Package 'sentencepiece'

May 9, 2026

Type Package

Title Text Tokenization using Byte Pair Encoding and Unigram Modelling

Version 0.2.5

Maintainer Jan Wijffels <jwijffels@bnosac.be>

Description

Unsupervised text tokenizer allowing to perform byte pair encoding and unigram modelling. Wraps the 'sentencepiece' library <https://github.com/google/sentencepiece> which provides a language independent tokenizer to split text in words and smaller subword units.
The techniques are explained in the paper ``SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing" by Taku Kudo and John Richardson (2018) <doi:10.18653/v1/D18-2012>.
Provides as well straightforward access to pretrained byte pair encoding models and subword embeddings trained on Wikipedia using 'word2vec',
as described in ``BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages" by Benjamin Heinzerling and Michael Strube (2018) <http://www.lrec-conf.org/proceedings/lrec2018/pdf/1049.pdf>.

URL https://github.com/bnosac/sentencepiece

License MPL-2.0

Encoding UTF-8

RoxygenNote 7.2.3

Depends R (>= 2.10)

Imports Rcpp (>= 0.11.5), stats

Suggests tokenizers.bpe, word2vec (>= 0.2.0)

LinkingTo Rcpp

SystemRequirements C++17

NeedsCompilation yes

Author Jan Wijffels [aut, cre, cph] (R wrapper),
BNOSAC [cph] (R wrapper),
Google Inc. [ctb, cph] (Files at src/sentencepiece/src (Apache License, Version 2.0),
The Abseil Authors [ctb, cph] (Files at src/third_party/absl (Apache

License, Version 2.0),
Google Inc. [ctb, cph] (Files at src/third_party/protobuf-lite (BSD-3 License)),
Kenton Varda (Google Inc.) [ctb, cph] (Files at src/third_party/protobuf-lite: coded_stream.cc, extension_set.cc, generated_message_util.cc, generated_message_util.cc, message_lite.cc, repeated_field.cc, wire_format_lite.cc, zero_copy_stream.cc, zero_copy_stream_impl_lite.cc, google/protobuf/extension_set.h, google/protobuf/generated_message_util.h, google/protobuf/wire_format_lite.h, google/protobuf/wire_format_lite_inl.h, google/protobuf/message_lite.h, google/protobuf/repeated_field.h, google/protobuf/io/coded_stream.h, google/protobuf/io/zero_copy_stream_impl_lite.h, google/protobuf/io/zero_copy_stream.h, google/protobuf/stubs/common.h, google/protobuf/stubs/hash.h, google/protobuf/stubs/once.h, google/protobuf/stubs/once.h.org (BSD-3 License)),
Sanjay Ghemawat (Google Inc.) [ctb, cph] (Design of files at src/third_party/protobuf-lite: coded_stream.cc, extension_set.cc, generated_message_util.cc, generated_message_util.cc, message_lite.cc, repeated_field.cc, wire_format_lite.cc, zero_copy_stream.cc, zero_copy_stream_impl_lite.cc, google/protobuf/extension_set.h, google/protobuf/generated_message_util.h, google/protobuf/wire_format_lite.h, google/protobuf/wire_format_lite_inl.h, google/protobuf/message_lite.h, google/protobuf/repeated_field.h, google/protobuf/io/coded_stream.h, google/protobuf/io/zero_copy_stream_impl_lite.h, google/protobuf/io/zero_copy_stream.h (BSD-3 License)),
Jeff Dean (Google Inc.) [ctb, cph] (Design of files at src/third_party/protobuf-lite: coded_stream.cc, extension_set.cc, generated_message_util.cc, generated_message_util.cc, message_lite.cc, repeated_field.cc, wire_format_lite.cc, zero_copy_stream.cc, zero_copy_stream_impl_lite.cc, google/protobuf/extension_set.h, google/protobuf/generated_message_util.h, google/protobuf/wire_format_lite.h, google/protobuf/wire_format_lite_inl.h, google/protobuf/message_lite.h, google/protobuf/repeated_field.h, google/protobuf/io/coded_stream.h, google/protobuf/io/zero_copy_stream_impl_lite.h, google/protobuf/io/zero_copy_stream.h (BSD-3 License)),
Laszlo Csomor (Google Inc.) [ctb, cph] (Files at src/third_party/protobuf-lite: io_win32.cc, google/protobuf/stubs/io_win32.h (BSD-3 License)),

