Package 'diffmatchpatch'

July 22, 2025

Type Package

Title String Diff, Match, and Patch Utilities

Version 0.1.0

Date 2021-04-10

Copyright Google Inc., Neil Fraser, Mike Slemmer, Sergey Nozhenko, Christian Leutloff, Colin Rundel

Description A wrapper for Google's 'diff-match-patch' library. It provides basic tools for computing diffs, finding fuzzy matches, and constructing / applying patches to strings.

Encoding UTF-8

Imports cli, Rcpp

LinkingTo Rcpp

RoxygenNote 7.1.1

License Apache License (>= 2)

URL https://github.com/rundel/diffmatchpatch

BugReports https://github.com/rundel/diffmatchpatch/issues

NeedsCompilation yes

Author Colin Rundel [aut, cre],
Google Inc. [cph] (diff_match_patch.h),
Neil Fraser [cph] (diff_match_patch.h),
Mike Slemmer [cph] (diff_match_patch.h),
Sergey Nozhenko [cph] (diff_match_patch.h),
Christian Leutloff [cph] (diff_match_patch.h)

Maintainer Colin Rundel <rundel@gmail.com>

Repository CRAN

Date/Publication 2021-04-16 07:00:05 UTC

Contents

`diff_make` *Compute diffs between text strings*

Description

The following functions are used to construct or work with diff(s) between text strings. Specifically, `diff_make()` computes the character level differences between the source string (`x`) and destination string (`y`). These diffs can be made more human friendly via a secondary cleaning process via the `cleanup` argument.

Once computed, diffs are represented using `diff_df` data frames, which consist of just two columns: `text` and `op`. Basic convenience functions for pretty printing of these are provided by the package.

The following helper functions are provided:

- `print()` - prints a diff using ANSI colors if available.
- `as.character()` - converts a diff (using ANSI colors if available) to a character vector.
- `diff_levenshtein()` calculates the Levenshtein distance of a diff.
- `diff_to_delta()` converts a diff to a delta string.
- `diff_from_delta()` creates a diff from a source string (`x`) and a `delta` string.
- `diff_to_html()` converts a diff to pretty HTML string.
- `diff_to_patch()` converts a diff to a patch string.
- `diff_text_source()` recovers the source string from a diff.
- `diff_text_dest()` recovers the destination string from a diff.

Usage

```
diff_make(x, y, cleanup = "semantic", checklines = TRUE)

diff_levenshtein(diff)

diff_to_delta(diff)

diff_from_delta(x, delta)

diff_to_html(diff)

diff_to_patch(diff)
```

```
diff_text_source(diff)

diff_text_dest(diff)
```

Arguments

x	The source string
y	The destination string
cleanup	Determines the cleanup method applied to the diffs. Allowed values include: semantic, lossless, efficiency, merge and none. See Details for the behavior of these methods.
checklines	Performance flag - if FALSE, then don't run a line-level diff first to identify the changed areas. If TRUE, run a faster slightly less optimal diff. Default: TRUE.
diff	A diff_df data frame.
delta	A delta string.

Details

Cleanup methods:

- semantic - Reduce the number of edits by eliminating semantically trivial equalities.
- semantic lossless - Look for single edits surrounded on both sides by equalities which can be shifted sideways to align the edit to a word boundary. e.g: The c**at c**ame. -> The **cat **came.
- efficiency - Reduce the number of edits by eliminating operationally trivial equalities.
- merge - Reorder and merge like edit sections. Merge equalities. Any edit section can move as long as it doesn't cross an equality.
- none - Do not apply any cleanup methods to the diffs.

Value

- diff_make() returns a diff_df data frame containing the diffs.
- diff_make() returns the Levenshtein distance as an integer.
- diff_to_delta() returns an character string.
- diff_from_delta() returns a diff_df data frame.
- diff_to_html() returns a character string.
- diff_to_patch() returns a character string.
- diff_text_source() returns a character string.
- diff_text_dest() returns a character string.

Examples

```
(d = diff_make("abcdef", "abchij"))

diff_levenshtein(d)

diff_to_html(d)

diff_text_source(d)

diff_text_dest(d)

diff_to_patch(d)

(delta = diff_to_delta(d))

diff_from_delta("abcdef", delta)
```

`dmp_options` *diffmatchpatch settings*

Description

Allows for examining or setting options that affect the behavior of the diff, match, and patch related functions in this package.

Usage

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

Arguments

`...`	No arguments returns all current options and their values. Character values retrieve a subset of options and the current values. Options can be set, using name = value. However, only the options named below are used. Options can also be passed by giving a single unnamed argument which is a named list.

Details

Available options:

- `diff_timeout` (float) - Number of seconds to map a diff before giving up (0 for infinity).
- `diff_edit_cost` (int) - Cost of an empty edit operation in terms of edit characters.
- `match_threshold` (float) - At what point is no match declared (0.0 = perfection, 1.0 = very loose).
- `match_distance` (int) - How far to search for a match (0 = exact location, 1000+ = broad match). A match this many characters away from the expected location will add 1.0 to the score (0.0 is a perfect match).

- patch_delete_threshold (float) - When deleting a large block of text (over ~64 characters), how close does the contents have to match the expected contents. (0.0 = perfection, 1.0 = very loose). Note that Match_Threshold controls how closely the end points of a delete need to match.
- patch_margin (int) - Chunk size for context length.
- match_max_bits (int) - The number of bits in an int.

Value

When getting options returns a named list of options and their current values, when setting options returns a named list of the previous value(s).

Examples

```
dmp_options()

dmp_options("diff_timeout")

prev = dmp_options(diff_timeout = 5)
prev
```

---

match_find *Fuzzy matching of a text string*

Description

Locate the best instance of pattern in the text near loc using the Bitap algorithm.Returns -1 if no match found. Assumes R's typical 1-based indexing for loc and the returned value.

This algorithm makes use of the match_distance and match_threshold options to determine the match. If these values are not set explicitly via the threshold and distance arguments - their value will use the currently set global option value.

Candidate matches are scored based on: a) the number of spelling differences between the pattern and the text and b) the distance between the candidate match and the expected location.

The match_distance option determines the relative importance of these two metrics.

Usage

```
match_find(text, pattern, loc = 1L, threshold = NULL, distance = NULL)
```

Arguments

text	The text to search.
pattern	The pattern to search for.
loc	The expected location of the pattern.
threshold	Threshold for determining a match (0 - perfect match, 1 - very loose).
distance	Distance from expected location scaling for score penalty.

Value

Index of best match or -1 for no match.

Examples

```
x = "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure
dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt
mollit anim id est laborum."

match_find(x, "Loren Ibsen")
match_find(x, "Loren Ibsen", threshold = 0.1)

match_find(x, "minimum")
match_find(x, "minimum", threshold = 0.4)
```

---

patch_make *Create and apply patches to a text string*

Description

Patches are constructed via `patch_make()` and applied using `patch_apply()`.

Usage

```
patch_make(x, y)

patch_apply(x, patch)
```

Arguments

`x`	The source string
`y`	The destination string
`patch`	A string representation of the patch(es).

Value

`patch_make()` returns a string representation of the patch(es).

- `patch_apply()` returns the patched version of the string `x`, the `matches` attribute contains logical values indicating which patches were successfully applied.

Examples

```
(p = patch_make("abcdef", "abchij"))

patch_apply("abcdef", p)

patch_apply("abc", p)

patch_apply("def", p)

patch_apply("hij", p)
```

Index