune_funnel_model\(\)](#), [train_tune_longformer_model\(\)](#), [train_tune_roberta_model\(\)](#)

create_funnel_model *Function for creating a new transformer based on Funnel Transformer*

Description

This function creates a transformer configuration based on the Funnel Transformer base architecture and a vocabulary based on WordPiece by using the python libraries 'transformers' and 'tokenizers'.

Usage

```
create_funnel_model(
  ml_framework = aifeducation_config$get_framework(),
  model_dir,
  vocab_raw_texts = NULL,
  vocab_size = 30522,
  vocab_do_lower_case = FALSE,
  max_position_embeddings = 512,
```

```

hidden_size = 768,
target_hidden_size = 64,
block_sizes = c(4, 4, 4),
num_attention_heads = 12,
intermediate_size = 3072,
num_decoder_layers = 2,
pooling_type = "mean",
hidden_act = "gelu",
hidden_dropout_prob = 0.1,
attention_probs_dropout_prob = 0.1,
activation_dropout = 0,
sustain_track = TRUE,
sustain_iso_code = NULL,
sustain_region = NULL,
sustain_interval = 15,
trace = TRUE,
pytorch_safetensors = TRUE
)

```

Arguments

ml_framework string Framework to use for training and inference. `ml_framework="tensorflow"` for 'tensorflow' and `ml_framework="pytorch"` for 'pytorch'.

model_dir string Path to the directory where the model should be saved.

vocab_raw_texts vector containing the raw texts for creating the vocabulary.

vocab_size int Size of the vocabulary.

vocab_do_lower_case bool TRUE if all words/tokens should be lower case.

max_position_embeddings int Number of maximal position embeddings. This parameter also determines the maximum length of a sequence which can be processed with the model.

hidden_size int Initial number of neurons in each layer.

target_hidden_size int Number of neurons in the final layer. This parameter determines the dimensionality of the resulting text embedding.

block_sizes vector of int determining the number and sizes of each block.

num_attention_heads int Number of attention heads.

intermediate_size int Number of neurons in the intermediate layer of the attention mechanism.

num_decoder_layers int Number of decoding layers.

pooling_type string "mean" for pooling with mean and "max" for pooling with maximum values.

hidden_act string name of the activation function.

| | |
|------------------------------|--|
| hidden_dropout_prob | double Ratio of dropout. |
| attention_probs_dropout_prob | double Ratio of dropout for attention probabilities. |
| activation_dropout | float Dropout probability between the layers of the feed-forward blocks. |
| sustain_track | bool If TRUE energy consumption is tracked during training via the python library codecarbon. |
| sustain_iso_code | string ISO code (Alpha-3-Code) for the country. This variable must be set if sustainability should be tracked. A list can be found on Wikipedia: https://en.wikipedia.org/wiki/List_of_ISO_3166_country_codes . |
| sustain_region | Region within a country. Only available for USA and Canada See the documentation of codecarbon for more information. https://mlco2.github.io/codecarbon/parameters.html |
| sustain_interval | integer Interval in seconds for measuring power usage. |
| trace | bool TRUE if information about the progress should be printed to the console. |
| pytorch_safetensors | bool If TRUE a 'pytorch' model is saved in safetensors format. If FALSE or 'safetensors' not available it is saved in the standard pytorch format (.bin). Only relevant for pytorch models. |

Value

This function does not return an object. Instead the configuration and the vocabulary of the new model are saved on disk.

Note

The model uses a configuration with `truncate_seq=TRUE` to avoid implementation problems with tensorflow.

To train the model, pass the directory of the model to the function [train_tune_funnel_model](#).

Model is created with `separate_cls=TRUE`, `truncate_seq=TRUE`, and `pool_q_only=TRUE`.

This models uses a WordPiece Tokenizer like BERT and can be trained with whole word masking. Transformer library may show a warning which can be ignored.

References

Dai, Z., Lai, G., Yang, Y. & Le, Q. V. (2020). Funnel-Transformer: Filtering out Sequential Redundancy for Efficient Language Processing. [doi:10.48550/arXiv.2006.03236](https://arxiv.org/abs/2006.03236)

Hugging Face documentation https://huggingface.co/docs/transformers/model_doc/funnel#funnel-transformer

See Also

Other Transformer: `create_bert_model()`, `create_deberta_v2_model()`, `create_longformer_model()`, `create_roberta_model()`, `train_tune_bert_model()`, `train_tune_deberta_v2_model()`, `train_tune_funnel_model()`, `train_tune_longformer_model()`, `train_tune_roberta_model()`

create_longformer_model

Function for creating a new transformer based on Longformer

Description

This function creates a transformer configuration based on the Longformer base architecture and a vocabulary based on Byte-Pair Encoding (BPE) tokenizer by using the python libraries 'transformers' and 'tokenizers'.

Usage

```
create_longformer_model(
    ml_framework = aifeducation_config$get_framework,
    model_dir,
    vocab_raw_texts = NULL,
    vocab_size = 30522,
    add_prefix_space = FALSE,
    trim_offsets = TRUE,
    max_position_embeddings = 512,
    hidden_size = 768,
    num_hidden_layer = 12,
    num_attention_heads = 12,
    intermediate_size = 3072,
    hidden_act = "gelu",
    hidden_dropout_prob = 0.1,
    attention_probs_dropout_prob = 0.1,
    attention_window = 512,
    sustain_track = TRUE,
    sustain_iso_code = NULL,
    sustain_region = NULL,
    sustain_interval = 15,
    trace = TRUE,
    pytorch_safetensors = TRUE
)
```

Arguments

| | |
|---------------------------|--|
| <code>ml_framework</code> | string Framework to use for training and inference. <code>ml_framework="tensorflow"</code> for 'tensorflow' and <code>ml_framework="pytorch"</code> for 'pytorch'. |
| <code>model_dir</code> | string Path to the directory where the model should be saved. |

| | |
|------------------------------|--|
| vocab_raw_texts | vector containing the raw texts for creating the vocabulary. |
| vocab_size | int Size of the vocabulary. |
| add_prefix_space | bool TRUE if an additional space should be insert to the leading words. |
| trim_offsets | bool TRUE trims the whitespaces from the produced offsets. |
| max_position_embeddings | int Number of maximal position embeddings. This parameter also determines the maximum length of a sequence which can be processed with the model. |
| hidden_size | int Number of neurons in each layer. This parameter determines the dimensionality of the resulting text embedding. |
| num_hidden_layer | int Number of hidden layers. |
| num_attention_heads | int Number of attention heads. |
| intermediate_size | int Number of neurons in the intermediate layer of the attention mechanism. |
| hidden_act | string name of the activation function. |
| hidden_dropout_prob | double Ratio of dropout |
| attention_probs_dropout_prob | double Ratio of dropout for attention probabilities. |
| attention_window | int Size of the window around each token for attention mechanism in every layer. |
| sustain_track | bool If TRUE energy consumption is tracked during training via the python library codecarbon. |
| sustain_iso_code | string ISO code (Alpha-3-Code) for the country. This variable must be set if sustainability should be tracked. A list can be found on Wikipedia: https://en.wikipedia.org/wiki/List_of_ISO_3166_country_codes . |
| sustain_region | Region within a country. Only available for USA and Canada See the documentation of codecarbon for more information. https://mlco2.github.io/codecarbon/parameters.html |
| sustain_interval | integer Interval in seconds for measuring power usage. |
| trace | bool TRUE if information about the progress should be printed to the console. |
| pytorch_safetensors | bool If TRUE a 'pytorch' model is saved in safetensors format. If FALSE or 'safetensors' not available it is saved in the standard pytorch format (.bin). Only relevant for pytorch models. |

Value

This function does not return an object. Instead the configuration and the vocabulary of the new model are saved on disk.

Note

To train the model, pass the directory of the model to the function [train_tune_longformer_model](#).

References

Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The Long-Document Transformer. [doi:10.48550/arXiv.2004.05150](https://doi.org/10.48550/arXiv.2004.05150)

Hugging Face Documentation https://huggingface.co/docs/transformers/model_doc/longformer#transformers.LongformerConfig

See Also

Other Transformer: [create_bert_model\(\)](#), [create_deberta_v2_model\(\)](#), [create_funnel_model\(\)](#), [create_roberta_model\(\)](#), [train_tune_bert_model\(\)](#), [train_tune_deberta_v2_model\(\)](#), [train_tune_funnel_model\(\)](#), [train_tune_longformer_model\(\)](#), [train_tune_roberta_model\(\)](#)

`create_roberta_model` *Function for creating a new transformer based on RoBERTa*

Description

This function creates a transformer configuration based on the RoBERTa base architecture and a vocabulary based on Byte-Pair Encoding (BPE) tokenizer by using the python libraries 'transformers' and 'tokenizers'.

Usage

```
create_roberta_model(  
    ml_framework = aifeducation_config$get_framework(),  
    model_dir,  
    vocab_raw_texts = NULL,  
    vocab_size = 30522,  
    add_prefix_space = FALSE,  
    trim_offsets = TRUE,  
    max_position_embeddings = 512,  
    hidden_size = 768,  
    num_hidden_layer = 12,  
    num_attention_heads = 12,  
    intermediate_size = 3072,  
    hidden_act = "gelu",  
    hidden_dropout_prob = 0.1,  
    attention_probs_dropout_prob = 0.1,  
    sustain_track = TRUE,  
    sustain_iso_code = NULL,  
    sustain_region = NULL,  
    sustain_interval = 15,  
)
```

```

    trace = TRUE,
    pytorch_safetensors = TRUE
)

```

Arguments

ml_framework string Framework to use for training and inference. ml_framework="tensorflow" for 'tensorflow' and ml_framework="pytorch" for 'pytorch'.

model_dir string Path to the directory where the model should be saved.

vocab_raw_texts vector containing the raw texts for creating the vocabulary.

vocab_size int Size of the vocabulary.

add_prefix_space bool TRUE if an additional space should be insert to the leading words.

trim_offsets bool If TRUE post processing trims offsets to avoid including whitespaces.

max_position_embeddings int Number of maximal position embeddings. This parameter also determines the maximum length of a sequence which can be processed with the model.

hidden_size int Number of neurons in each layer. This parameter determines the dimensionality of the resulting text embedding.

num_hidden_layer int Number of hidden layers.

num_attention_heads int Number of attention heads.

intermediate_size int Number of neurons in the intermediate layer of the attention mechanism.

hidden_act string name of the activation function.

hidden_dropout_prob double Ratio of dropout.

attention_probs_dropout_prob double Ratio of dropout for attention probabilities.

sustain_track bool If TRUE energy consumption is tracked during training via the python library codecarbon.

sustain_iso_code string ISO code (Alpha-3-Code) for the country. This variable must be set if sustainability should be tracked. A list can be found on Wikipedia: https://en.wikipedia.org/wiki/List_of_ISO_3166_country_codes.

sustain_region Region within a country. Only available for USA and Canada See the documentation of codecarbon for more information. <https://mlco2.github.io/codecarbon/parameters.html>

sustain_interval integer Interval in seconds for measuring power usage.

trace bool TRUE if information about the progress should be printed to the console.

pytorch_safetensors bool If TRUE a 'pytorch' model is saved in safetensors format. If FALSE or 'safetensors' not available it is saved in the standard pytorch format (.bin). Only relevant for pytorch models.

Value

This function does not return an object. Instead the configuration and the vocabulary of the new model are saved on disk.

Note

To train the model, pass the directory of the model to the function `train_tune_roberta_model`.

References

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. doi:10.48550/arXiv.1907.11692

Hugging Face Documentation https://huggingface.co/docs/transformers/model_doc/roberta#transformers.RobertaConfig

See Also

Other Transformer: `create_bert_model()`, `create_deberta_v2_model()`, `create_funnel_model()`, `create_longformer_model()`, `train_tune_bert_model()`, `train_tune_deberta_v2_model()`, `train_tune_funnel_model()`, `train_tune_longformer_model()`, `train_tune_roberta_model()`

create_synthetic_units

Create synthetic units

Description

Function for creating synthetic cases in order to balance the data for training with `TextEmbeddingClassifierNeuralNet`. This is an auxiliary function for use with `get_synthetic_cases` to allow parallel computations.

Usage

```
create_synthetic_units(embedding, target, k, max_k, method, cat, cat_freq)
```

Arguments

| | |
|-----------|---|
| embedding | Named data.frame containing the text embeddings. In most cases this object is taken from <code>EmbeddedText\$embeddings</code> . |
| target | Named factor containing the labels/categories of the corresponding cases. |
| k | int The number of nearest neighbors during sampling process. |
| max_k | int The maximum number of nearest neighbors during sampling process. |
| method | vector containing strings of the requested methods for generating new cases. Currently "smote", "dbsmote", and "adas" from the package smotefamily are available. |

cat string The category for which new cases should be created.
 cat_freq Object of class "table" containing the absolute frequencies of every category/label.