Wink Saville (Google Inc.) [ctb, cph] (Files at
src/third_party/protobuf-lite: message_lite.cc,
google/protobuf/wire_format_lite.h,
google/protobuf/wire_format_lite_inl.h,
google/protobuf/message_lite.h (BSD-3 License)),
Jim Meehan (Google Inc.) [ctb, cph] (Files at
src/third_party/protobuf-lite: structurally_valid.cc (BSD-3
License)),
Chris Atenasio (Google Inc.) [ctb, cph] (Files at
src/third_party/protobuf-lite: google/protobuf/wire_format_lite.h
(BSD-3 License)),
Jason Hsueh (Google Inc.) [ctb, cph] (Files at
src/third_party/protobuf-lite:
google/protobuf/io/coded_stream_inl.h (BSD-3 License)),
Anton Carver (Google Inc.) [ctb, cph] (Files at
src/third_party/protobuf-lite: google/protobuf/stubs/map_util.h
(BSD-3 License)),
Maxim Lifantsev (Google Inc.) [ctb, cph] (Files at
src/third_party/protobuf-lite: google/protobuf/stubs/mathlimits.h
(BSD-3 License)),
Susumu Yata [ctb, cph] (Files at src/third_party/darts_clone (BSD-3
License),
Daisuke Okanohara [ctb, cph] (File src/third_party/esaxx/esa.hxx (MIT
License)),
Yuta Mori [ctb, cph] (File src/third_party/esaxx/sais.hxx (MIT
License)),
Benjamin Heinzerling [ctb, cph] (Files
data/models/nl.wiki.bpe.vs1000.d25.w2v.txt,
data/models/nl.wiki.bpe.vs1000.d25.w2v.bin and
data/models/nl.wiki.bpe.vs1000.model (MIT License))

Repository CRAN

Date/Publication 2026-02-09 14:40:02 UTC

Contents

`BPEembed` *Tokenise and embed text alongside a Sentencepiece and Word2vec model*

Description

Use a sentencepiece model to tokenise text and get the embeddings of these

Usage

```
BPEembed(
  file_sentencepiece = x$file_model,
  file_word2vec = x$glove.bin$file_model,
  x,
  normalize = TRUE
)
```

Arguments

`file_sentencepiece`	the path to the file containing the sentencepiece model
`file_word2vec`	the path to the file containing the word2vec embeddings
`x`	the result of a call to `sentencepiece_download_model`. If this is provided, arguments `file_sentencepiece` and `file_word2vec` will not be used.
`normalize`	passed on to `read.wordvectors` to read in `file_word2vec`. Defaults to `TRUE`.

Value

an object of class BPEembed which is a list with elements

- model: a sentencepiece model as loaded with `sentencepiece_load_model`
- embedding: a matrix with embeddings as loaded with `read.wordvectors`
- dim: the dimension of the embedding
- n: the number of elements in the vocabulary
- file_sentencepiece: the sentencepiece model file
- file_word2vec: the word2vec embedding file

See Also

`predict.BPEembed`, `sentencepiece_load_model`, `sentencepiece_download_model`, `read.wordvectors`

