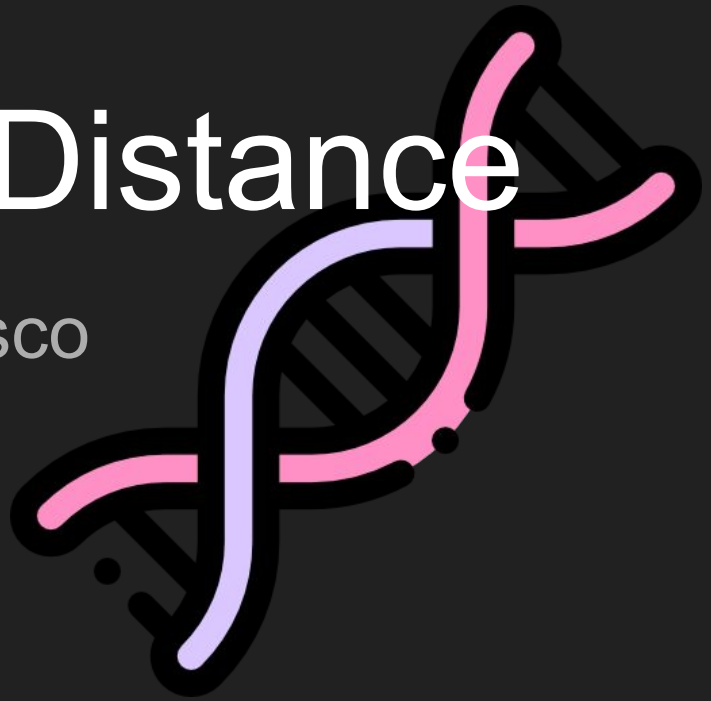
Levenshtein Distance

Nick Sebasco



Definition

The Levenshtein distance, also known as the minimum edit distance is a way of assessing how different two strings are.

Strings are a data structure (โครงสร้างข้อมูล) defined as an order collection of characters. In many programming languages the sequence of characters is enclosed with quotes.

Actual	OCR
1AY4DF	LAYADF

Levenshtein Distance	2

Motivation

In euclidean space there are straight forward distance metrics that look at how similar the components making up two vectors are. For instance, the **Manhattan** distance:

$D_{manhattan} = \Sigma |x_i - y_i|$

x = <1, 2, 0>
y = <0,0,-1>

$D_{manhattan} = |1 - 0| + | 2 - 0| + |0 + 1| = 4$

It is tempting to try a similar strategy for strings. Let's compute a string distance which adds 1 to the distance every time two characters at the same position are different and a 0 each time they are the same. Clearly more similar strings will have smaller distances.

$D_{string} = \Sigma$ 1 if $x_i = y_i$ else 0

u = "กาม"
x = "กออ"
y = "าม"

D_{ux} = ก=ก + า=อ + อ=ม + อ=_ = 3
D_{uy} = า=ก + ม=า + า=ม = 3

Is that the best we can do?

The string distance metric in the last example seems to miss the fact that **y** and **u** have more similar characters that **x** and **u**. Also **y** is the same length as **u**. Thus **y** and **u** appear to be more similar. If we consider the more general question of transforming one string into another and restrict ourselves to the operations **add**, **delete**, and **insert,** then we can create a more robust string similarity metric.

String	Operation	Position	Result
ถนน	delete	1	นน
นก	Insert,insert,insert	1,2,3	รังนก
มะนาว	edit	3	มะตาว

Levenshtein Distance

u = “กาม”

x = “กออ”

y = “าม”

D_{ux} = 1) Edit **อ** ➡ า 2) Edit **อ** ➡ **ม** 3) Delete **อ** = 3

D_{uy} = 1) Insert **ก** = กามา 2) Delete **า** = กาม = **2**

$D_{uy} < D_{ux}$ ⟹ **u** more similar to **y**.

Algorithm

1. Given two strings **u** and **v** of length **m** and **n** respectively. Initialize an **m+1**x**n+1** matrix **M**.
2. Let $\mathbf{r}_i$ be the $\mathbf{i}^{th}$ row of **M** and $\mathbf{c}_i$ be the $\mathbf{i}^{th}$ column of M. Then $\mathbf{r}_1$= [0, …, **m**] and

   $\mathbf{c}_1^T$ = [0, …, **n**]

3. Iterate over **u** from **i** = 1 … **m** and iterate over **v** from **j** = 1 … **n**.
4. if **u[i]** = **v[j]** then cost = 0 else **u[i]** =/= **v[j]** then cost = 1
5. Set cell **M[i, j]** of the matrix equal to the minimum of:

   I. The cell (one row above): **M[i-1,j]** + 1.

   II. The cell (one column to the left) 1: **M[i,j-1]** + 1.

   III. The cell (one row above and one column to the left): **M[i-1,j-1]** + cost.

6. After the matrix has been filled, the final value in the matrix will be the distance!

bed

lead

		l	e	a	d
	0	1	2	3	4
b	1	1	2	3	4
e	2	2	1	2	3
d	3	3	2	2	**2**

Figure: *Computing the Levenshtein distance*

Python Implementation

```python
#Author: Nick Sebasco
#Date: 8/23

import numpy as np

def lev(s1: str, s2: str) -> np.matrix:
    '''AKA: Edit distance
    levenshtein distance between two sequences s1, s2 is the minimum number of single
    element (insert/ delete/ substitution) required to transform s1 -> s2.
    '''
    # 1. initialize matrix
    T = np.zeros((len(s1) + 1, len(s2) + 1))
    for i in range(max(len(s1)+1, len(s2)+1)):
        if i < len(s1)+1:
            T[i, 0] = i
        if i < len(s2)+1:
            T[0, i] = i
    # 2.
    for i in range(len(s1)):
        for j in range(len(s2)):
            T[i + 1, j + 1] = min(T[i, j + 1], T[i + 1, j], T[i, j]) + 1 if s1[i] != s2[j] else T[i, j]
    return T

print(lev("bed","lead"))
```

Application: Covid-19 Variants

The string alpha represents a random snippet of RNA base pairs from the Covid-19 alpha variant, see the reference below for a link to the full genome [1]. Let $mutant_1$ and $mutant_2$ represent disparate mutations of the alpha variant. Use the levenshtein distance to calculate which mutation is most similar.

Variant	Base Pairs
alpha	attaaaggtt tataccttcc
$mutant_1$	caggatagctt atacctaacc
$mutant_2$	ctaaaagtt tatagttcc

[1] Covid-19 Alpha variant genome: https://www.ncbi.nlm.nih.gov/nuccore/MN908947.3