Distance between words

Mathematics for Sustainable Development

Year 2021-2022

By Szekelyhidi Deborah and Groșan-Cerneșteanu Alexia

10th Grade at Colegiul Național “Mihai Eminescu” Satu Mare

Coordinating teachers: Marciuc Daly and Șandor Nicoleta

I. The text of the problem

“How does my spell checker propose a list of words when I make a spelling mistake?”

Spell Checking

- A spell checker works by searching for a given string in its dictionary of known strings.
- Any string not found in the dictionary is deemed an alleged misspelling.
- To correct a misspelling, the spell checker assumes that your misspelling must be a mutation of one of the strings in its dictionary. To suggest a correction, the spell checker searches its dictionary for strings similar to the misspelling. It then orders the potential corrections by their similarity and likelihood score. The known string nearest to your misspelling is the suggested correction.

II. Approach

After understanding the requirement of the chosen topic, we decided to make use of an algorithm that would facilitate our task. In order to accomplish that, we had to do some manual calculations and approach different methods, that would all lead to the same conclusion.

III. Work method

a. Basic intuition

Your instincts are correct.

Set similarity

- You intuitively know that the strings *tomatoes* and *potatoes* are similar. So are *potato* and *potatoes*. The question is – how similar? How do you measure similarity? Is the string *potatoes* closer to *potato* or to *tomatoes*?

- Consider the strings below, which are expressed as sets.

- *Problem:* compare set A (potatoes) to set B (potato) and set C (tomatoes).

Set	String	Elements	Element Count
A	potatoes	{p, o, t, a, t, o, e, s}	8
B	potato	{p, o, t, a, t, o}	6
C	tomatoes	{t, o, m, a, t, o, s}	7

Fig. 1

- The more elements that sets A and B share, the more alike they must be.

- Your measurement is expressed as: Counting (A intersection B)

  - A (potatoes) and B (potato) share six elements.

  - A (potatoes) and C (tomatoes) also share six elements.

- This simple string similarity measure suggests that *potatoes* is equally similar to both *potato* and *tomatoes*, but that is not true. This means that this simple similarity measure is unfortunately not discriminating enough.

b. The Jaccard Coefficient

It calculates the fraction of elements found in both sets.

The similarity between two sets A & B is given by:

- **Count(A intersect B) / Count(A union B)**
- The set of all unique elements in (A, B) => (A union B).

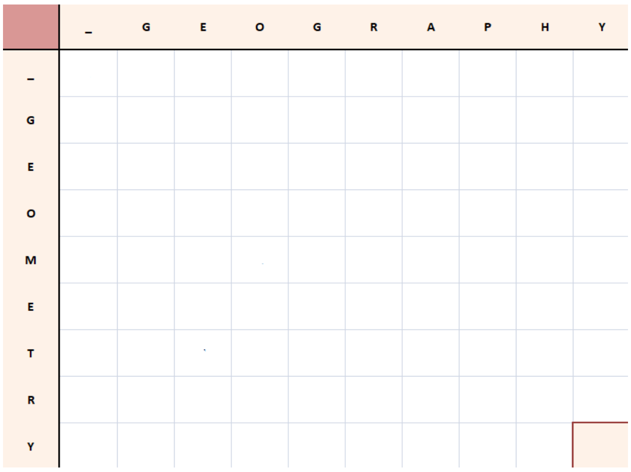
A (Potatoes)	Count (Intersection)	Count (Union)	Score
B (Potato)	6	8	6/8=0.75
C (Tomatoes)	6	9	6/9=0.66

Fig. 2

- The set of common elements in (A, B) => (A intersection B**)**
- The highest result of the fractions will represent the most probable word to be suggested by the spell checker.

c. The Levenshtein Distance

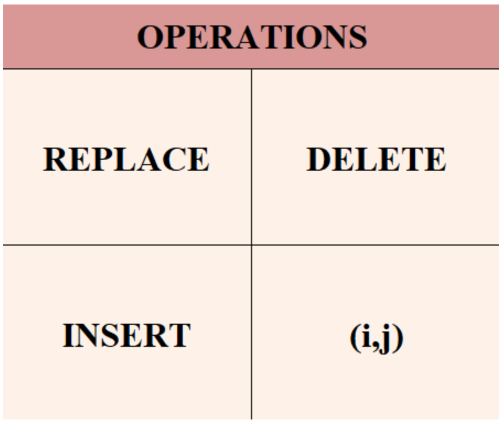
In order to compare two words, they have to be placed into a matrix, so as to check each possible cross-section. The word that needs to be converted in the horizontal direction, and the one that is converted into, in the vertical direction.