Examples

```
##
## Example loading model from disk
##
folder    <- system.file(package = "sentencepiece", "models")
embedding <- file.path(folder, "nl.wiki.bpe.vs1000.d25.w2v.bin")
model     <- file.path(folder, "nl.wiki.bpe.vs1000.model")
encoder   <- BPEembed(model, embedding)

## Do tokenisation with the sentencepiece model + embed these
txt    <- c("De eigendomsoverdracht aan de deelstaten is ingewikkeld.",
            "On est d'accord sur le prix de la biere?")
values <- predict(encoder, txt, type = "encode")
str(values)
values

txt <- rownames(values[[1]])
predict(encoder, txt, type = "decode")
txt <- lapply(values, FUN = rownames)
predict(encoder, txt, type = "decode")
```

`BPEembedder` *Build a BPEembed model containing a Sentencepiece and Word2vec model*

Description

Build a sentencepiece model on text and build a matching word2vec model on the sentencepiece vocabulary

Usage

```
BPEembedder(
  x,
  tokenizer = c("bpe", "char", "unigram", "word"),
  args = list(vocab_size = 8000, coverage = 0.9999),
  ...
)
```

Arguments

`x`	a data.frame with columns doc_id and text
`tokenizer`	character string with the type of sentencepiece tokenizer. Either 'bpe', 'char', 'unigram' or 'word' for Byte Pair Encoding, Character level encoding, Unigram encoding or pretokenised word encoding. Defaults to 'bpe' (Byte Pair Encoding). Passed on to `sentencepiece`
`args`	a list of arguments passed on to `sentencepiece`
`...`	arguments passed on to `word2vec` for training a word2vec model

Value

an object of class BPEembed which is a list with elements

- model: a sentencepiece model as loaded with sentencepiece_load_model
- embedding: a matrix with embeddings as loaded with read.wordvectors
- dim: the dimension of the embedding
- n: the number of elements in the vocabulary
- file_sentencepiece: the sentencepiece model file
- file_word2vec: the word2vec embedding file

See Also

sentencepiece, word2vec, predict.BPEembed

Examples

```
library(tokenizers.bpe)
data(belgium_parliament, package = "tokenizers.bpe")
x     <- subset(belgium_parliament, language %in% "dutch")
model <- BPEembedder(x, tokenizer = "bpe", args = list(vocab_size = 1000),
                     type = "cbow", dim = 20, iter = 10)
model

txt    <- c("De eigendomsoverdracht aan de deelstaten is ingewikkeld.")
values <- predict(model, txt, type = "encode")
```

---

predict.BPEembed *Encode and Decode alongside a BPEembed model*

Description

Use the sentencepiece model to either

- encode: tokenise and embed text
- decode: get the untokenised text back of tokenised data
- tokenize: only tokenize alongside the sentencepiece model

Usage

```
## S3 method for class 'BPEembed'
predict(object, newdata, type = c("encode", "decode", "tokenize"), ...)
```

Arguments

object	an object of class BPEembed as returned by BPEembed
newdata	a character vector of text to encode or a character vector of encoded tokens to decode or a list of those
type	character string, either 'encode', 'decode' or 'tokenize'
...	further arguments passed on to the methods

Value

- in case type is set to 'encode': a list of matrices containing embeddings of the text which is tokenised with sentencepiece_encode
- in case type is set to 'decode': a character vector of decoded text as returned by sentencepiece_decode
- in case type is set to 'tokenize': a tokenised sentencepiece_encode

See Also

BPEembed, sentencepiece_decode, sentencepiece_encode

Examples

```
embedding <- system.file(package = "sentencepiece", "models",
                         "nl.wiki.bpe.vs1000.d25.w2v.bin")
model     <- system.file(package = "sentencepiece", "models",
                         "nl.wiki.bpe.vs1000.model")
encoder   <- BPEembed(model, embedding)

txt      <- c("De eigendomsoverdracht aan de deelstaten is ingewikkeld.",
              "On est d'accord sur le prix de la biere?")
values   <- predict(encoder, txt, type = "encode")
str(values)
values

txt <- rownames(values[[1]])
predict(encoder, txt, type = "decode")
txt <- lapply(values, FUN = rownames)
predict(encoder, txt, type = "decode")
txt <- c("De eigendomsoverdracht aan de deelstaten is ingewikkeld.",
         "On est d'accord sur le prix de la biere?")
predict(encoder, txt, type = "tokenize", "subwords")
predict(encoder, txt, type = "tokenize", "ids")
```

---

read_word2vec *Read a word2vec embedding file*

Description

Read a word2vec embedding file as a dense matrix. This uses read.wordvectors from the word2vec package.