Value

Returns a list which contains the text embeddings of the new synthetic cases as a named data.frame and their labels as a named factor.

See Also

Other Auxiliary Functions: [array_to_matrix\(\)](#), [calc_standard_classification_measures\(\)](#), [check_embedding_models\(\)](#), [clean_pytorch_log_transformers\(\)](#), [create_iota2_mean_object\(\)](#), [generate_id\(\)](#), [get_coder_metrics\(\)](#), [get_folds\(\)](#), [get_n_chunks\(\)](#), [get_stratified_train_test_split\(\)](#), [get_synthetic_cases\(\)](#), [get_train_test_split\(\)](#), [is.null_or_na\(\)](#), [matrix_to_array_c\(\)](#), [split_labeled_unlabeled\(\)](#), [summarize_tracked_sustainability\(\)](#), [to_categorical_c\(\)](#)

| | |
|--------------|----------------------|
| EmbeddedText | <i>Embedded text</i> |
|--------------|----------------------|

Description

Object of class [R6](#) which stores the text embeddings generated by an object of class [TextEmbeddingModel](#) via the method `embed()`.

Value

Returns an object of class `EmbeddedText`. These objects are used for storing and managing the text embeddings created with objects of class [TextEmbeddingModel](#). Objects of class `EmbeddedText` serve as input for classifiers of class [TextEmbeddingClassifierNeuralNet](#). The main aim of this class is to provide a structured link between embedding models and classifiers. Since objects of this class save information on the text embedding model that created the text embedding it ensures that only embedding generated with same embedding model are combined. Furthermore, the stored information allows classifiers to check if embeddings of the correct text embedding model are used for training and predicting.

Public fields

`embeddings` ('data.frame()')
 data.frame containing the text embeddings for all chunks. Documents are in the rows. Embedding dimensions are in the columns.

Methods**Public methods:**

- [EmbeddedText\\$new\(\)](#)
- [EmbeddedText\\$get_model_info\(\)](#)
- [EmbeddedText\\$get_model_label\(\)](#)

- [EmbeddedText\\$clone\(\)](#)

Method new(): Creates a new object representing text embeddings.

Usage:

```
EmbeddedText$new(
  model_name = NA,
  model_label = NA,
  model_date = NA,
  model_method = NA,
  model_version = NA,
  model_language = NA,
  param_seq_length = NA,
  param_chunks = NULL,
  param_overlap = NULL,
  param_emb_layer_min = NULL,
  param_emb_layer_max = NULL,
  param_emb_pool_type = NULL,
  param_aggregation = NULL,
  embeddings
)
```

Arguments:

`model_name` string Name of the model that generates this embedding.

`model_label` string Label of the model that generates this embedding.

`model_date` string Date when the embedding generating model was created.

`model_method` string Method of the underlying embedding model.

`model_version` string Version of the model that generated this embedding.

`model_language` string Language of the model that generated this embedding.

`param_seq_length` int Maximum number of tokens that processes the generating model for a chunk.

`param_chunks` int Maximum number of chunks which are supported by the generating model.

`param_overlap` int Number of tokens that were added at the beginning of the sequence for the next chunk by this model.

`param_emb_layer_min` int or string determining the first layer to be included in the creation of embeddings.

`param_emb_layer_max` int or string determining the last layer to be included in the creation of embeddings.

`param_emb_pool_type` string determining the method for pooling the token embeddings within each layer.

`param_aggregation` string Aggregation method of the hidden states. Deprecated. Only included for backward compatibility.

`embeddings` data.frame containing the text embeddings.

Returns: Returns an object of class [EmbeddedText](#) which stores the text embeddings produced by an objects of class [TextEmbeddingModel](#). The object serves as input for objects of class [TextEmbeddingClassifierNeuralNet](#).

Method `get_model_info()`: Method for retrieving information about the model that generated this embedding.

Usage:

```
EmbeddedText$get_model_info()
```

Returns: list contains all saved information about the underlying text embedding model.

Method `get_model_label()`: Method for retrieving the label of the model that generated this embedding.

Usage:

```
EmbeddedText$get_model_label()
```

Returns: string Label of the corresponding text embedding model

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
EmbeddedText$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other Text Embedding: [TextEmbeddingModel](#), [combine_embeddings\(\)](#)

get_coder_metrics *Calculate reliability measures based on content analysis*

Description

This function calculates different reliability measures which are based on the empirical research method of content analysis.

Usage

```
get_coder_metrics(  
  true_values = NULL,  
  predicted_values = NULL,  
  return_names_only = FALSE  
)
```

Arguments

`true_values` factor containing the true labels/categories.

`predicted_values`
 factor containing the predicted labels/categories.

`return_names_only`
 bool If TRUE returns only the names of the resulting vector. Use FALSE to request computation of the values.

Value

If `return_names_only=FALSE` returns a vector with the following reliability measures: #'

- **iota_index**: Iota Index from the Iota Reliability Concept Version 2.
- **min_iota2**: Minimal Iota from Iota Reliability Concept Version 2.
- **avg_iota2**: Average Iota from Iota Reliability Concept Version 2.
- **max_iota2**: Maximum Iota from Iota Reliability Concept Version 2.
- **min_alpha**: Minimal Alpha Reliability from Iota Reliability Concept Version 2.
- **avg_alpha**: Average Alpha Reliability from Iota Reliability Concept Version 2.
- **max_alpha**: Maximum Alpha Reliability from Iota Reliability Concept Version 2.
- **static_iota_index**: Static Iota Index from Iota Reliability Concept Version 2.
- **dynamic_iota_index**: Dynamic Iota Index Iota Reliability Concept Version 2.
- **kalpha_nominal**: Krippendorff's Alpha for nominal variables.
- **kalpha_ordinal**: Krippendorff's Alpha for ordinal variables.
- **kendall**: Kendall's coefficient of concordance W.
- **kappa2_unweighted**: Cohen's Kappa unweighted.
- **kappa2_equal_weighted**: Weighted Cohen's Kappa with equal weights.
- **kappa2_squared_weighted**: Weighted Cohen's Kappa with squared weights.
- **kappa_fleiss**: Fleiss' Kappa for multiple raters without exact estimation.
- **percentage_agreement**: Percentage Agreement.
- **balanced_accuracy**: Average accuracy within each class.
- **gwet_ac**: Gwet's AC1/AC2 agreement coefficient.

If `return_names_only=TRUE` returns only the names of the vector elements.

See Also

Other Auxiliary Functions: `array_to_matrix()`, `calc_standard_classification_measures()`, `check_embedding_models()`, `clean_pytorch_log_transformers()`, `create_iota2_mean_object()`, `create_synthetic_units()`, `generate_id()`, `get_folds()`, `get_n_chunks()`, `get_stratified_train_test_split()`, `get_synthetic_cases()`, `get_train_test_split()`, `is.null_or_na()`, `matrix_to_array_c()`, `split_labeled_unlabeled()`, `summarize_tracked_sustainability()`, `to_categorical_c()`

| | |
|--------------|---|
| get_n_chunks | <i>Get the number of chunks/sequences for each case</i> |
|--------------|---|

Description

Function for calculating the number of chunks/sequences for every case

Usage

```
get_n_chunks(text_embeddings, features, times)
```

Arguments

| | |
|-----------------|--|
| text_embeddings | data.frame containing the text embeddings. |
| features | int Number of features within each sequence. |
| times | int Number of sequences |

Value

Namedvector of integers representing the number of chunks/sequences for every case.

See Also

Other Auxiliary Functions: [array_to_matrix\(\)](#), [calc_standard_classification_measures\(\)](#), [check_embedding_models\(\)](#), [clean_pytorch_log_transformers\(\)](#), [create_iota2_mean_object\(\)](#), [create_synthetic_units\(\)](#), [generate_id\(\)](#), [get_coder_metrics\(\)](#), [get_folds\(\)](#), [get_stratified_train_test_split\(\)](#), [get_synthetic_cases\(\)](#), [get_train_test_split\(\)](#), [is.null_or_na\(\)](#), [matrix_to_array_c\(\)](#), [split_labeled_unlabeled\(\)](#), [summarize_tracked_sustainability\(\)](#), [to_categorical_c\(\)](#)

| | |
|---------------------|---|
| get_synthetic_cases | <i>Create synthetic cases for balancing training data</i> |
|---------------------|---|

Description

This function creates synthetic cases for balancing the training with an object of the class [TextEmbeddingClassifierNeuralNet](#).

Usage

```
get_synthetic_cases(
  embedding,
  times,
  features,
  target,
  method = c("smote"),
  max_k = 6
)
```

Arguments

| | |
|-----------|---|
| embedding | Named data.frame containing the text embeddings. In most cases, this object is taken from <code>EmbeddedText\$embeddings</code> . |
| times | int for the number of sequences/times. |
| features | int for the number of features within each sequence. |
| target | Named factor containing the labels of the corresponding embeddings. |
| method | vector containing strings of the requested methods for generating new cases. Currently "smote", "dbsmote", and "adas" from the package smotefamily are available. |
| max_k | int The maximum number of nearest neighbors during sampling process. |

Value

list with the following components.

- `synthetic_embeddings`: Named data.frame containing the text embeddings of the synthetic cases.
- `synthetic_targets` Named factor containing the labels of the corresponding synthetic cases.
- `n_synthetic_units` table showing the number of synthetic cases for every label/category.

See Also

Other Auxiliary Functions: `array_to_matrix()`, `calc_standard_classification_measures()`, `check_embedding_models()`, `clean_pytorch_log_transformers()`, `create_iota2_mean_object()`, `create_synthetic_units()`, `generate_id()`, `get_coder_metrics()`, `get_folds()`, `get_n_chunks()`, `get_stratified_train_test_split()`, `get_train_test_split()`, `is.null_or_na()`, `matrix_to_array_c()`, `split_labeled_unlabeled()`, `summarize_tracked_sustainability()`, `to_categorical_c()`

install_py_modules *Installing necessary python modules to an environment*

Description

Function for installing the necessary python modules

Usage

```
install_py_modules(
  envname = "aifeducation",
  install = "pytorch",
  tf_version = "<=2.15",
  pytorch_cuda_version = "12.1",
  python_version = "3.9",
  remove_first = FALSE,
  cpu_only = FALSE
)
```

Arguments

| | |
|----------------------|---|
| envname | string Name of the environment where the packages should be installed. |
| install | character determining which machine learning frameworks should be installed. install="all" for 'pytorch' and 'tensorflow'. install="pytorch" for 'pytorch', and install="tensorflow" for 'tensorflow'. |
| tf_version | string determining the desired version of 'tensorflow'. |
| pytorch_cuda_version | string determining the desired version of 'cuda' for 'PyTorch'. |
| python_version | string Python version to use. |
| remove_first | bool If TRUE removes the environment completely before recreating the environment and installing the packages. If FALSE the packages are installed in the existing environment without any prior changes. |
| cpu_only | bool TRUE installs the cpu only version of the machine learning frameworks. |

Value

Returns no values or objects. Function is used for installing the necessary python libraries in a conda environment.

See Also

Other Installation and Configuration: [AifeducationConfiguration](#), [aifeducation_config](#), [check_aif_py_modules\(\)](#), [set_config_cpu_only\(\)](#), [set_config_gpu_low_memory\(\)](#), [set_config_os_envIRON_logger\(\)](#), [set_config_tf_logger\(\)](#), [set_transformers_logger\(\)](#)

| | |
|---------------|---|
| load_ai_model | <i>Loading models created with 'aifeducation'</i> |
|---------------|---|

Description

Function for loading models created with 'aifeducation'.

Usage

```
load_ai_model(model_dir, ml_framework = aifeducation_config$get_framework())
```

Arguments

| | |
|--------------|---|
| model_dir | Path to the directory where the model is stored. |
| ml_framework | string Determines the machine learning framework for using the model. Possible are ml_framework="pytorch" for 'pytorch', ml_framework="tensorflow" for 'tensorflow', and ml_framework="auto". for using the framework used when saving the model. |

Value

Returns an object of class [TextEmbeddingClassifierNeuralNet](#) or [TextEmbeddingModel](#).

See Also

Other Saving and Loading: [save_ai_model\(\)](#)

matrix_to_array_c *Reshape matrix to array*

Description

Function written in C++ for reshaping a matrix containing sequential data into an array for use with keras.

