Package 'piecemaker'

March 3, 2022

Title Tools for Preparing Text for Tokenizers

Version 1.0.1

Description Tokenizers break text into pieces that are more usable by machine learning models. Many tokenizers share some preparation steps. This package provides those shared steps, along with a simple tokenizer.

License Apache License (>= 2)

Encoding UTF-8

RoxygenNote 7.1.2

URL https://github.com/macmillancontentscience/piecemaker

BugReports https://github.com/macmillancontentscience/piecemaker/issues

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Imports rlang (>= 0.4.2), stringi, stringr

Depends R (>= 2.10)

NeedsCompilation no

Author Jon Harmon [aut, cre] (<https://orcid.org/0000-0003-4781-4346>), Jonathan Bratt [aut] (<https://orcid.org/0000-0003-2859-0076>), Bedford Freeman & Worth Pub Grp LLC DBA Macmillan Learning [cph]

Maintainer Jon Harmon <jonthegeek@gmail.com>

Repository CRAN

Date/Publication 2022-03-03 15:50:06 UTC

R topics documented:

prepare_and_tokenize 2
prepare_text 3
remove_control_characters 4
remove_diacritics 5
remove_replacement_characters 5
space_cjk 6

space_punctuation 7
squish_whitespace 7
tokenize_space 8
validate_utf8 9

Index **10**

`prepare_and_tokenize` *Split Text on Spaces*

Description

This is an extremely simple tokenizer that simply splits text on spaces. It also optionally applies the cleaning processes from `prepare_text`.

Usage

```
prepare_and_tokenize(text, prepare = TRUE, ...)
```

Arguments

`text`	A character vector to clean.
`prepare`	Logical; should the text be passed through `prepare_text`?
`...`	Arguments passed on to `prepare_text` `squish_whitespace` Logical scalar; squish whitespace characters (using `str_squish`)? `remove_control_characters` Logical scalar; remove control characters? `remove_replacement_characters` Logical scalar; remove the "replacement character", `U-FFFD`? `remove_diacritics` Logical scalar; remove diacritical marks (accents, etc) from characters? `space_cjk` Logical scalar; add spaces around Chinese/Japanese/Korean ideographs? `space_punctuation` Logical scalar; add spaces around punctuation (to make it easier to keep punctuation during tokenization)? `remove_terminal_hyphens` Logical; should hyphens at the end of lines after a word be removed? For example, "un-\nbroken" would become "unbroken". `space_hyphens` Logical; treat hyphens between letters and at the start/end of words as punctuation? Other hyphens are always treated as punctuation. `space_abbreviations` Logical; treat apostrophes between letters as punctuation? Other apostrophes are always treated as punctuation.

Value

The text as a list of character vectors. Each element of each vector is roughly equivalent to a word.

Examples

```
prepare_and_tokenize("This is some text.")
prepare_and_tokenize("This is some text.", space_punctuation = FALSE)
```

prepare_text *Prepare Text for Tokenization*

Description

This function combines the other functions in this package to prepare text for tokenization. The text gets converted to valid UTF-8 (if possible), and then various cleaning functions are applied.

Usage

```
prepare_text(
  text,
  squish_whitespace = TRUE,
  remove_terminal_hyphens = TRUE,
  remove_control_characters = TRUE,
  remove_replacement_characters = TRUE,
  remove_diacritics = TRUE,
  space_cjk = TRUE,
  space_punctuation = TRUE,
  space_hyphens = TRUE,
  space_abbreviations = TRUE
)
```

Arguments

text	A character vector to clean.
squish_whitespace	Logical scalar; squish whitespace characters (using str_squish)?
remove_terminal_hyphens	Logical; should hyphens at the end of lines after a word be removed? For example, "un-\nbroken" would become "unbroken".
remove_control_characters	Logical scalar; remove control characters?
remove_replacement_characters	Logical scalar; remove the "replacement character", U-FFFD?
remove_diacritics	Logical scalar; remove diacritical marks (accents, etc) from characters?
space_cjk	Logical scalar; add spaces around Chinese/Japanese/Korean ideographs?
space_punctuation	Logical scalar; add spaces around punctuation (to make it easier to keep punctuation during tokenization)?
space_hyphens	Logical; treat hyphens between letters and at the start/end of words as punctuation? Other hyphens are always treated as punctuation.
space_abbreviations	Logical; treat apostrophes between letters as punctuation? Other apostrophes are always treated as punctuation.

Value

The character vector, cleaned as specified.

Examples

```
piece1 <- " This is a    \n\nfa\xE7ile\n\n    example.\n"
# Specify encoding so this example behaves the same on all systems.
Encoding(piece1) <- "latin1"
example_text <- paste(
  piece1,
  "It has the bell character, \a, and the replacement character,",
  intToUtf8(65533)
)
prepare_text(example_text)
prepare_text(example_text, squish_whitespace = FALSE)
prepare_text(example_text, remove_control_characters = FALSE)
prepare_text(example_text, remove_replacement_characters = FALSE)
prepare_text(example_text, remove_diacritics = FALSE)
```

---

remove_control_characters

Remove Non-Character Characters

Description

Unicode includes several control codes, such as `U+0000` (NULL, used in null-terminated strings) and `U+000D` (carriage return). This function removes all such characters from text.

