NAME

Text::Levenshtein – calculate the Levenshtein edit distance between two strings

SYNOPSIS

```
 use Text::Levenshtein qw(distance);

 print distance("foo","four");
 # prints "2"

 my @words     = qw/ four foo bar /;
 my @distances = distance("foo",@words);

 print "@distances";
 # prints "2 0 3"
```

DESCRIPTION

This module implements the Levenshtein edit distance, which measures the difference between two strings, in terms of the *edit distance*. This distance is the number of substitutions, deletions or insertions ("edits") needed to transform one string into the other one (and vice versa). When two strings have distance 0, they are the same.

To learn more about the Levenshtein metric, have a look at the wikipedia page <http://en.wikipedia.org/wiki/Levenshtein_distance>.

distance()

The simplest usage will take two strings and return the edit distance:

```
 $distance = distance('brown', 'green');
 # returns 3, as 'r' and 'n' don't change
```

Instead of a single second string, you can pass a list of strings. Each string will be compared to the first string passed, and a list of the edit distances returned:

```
 @words     = qw/ green trainee brains /;
 @distances = distances('brown', @words);
 # returns (3, 5, 3)
```

fastdistance()

Previous versions of this module provided an alternative implementation, in the function `fastdistance()`. This function is still provided, for backwards compatibility, but they now run the same function to calculate the edit distance.

Unlike `distance()`, `fastdistance()` only takes two strings, and returns the edit distance between them.

ignore_diacritics

Both the `distance()` and `fastdistance()` functions can take a hashref with optional arguments, as the final argument. At the moment the only option is `ignore_diacritics`. If this is true, then any diacritics are ignored when calculating edit distance. For example, "cafe" and "café" normally have an edit distance of 1, but when diacritics are ignored, the distance will be 0:

```
 use Text::Levenshtein 0.11 qw/ distance /;
 $distance = distance($word1, $word2, {ignore_diacritics => 1});
```

If you turn on this option, then Unicode::Collate will be loaded, and used when comparing characters in the words.

Early version of `Text::Levenshtein` didn't support this version, so you should require version 0.11 or later, as above.

SEE ALSO

There are many different modules on CPAN for calculating the edit distance between two strings. Here's just a selection.

Text::LevenshteinXS and Text::Levenshtein::XS are both versions of the Levenshtein algorithm that require a C compiler, but will be a lot faster than this module.

The Damerau-Levenshtein edit distance is like the Levenshtein distance, but in addition to insertion, deletion and substitution, it also considers the transposition of two adjacent characters to be a single edit. The module Text::Levenshtein::Damerau defaults to using a pure perl implementation, but if you've installed Text::Levenshtein::Damerau::XS then it will be a lot quicker.

Text::WagnerFischer is an implementation of the Wagner-Fischer edit distance, which is similar to the Levenshtein, but applies different weights to each edit type.

Text::Brew is an implementation of the Brew edit distance, which is another algorithm based on edit weights.

Text::Fuzzy provides a number of operations for partial or fuzzy matching of text based on edit distance. Text::Fuzzy::PP is a pure perl implementation of the same interface.

String::Similarity takes two strings and returns a value between 0 (meaning entirely different) and 1 (meaning identical). Apparently based on edit distance.

Text::Dice calculates Dice's coefficient <https://en.wikipedia.org/wiki/Sørensen–Dice_coefficient> for two strings. This formula was originally developed to measure the similarity of two different populations in ecological research.

REPOSITORY

<https://github.com/neilbowers/Text−Levenshtein>

AUTHOR

Dree Mistrut originally wrote this module and released it to CPAN in 2002.

Josh Goldberg then took over maintenance and released versions between 2004 and 2008.

Neil Bowers (NEILB on CPAN) is now maintaining this module. Version 0.07 was a complete rewrite, based on one of the algorithms on the wikipedia page.

COPYRIGHT AND LICENSE