	_	G	E	O	G	R	A	P	H	Y
_										
G										
E										
O										
M										
E										
T										
R										
Y										

Fig. 3

To complete this table three types of edits are allowed. (Fig. 4)

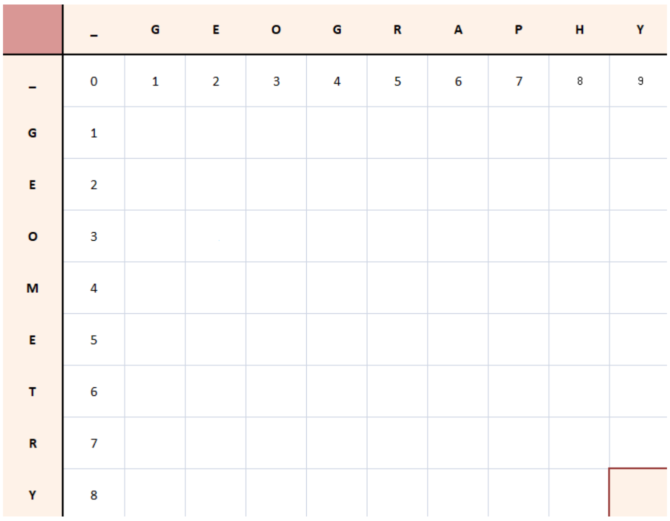


OPERATIONS	
REPLACE	DELETE
INSERT	(i,j)

Fig. 4

To apply an edit the digits from the left, from above and the oblique one that points to the left upper corner that surrounds the empty cell will be taken into consideration. The edits work under the following rules:

- for the first horizontal/vertical lines of cells these will be numerotated with the values from 0-to how many letters the respective word has; (Fig. 5)



	_	G	E	O	G	R	A	P	H	Y
_	0	1	2	3	4	5	6	7	8	9
G	1									
E	2									
O	3									
M	4									
E	5									
T	6									
R	7									
Y	8									

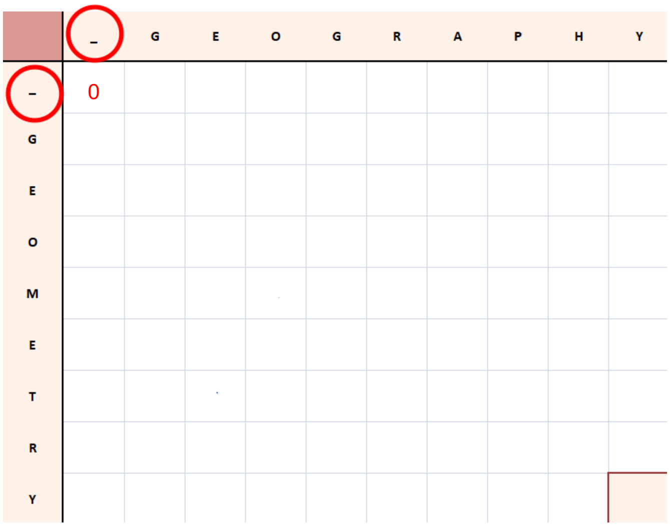
Fig. 5

- if the values of the three digits mentioned above are all different from each other then the lowest number of the three *+1* will represent the new number;

- if the values from the left and the one from above are equal and also the characters of the two words are also the same, than the value from the oblique cell will be copied;

- if the values from the left and the one from above are equal but the characters of the two words are different, then the value from the oblique cell *+1* will be inserted.

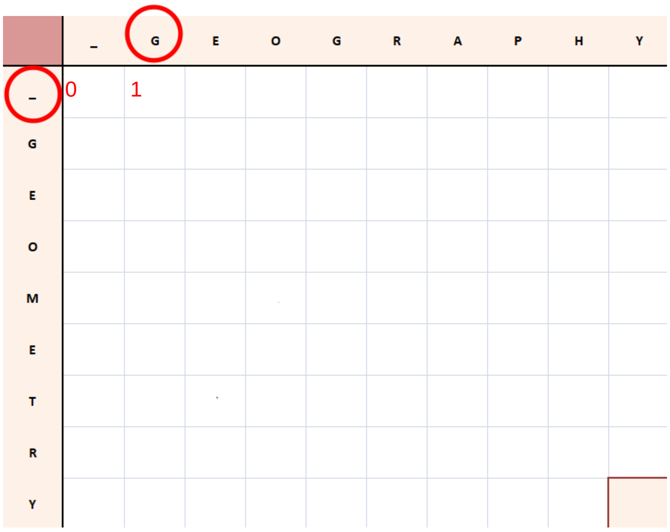
Starting off with two empty strings, to convert an empty string into an empty string, zero operations are needed, so *0* will be written in the cell. (Fig. 6)