Usage

```
matrix_to_array_c(matrix, times, features)
```

Arguments

| | |
|----------|--|
| matrix | matrix containing the sequential data. |
| times | uword Number of sequences. |
| features | uword Number of features within each sequence. |

Value

Returns an array. The first dimension corresponds to the cases, the second to the times, and the third to the features.

See Also

Other Auxiliary Functions: [array_to_matrix\(\)](#), [calc_standard_classification_measures\(\)](#), [check_embedding_models\(\)](#), [clean_pytorch_log_transformers\(\)](#), [create_iota2_mean_object\(\)](#), [create_synthetic_units\(\)](#), [generate_id\(\)](#), [get_coder_metrics\(\)](#), [get_folds\(\)](#), [get_n_chunks\(\)](#), [get_stratified_train_test_split\(\)](#), [get_synthetic_cases\(\)](#), [get_train_test_split\(\)](#), [is.null_or_na\(\)](#), [split_labeled_unlabeled\(\)](#), [summarize_tracked_sustainability\(\)](#), [to_categorical_c\(\)](#)

| | |
|---------------|--|
| save_ai_model | <i>Saving models created with 'aifeducation'</i> |
|---------------|--|

Description

Function for saving models created with 'aifeducation'.

Usage

```
save_ai_model(  
    model,  
    model_dir,  
    dir_name = NULL,  
    save_format = "default",  
    append_ID = TRUE  
)
```

Arguments

| | |
|-------------|--|
| model | Object of class TextEmbeddingClassifierNeuralNet or TextEmbeddingModel which should be saved. |
| model_dir | Path to the directory where the should model is stored. |
| dir_name | Name of the folder that will be created at model_dir. If dir_name=NULL the model's name will be used. If additionally append_ID=TRUE the models's name and ID will be used for generating a name for that directory. |
| save_format | Only relevant for TextEmbeddingClassifierNeuralNet . Format for saving the model. For 'tensorflow'/'keras' models "keras" for 'Keras v3 format', "tf" for SavedModel or "h5" for HDF5. For 'pytorch' models "safetensors" for 'safetensors' or "pt" for 'pytorch via pickle'. Use "default" for the standard format. This is keras for 'tensorflow'/'keras' models and safetensors for 'pytorch' models. |
| append_ID | bool TRUE if the ID should be appended to the model directory for saving purposes. FALSE if not. |

Value

Function does not return a value. It saves the model to disk.

No return value, called for side effects.

See Also

Other Saving and Loading: [load_ai_model\(\)](#)

set_config_cpu_only *Setting cpu only for 'tensorflow'*

Description

This functions configures 'tensorflow' to use only cpus.

Usage

```
set_config_cpu_only()
```

Value

This function does not return anything. It is used for its side effects.

Note

```
os$environ$setdefault("CUDA_VISIBLE_DEVICES", "-1")
```

See Also

Other Installation and Configuration: [AifeducationConfiguration](#), [aifeducation_config](#), [check_aif_py_modules\(\)](#), [install_py_modules\(\)](#), [set_config_gpu_low_memory\(\)](#), [set_config_os_environ_logger\(\)](#), [set_config_tf_logger\(\)](#), [set_transformers_logger\(\)](#)

set_config_gpu_low_memory
Setting gpus' memory usage

Description

This function changes the memory usage of the gpus to allow computations on machines with small memory. With this function, some computations of large models may be possible but the speed of computation decreases.

Usage

```
set_config_gpu_low_memory()
```

Value

This function does not return anything. It is used for its side effects.

Note

This function sets TF_GPU_ALLOCATOR to "cuda_malloc_async" and sets memory growth to TRUE.

See Also

Other Installation and Configuration: [AifeducationConfiguration](#), [aifeducation_config](#), [check_aif_py_modules\(\)](#), [install_py_modules\(\)](#), [set_config_cpu_only\(\)](#), [set_config_os_environ_logger\(\)](#), [set_config_tf_logger\(\)](#), [set_transformers_logger\(\)](#)

set_config_os_environ_logger

Sets the level for logging information in tensor flow.

Description

This function changes the level for logging information with 'tensorflow' via the os environment. This function must be called before importing 'tensorflow'.

Usage

```
set_config_os_environ_logger(level = "ERROR")
```

Arguments

| | |
|-------|---|
| level | string Minimal level that should be printed to console. Four levels are available: INFO, WARNING, ERROR and NONE. |
|-------|---|

Value

This function does not return anything. It is used for its side effects.

See Also

Other Installation and Configuration: [AifeducationConfiguration](#), [aifeducation_config](#), [check_aif_py_modules\(\)](#), [install_py_modules\(\)](#), [set_config_cpu_only\(\)](#), [set_config_gpu_low_memory\(\)](#), [set_config_tf_logger\(\)](#), [set_transformers_logger\(\)](#)

set_config_tf_logger *Sets the level for logging information in tensor flow.*

Description

This function changes the level for logging information with 'tensorflow'.

Usage

```
set_config_tf_logger(level = "ERROR")
```

Arguments

level string Minimal level that should be printed to console. Five levels are available: FATAL, ERROR, WARN, INFO, and DEBUG.

Value

This function does not return anything. It is used for its side effects.

See Also

Other Installation and Configuration: [AifeducationConfiguration](#), [aifeducation_config](#), [check_aif_py_modules\(\)](#), [install_py_modules\(\)](#), [set_config_cpu_only\(\)](#), [set_config_gpu_low_memory\(\)](#), [set_config_os_envIRON_logger](#), [set_transformers_logger\(\)](#)

set_transformers_logger

Sets the level for logging information of the 'transformers' library.

Description

This function changes the level for logging information of the 'transformers' library. It influences the output printed to console for creating and training transformer models as well as [TextEmbeddingModels](#).

Usage

```
set_transformers_logger(level = "ERROR")
```

Arguments

level string Minimal level that should be printed to console. Four levels are available: INFO, WARNING, ERROR and DEBUG

Value

This function does not return anything. It is used for its side effects.

See Also

Other Installation and Configuration: [AifeducationConfiguration](#), [aifeducation_config](#), [check_aif_py_modules\(\)](#), [install_py_modules\(\)](#), [set_config_cpu_only\(\)](#), [set_config_gpu_low_memory\(\)](#), [set_config_os_envIRON_logger](#), [set_config_tf_logger\(\)](#)

```
start_aifeducation_studio
```

Aifeducation Studio

Description

Functions starts a shiny app that represents Aifeducation Studio

Usage

```
start_aifeducation_studio()
```

Value

This function does nothing return. It is used to start a shiny app.

```
TextEmbeddingClassifierNeuralNet
```

Text embedding classifier with a neural net

Description

Abstract class for neural nets with 'keras'/'tensorflow' and 'pytorch'.

Value

Objects of this class are used for assigning texts to classes/categories. For the creation and training of a classifier an object of class [EmbeddedText](#) and a factor are necessary. The object of class [EmbeddedText](#) contains the numerical text representations (text embeddings) of the raw texts generated by an object of class [TextEmbeddingModel](#). The factor contains the classes/categories for every text. Missing values (unlabeled cases) are supported. For predictions an object of class [EmbeddedText](#) has to be used which was created with the same text embedding model as for training.

Public fields

```
model ('tensorflow_model()')
```

Field for storing the tensorflow model after loading.

```
model_config ('list()')
```

List for storing information about the configuration of the model. This information is used to predict new data.

- `model_config$n_rec`: Number of recurrent layers.
- `model_config$n_hidden`: Number of dense layers.
- `model_config$target_levels`: Levels of the target variable. Do not change this manually.

- `model_config$input_variables`: Order and name of the input variables. Do not change this manually.
- `model_config$init_config`: List storing all parameters passed to method `new()`.

`last_training` ('list()')

List for storing the history and the results of the last training. This information will be overwritten if a new training is started.

- `last_training$learning_time`: Duration of the training process.
- `config$history`: History of the last training.
- `config$data`: Object of class table storing the initial frequencies of the passed data.
- `config$data_pb`: 1 Matrix storing the number of additional cases (test and training) added during balanced pseudo-labeling. The rows refer to folds and final training. The columns refer to the steps during pseudo-labeling.
- `config$data_bsc_test`: Matrix storing the number of cases for each category used for testing during the phase of balanced synthetic units. Please note that the frequencies include original and synthetic cases. In case the number of original and synthetic cases exceeds the limit for the majority classes, the frequency represents the number of cases created by cluster analysis.
- `config$date`: Time when the last training finished.
- `config$config`: List storing which kind of estimation was requested during the last training.
 - `config$config$use_bsc`: TRUE if balanced synthetic cases were requested. FALSE if not.
 - `config$config$use_baseline`: TRUE if baseline estimation were requested. FALSE if not.
 - `config$config$use_bp1`: TRUE if balanced, pseudo-labeling cases were requested. FALSE if not.

`reliability` ('list()')

List for storing central reliability measures of the last training.

- `reliability$test_metric`: Array containing the reliability measures for the validation data for every fold, method, and step (in case of pseudo-labeling).
- `reliability$test_metric_mean`: Array containing the reliability measures for the validation data for every method and step (in case of pseudo-labeling). The values represent the mean values for every fold.
- `reliability$raw_iota_objects`: List containing all `iota_object` generated with the package `iotarelr` for every fold at the start and the end of the last training.
 - `reliability$raw_iota_objects$iota_objects_start`: List of objects with class `iotarelr_iota2` containing the estimated iota reliability of the second generation for the baseline model for every fold. If the estimation of the baseline model is not requested, the list is set to NULL.
 - `reliability$raw_iota_objects$iota_objects_end`: List of objects with class `iotarelr_iota2` containing the estimated iota reliability of the second generation for the final model for every fold. Depending of the requested training method these values refer to the baseline model, a trained model on the basis of balanced synthetic cases, balanced pseudo labeling or a combination of balanced synthetic cases with pseudo labeling.

- `reliability$raw_iota_objects$iota_objects_start_free`: List of objects with class `iotarelr_iota2` containing the estimated iota reliability of the second generation for the baseline model for every fold. If the estimation of the baseline model is not requested, the list is set to `NULL`. Please note that the model is estimated without forcing the Assignment Error Matrix to be in line with the assumption of weak superiority.
- `reliability$raw_iota_objects$iota_objects_end_free`: List of objects with class `iotarelr_iota2` containing the estimated iota reliability of the second generation for the final model for every fold. Depending of the requested training method, these values refer to the baseline model, a trained model on the basis of balanced synthetic cases, balanced pseudo-labeling or a combination of balanced synthetic cases and pseudo-labeling. Please note that the model is estimated without forcing the Assignment Error Matrix to be in line with the assumption of weak superiority.
- `reliability$iota_object_start`: Object of class `iotarelr_iota2` as a mean of the individual objects for every fold. If the estimation of the baseline model is not requested, the list is set to `NULL`.
- `reliability$iota_object_start_free`: Object of class `iotarelr_iota2` as a mean of the individual objects for every fold. If the estimation of the baseline model is not requested, the list is set to `NULL`. Please note that the model is estimated without forcing the Assignment Error Matrix to be in line with the assumption of weak superiority.
- `reliability$iota_object_end`: Object of class `iotarelr_iota2` as a mean of the individual objects for every fold. Depending on the requested training method, this object refers to the baseline model, a trained model on the basis of balanced synthetic cases, balanced pseudo-labeling or a combination of balanced synthetic cases and pseudo-labeling.
- `reliability$iota_object_end_free`: Object of class `iotarelr_iota2` as a mean of the individual objects for every fold. Depending on the requested training method, this object refers to the baseline model, a trained model on the basis of balanced synthetic cases, balanced pseudo-labeling or a combination of balanced synthetic cases and pseudo-labeling. Please note that the model is estimated without forcing the Assignment Error Matrix to be in line with the assumption of weak superiority.
- `reliability$standard_measures_end`: Object of class `list` containing the final measures for precision, recall, and f1 for every fold. Depending of the requested training method, these values refer to the baseline model, a trained model on the basis of balanced synthetic cases, balanced pseudo-labeling or a combination of balanced synthetic cases and pseudo-labeling.
- `reliability$standard_measures_mean`: matrix containing the mean measures for precision, recall, and f1 at the end of every fold.