Usage

```
read_word2vec(
  x,
  type = c("txt", "bin"),
  n = .Machine$integer.max,
  encoding = "UTF-8",
  normalize = TRUE
)
```

Arguments

x	path to the file
type	either 'bin' or 'txt' indicating the `file` is a binary file or a text file
n	integer, indicating to limit the number of words to read in. Defaults to reading all words.
encoding	encoding to be assumed for the words. Defaults to 'UTF-8'
normalize	logical indicating to normalize the embeddings by dividing by the factor (sqrt(sum(x . x) / length(x))). Defaults to TRUE.

Value

a matrix with one row per token containing the embedding of the token

See Also

`read.wordvectors`

Examples

```
folder    <- system.file(package = "sentencepiece", "models")
embedding <- file.path(folder, "nl.wiki.bpe.vs1000.d25.w2v.bin")
embedding <- read_word2vec(embedding, type = "bin")
head(embedding)
embedding <- file.path(folder, "nl.wiki.bpe.vs1000.d25.w2v.txt")
embedding <- read_word2vec(embedding, type = "txt")
head(embedding, n = 10)
```

---

sentencepiece *Construct a Sentencepiece model*

Description

Construct a Sentencepiece model on text.

Usage

```
sentencepiece(
  x,
  type = c("bpe", "char", "unigram", "word"),
  vocab_size = 8000,
  coverage = 0.9999,
  model_prefix = "sentencepiece",
  model_dir = tempdir(),
  threads = 1L,
  args,
  verbose = FALSE
)
```

Arguments

x	a character vector of path(s) to the text files containing training data
type	either one of 'bpe', 'char', 'unigram' or 'word' for Byte Pair Encoding, Character level encoding, Unigram encoding or pretokenised word encoding. Defaults to 'bpe' (Byte Pair Encoding).
vocab_size	integer indicating the number of tokens in the final vocabulary. Defaults to 8000.
coverage	fraction of characters covered by the model. Must be in the range [0, 1]. A good value to use is about 0.9999.
model_prefix	character string with the name of the model. Defaults to 'sentencepiece'. When executing the function 2 files will be created in the directory specified by model_dir, namely sentencepiece.model with the model and sentencepiece.vocab containing the vocabulary of the model. You can change the name of the model by providing the model_prefix argument.
model_dir	directory where the model will be saved. Defaults to the temporary directory (tempdir())
threads	integer indicating number of threads to use when building the model
args	character string with arguments passed on to sentencepiece::SentencePieceTrainer::Train (for expert use only)
verbose	logical indicating to show progress of sentencepiece training. Defaults to FALSE.

Value

an object of class sentencepiece which is defined at sentencepiece_load_model

See Also

sentencepiece_load_model

Examples

```
library(tokenizers.bpe)
data(belgium_parliament, package = "tokenizers.bpe")
path   <- "traindata.txt"
```

```
folder <- getwd()

writeLines(belgium_parliament$text, con = path)


model <- sentencepiece(path, type = "char",
                       model_dir = folder, verbose = TRUE)
model <- sentencepiece(path, type = "unigram", vocab_size = 20000,
                       model_dir = folder, verbose = TRUE)
model <- sentencepiece(path, type = "bpe", vocab_size = 4000,
                       model_dir = folder, verbose = TRUE)

txt <- c("De eigendomsoverdracht aan de deelstaten is ingewikkeld.",
         "On est d'accord sur le prix de la biere?")
sentencepiece_encode(model, x = txt, type = "subwords")
sentencepiece_encode(model, x = txt, type = "ids")


model <- sentencepiece_load_model(file.path(folder, "sentencepiece.model"))
sentencepiece_encode(model, x = txt, type = "subwords")
sentencepiece_encode(model, x = txt, type = "ids")
```

