Measure distance between 2 words by simple calculation

Edward Ma
Oct 7, 2018 · 4 min read

"brown rail train" by Johannes Plenio on Unsplash

Calculating word distance in NLP is a routine task. Whenever you want to find the most nearest word or measuring metrics, word distance is a one of the way to achieve it.

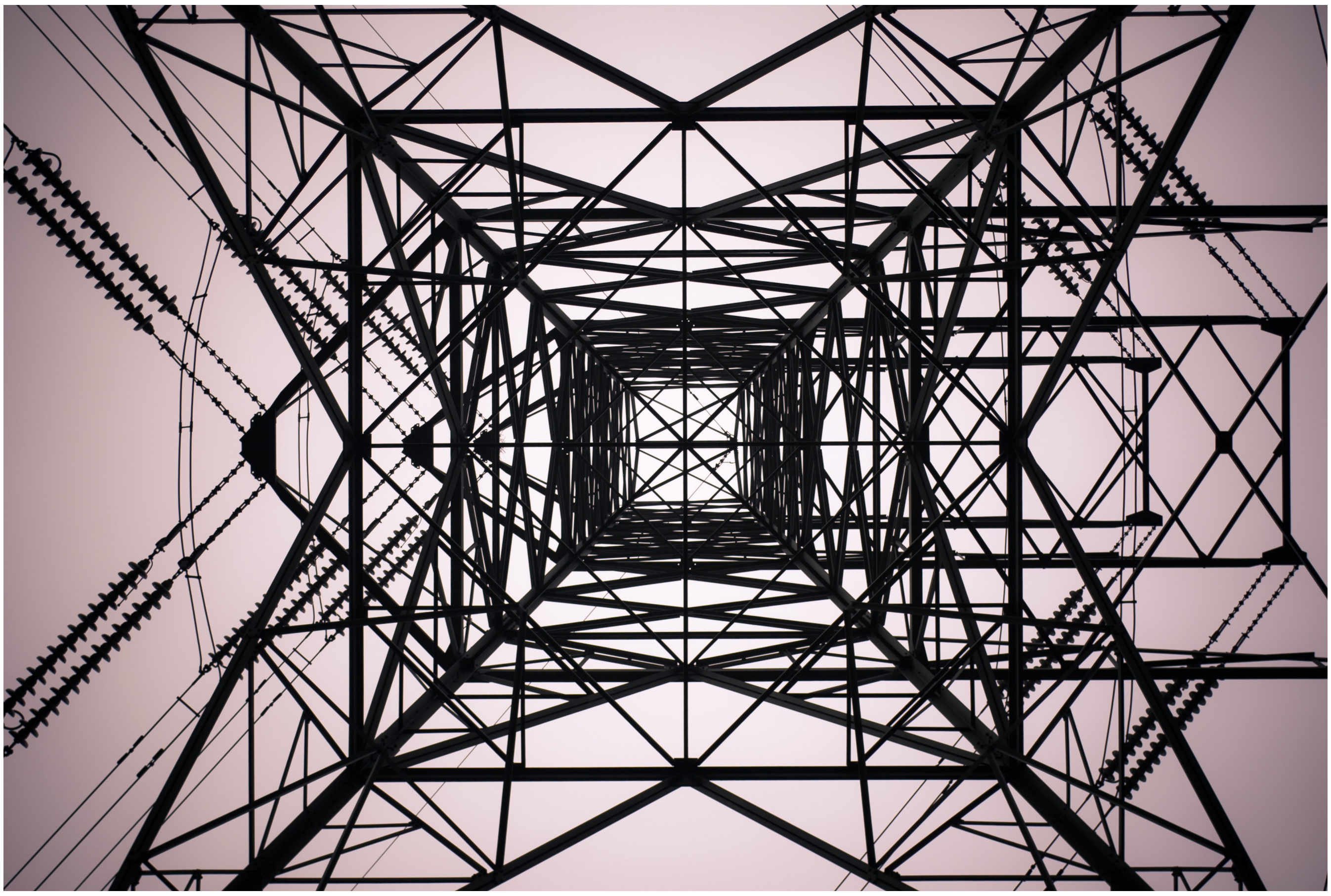
In my previous project, identifying target signal from OCR result is the critical part. To capture more valuable information, I have to tolerance a minor error from OCR result. Otherwise, I miss lots of important signal.

Different from previous distance between word embeddings, string distance is calculating the minimum number of deletion, insertion or substitution required to change from one word to another word. There are numerous way to calculating distance while I will focus on two measurements in this sharing.

After reading this article, you will understand:

- Levenshtein Distance
- Longest Common Subsequence Distance
- Take Away

Levenshtein Distance



"black electrical tower" by Shane Rounce on Unsplash

This method was invented by Vladimir Levenshtein in 1965. The value refer to the minimum number of action (deletion, insertion and substitution) required to transform from one word to another word.

$$\begin{cases} \max(i,j) & \text{if } \min(i,j)=0, \end{cases}$$

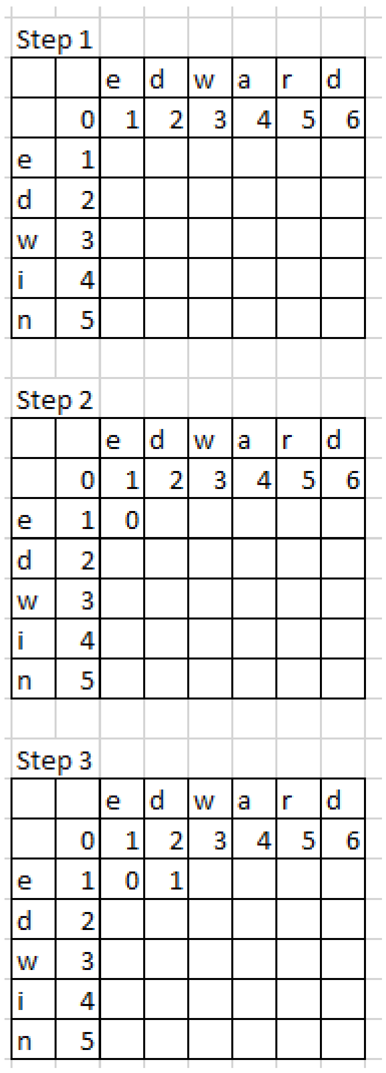
$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \min \begin{cases} \operatorname{lev}_{a,b}(i,j-1)+1 \\ \operatorname{lev}_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