Methods

Public methods:

- `TextEmbeddingClassifierNeuralNet$new()`
- `TextEmbeddingClassifierNeuralNet$train()`
- `TextEmbeddingClassifierNeuralNet$predict()`
- `TextEmbeddingClassifierNeuralNet$check_embedding_model()`
- `TextEmbeddingClassifierNeuralNet$get_model_info()`
- `TextEmbeddingClassifierNeuralNet$get_text_embedding_model()`

- `TextEmbeddingClassifierNeuralNet$set_publication_info()`
- `TextEmbeddingClassifierNeuralNet$get_publication_info()`
- `TextEmbeddingClassifierNeuralNet$set_software_license()`
- `TextEmbeddingClassifierNeuralNet$get_software_license()`
- `TextEmbeddingClassifierNeuralNet$set_documentation_license()`
- `TextEmbeddingClassifierNeuralNet$get_documentation_license()`
- `TextEmbeddingClassifierNeuralNet$set_model_description()`
- `TextEmbeddingClassifierNeuralNet$get_model_description()`
- `TextEmbeddingClassifierNeuralNet$save_model()`
- `TextEmbeddingClassifierNeuralNet$load_model()`
- `TextEmbeddingClassifierNeuralNet$get_package_versions()`
- `TextEmbeddingClassifierNeuralNet$get_sustainability_data()`
- `TextEmbeddingClassifierNeuralNet$get_ml_framework()`
- `TextEmbeddingClassifierNeuralNet$clone()`

Method `new()`: Creating a new instance of this class.

Usage:

```
TextEmbeddingClassifierNeuralNet$new(
  ml_framework = aifeducation_config$get_framework(),
  name = NULL,
  label = NULL,
  text_embeddings = NULL,
  targets = NULL,
  hidden = c(128),
  rec = c(128),
  self_attention_heads = 0,
  intermediate_size = NULL,
  attention_type = "fourier",
  add_pos_embedding = TRUE,
  rec_dropout = 0.1,
  repeat_encoder = 1,
  dense_dropout = 0.4,
  recurrent_dropout = 0.4,
  encoder_dropout = 0.1,
  optimizer = "adam"
)
```

Arguments:

`ml_framework` string Framework to use for training and inference. `ml_framework="tensorflow"` for 'tensorflow' and `ml_framework="pytorch"` for 'pytorch'

`name` Character Name of the new classifier. Please refer to common name conventions. Free text can be used with parameter `label`.

`label` Character Label for the new classifier. Here you can use free text.

`text_embeddings` An object of class `TextEmbeddingModel`.

`targets` factor containing the target values of the classifier.

hidden_vector vector containing the number of neurons for each dense layer. The length of the vector determines the number of dense layers. If you want no dense layer, set this parameter to NULL.

rec_vector vector containing the number of neurons for each recurrent layer. The length of the vector determines the number of dense layers. If you want no dense layer, set this parameter to NULL.

self_attention_heads integer determining the number of attention heads for a self-attention layer. Only relevant if attention_type="multihead"

intermediate_size int determining the size of the projection layer within a each transformer encoder.

attention_type string Choose the relevant attention type. Possible values are "fourier" and multihead.

add_pos_embedding bool TRUE if positional embedding should be used.

rec_dropout double ranging between 0 and lower 1, determining the dropout between bidirectional gru layers.

repeat_encoder int determining how many times the encoder should be added to the network.

dense_dropout double ranging between 0 and lower 1, determining the dropout between dense layers.

recurrent_dropout double ranging between 0 and lower 1, determining the recurrent dropout for each recurrent layer. Only relevant for keras models.

encoder_dropout double ranging between 0 and lower 1, determining the dropout for the dense projection within the encoder layers.

optimizer Object of class keras.optimizers.

Returns: Returns an object of class [TextEmbeddingClassifierNeuralNet](#) which is ready for training.

Method train(): Method for training a neural net.

Usage:

```
TextEmbeddingClassifierNeuralNet$train(
  data_embeddings,
  data_targets,
  data_n_test_samples = 5,
  balance_class_weights = TRUE,
  use_baseline = TRUE,
  bsl_val_size = 0.25,
  use_bsc = TRUE,
  bsc_methods = c("dbsmote"),
  bsc_max_k = 10,
  bsc_val_size = 0.25,
  bsc_add_all = FALSE,
  use_bpl = TRUE,
  bpl_max_steps = 3,
  bpl_epochs_per_step = 1,
  bpl_dynamic_inc = FALSE,
  bpl_balance = FALSE,
```

```

    bpl_max = 1,
    bpl_anchor = 1,
    bpl_min = 0,
    bpl_weight_inc = 0.02,
    bpl_weight_start = 0,
    bpl_model_reset = FALSE,
    sustain_track = TRUE,
    sustain_iso_code = NULL,
    sustain_region = NULL,
    sustain_interval = 15,
    epochs = 40,
    batch_size = 32,
    dir_checkpoint,
    trace = TRUE,
    keras_trace = 2,
    pytorch_trace = 2,
    n_cores = 2
)

```

Arguments:

`data_embeddings` Object of class `TextEmbeddingModel`.

`data_targets` Factor containing the labels for cases stored in `data_embeddings`. Factor must be named and has to use the same names used in `data_embeddings`.

`data_n_test_samples` int determining the number of cross-fold samples.

`balance_class_weights` bool If TRUE class weights are generated based on the frequencies of the training data with the method Inverse Class Frequency'. If FALSE each class has the weight 1.

`use_baseline` bool TRUE if the calculation of a baseline model is requested. This option is only relevant for `use_bsc=TRUE` or `use_pbl=TRUE`. If both are FALSE, a baseline model is calculated.

`bsl_val_size` double between 0 and 1, indicating the proportion of cases of each class which should be used for the validation sample during the estimation of the baseline model. The remaining cases are part of the training data.

`use_bsc` bool TRUE if the estimation should integrate balanced synthetic cases. FALSE if not.

`bsc_methods` vector containing the methods for generating synthetic cases via 'smotefamily'. Multiple methods can be passed. Currently `bsc_methods=c("adas")`, `bsc_methods=c("smote")` and `bsc_methods=c("dbsmote")` are possible.

`bsc_max_k` int determining the maximal number of k which is used for creating synthetic units.

`bsc_val_size` double between 0 and 1, indicating the proportion of cases of each class which should be used for the validation sample during the estimation with synthetic cases.

`bsc_add_all` bool If FALSE only synthetic cases necessary to fill the gab between the class and the major class are added to the data. If TRUE all generated synthetic cases are added to the data.

`use_pbl` bool TRUE if the estimation should integrate balanced pseudo-labeling. FALSE if not.

`bpl_max_steps` int determining the maximum number of steps during pseudo-labeling.

`bpl_epochs_per_step` int Number of training epochs within every step.

bpl_dynamic_inc bool If TRUE, only a specific percentage of cases is included during each step. The percentage is determined by $step/bpl_max_steps$. If FALSE, all cases are used.

bpl_balance bool If TRUE, the same number of cases for every category/class of the pseudo-labeled data are used with training. That is, the number of cases is determined by the minor class/category.

bpl_max double between 0 and 1, setting the maximal level of confidence for considering a case for pseudo-labeling.

bpl_anchor double between 0 and 1 indicating the reference point for sorting the new cases of every label. See notes for more details.

bpl_min double between 0 and 1, setting the minimal level of confidence for considering a case for pseudo-labeling.

bpl_weight_inc double value how much the sample weights should be increased for the cases with pseudo-labels in every step.

bpl_weight_start double Starting value for the weights of the unlabeled cases.

bpl_model_reset bool If TRUE, model is re-initialized at every step.

sustain_track bool If TRUE energy consumption is tracked during training via the python library codecarbon.

sustain_iso_code string ISO code (Alpha-3-Code) for the country. This variable must be set if sustainability should be tracked. A list can be found on Wikipedia: https://en.wikipedia.org/wiki/List_of_ISO_3166_country_codes.

sustain_region Region within a country. Only available for USA and Canada See the documentation of codecarbon for more information. <https://mlco2.github.io/codecarbon/parameters.html>

sustain_interval integer Interval in seconds for measuring power usage.

epochs int Number of training epochs.

batch_size int Size of batches.

dir_checkpoint string Path to the directory where the checkpoint during training should be saved. If the directory does not exist, it is created.

trace bool TRUE, if information about the estimation phase should be printed to the console.

keras_trace int `keras_trace=0` does not print any information about the training process from keras on the console.

pytorch_trace int `pytorch_trace=0` does not print any information about the training process from pytorch on the console. `pytorch_trace=1` prints a progress bar. `pytorch_trace=2` prints one line of information for every epoch.

n_cores int Number of cores used for creating synthetic units.

Details:

- **bsc_max_k**: All values from 2 up to `bsc_max_k` are successively used. If the number of `bsc_max_k` is too high, the value is reduced to a number that allows the calculating of synthetic units.
- **bpl_anchor**: With the help of this value, the new cases are sorted. For this aim, the distance from the anchor is calculated and all cases are arranged into an ascending order.

Returns: Function does not return a value. It changes the object into a trained classifier.

Method `predict()`: Method for predicting new data with a trained neural net.

Usage:

```
TextEmbeddingClassifierNeuralNet$predict(newdata, batch_size = 32, verbose = 1)
```

Arguments:

`newdata` Object of class `TextEmbeddingModel` or `data.frame` for which predictions should be made.

`batch_size` `int` Size of batches.

`verbose` `int` `verbose=0` does not cat any information about the training process from keras on the console. `verbose=1` prints a progress bar. `verbose=2` prints one line of information for every epoch.

Returns: Returns a `data.frame` containing the predictions and the probabilities of the different labels for each case.

Method `check_embedding_model()`: Method for checking if the provided text embeddings are created with the same `TextEmbeddingModel` as the classifier.

Usage:

```
TextEmbeddingClassifierNeuralNet$check_embedding_model(text_embeddings)
```

Arguments:

`text_embeddings` Object of class `EmbeddedText`.

Returns: TRUE if the underlying `TextEmbeddingModel` are the same. FALSE if the models differ.

Method `get_model_info()`: Method for requesting the model information

Usage:

```
TextEmbeddingClassifierNeuralNet$get_model_info()
```

Returns: list of all relevant model information

Method `get_text_embedding_model()`: Method for requesting the text embedding model information

Usage:

```
TextEmbeddingClassifierNeuralNet$get_text_embedding_model()
```

Returns: list of all relevant model information on the text embedding model underlying the classifier

Method `set_publication_info()`: Method for setting publication information of the classifier

Usage:

```
TextEmbeddingClassifierNeuralNet$set_publication_info(
  authors,
  citation,
  url = NULL
)
```

Arguments:

`authors` List of authors.

`citation` Free text citation.

`url` URL of a corresponding homepage.

Returns: Function does not return a value. It is used for setting the private members for publication information.

Method `get_publication_info()`: Method for requesting the bibliographic information of the classifier.

Usage:

```
TextEmbeddingClassifierNeuralNet$get_publication_info()
```

Returns: list with all saved bibliographic information.

Method `set_software_license()`: Method for setting the license of the classifier.

Usage:

```
TextEmbeddingClassifierNeuralNet$set_software_license(license = "GPL-3")
```

Arguments:

`license` string containing the abbreviation of the license or the license text.

Returns: Function does not return a value. It is used for setting the private member for the software license of the model.

Method `get_software_license()`: Method for getting the license of the classifier.

Usage:

```
TextEmbeddingClassifierNeuralNet$get_software_license()
```

Arguments:

`license` string containing the abbreviation of the license or the license text.

Returns: string representing the license for the software.

Method `set_documentation_license()`: Method for setting the license of the classifier's documentation.

Usage:

```
TextEmbeddingClassifierNeuralNet$set_documentation_license(  
  license = "CC BY-SA"  
)
```

Arguments:

`license` string containing the abbreviation of the license or the license text.

Returns: Function does not return a value. It is used for setting the private member for the documentation license of the model.

Method `get_documentation_license()`: Method for getting the license of the classifier's documentation.

Usage:

```
TextEmbeddingClassifierNeuralNet$get_documentation_license()
```

Arguments:

`license` string containing the abbreviation of the license or the license text.

Returns: Returns the license as a string.

Method `set_model_description()`: Method for setting a description of the classifier.

Usage:

```

TextEmbeddingClassifierNeuralNet$set_model_description(
  eng = NULL,
  native = NULL,
  abstract_eng = NULL,
  abstract_native = NULL,
  keywords_eng = NULL,
  keywords_native = NULL
)

```

Arguments:

`eng` string A text describing the training of the learner, its theoretical and empirical background, and the different output labels in English.

`native` string A text describing the training of the learner, its theoretical and empirical background, and the different output labels in the native language of the classifier.

`abstract_eng` string A text providing a summary of the description in English.

`abstract_native` string A text providing a summary of the description in the native language of the classifier.

`keywords_eng` vector of keyword in English.

`keywords_native` vector of keyword in the native language of the classifier.

Returns: Function does not return a value. It is used for setting the private members for the description of the model.