`sentencepiece_decode` *Decode encoded sequences back to text*

Description

Decode a sequence of Sentencepiece ids into text again

Usage

```
sentencepiece_decode(model, x)
```

Arguments

`model`	an object of class `sentencepiece` as returned by `sentencepiece_load_model` or `sentencepiece`
`x`	an integer vector of Sentencepiece id's or a list of these

Value

a character vector of detokenised text or if you encoded with `nbest`, a list of these

Examples

```
model <- system.file(package = "sentencepiece", "models", "nl-fr-dekamer.model")
model <- sentencepiece_load_model(file = model)

txt <- c("De eigendomsoverdracht aan de deelstaten is ingewikkeld.",
         "On est d'accord sur le prix de la biere?")

x <- sentencepiece_encode(model, x = txt, type = "subwords")
sentencepiece_decode(model, x)
x <- sentencepiece_encode(model, x = txt, type = "ids")
sentencepiece_decode(model, x)

model <- system.file(package = "sentencepiece", "models",
                     "nl-fr-dekamer-unigram.model")
model <- sentencepiece_load_model(file = model)
x <- sentencepiece_encode(model, x = txt, type = "subwords", nbest = 3)
sentencepiece_decode(model, x)
x <- sentencepiece_encode(model, x = txt, type = "subwords",
                          nbest = 3, alpha = 0.1)
sentencepiece_decode(model, x)
```

---

sentencepiece_download_model

Download a Sentencepiece model

Description

Download pretrained models built on Wikipedia made available at https://bpemb.h-its.org through https://github.com/bheinzerling/bpemb. These models contain Byte Pair Encoded models trained with sentencepiece as well as Glove embeddings of these Byte Pair subwords. Models for 275 languages are available.

Usage

```
sentencepiece_download_model(
  language,
  vocab_size,
  dim,
  model_dir = system.file(package = "sentencepiece", "models")
)
```

Arguments

language	a character string with the language name. This can be either a plain language or a wikipedia shorthand. Possible values can be found by looking at the examples or typing sentencepiece:::.bpemb$languages If you provide multi it downloads the multilingual model available at https://bpemb.h-its.org/multi/

vocab_size	integer indicating the number of tokens in the final vocabulary. Defaults to 5000. Possible values depend on the language. To inspect possible values, type sentencepiece:::.bpemb$vocab_sizes and look to your language of your choice.
dim	dimension of the embedding. Either 25, 50, 100, 200 or 300.
model_dir	path to the location where the model will be downloaded to. Defaults to system.file(package = "sentencepiece", "models").

Value

a list with elements

- language: the provided language
- wikicode: the wikipedia code of the provided language
- file_model: the path to the downloaded Sentencepiece model
- url: the url where the Sentencepiece model was fetched from
- download_failed: logical, indicating if the download failed
- download_message: a character string with possible download failure information
- glove: a list with elements file_model, url, download_failed and download_message indicating the path to the Glove embeddings in txt format. Only present if the dim argument is provided in the function. Otherwise the embeddings will not be downloaded
- glove.bin: a list with elements file_model, url, download_failed and download_message indicating the path to the Glove embeddings in bin format. Only present if the dim argument is provided in the function. Otherwise the embeddings will not be downloaded

