How can I calculate the Levenshtein Distance in R, and what are some examples of its application?

Authored by

stats writer

April 23, 2024

The Levenshtein Distance is a measure of the similarity between two strings, often used in natural language processing and spell checking. In R, the Levenshtein Distance can be calculated using the library "stringdist" which provides functions to measure the distance between two strings. This can be done by using the function "stringdist" with the argument "method = "lv". The resulting distance is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one string into the other.

The Levenshtein Distance has various applications, such as in spell checking algorithms where it helps identify and correct misspelled words. It is also used in DNA sequencing to measure the similarity between two genetic sequences. Additionally, it is used in text mining to identify similar words or phrases in large datasets. Overall, the Levenshtein Distance is a useful tool for comparing and analyzing text data in various fields.

Calculate Levenshtein Distance in R (With Examples)

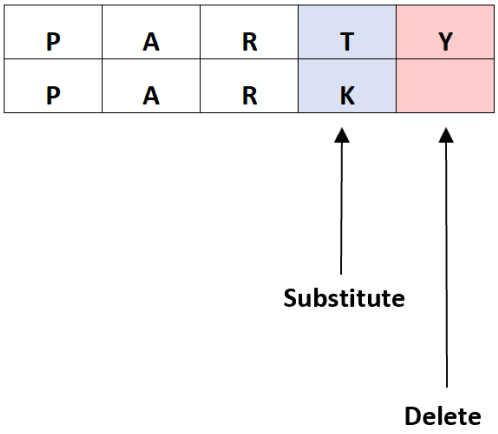
The Levenshtein distance between two strings is the minimum number of single-character edits required to turn one word into the other.

The word "edits" includes substitutions, insertions, and deletions.

For example, suppose we have the following two words:

PARTYPARK

The Levenshtein distance between the two words (i.e. the number of edits we have to make to turn one word into the other) would be 2:



In practice, the Levenshtein distance is used in many different applications including approximate string matching, spell-checking, and natural language processing.

This tutorial explains how to calculate the Levenshtein distance between strings in R by using the function from the stringdist package in R.

This function uses the following basic syntax:

```
#load stringdist package
library(stringdist)

#calculate Levenshtein distance between two strings
stringdist("string1", "string2", method = "lv")
```

Note that this function can calculate many different distance metrics. By specifying method = "lv", we tell the function to calculate the Levenshtein distance.

Example 1: Levenshtein Distance Between Two Strings

The following code shows how to calculate the Levenshtein distance between the two strings "party" and "park" using the stringdist() function:

```
#load stringdist package
library(stringdist)

#calculate Levenshtein distance between two strings
stringdist('party', 'park', method = 'lv')

2
```

The Levenshtein distance turns out to be 2.

Example 2: Levenshtein Distance Between Two Vectors

The following code shows how to calculate the Levenshtein distance between every pairwise combination of strings in two different vectors:

```
#load stringdist package
```

```
library(stringdist)

#define vectors
a <- c('Mavs', 'Spurs', 'Lakers', 'Cavs')
b <- c('Rockets', 'Pacers', 'Warriors', 'Celtics')

#calculate Levenshtein distance between two vectors
stringdist(a, b, method='lv')

6 4 5 5
```

The way to interpret the output is as follows:

The Levenshtein distance between 'Mavs' and 'Rockets' is 6.The Levenshtein distance between 'Spurs' and 'Pacers' is 4.The Levenshtein distance between 'Lakers' and 'Warriors' is 5.The Levenshtein distance between 'Cavs' and 'Celtics' is 5.

Example 3: Levenshtein Distance Between Data Frame Columns

The following code shows how to calculate the Levenshtein distance between every pairwise combination of strings in two different columns of a data frame:

```
#load stringdist package
library(stringdist)

#define data
data <- data.frame(a = c('Mavs', 'Spurs', 'Lakers', 'Cavs'),
b = c('Rockets', 'Pacers', 'Warriors', 'Celtics'))

#calculate Levenshtein distance
stringdist(data$a, data$b, method='lv')

6 4 5 5
```

We could then append the Levenshtein distance as a new column in the data frame if we'd like:

```
#save Levenshtein distance as vector
lev <- stringdist(data$a, data$b, method='lv')

#append Levenshtein distance as new column
data$lev <- lev

#view data frame
data

a b lev
1 Mavs Rockets 6
```

2 Spurs Pacers 4

3 Lakers Warriors 5

4 Cavs Celtics 5