Method `get_model_description()`: Method for requesting the model description.

Usage:

```
TextEmbeddingClassifierNeuralNet$get_model_description()
```

Returns: list with the description of the classifier in English and the native language.

Method `save_model()`: Method for saving a model to 'Keras v3 format', 'tensorflow' SavedModel format or h5 format.

Usage:

```
TextEmbeddingClassifierNeuralNet$save_model(dir_path, save_format = "default")
```

Arguments:

`dir_path` string() Path of the directory where the model should be saved.

`save_format` Format for saving the model. For 'tensorflow'/'keras' models "keras" for 'Keras v3 format', "tf" for SavedModel or "h5" for HDF5. For 'pytorch' models "safetensors" for 'safetensors' or "pt" for 'pytorch' via pickle. Use "default" for the standard format. This is keras for 'tensorflow'/'keras' models and safetensors for 'pytorch' models.

Returns: Function does not return a value. It saves the model to disk.

Method `load_model()`: Method for importing a model from 'Keras v3 format', 'tensorflow' SavedModel format or h5 format.

Usage:

```
TextEmbeddingClassifierNeuralNet$load_model(dir_path, ml_framework = "auto")
```

Arguments:

`dir_path` string() Path of the directory where the model is saved.

`ml_framework` string Determines the machine learning framework for using the model. Possible are `ml_framework="pytorch"` for 'pytorch', `ml_framework="tensorflow"` for 'tensorflow', and `ml_framework="auto"`.

Returns: Function does not return a value. It is used to load the weights of a model.

Method `get_package_versions()`: Method for requesting a summary of the R and python packages' versions used for creating the classifier.

Usage:

```
TextEmbeddingClassifierNeuralNet$get_package_versions()
```

Returns: Returns a list containing the versions of the relevant R and python packages.

Method `get_sustainability_data()`: Method for requesting a summary of tracked energy consumption during training and an estimate of the resulting CO2 equivalents in kg.

Usage:

```
TextEmbeddingClassifierNeuralNet$get_sustainability_data()
```

Returns: Returns a list containing the tracked energy consumption, CO2 equivalents in kg, information on the tracker used, and technical information on the training infrastructure.

Method `get_ml_framework()`: Method for requesting the machine learning framework used for the classifier.

Usage:

```
TextEmbeddingClassifierNeuralNet$get_ml_framework()
```

Returns: Returns a string describing the machine learning framework used for the classifier

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
TextEmbeddingClassifierNeuralNet$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

TextEmbeddingModel *Text embedding model*

Description

This [R6](#) class stores a text embedding model which can be used to tokenize, encode, decode, and embed raw texts. The object provides a unique interface for different text processing methods.

Value

Objects of class `TextEmbeddingModel` transform raw texts into numerical representations which can be used for downstream tasks. For this aim objects of this class allow to tokenize raw texts, to encode tokens to sequences of integers, and to decode sequences of integers back to tokens.

Public fields

`last_training` ('list()')

List for storing the history and the results of the last training. This information will be overwritten if a new training is started.

Methods**Public methods:**

- `TextEmbeddingModel$new()`
- `TextEmbeddingModel$load_model()`
- `TextEmbeddingModel$save_model()`
- `TextEmbeddingModel$encode()`
- `TextEmbeddingModel$decode()`
- `TextEmbeddingModel$get_special_tokens()`
- `TextEmbeddingModel$embed()`
- `TextEmbeddingModel$fill_mask()`
- `TextEmbeddingModel$set_publication_info()`
- `TextEmbeddingModel$get_publication_info()`
- `TextEmbeddingModel$set_software_license()`
- `TextEmbeddingModel$get_software_license()`
- `TextEmbeddingModel$set_documentation_license()`
- `TextEmbeddingModel$get_documentation_license()`
- `TextEmbeddingModel$set_model_description()`
- `TextEmbeddingModel$get_model_description()`
- `TextEmbeddingModel$get_model_info()`
- `TextEmbeddingModel$get_package_versions()`
- `TextEmbeddingModel$get_basic_components()`
- `TextEmbeddingModel$get_bow_components()`
- `TextEmbeddingModel$get_transformer_components()`
- `TextEmbeddingModel$get_sustainability_data()`
- `TextEmbeddingModel$get_ml_framework()`
- `TextEmbeddingModel$clone()`

Method `new()`: Method for creating a new text embedding model

Usage:

```
TextEmbeddingModel$new(
  model_name = NULL,
  model_label = NULL,
  model_version = NULL,
  model_language = NULL,
  method = NULL,
  ml_framework = aifeducation_config$get_framework()$TextEmbeddingFramework,
  max_length = 0,
  chunks = 1,
```

```

overlap = 0,
emb_layer_min = "middle",
emb_layer_max = "2_3_layer",
emb_pool_type = "average",
model_dir,
bow_basic_text_rep,
bow_n_dim = 10,
bow_n_cluster = 100,
bow_max_iter = 500,
bow_max_iter_cluster = 500,
bow_cr_criterion = 1e-08,
bow_learning_rate = 1e-08,
trace = FALSE
)

```

Arguments:

`model_name` string containing the name of the new model.

`model_label` string containing the label/title of the new model.

`model_version` string version of the model.

`model_language` string containing the language which the model represents (e.g., English).

`method` string determining the kind of embedding model. Currently the following models are supported: `method="bert"` for Bidirectional Encoder Representations from Transformers (BERT), `method="roberta"` for A Robustly Optimized BERT Pretraining Approach (RoBERTa), `method="longformer"` for Long-Document Transformer, `method="funnel"` for Funnel-Transformer, `method="deberta_v2"` for Decoding-enhanced BERT with Disentangled Attention (DeBERTa V2), `method="glove"` for GlobalVector Clusters, and `method="lda"` for topic modeling. See details for more information.

`ml_framework` string Framework to use for the model. `ml_framework="tensorflow"` for 'tensorflow' and `ml_framework="pytorch"` for 'pytorch'. Only relevant for transformer models.

`max_length` int determining the maximum length of token sequences used in transformer models. Not relevant for the other methods.

`chunks` int Maximum number of chunks. Only relevant for transformer models.

`overlap` int determining the number of tokens which should be added at the beginning of the next chunk. Only relevant for BERT models.

`emb_layer_min` int or string determining the first layer to be included in the creation of embeddings. An integer corresponds to the layer number. The first layer has the number 1. Instead of an integer the following strings are possible: "start" for the first layer, "middle" for the middle layer, "2_3_layer" for the layer two-third layer, and "last" for the last layer.

`emb_layer_max` int or string determining the last layer to be included in the creation of embeddings. An integer corresponds to the layer number. The first layer has the number 1. Instead of an integer the following strings are possible: "start" for the first layer, "middle" for the middle layer, "2_3_layer" for the layer two-third layer, and "last" for the last layer.

`emb_pool_type` string determining the method for pooling the token embeddings within each layer. If "cls" only the embedding of the CLS token is used. If "average" the token embedding of all tokens are averaged (excluding padding tokens).

model_dir string path to the directory where the BERT model is stored.
bow_basic_text_rep object of class `basic_text_rep` created via the function `bow_pp_create_basic_text_rep`.
 Only relevant for `method="glove_cluster"` and `method="lda"`.
bow_n_dim int Number of dimensions of the GlobalVector or number of topics for LDA.
bow_n_cluster int Number of clusters created on the basis of GlobalVectors. Parameter is not relevant for `method="lda"` and `method="bert"`
bow_max_iter int Maximum number of iterations for fitting GlobalVectors and Topic Models.
bow_max_iter_cluster int Maximum number of iterations for fitting cluster if `method="glove"`.
bow_cr_criterion double convergence criterion for GlobalVectors.
bow_learning_rate double initial learning rate for GlobalVectors.
trace bool TRUE prints information about the progress. FALSE does not.

Details:

- `method`: In the case of `method="bert"`, `method="roberta"`, and `method="longformer"`, a pretrained transformer model must be supplied via `model_dir`. For `method="glove"` and `method="lda"` a new model will be created based on the data provided via `bow_basic_text_rep`. The original algorithm for GlobalVectors provides only word embeddings, not text embeddings. To achieve text embeddings the words are clustered based on their word embeddings with `kmeans`.

Returns: Returns an object of class `TextEmbeddingModel`.

Method `load_model()`: Method for loading a transformers model into R.

Usage:

```
TextEmbeddingModel$load_model(model_dir, ml_framework = "auto")
```

Arguments:

model_dir string containing the path to the relevant model directory.
ml_framework string Determines the machine learning framework for using the model. Possible are `ml_framework="pytorch"` for 'pytorch', `ml_framework="tensorflow"` for 'tensorflow', and `ml_framework="auto"`.

Returns: Function does not return a value. It is used for loading a saved transformer model into the R interface.

Method `save_model()`: Method for saving a transformer model on disk. Relevant only for transformer models.

Usage:

```
TextEmbeddingModel$save_model(model_dir, save_format = "default")
```

Arguments:

model_dir string containing the path to the relevant model directory.
save_format Format for saving the model. For 'tensorflow'/'keras' models "h5" for HDF5. For 'pytorch' models "safetensors" for 'safetensors' or "pt" for 'pytorch' via pickle. Use "default" for the standard format. This is h5 for 'tensorflow'/'keras' models and safetensors for 'pytorch' models.

Returns: Function does not return a value. It is used for saving a transformer model to disk.

Method `encode()`: Method for encoding words of raw texts into integers.

Usage:

```
TextEmbeddingModel$encode(
  raw_text,
  token_encodings_only = FALSE,
  to_int = TRUE,
  trace = FALSE
)
```

Arguments:

`raw_text` vector containing the raw texts.

`token_encodings_only` bool If TRUE, only the token encodings are returned. If FALSE, the complete encoding is returned which is important for BERT models.

`to_int` bool If TRUE the integer ids of the tokens are returned. If FALSE the tokens are returned. Argument only applies for transformer models and if `token_encodings_only==TRUE`.

`trace` bool If TRUE, information of the progress is printed. FALSE if not requested.

Returns: list containing the integer sequences of the raw texts with special tokens.

Method `decode()`: Method for decoding a sequence of integers into tokens

Usage:

```
TextEmbeddingModel$decode(int_sequence, to_token = FALSE)
```

Arguments:

`int_sequence` list containing the integer sequences which should be transformed to tokens or plain text.

`to_token` bool If FALSE a plain text is returned. if TRUE a sequence of tokens is returned. Argument only relevant if the model is based on a transformer.

Returns: list of token sequences

Method `get_special_tokens()`: Method for receiving the special tokens of the model

Usage:

```
TextEmbeddingModel$get_special_tokens()
```

Returns: Returns a matrix containing the special tokens in the rows and their type, token, and id in the columns.

Method `embed()`: Method for creating text embeddings from raw texts

In the case of using a GPU and running out of memory reduce the batch size or restart R and switch to use cpu only via [set_config_cpu_only](#).

Usage:

```
TextEmbeddingModel$embed(
  raw_text = NULL,
  doc_id = NULL,
  batch_size = 8,
  trace = FALSE
)
```

Arguments:

`raw_text` vector containing the raw texts.

`doc_id` vector containing the corresponding IDs for every text.

`batch_size` int determining the maximal size of every batch.

`trace` bool TRUE, if information about the progression should be printed on console.

Returns: Method returns a R6 object of class `EmbeddedText`. This object contains the embeddings as a `data.frame` and information about the model creating the embeddings.

Method `fill_mask()`: Method for calculating tokens behind mask tokens.

Usage:

```
TextEmbeddingModel$fill_mask(text, n_solutions = 5)
```

Arguments:

`text` string Text containing mask tokens.

`n_solutions` int Number estimated tokens for every mask.

Returns: Returns a list containing a `data.frame` for every mask. The `data.frame` contains the solutions in the rows and reports the score, token id, and token string in the columns.

Method `set_publication_info()`: Method for setting the bibliographic information of the model.

Usage:

```
TextEmbeddingModel$set_publication_info(type, authors, citation, url = NULL)
```

Arguments:

`type` string Type of information which should be changed/added. `type="developer"`, and `type="modifier"` are possible.

`authors` List of people.

`citation` string Citation in free text.

`url` string Corresponding URL if applicable.

Returns: Function does not return a value. It is used to set the private members for publication information of the model.

Method `get_publication_info()`: Method for getting the bibliographic information of the model.

Usage:

```
TextEmbeddingModel$get_publication_info()
```

Returns: list of bibliographic information.

Method `set_software_license()`: Method for setting the license of the model

Usage:

```
TextEmbeddingModel$set_software_license(license = "GPL-3")
```

Arguments:

`license` string containing the abbreviation of the license or the license text.