Usage

```
remove_control_characters(text)
```

Arguments

`text`	A character vector to clean.

Details

Note: We highly recommend that you first condense all space-like characters (including new lines) before removing control codes. You can easily do so with `str_squish`. We also recommend validating text at the start of any cleaning process using `validate_utf8`.

Value

The character vector without control characters.

Examples

```
remove_control_characters("Line 1\nLine2")
```

remove_diacritics *Remove Diacritical Marks on Characters*

Description

Accent characters and other diacritical marks are often difficult to type, and thus can be missing from text. To normalize the various ways a user might spell a word that should have a diacritical mark, you can convert all such characters to their simpler equivalent character.

Usage

```
remove_diacritics(text)
```

Arguments

text	A character vector to clean.

Value

The character vector with simpler character representations.

Examples

```
# This text can appear differently between machines if we aren't careful, so
# we explicitly encode the desired characters.
sample_text <- "fa\u00e7ile r\u00e9sum\u00e9"
sample_text
remove_diacritics(sample_text)
```

remove_replacement_characters *Remove the Unicode Replacement Character*

Description

The replacement character, U+FFFD, is used to mark characters that could not be loaded. These characters might be a sign of encoding issues, so it is advisable to investigate and try to eliminate any cases in your text, but in the end these characters will almost definitely confuse downstream processes.

Usage

```
remove_replacement_characters(text)
```

Arguments

text	A character vector to clean.

Value

The character vector with replacement characters removed.

Examples

```
remove_replacement_characters(
  paste(
    "The replacement character:",
    intToUtf8(65533)
  )
)
```

---

space_cjk *Add Spaces Around CJK Ideographs*

Description

To tokenize Chinese, Japanese, and Korean (CJK) characters, it's convenient to add spaces around the characters.

Usage

```
space_cjk(text)
```

Arguments

text	A character vector to clean.

Value

A character vector the same length as the input text, with spaces added between ideographs.

Examples

```
to_space <- intToUtf8(13312:13320)
to_space
space_cjk(to_space)
```

space_punctuation *Add Spaces Around Punctuation*

Description

To keep punctuation during tokenization, it's convenient to add spacing around punctuation. This function does that, with options to keep certain types of punctuation together as part of the word.

Usage

```
space_punctuation(text, space_hyphens = TRUE, space_abbreviations = TRUE)
```

Arguments

text	A character vector to clean.
space_hyphens	Logical; treat hyphens between letters and at the start/end of words as punctuation? Other hyphens are always treated as punctuation.
space_abbreviations	Logical; treat apostrophes between letters as punctuation? Other apostrophes are always treated as punctuation.

Value

A character vector the same length as the input text, with spaces added around punctuation characters.

Examples

```
to_space <- "This is some 'gosh-darn' $5 text. Isn't it lovely?"
to_space
space_punctuation(to_space)
space_punctuation(to_space, space_hyphens = FALSE)
space_punctuation(to_space, space_abbreviations = FALSE)
```

squish_whitespace *Remove Extra Whitespace*

Description

This function is mostly a wrapper around str_squish, with the additional option to remove hyphens at the ends of lines.

Usage

```
squish_whitespace(text, remove_terminal_hyphens = TRUE)
```

Arguments

`text`	A character vector to clean.
`remove_terminal_hyphens`	Logical; should hyphens at the end of lines after a word be removed? For example, "un-\nbroken" would become "unbroken".

Value

The character vector with spacing at the start and end removed, and with internal spacing reduced to a single space character each.

Examples

```
sample_text <- "This  had many space char-\\nacters."
squish_whitespace(sample_text)
```

`tokenize_space` *Break Text at Spaces*

Description

This is an extremely simple tokenizer, breaking only and exactly on the space character. This tokenizer is intended to work in tandem with `prepare_text`, so that spaces are cleaned up and inserted as necessary before the tokenizer runs. This function and `prepare_text` are combined together in `prepare_and_tokenize`.

Usage

```
tokenize_space(text)
```

Arguments

`text`	A character vector to clean.

Value

The text as a list of character vectors (one vector per element of `text`). Each element of each vector is roughly equivalent to a word.

Examples

```
tokenize_space("This is some text.")
```

validate_utf8 *Clean Up Text to UTF-8*

Description

Text cleaning works best if the encoding is known. This function attempts to convert text to UTF-8 encoding, and provides an informative error if that is not possible.

Usage

```
validate_utf8(text)
```

Arguments

text	A character vector to clean.

Value

The text with formal UTF-8 encoding, if possible.

Examples

```
text <- "fa\xE7ile"
# Specify the encoding so the example is the same on all systems.
Encoding(text) <- "latin1"
validate_utf8(text)
```

Index

prepare_and_tokenize, 2, *8*
prepare_text, *2*, 3, *8*

remove_control_characters, 4
remove_diacritics, 5
remove_replacement_characters, 5

space_cjk, 6
space_punctuation, 7
squish_whitespace, 7
str_squish, *2–4*, *7*

tokenize_space, 8

validate_utf8, *4*, 9