	_	G	E	O	G	R	A	P	H	Y
_	0									
G										
E										
O										
M										
E										
T										
R										
Y										

Fig. 6

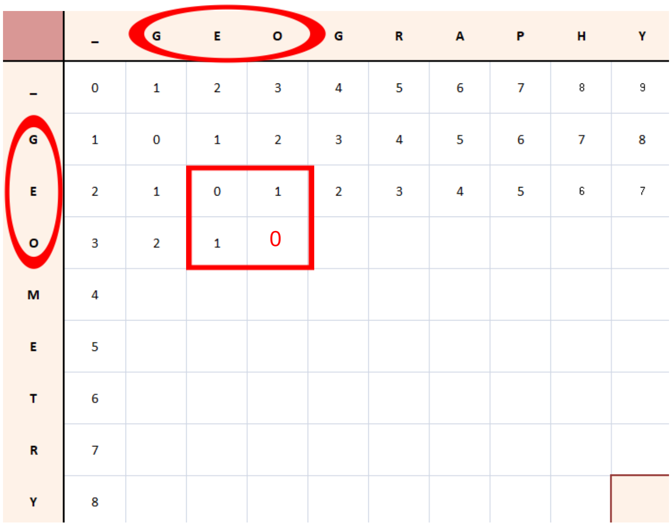
Then, to convert the letter “G” into an empty string, a deletion is requaried so *1* will be written in the corresponding cell. (Fig. 7)



	_	G	E	O	G	R	A	P	H	Y
_	0	1								
G										
E										
O										
M										
E										
T										
R										
Y										

Fig. 7

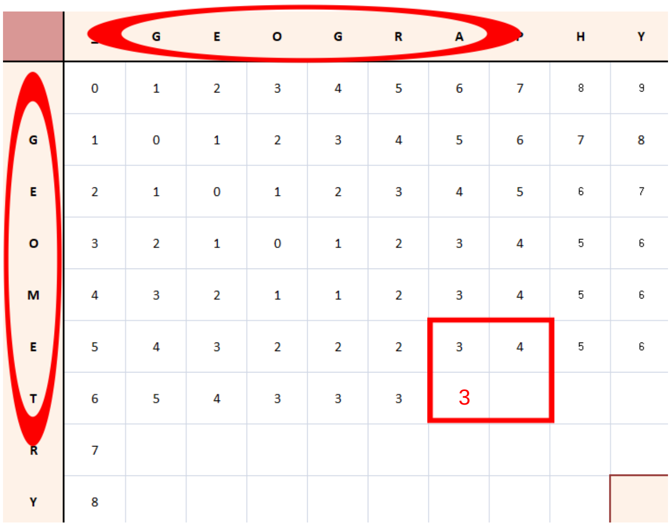
Moving forward, working with the letters “GE”, converting them into an empty string will again require deletion, the only difference is that this time instead of one operation there will be two. Similarly, this rule is applied to all of the letters on the first horizontal and vertical string. (Fig. 5)



	_	G	E	O	G	R	A	P	H	Y
_	0	1	2	3	4	5	6	7	8	9
G	1	0	1	2	3	4	5	6	7	8
E	2	1	0	1	2	3	4	5	6	7
O	3	2	1	0						
M	4									
E	5									
T	6									
R	7									
Y	8									

Fig. 8

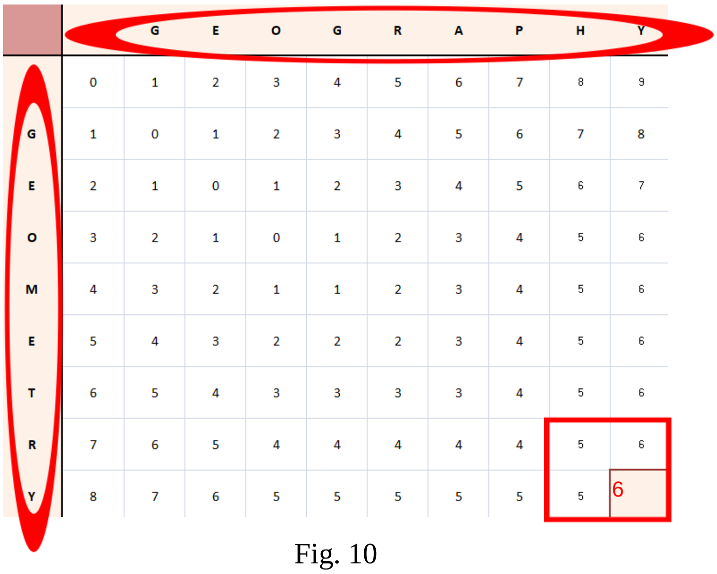
It is noticeable that in order to convert the horizontal string “GEO” into the vertical one “GEO” there are no edist required, so *0* will be inserted in the cell. (Fig. 8)



	_	G	E	O	G	R	A	P	H	Y
_	0	1	2	3	4	5	6	7	8	9
G	1	0	1	2	3	4	5	6	7	8
E	2	1	0	1	2	3	4	5	6	7
O	3	2	1	0	1	2	3	4	5	6
M	4	3	2	1	1	2	3	4	5	6
E	5	4	3	2	2	2	3	4	5	6
T	6	5	4	3	3	3	3			
R	7									
Y	8									