Returns: Function does not return a value. It is used for setting the private member for the software license of the model.

Method `get_software_license()`: Method for requesting the license of the model

Usage:

```
TextEmbeddingModel$get_software_license()
```

Returns: string License of the model

Method `set_documentation_license()`: Method for setting the license of models' documentation.

Usage:

```
TextEmbeddingModel$set_documentation_license(license = "CC BY-SA")
```

Arguments:

`license` string containing the abbreviation of the license or the license text.

Returns: Function does not return a value. It is used to set the private member for the documentation license of the model.

Method `get_documentation_license()`: Method for getting the license of the models' documentation.

Usage:

```
TextEmbeddingModel$get_documentation_license()
```

Arguments:

`license` string containing the abbreviation of the license or the license text.

Method `set_model_description()`: Method for setting a description of the model

Usage:

```
TextEmbeddingModel$set_model_description(
  eng = NULL,
  native = NULL,
  abstract_eng = NULL,
  abstract_native = NULL,
  keywords_eng = NULL,
  keywords_native = NULL
)
```

Arguments:

`eng` string A text describing the training of the classifier, its theoretical and empirical background, and the different output labels in English.

`native` string A text describing the training of the classifier, its theoretical and empirical background, and the different output labels in the native language of the model.

`abstract_eng` string A text providing a summary of the description in English.

`abstract_native` string A text providing a summary of the description in the native language of the classifier.

`keywords_eng` vector of keywords in English.

`keywords_native` vector of keywords in the native language of the classifier.

Returns: Function does not return a value. It is used to set the private members for the description of the model.

Method `get_model_description()`: Method for requesting the model description.

Usage:

```
TextEmbeddingModel$get_model_description()
```

Returns: list with the description of the model in English and the native language.

Method `get_model_info()`: Method for requesting the model information

Usage:

```
TextEmbeddingModel$get_model_info()
```

Returns: list of all relevant model information

Method `get_package_versions()`: Method for requesting a summary of the R and python packages' versions used for creating the classifier.

Usage:

```
TextEmbeddingModel$get_package_versions()
```

Returns: Returns a list containing the versions of the relevant R and python packages.

Method `get_basic_components()`: Method for requesting the part of interface's configuration that is necessary for all models.

Usage:

```
TextEmbeddingModel$get_basic_components()
```

Returns: Returns a list.

Method `get_bow_components()`: Method for requesting the part of interface's configuration that is necessary bag-of-words models.

Usage:

```
TextEmbeddingModel$get_bow_components()
```

Returns: Returns a list.

Method `get_transformer_components()`: Method for requesting the part of interface's configuration that is necessary for transformer models.

Usage:

```
TextEmbeddingModel$get_transformer_components()
```

Returns: Returns a list.

Method `get_sustainability_data()`: Method for requesting a log of tracked energy consumption during training and an estimate of the resulting CO2 equivalents in kg.

Usage:

```
TextEmbeddingModel$get_sustainability_data()
```

Returns: Returns a matrix containing the tracked energy consumption, CO2 equivalents in kg, information on the tracker used, and technical information on the training infrastructure for every training run.

Method `get_ml_framework()`: Method for requesting the machine learning framework used for the classifier.

Usage:

```
TextEmbeddingModel$get_ml_framework()
```

Returns: Returns a string describing the machine learning framework used for the classifier

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
TextEmbeddingModel$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other Text Embedding: [EmbeddedText](#), [combine_embeddings\(\)](#)

| | |
|------------------|---|
| to_categorical_c | <i>Transforming classes to one-hot encoding</i> |
|------------------|---|

Description

Function written in C++ transforming a vector of classes (int) into a binary class matrix.

Usage

```
to_categorical_c(class_vector, n_classes)
```

Arguments

class_vector vector containing integers for every class. The integers must range from 0 to n_classes-1.

n_classes int Total number of classes.

Value

Returns a matrix containing the binary representation for every class.

See Also

Other Auxiliary Functions: [array_to_matrix\(\)](#), [calc_standard_classification_measures\(\)](#), [check_embedding_models\(\)](#), [clean_pytorch_log_transformers\(\)](#), [create_iota2_mean_object\(\)](#), [create_synthetic_units\(\)](#), [generate_id\(\)](#), [get_coder_metrics\(\)](#), [get_folds\(\)](#), [get_n_chunks\(\)](#), [get_stratified_train_test_split\(\)](#), [get_synthetic_cases\(\)](#), [get_train_test_split\(\)](#), [is.null_or_na\(\)](#), [matrix_to_array_c\(\)](#), [split_labeled_unlabeled\(\)](#), [summarize_tracked_sustainability\(\)](#)

train_tune_bert_model *Function for training and fine-tuning a BERT model*

Description

This function can be used to train or fine-tune a transformer based on BERT architecture with the help of the python libraries 'transformers', 'datasets', and 'tokenizers'.

Usage

```
train_tune_bert_model(
    ml_framework = aifeducation_config$get_framework(),
    output_dir,
    model_dir_path,
    raw_texts,
    p_mask = 0.15,
    whole_word = TRUE,
    val_size = 0.1,
    n_epoch = 1,
    batch_size = 12,
    chunk_size = 250,
    full_sequences_only = FALSE,
    min_seq_len = 50,
    learning_rate = 0.003,
    n_workers = 1,
    multi_process = FALSE,
    sustain_track = TRUE,
    sustain_iso_code = NULL,
    sustain_region = NULL,
    sustain_interval = 15,
    trace = TRUE,
    keras_trace = 1,
    pytorch_trace = 1,
    pytorch_safetensors = TRUE
)
```

Arguments

| | |
|----------------|--|
| ml_framework | string Framework to use for training and inference. ml_framework="tensorflow" for 'tensorflow' and ml_framework="pytorch" for 'pytorch'. |
| output_dir | string Path to the directory where the final model should be saved. If the directory does not exist, it will be created. |
| model_dir_path | string Path to the directory where the original model is stored. |
| raw_texts | vector containing the raw texts for training. |
| p_mask | double Ratio determining the number of words/tokens for masking. |

| | |
|---------------------|--|
| whole_word | bool TRUE if whole word masking should be applied. If FALSE token masking is used. |
| val_size | double Ratio determining the amount of token chunks used for validation. |
| n_epoch | int Number of epochs for training. |
| batch_size | int Size of batches. |
| chunk_size | int Size of every chunk for training. |
| full_sequences_only | bool TRUE for using only chunks with a sequence length equal to chunk_size. |
| min_seq_len | int Only relevant if full_sequences_only=FALSE. Value determines the minimal sequence length for inclusion in training process. |
| learning_rate | double Learning rate for adam optimizer. |
| n_workers | int Number of workers. Only relevant if ml_framework="tensorflow". |
| multi_process | bool TRUE if multiple processes should be activated. Only relevant if ml_framework="tensorflow". |
| sustain_track | bool If TRUE energy consumption is tracked during training via the python library codecarbon. |
| sustain_iso_code | string ISO code (Alpha-3-Code) for the country. This variable must be set if sustainability should be tracked. A list can be found on Wikipedia: https://en.wikipedia.org/wiki/List_of_ISO_3166_country_codes . |
| sustain_region | Region within a country. Only available for USA and Canada See the documentation of codecarbon for more information. https://mlco2.github.io/codecarbon/parameters.html |
| sustain_interval | integer Interval in seconds for measuring power usage. |
| trace | bool TRUE if information on the progress should be printed to the console. |
| keras_trace | int keras_trace=0 does not print any information about the training process from keras on the console. keras_trace=1 prints a progress bar. keras_trace=2 prints one line of information for every epoch. Only relevant if ml_framework="tensorflow". |
| pytorch_trace | int pytorch_trace=0 does not print any information about the training process from pytorch on the console. pytorch_trace=1 prints a progress bar. |
| pytorch_safetensors | bool If TRUE a 'pytorch' model is saved in safetensors format. If FALSE or 'safetensors' not available it is saved in the standard pytorch format (.bin). Only relevant for pytorch models. |

Value

This function does not return an object. Instead the trained or fine-tuned model is saved to disk.

Note

This models uses a WordPiece Tokenizer like BERT and can be trained with whole word masking. Transformer library may show a warning which can be ignored.

Pre-Trained models which can be fine-tuned with this function are available at <https://huggingface.co/>.

New models can be created via the function `create_bert_model`.

Training of the model makes use of dynamic masking in contrast to the original paper where static masking was applied.

References

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In J. Burstein, C. Doran, & T. Solorio (Eds.), Proceedings of the 2019 Conference of the North (pp. 4171–4186). Association for Computational Linguistics. doi:10.18653/v1/N191423

Hugging Face documentation https://huggingface.co/docs/transformers/model_doc/bert#transformers.TFBertForMaskedLM

See Also

Other Transformer: `create_bert_model()`, `create_deberta_v2_model()`, `create_funnel_model()`, `create_longformer_model()`, `create_roberta_model()`, `train_tune_deberta_v2_model()`, `train_tune_funnel_model()`, `train_tune_longformer_model()`, `train_tune_roberta_model()`

train_tune_deberta_v2_model

Function for training and fine-tuning a DeBERTa-V2 model

Description

This function can be used to train or fine-tune a transformer based on DeBERTa-V2 architecture with the help of the python libraries 'transformers', 'datasets', and 'tokenizers'.

Usage

```
train_tune_deberta_v2_model(  
    ml_framework = aifeducation_config$get_framework(),  
    output_dir,  
    model_dir_path,  
    raw_texts,  
    p_mask = 0.15,  
    whole_word = TRUE,  
    val_size = 0.1,  
    n_epoch = 1,  
    batch_size = 12,  
    chunk_size = 250,  
    full_sequences_only = FALSE,  
    min_seq_len = 50,  
    learning_rate = 0.03,
```



```

n_workers = 1,
multi_process = FALSE,
sustain_track = TRUE,
sustain_iso_code = NULL,
sustain_region = NULL,
sustain_interval = 15,
trace = TRUE,
keras_trace = 1,
pytorch_trace = 1,
pytorch_safetensors = TRUE
)

```

Arguments

| | |
|----------------------------------|--|
| <code>ml_framework</code> | string Framework to use for training and inference. <code>ml_framework="tensorflow"</code> for 'tensorflow' and <code>ml_framework="pytorch"</code> for 'pytorch'. |
| <code>output_dir</code> | string Path to the directory where the final model should be saved. If the directory does not exist, it will be created. |
| <code>model_dir_path</code> | string Path to the directory where the original model is stored. |
| <code>raw_texts</code> | vector containing the raw texts for training. |
| <code>p_mask</code> | double Ratio determining the number of words/tokens for masking. |
| <code>whole_word</code> | bool TRUE if whole word masking should be applied. If FALSE token masking is used. |
| <code>val_size</code> | double Ratio determining the amount of token chunks used for validation. |
| <code>n_epoch</code> | int Number of epochs for training. |
| <code>batch_size</code> | int Size of batches. |
| <code>chunk_size</code> | int Size of every chunk for training. |
| <code>full_sequences_only</code> | bool TRUE for using only chunks with a sequence length equal to <code>chunk_size</code> . |
| <code>min_seq_len</code> | int Only relevant if <code>full_sequences_only=FALSE</code> . Value determines the minimal sequence length for inclusion in training process. |
| <code>learning_rate</code> | bool Learning rate for adam optimizer. |
| <code>n_workers</code> | int Number of workers. Only relevant if <code>ml_framework="tensorflow"</code> . |
| <code>multi_process</code> | bool TRUE if multiple processes should be activated. Only relevant if <code>ml_framework="tensorflow"</code> . |
| <code>sustain_track</code> | bool If TRUE energy consumption is tracked during training via the python library <code>codecarbon</code> . |
| <code>sustain_iso_code</code> | string ISO code (Alpha-3-Code) for the country. This variable must be set if sustainability should be tracked. A list can be found on Wikipedia: https://en.wikipedia.org/wiki/List_of_ISO_3166_country_codes . |
| <code>sustain_region</code> | Region within a country. Only available for USA and Canada See the documentation of <code>codecarbon</code> for more information. https://mlco2.github.io/codecarbon/parameters.html |

| | |
|---------------------|---|
| sustain_interval | integer Interval in seconds for measuring power usage. |
| trace | bool TRUE if information on the progress should be printed to the console. |
| keras_trace | int keras_trace=0 does not print any information about the training process from keras on the console. keras_trace=1 prints a progress bar. keras_trace=2 prints one line of information for every epoch. Only relevant if ml_framework="tensorflow". |
| pytorch_trace | int pytorch_trace=0 does not print any information about the training process from pytorch on the console. pytorch_trace=1 prints a progress bar. |
| pytorch_safetensors | bool If TRUE a 'pytorch' model is saved in safetensors format. If FALSE or 'safetensors' not available it is saved in the standard pytorch format (.bin). Only relevant for pytorch models. |

Value

This function does not return an object. Instead the trained or fine-tuned model is saved to disk.