See Also

sentencepiece_load_model

Examples

```
path <- getwd()



##
## Download only the tokeniser model
##
dl <- sentencepiece_download_model("Russian", vocab_size = 50000, model_dir = path)
dl <- sentencepiece_download_model("English", vocab_size = 100000, model_dir = path)
dl <- sentencepiece_download_model("French", vocab_size = 25000, model_dir = path)
dl <- sentencepiece_download_model("multi", vocab_size = 320000, model_dir = path)
dl <- sentencepiece_download_model("Vlaams", vocab_size = 1000, model_dir = path)
dl <- sentencepiece_download_model("Dutch", vocab_size = 25000, model_dir = path)
dl <- sentencepiece_download_model("nl", vocab_size = 25000, model_dir = path)
str(dl)
model     <- sentencepiece_load_model(dl$file_model)

##
```

```
## Download the tokeniser model + Glove embeddings of Byte Pairs
##
dl <- sentencepiece_download_model("nl", vocab_size = 1000, dim = 50, model_dir = path)
str(dl)
model     <- sentencepiece_load_model(dl$file_model)
embedding <- read_word2vec(dl$glove$file_model)



dl <- sentencepiece_download_model("nl", vocab_size = 1000, dim = 25,
                                   model_dir = tempdir())
str(dl)
```

`sentencepiece_encode` *Tokenise text alongside a Sentencepiece model*

Description

Tokenise text alongside a Sentencepiece model

Usage

```
sentencepiece_encode(
  model,
  x,
  type = c("subwords", "ids"),
  nbest = -1L,
  alpha = 0.1
)
```

Arguments

`model`	an object of class `sentencepiece` as returned by `sentencepiece_load_model` or `sentencepiece`
`x`	a character vector of text (in UTF-8 Encoding)
`type`	a character string, either 'subwords' or 'ids' to get the subwords or the corresponding ids of these subwords as defined in the vocabulary of the model. Defaults to 'subwords'.
`nbest`	integer indicating the number of segmentations to extract. See the details. The argument is not used if you do not provide a value for it.
`alpha`	smoothing parameter to perform subword regularisation. Typical values are 0.1, 0.2 or 0.5. See the details. The argument is not used if you do not provide a value for it or do not provide a value for `nbest`.

Details

If you specify alpha to perform subword regularisation, keep in mind the following.
When alpha is 0.0, one segmentation is uniformly sampled from the nbest or lattice. The best Viterbi segmentation is more likely sampled when setting larger alpha values like 0.1.

- If you provide a positive value for nbest, approximately samples one segmentation from nbest candidates.
- If you provide a negative value for nbest, samples one segmentation from the hypotheses (Lattice) according to the generation probabilities using forward-filtering and backward-sampling algorithm.

nbest and alpha correspond respectively to the parameter l and in alpha in the paper https://arxiv.org/abs/1804.10959 where (nbest < 0 means l = infinity).

If the model is a BPE model, alpha is the merge probability p explained in https://arxiv.org/abs/1910.13267. In a BPE model, nbest-based sampling is not supported so the nbest parameter is ignored although it still needs to be provided if you want to make use of alpha.

Value

a list with tokenised text, one for each element of x unless you provide nbest without providing alpha in which case the result is a list of list of nbest tokenised texts

Examples

```
model <- system.file(package = "sentencepiece", "models", "nl-fr-dekamer.model")
model <- sentencepiece_load_model(file = model)

txt <- c("De eigendomsoverdracht aan de deelstaten is ingewikkeld.",
         "On est d'accord sur le prix de la biere?")
sentencepiece_encode(model, x = txt, type = "subwords")
sentencepiece_encode(model, x = txt, type = "ids")

## Examples using subword regularisation
model <- system.file(package = "sentencepiece", "models", "nl-fr-dekamer-unigram.model")
model <- sentencepiece_load_model(file = model)

txt <- c("Goed zo",
         "On est d'accord")
sentencepiece_encode(model, x = txt, type = "subwords", nbest = 4)
sentencepiece_encode(model, x = txt, type = "ids", nbest = 4)
sentencepiece_encode(model, x = txt, type = "subwords", nbest = 2)
sentencepiece_encode(model, x = txt, type = "ids", nbest = 2)
sentencepiece_encode(model, x = txt, type = "subwords", nbest = 1)
sentencepiece_encode(model, x = txt, type = "ids", nbest = 1)
sentencepiece_encode(model, x = txt, type = "subwords", nbest = 4, alpha = 0.1)
sentencepiece_encode(model, x = txt, type = "ids", nbest = 4, alpha = 0.1)
sentencepiece_encode(model, x = txt, type = "subwords", nbest = -1, alpha = 0.1)
sentencepiece_encode(model, x = txt, type = "ids", nbest = -1, alpha = 0.1)
sentencepiece_encode(model, x = txt, type = "subwords", nbest = -1, alpha = 0)
```

```
sentencepiece_encode(model, x = txt, type = "ids", nbest = -1, alpha = 0)
```