Fig. 9

According to the rule, since the two substrings are a mismatch, the lowest value of the ones surrounding the empty cell *+1* will be written. In this case *2* is the smallest number, so *3* will be the edit distance between “GEOGRA” and “GEOMET”. (Fig. 9)



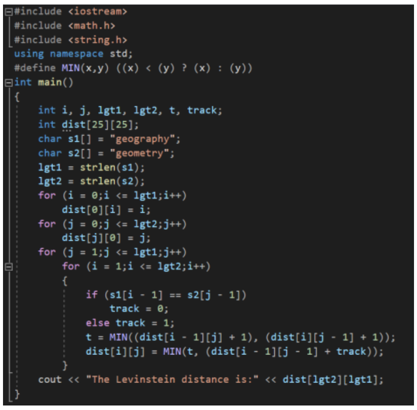
		G	E	O	G	R	A	P	H	Y
	0	1	2	3	4	5	6	7	8	9
G	1	0	1	2	3	4	5	6	7	8
E	2	1	0	1	2	3	4	5	6	7
O	3	2	1	0	1	2	3	4	5	6
M	4	3	2	1	1	2	3	4	5	6
E	5	4	3	2	2	2	3	4	5	6
T	6	5	4	3	3	3	3	4	5	6
R	7	6	5	4	4	4	4	4	5	6
Y	8	7	6	5	5	5	5	5	5	6

Fig. 10

Moving on doing the same operations and using the same concept. Whenever there are two equal minimum values, it does not matter which one is chosen, applying the rule will lead to the same result. According to the table, the distance between “GEOGRAPHY” and “GEOMETRY” is *6*, so that means that there are 6 edits required in order to convert one word into the other one. (Fig. 10)

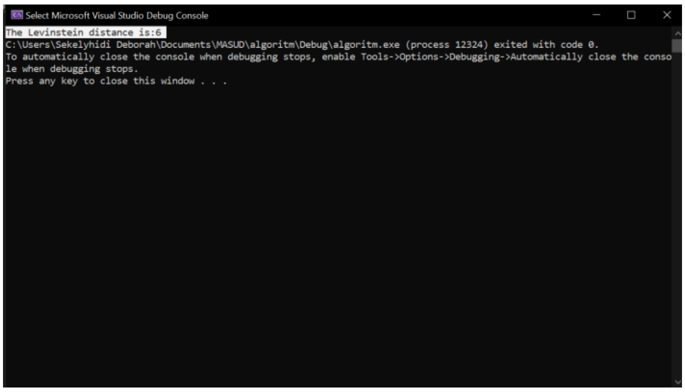
IV. Demonstration

The following algorithm is an example of a program that calculates the distance between the words proposed.



```cpp
#include <iostream>
#include <math.h>
#include <string.h>
using namespace std;
#define MIN(x,y) ((x) < (y) ? (x) : (y))
int main()
{
    int i, j, lgt1, lgt2, t, track;
    int dist[25][25];
    char s1[] = "geography";
    char s2[] = "geometry";
    lgt1 = strlen(s1);
    lgt2 = strlen(s2);
    for (i = 0;i <= lgt1;i++)
        dist[0][i] = i;
    for (j = 0;j <= lgt2;j++)
        dist[j][0] = j;
    for (j = 1;j <= lgt1;j++)
        for (i = 1;i <= lgt2;i++)
        {
            if (s1[i - 1] == s2[j - 1])
                track = 0;
            else track = 1;
            t = MIN((dist[i - 1][j] + 1), (dist[i][j - 1] + 1));
            dist[i][j] = MIN(t, (dist[i - 1][j - 1] + track));
        }
    cout << "The Levinstein distance is:" << dist[lgt2][lgt1];
}
```

Fig. 11



```
The Levinstein distance is:6
C:\Users\Sekelyhidi Deborah\Documents\MASUD\algoritm\Debug\algoritm.exe (process 12324) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

Fig. 12

V. Conclusion

The distance between two words is defined by the number of edits that are required to convert the character string into the other one. Therefore, even though there are several methods of calculating the distance between two words, the most efficient one is the Levenshtein distance.

VI. Bibliography

- https://medium.com/@ethannam/understanding-the-levenshtein-distance-equation-for-beginners-c4285a5604f0
- https://blog.paperspace.com/measuring-text-similarity-using-levenshtein-distance/
- https://towardsdatascience.com/measure-distance-between-2-words-by-simple-calculation-a97cf4993305
- https://en.wikipedia.org/wiki/Jaccard_index
- https://datascience.stackexchange.com/questions/5121/applications-and-differences-for-jaccard-similarity-and-cosine-similarity