Note

Pre-Trained models which can be fine-tuned with this function are available at <https://huggingface.co/>. New models can be created via the function [create_deberta_v2_model](#).

Training of this model makes use of dynamic masking.

References

He, P., Liu, X., Gao, J. & Chen, W. (2020). DeBERTa: Decoding-enhanced BERT with Disentangled Attention. [doi:10.48550/arXiv.2006.03654](https://arxiv.org/abs/2006.03654)

Hugging Face Documentation https://huggingface.co/docs/transformers/model_doc/deberta-v2#debertav2

See Also

Other Transformer: [create_bert_model\(\)](#), [create_deberta_v2_model\(\)](#), [create_funnel_model\(\)](#), [create_longformer_model\(\)](#), [create_roberta_model\(\)](#), [train_tune_bert_model\(\)](#), [train_tune_funnel_model\(\)](#), [train_tune_longformer_model\(\)](#), [train_tune_roberta_model\(\)](#)

train_tune_funnel_model

Function for training and fine-tuning a Funnel Transformer model

Description

This function can be used to train or fine-tune a transformer based on Funnel Transformer architecture with the help of the python libraries 'transformers', 'datasets', and 'tokenizers'.

Usage

```

train_tune_funnel_model(
    ml_framework = aifeducation_config$get_framework(),
    output_dir,
    model_dir_path,
    raw_texts,
    p_mask = 0.15,
    whole_word = TRUE,
    val_size = 0.1,
    n_epoch = 1,
    batch_size = 12,
    chunk_size = 250,
    min_seq_len = 50,
    full_sequences_only = FALSE,
    learning_rate = 0.003,
    n_workers = 1,
    multi_process = FALSE,
    sustain_track = TRUE,
    sustain_iso_code = NULL,
    sustain_region = NULL,
    sustain_interval = 15,
    trace = TRUE,
    keras_trace = 1,
    pytorch_trace = 1,
    pytorch_safetensors = TRUE
)

```

Arguments

| | |
|-----------------------------|--|
| <code>ml_framework</code> | string Framework to use for training and inference. <code>ml_framework="tensorflow"</code> for 'tensorflow' and <code>ml_framework="pytorch"</code> for 'pytorch'. |
| <code>output_dir</code> | string Path to the directory where the final model should be saved. If the directory does not exist, it will be created. |
| <code>model_dir_path</code> | string Path to the directory where the original model is stored. |
| <code>raw_texts</code> | vector containing the raw texts for training. |
| <code>p_mask</code> | double Ratio determining the number of words/tokens for masking. |
| <code>whole_word</code> | bool TRUE if whole word masking should be applied. If FALSE token masking is used. |
| <code>val_size</code> | double Ratio determining the amount of token chunks used for validation. |
| <code>n_epoch</code> | int Number of epochs for training. |
| <code>batch_size</code> | int Size of batches. |
| <code>chunk_size</code> | int Size of every chunk for training. |
| <code>min_seq_len</code> | int Only relevant if <code>full_sequences_only=FALSE</code> . Value determines the minimal sequence length for inclusion in training process. |

| | |
|---------------------|--|
| full_sequences_only | bool TRUE if only token sequences with a length equal to chunk_size should be used for training. |
| learning_rate | double Learning rate for adam optimizer. |
| n_workers | int Number of workers. |
| multi_process | bool TRUE if multiple processes should be activated. |
| sustain_track | bool If TRUE energy consumption is tracked during training via the python library codecarbon. |
| sustain_iso_code | string ISO code (Alpha-3-Code) for the country. This variable must be set if sustainability should be tracked. A list can be found on Wikipedia: https://en.wikipedia.org/wiki/List_of_ISO_3166_country_codes . |
| sustain_region | Region within a country. Only available for USA and Canada See the documentation of codecarbon for more information. https://mlco2.github.io/codecarbon/parameters.html |
| sustain_interval | integer Interval in seconds for measuring power usage. |
| trace | bool TRUE if information on the progress should be printed to the console. |
| keras_trace | int keras_trace=0 does not print any information about the training process from keras on the console. keras_trace=1 prints a progress bar. keras_trace=2 prints one line of information for every epoch. |
| pytorch_trace | int pytorch_trace=0 does not print any information about the training process from pytorch on the console. pytorch_trace=1 prints a progress bar. |
| pytorch_safetensors | bool If TRUE a 'pytorch' model is saved in safetensors format. If FALSE or 'safetensors' not available it is saved in the standard pytorch format (.bin). Only relevant for pytorch models. |

Value

This function does not return an object. Instead the trained or fine-tuned model is saved to disk.

Note

if `aug_vocab_by > 0` the raw text is used for training a WordPiece tokenizer. At the end of this process, additional entries are added to the vocabulary that are not part of the original vocabulary. This is in an experimental state.

Pre-Trained models which can be fine-tuned with this function are available at <https://huggingface.co/>.

New models can be created via the function [create_funnel_model](#).

Training of the model makes use of dynamic masking.

References

Dai, Z., Lai, G., Yang, Y. & Le, Q. V. (2020). Funnel-Transformer: Filtering out Sequential Redundancy for Efficient Language Processing. [doi:10.48550/arXiv.2006.03236](https://doi.org/10.48550/arXiv.2006.03236)

Hugging Face documentation https://huggingface.co/docs/transformers/model_doc/funnel#funnel-transformer

See Also

Other Transformer: [create_bert_model\(\)](#), [create_deberta_v2_model\(\)](#), [create_funnel_model\(\)](#), [create_longformer_model\(\)](#), [create_roberta_model\(\)](#), [train_tune_bert_model\(\)](#), [train_tune_deberta_v2_model\(\)](#), [train_tune_longformer_model\(\)](#), [train_tune_roberta_model\(\)](#)

train_tune_longformer_model

Function for training and fine-tuning a Longformer model

Description

This function can be used to train or fine-tune a transformer based on Longformer architecture with the help of the python libraries 'transformers', 'datasets', and 'tokenizers'.

Usage

```
train_tune_longformer_model(  
    ml_framework = aifeducation_config$get_framework,  
    output_dir,  
    model_dir_path,  
    raw_texts,  
    p_mask = 0.15,  
    val_size = 0.1,  
    n_epoch = 1,  
    batch_size = 12,  
    chunk_size = 250,  
    full_sequences_only = FALSE,  
    min_seq_len = 50,  
    learning_rate = 0.03,  
    n_workers = 1,  
    multi_process = FALSE,  
    sustain_track = TRUE,  
    sustain_iso_code = NULL,  
    sustain_region = NULL,  
    sustain_interval = 15,  
    trace = TRUE,  
    keras_trace = 1,  
    pytorch_trace = 1,  
    pytorch_safetensors = TRUE  
)
```

Arguments

| | |
|----------------------------------|--|
| <code>ml_framework</code> | string Framework to use for training and inference. <code>ml_framework="tensorflow"</code> for 'tensorflow' and <code>ml_framework="pytorch"</code> for 'pytorch'. |
| <code>output_dir</code> | string Path to the directory where the final model should be saved. If the directory does not exist, it will be created. |
| <code>model_dir_path</code> | string Path to the directory where the original model is stored. |
| <code>raw_texts</code> | vector containing the raw texts for training. |
| <code>p_mask</code> | double Ratio determining the number of words/tokens for masking. |
| <code>val_size</code> | double Ratio determining the amount of token chunks used for validation. |
| <code>n_epoch</code> | int Number of epochs for training. |
| <code>batch_size</code> | int Size of batches. |
| <code>chunk_size</code> | int Size of every chunk for training. |
| <code>full_sequences_only</code> | bool TRUE for using only chunks with a sequence length equal to <code>chunk_size</code> . |
| <code>min_seq_len</code> | int Only relevant if <code>full_sequences_only=FALSE</code> . Value determines the minimal sequence length for inclusion in training process. |
| <code>learning_rate</code> | bool Learning rate for adam optimizer. |
| <code>n_workers</code> | int Number of workers. Only relevant if <code>ml_framework="tensorflow"</code> . |
| <code>multi_process</code> | bool TRUE if multiple processes should be activated. Only relevant if <code>ml_framework="tensorflow"</code> . |
| <code>sustain_track</code> | bool If TRUE energy consumption is tracked during training via the python library <code>codecarbon</code> . |
| <code>sustain_iso_code</code> | string ISO code (Alpha-3-Code) for the country. This variable must be set if sustainability should be tracked. A list can be found on Wikipedia: https://en.wikipedia.org/wiki/List_of_ISO_3166_country_codes . |
| <code>sustain_region</code> | Region within a country. Only available for USA and Canada See the documentation of <code>codecarbon</code> for more information. https://mlco2.github.io/codecarbon/parameters.html |
| <code>sustain_interval</code> | integer Interval in seconds for measuring power usage. |
| <code>trace</code> | bool TRUE if information on the progress should be printed to the console. |
| <code>keras_trace</code> | int <code>keras_trace=0</code> does not print any information about the training process from keras on the console. <code>keras_trace=1</code> prints a progress bar. <code>keras_trace=2</code> prints one line of information for every epoch. Only relevant if <code>ml_framework="tensorflow"</code> . |
| <code>pytorch_trace</code> | int <code>pytorch_trace=0</code> does not print any information about the training process from pytorch on the console. <code>pytorch_trace=1</code> prints a progress bar. |
| <code>pytorch_safetensors</code> | bool If TRUE a 'pytorch' model is saved in <code>safetensors</code> format. If FALSE or 'safetensors' not available it is saved in the standard pytorch format (.bin). Only relevant for pytorch models. |

Value

This function does not return an object. Instead the trained or fine-tuned model is saved to disk.

Note

Pre-Trained models which can be fine-tuned with this function are available at <https://huggingface.co/>. New models can be created via the function `create_roberta_model`.

Training of this model makes use of dynamic masking.

References

Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The Long-Document Transformer. doi:10.48550/arXiv.2004.05150

Hugging Face Documentation https://huggingface.co/docs/transformers/model_doc/longformer#transformers.LongformerConfig

See Also

Other Transformer: `create_bert_model()`, `create_deberta_v2_model()`, `create_funnel_model()`, `create_longformer_model()`, `create_roberta_model()`, `train_tune_bert_model()`, `train_tune_deberta_v2_model()`, `train_tune_funnel_model()`, `train_tune_roberta_model()`

`train_tune_roberta_model`

Function for training and fine-tuning a RoBERTa model

Description

This function can be used to train or fine-tune a transformer based on RoBERTa architecture with the help of the python libraries 'transformers', 'datasets', and 'tokenizers'.

Usage

```
train_tune_roberta_model(  
    ml_framework = aifeducation_config$get_framework(),  
    output_dir,  
    model_dir_path,  
    raw_texts,  
    p_mask = 0.15,  
    val_size = 0.1,  
    n_epoch = 1,  
    batch_size = 12,  
    chunk_size = 250,  
    full_sequences_only = FALSE,  
    min_seq_len = 50,  
    learning_rate = 0.03,
```

```

n_workers = 1,
multi_process = FALSE,
sustain_track = TRUE,
sustain_iso_code = NULL,
sustain_region = NULL,
sustain_interval = 15,
trace = TRUE,
keras_trace = 1,
pytorch_trace = 1,
pytorch_safetensors = TRUE
)

```

Arguments

| | |
|----------------------------------|--|
| <code>ml_framework</code> | string Framework to use for training and inference. <code>ml_framework="tensorflow"</code> for 'tensorflow' and <code>ml_framework="pytorch"</code> for 'pytorch'. |
| <code>output_dir</code> | string Path to the directory where the final model should be saved. If the directory does not exist, it will be created. |
| <code>model_dir_path</code> | string Path to the directory where the original model is stored. |
| <code>raw_texts</code> | vector containing the raw texts for training. |
| <code>p_mask</code> | double Ratio determining the number of words/tokens for masking. |
| <code>val_size</code> | double Ratio determining the amount of token chunks used for validation. |
| <code>n_epoch</code> | int Number of epochs for training. |
| <code>batch_size</code> | int Size of batches. |
| <code>chunk_size</code> | int Size of every chunk for training. |
| <code>full_sequences_only</code> | bool TRUE for using only chunks with a sequence length equal to <code>chunk_size</code> . |
| <code>min_seq_len</code> | int Only relevant if <code>full_sequences_only=FALSE</code> . Value determines the minimal sequence length for inclusion in training process. |
| <code>learning_rate</code> | bool Learning rate for adam optimizer. |
| <code>n_workers</code> | int Number of workers. Only relevant if <code>ml_framework="tensorflow"</code> . |
| <code>multi_process</code> | bool TRUE if multiple processes should be activated. Only relevant if <code>ml_framework="tensorflow"</code> . |
| <code>sustain_track</code> | bool If TRUE energy consumption is tracked during training via the python library <code>codecarbon</code> . |
| <code>sustain_iso_code</code> | string ISO code (Alpha-3-Code) for the country. This variable must be set if sustainability should be tracked. A list can be found on Wikipedia: https://en.wikipedia.org/wiki/List_of_ISO_3166_country_codes . |
| <code>sustain_region</code> | Region within a country. Only available for USA and Canada See the documentation of <code>codecarbon</code> for more information. https://mlco2.github.io/codecarbon/parameters.html |
| <code>sustain_interval</code> | integer Interval in seconds for measuring power usage. |
| <code>trace</code> | bool TRUE if information on the progress should be printed to the console. |

`keras_trace` int `keras_trace=0` does not print any information about the training process from keras on the console. `keras_trace=1` prints a progress bar. `keras_trace=2` prints one line of information for every epoch. Only relevant if `ml_framework="tensorflow"`.