`sentencepiece_load_model` *Load a Sentencepiece model*

Description

Load a Sentencepiece model which either was trained with `sentencepiece` or which you have found in the wild.

Usage

```
sentencepiece_load_model(file = "sentencepiece.model")
```

Arguments

`file`	path to the file containing the Sentencepiece model

Value

an object of class `sentencepiece` which is a list with elements

- model: an Rcpp pointer to the model
- model_path: the path to the model
- vocab_size: the size of the Sentencepiece vocabulary
- vocabulary: the Sentencepiece vocabulary which is a data.frame with columns id and subword

Examples

```
model <- system.file(package = "sentencepiece", "models", "nl-fr-dekamer.model")
model <- sentencepiece_load_model(file = model)

txt <- c("De eigendomsoverdracht aan de deelstaten is ingewikkeld.",
         "On est d'accord sur le prix de la biere?")
sentencepiece_encode(model, x = txt, type = "subwords")
sentencepiece_encode(model, x = txt, type = "ids")
```

txt_remove_ *Remove prefixed underscore*

Description

Remove prefixed underscore unicode character 'LOWER ONE EIGHTH BLOCK' (U+2581)

Usage

```
txt_remove_(x, replacement = "")
```

Arguments

x	a character vector
replacement	character string how to replace the underscore. Defaults to the empty string.

Value

x where the prefixed underscore is removed

Examples

```
x <- c("\u2581word", "hello", "_regularunderscore")
x
txt_remove_(x)
```

wordpiece_encode *Wordpiece encoding*

Description

Wordpiece encoding, usefull for BERT-style tokenisation. Experimental version mimicing class WordpieceTokenizer from https://github.com/huggingface/transformers/blob/master/src/transformers/models/bert/tokenization_bert.py

Usage

```
wordpiece_encode(
  x,
  vocabulary = character(),
  type = c("subwords", "ids"),
  unk_token = "[UNK]",
  max_input_chars_per_word = 100L
)
```

Arguments

x	a character vector with text which can be splitted based on white space to obtain words
vocabulary	a character vector of the vocabulary
type	a character string, either 'subwords' or 'ids' to get the subwords or the corresponding ids of these subwords as defined in the vocabulary of the model. Defaults to 'subwords'.
unk_token	character string with a value for a token which is not part of the vocabulary. Defaults to '[UNK]'
max_input_chars_per_word	integer. A word which is longer than this specified number of characters will be set to the unknown token.

Value

a list of subword tokens

Examples

```
wordpiece_encode("unaffable", vocabulary = c("un", "##aff", "##able"))
wordpiece_encode(x = c("unaffable", "unaffableun"),
                 vocabulary = c("un", "##aff", "##able"))
wordpiece_encode(x = c("unaffable", "unaffableun", "unknown territory"),
                 vocabulary = c("un", "##aff", "##able", "##un"))
wordpiece_encode(x = c("unaffable", "unaffableun", "unknown territory"),
                 vocabulary = c("un", "##aff", "##able", "##un"),
                 type = "ids")
```

Index