`pytorch_trace` int `pytorch_trace=0` does not print any information about the training process from pytorch on the console. `pytorch_trace=1` prints a progress bar.

`pytorch_safetensors` bool If TRUE a 'pytorch' model is saved in safetensors format. If FALSE or 'safetensors' not available it is saved in the standard pytorch format (.bin). Only relevant for pytorch models.

Value

This function does not return an object. Instead the trained or fine-tuned model is saved to disk.

Note

Pre-Trained models which can be fine-tuned with this function are available at <https://huggingface.co/>. New models can be created via the function [create_roberta_model](#).

Training of this model makes use of dynamic masking.

References

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. [doi:10.48550/arXiv.1907.11692](https://arxiv.org/abs/1907.11692)

Hugging Face Documentation https://huggingface.co/docs/transformers/model_doc/roberta#transformers.RobertaConfig

See Also

Other Transformer: [create_bert_model\(\)](#), [create_deberta_v2_model\(\)](#), [create_funnel_model\(\)](#), [create_longformer_model\(\)](#), [create_roberta_model\(\)](#), [train_tune_bert_model\(\)](#), [train_tune_deberta_v2_model\(\)](#), [train_tune_funnel_model\(\)](#), [train_tune_longformer_model\(\)](#)

update_aifeducation_progress_bar

Update master progress bar in aifeducation shiny app.

Description

This function updates the master progress bar in aifeducation shiny app. The progress bar reports the current state of the overall process.

Usage

`update_aifeducation_progress_bar(value, total, title = NULL)`

Arguments

| | | |
|-------|--------|---|
| value | int | Value describing the current step of the process. |
| total | int | Total number of steps of the process. |
| title | string | Title displaying in the top of the progress bar. |

Value

Function does nothing returns. It updates the progress bar with the id "pgr_bar_aifeducation".

See Also

Other Auxiliary GUI Functions: [update_aifeducation_progress_bar_epochs\(\)](#), [update_aifeducation_progress_bar_epochs\(\)](#)

update_aifeducation_progress_bar_epochs

Update epoch progress bar in aifeducation shiny app.

Description

This function updates the epoch progress bar in aifeducation shiny app. The progress bar reports the current state of the overall process.

Usage

```
update_aifeducation_progress_bar_epochs(value, total, title = NULL)
```

Arguments

| | | |
|-------|--------|---|
| value | int | Value describing the current step of the process. |
| total | int | Total number of steps of the process. |
| title | string | Title displaying in the top of the progress bar. |

Details

This function is called very often during training a model. Thus, the function does not check the requirements for updating the progress bar to reduce computational time. The check for fulfilling the necessary conditions must be implemented separately.

Value

Function does nothing returns. It updates the progress bar with the id "pgr_bar_aifeducation_epochs".

See Also

Other Auxiliary GUI Functions: [update_aifeducation_progress_bar\(\)](#), [update_aifeducation_progress_bar_steps\(\)](#)

`update_aifeducation_progress_bar_steps`*Update step/batch progress bar in aifeducation shiny app.*

Description

This function updates the step/batch progress bar in aifeducation shiny app. The progress bar reports the current state of the overall process.

Usage

```
update_aifeducation_progress_bar_steps(value, total, title = NULL)
```

Arguments

| | |
|--------------------|--|
| <code>value</code> | <code>int</code> Value describing the current step of the process. |
| <code>total</code> | <code>int</code> Total number of steps of the process. |
| <code>title</code> | <code>string</code> Title displaying in the top of the progress bar. |

Details

This function is called very often during training a model. Thus, the function does not check the requirements for updating the progress bar to reduce computational time. The check for fulfilling the necessary conditions must be implemented separately.

Value

Function does nothing returns. It updates the progress bar with the id "pgr_bar_aifeducation_steps".

See Also

Other Auxiliary GUI Functions: [update_aifeducation_progress_bar\(\)](#), [update_aifeducation_progress_bar_epoch](#)

Index

- * **Auxiliary Functions**
 - [array_to_matrix](#), 5
 - [calc_standard_classification_measures](#), 9
 - [create_synthetic_units](#), 22
 - [get_coder_metrics](#), 25
 - [get_n_chunks](#), 27
 - [get_synthetic_cases](#), 27
 - [matrix_to_array_c](#), 30
 - [to_categorical_c](#), 53
 - * **Auxiliary GUI Functions**
 - [update_aifeducation_progress_bar](#), 65
 - [update_aifeducation_progress_bar_epochs](#), 66
 - [update_aifeducation_progress_bar_steps](#), 67
 - * **Classification**
 - [TextEmbeddingClassifierNeuralNet](#), 35
 - * **Graphical User Interface**
 - [start_aifeducation_studio](#), 35
 - * **Installation and Configuration**
 - [aifeducation_config](#), 4
 - [AifeducationConfiguration](#), 3
 - [check_aif_py_modules](#), 10
 - [install_py_modules](#), 28
 - [set_config_cpu_only](#), 32
 - [set_config_gpu_low_memory](#), 32
 - [set_config_os_envIRON_logger](#), 33
 - [set_config_tf_logger](#), 33
 - [set_transformers_logger](#), 34
 - * **Preparation**
 - [bow_pp_create_basic_text_rep](#), 6
 - [bow_pp_create_vocab_draft](#), 8
 - * **Saving and Loading**
 - [load_ai_model](#), 29
 - [save_ai_model](#), 31
 - * **Text Embedding**
 - [combine_embeddings](#), 10
 - [EmbeddedText](#), 23
 - [TextEmbeddingModel](#), 45
 - * **Transformer**
 - [create_bert_model](#), 11
 - [create_deberta_v2_model](#), 13
 - [create_funnel_model](#), 15
 - [create_longformer_model](#), 18
 - [create_roberta_model](#), 20
 - [train_tune_bert_model](#), 54
 - [train_tune_deberta_v2_model](#), 56
 - [train_tune_funnel_model](#), 58
 - [train_tune_longformer_model](#), 61
 - [train_tune_roberta_model](#), 63
 - * **datasets**
 - [aifeducation_config](#), 4
- [aifeducation_config](#), 4, 4, 10, 29, 32–34
- [AifeducationConfiguration](#), 3, 5, 10, 29, 32–34
- [array_to_matrix](#), 5, 9, 23, 26–28, 30, 53
- [bow_pp_create_basic_text_rep](#), 6, 9, 48
- [bow_pp_create_vocab_draft](#), 6, 7, 8
- [calc_standard_classification_measures](#), 5, 9, 23, 26–28, 30, 53
- [check_aif_py_modules](#), 4, 5, 10, 29, 32–34
- [check_embedding_models](#), 5, 9, 23, 26–28, 30, 53
- [clean_pytorch_log_transformers](#), 5, 9, 23, 26–28, 30, 53
- [combine_embeddings](#), 10, 25, 53
- [create_bert_model](#), 11, 15, 18, 20, 22, 56, 58, 61, 63, 65
- [create_deberta_v2_model](#), 13, 13, 18, 20, 22, 56, 58, 61, 63, 65
- [create_funnel_model](#), 13, 15, 15, 20, 22, 56, 58, 60, 61, 63, 65

- create_iota2_mean_object, [5](#), [9](#), [23](#), [26–28](#), [30](#), [53](#)
- create_longformer_model, [13](#), [15](#), [18](#), [18](#), [22](#), [56](#), [58](#), [61](#), [63](#), [65](#)
- create_roberta_model, [13](#), [15](#), [18](#), [20](#), [20](#), [56](#), [58](#), [61](#), [63](#), [65](#)
- create_synthetic_units, [5](#), [9](#), [22](#), [26–28](#), [30](#), [53](#)

- EmbeddedText, [11](#), [22](#), [23](#), [24](#), [28](#), [35](#), [42](#), [50](#), [53](#)

- generate_id, [5](#), [9](#), [23](#), [26–28](#), [30](#), [53](#)
- get_coder_metrics, [5](#), [9](#), [23](#), [25](#), [27](#), [28](#), [30](#), [53](#)
- get_folds, [5](#), [9](#), [23](#), [26–28](#), [30](#), [53](#)
- get_n_chunks, [5](#), [9](#), [23](#), [26](#), [27](#), [28](#), [30](#), [53](#)
- get_stratified_train_test_split, [5](#), [9](#), [23](#), [26–28](#), [30](#), [53](#)
- get_synthetic_cases, [5](#), [9](#), [22](#), [23](#), [26](#), [27](#), [27](#), [30](#), [53](#)
- get_train_test_split, [5](#), [9](#), [23](#), [26–28](#), [30](#), [53](#)

- install_py_modules, [4](#), [5](#), [10](#), [28](#), [32–34](#)
- is.null_or_na, [5](#), [9](#), [23](#), [26–28](#), [30](#), [53](#)

- load_ai_model, [29](#), [31](#)

- matrix_to_array_c, [5](#), [9](#), [23](#), [26–28](#), [30](#), [53](#)

- R6, [23](#), [45](#), [50](#)

- save_ai_model, [30](#), [31](#)
- set_config_cpu_only, [4](#), [5](#), [10](#), [29](#), [32](#), [33](#), [34](#), [49](#)
- set_config_gpu_low_memory, [4](#), [5](#), [10](#), [29](#), [32](#), [32](#), [33](#), [34](#)
- set_config_os_envIRON_logger, [4](#), [5](#), [10](#), [29](#), [32](#), [33](#), [33](#), [34](#)
- set_config_tf_logger, [4](#), [5](#), [10](#), [29](#), [32](#), [33](#), [33](#), [34](#)
- set_transformers_logger, [4](#), [5](#), [10](#), [29](#), [32–34](#), [34](#)
- split_labeled_unlabeled, [5](#), [9](#), [23](#), [26–28](#), [30](#), [53](#)
- start_aifeducation_studio, [35](#)
- summarize_tracked_sustainability, [5](#), [9](#), [23](#), [26–28](#), [30](#), [53](#)

- TextEmbeddingClassifierNeuralNet, [4](#), [22–24](#), [27](#), [30](#), [31](#), [35](#), [39](#)
- TextEmbeddingModel, [4–6](#), [11](#), [23–25](#), [30](#), [31](#), [34](#), [35](#), [42](#), [45](#), [48](#)
- to_categorical_c, [5](#), [9](#), [23](#), [26–28](#), [30](#), [53](#)
- train_tune_bert_model, [13](#), [15](#), [18](#), [20](#), [22](#), [54](#), [58](#), [61](#), [63](#), [65](#)
- train_tune_deberta_v2_model, [13](#), [15](#), [18](#), [20](#), [22](#), [56](#), [56](#), [61](#), [63](#), [65](#)
- train_tune_funnel_model, [13](#), [15](#), [17](#), [18](#), [20](#), [22](#), [56](#), [58](#), [58](#), [63](#), [65](#)
- train_tune_longformer_model, [13](#), [15](#), [18](#), [20](#), [22](#), [56](#), [58](#), [61](#), [61](#), [65](#)
- train_tune_roberta_model, [13](#), [15](#), [18](#), [20](#), [22](#), [56](#), [58](#), [61](#), [63](#), [63](#)

- update_aifeducation_progress_bar, [65](#), [66](#), [67](#)
- update_aifeducation_progress_bar_epochs, [66](#), [66](#), [67](#)
- update_aifeducation_progress_bar_steps, [66](#), [67](#)