From https://en.wikipedia.org/wiki/Levenshtein_distance

To calculate the Levenshtein Distance, we have to go through every single word and using the minimum + 1 from

- (i-1, j): It means left box (deletion)
- (i, j-1): It means upper box (insertion)
- (i-1, j-1): It means left upper box (substitution)

In the following example, I will transform "edward" to "edwin" and calculating Levenshtein Distance.



Step 1

		e	d	w	a	r	d
	0	1	2	3	4	5	6
e	1						
d	2						
w	3						
i	4						
n	5						

Step 2

		e	d	w	a	r	d
	0	1	2	3	4	5	6
e	1	0					
d	2						
w	3						
i	4						
n	5						

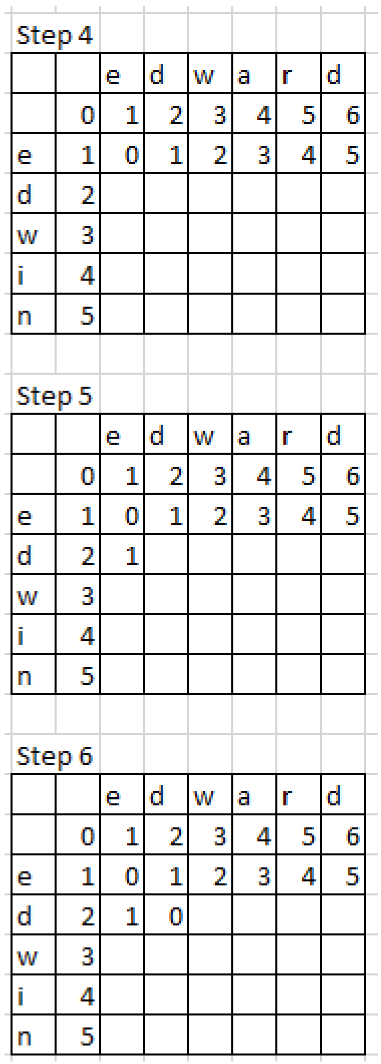
Step 3

		e	d	w	a	r	d
	0	1	2	3	4	5	6
e	1	0	1				
d	2						
w	3						
i	4						
n	5						

Step 1: Assign number from 0 to corresponding number for two words

Step 2: Since “e” is equal to “e”, the value is 0

Step 3: “d” is not equal to “e”, so find the minimum number from left (deletion), diagonal (substitution) and upper (insertion). So it is 0 + 1 = 1



Step 4

		e	d	w	a	r	d
	0	1	2	3	4	5	6
e	1	0	1	2	3	4	5
d	2						
w	3						
i	4						
n	5						

Step 5

		e	d	w	a	r	d
	0	1	2	3	4	5	6
e	1	0	1	2	3	4	5
d	2	1					
w	3						
i	4						
n	5						

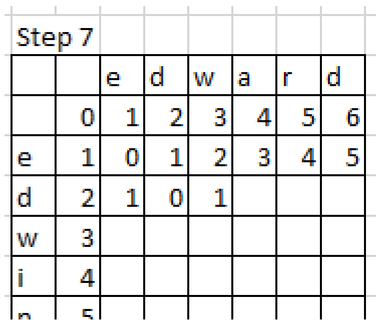
Step 6

		e	d	w	a	r	d
	0	1	2	3	4	5	6
e	1	0	1	2	3	4	5
d	2	1	0				
w	3						
i	4						
n	5						

Step 4: Repeat Step 3 for the following line

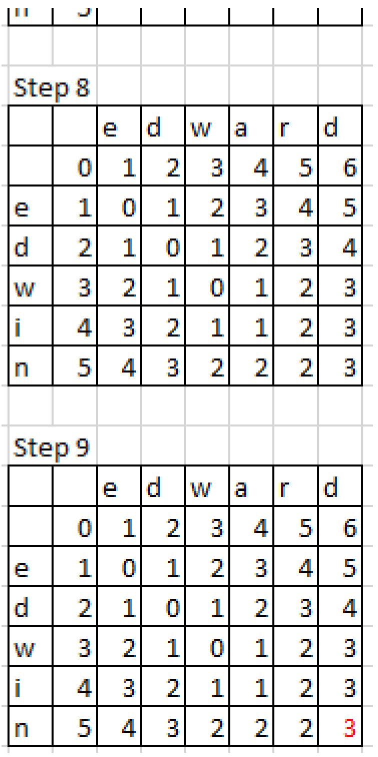
Step 5: “e” is not equal to “d”, so find the minimum number from left (deletion), diagonal (substitution) and upper (insertion). So it is 0 + 1 = 1

Step 6: Since “d” is equal to “d”, so copy the diagonal value which is 0



Step 7

		e	d	w	a	r	d
	0	1	2	3	4	5	6
e	1	0	1	2	3	4	5
d	2	1	0	1			
w	3						
i	4						
n	5						

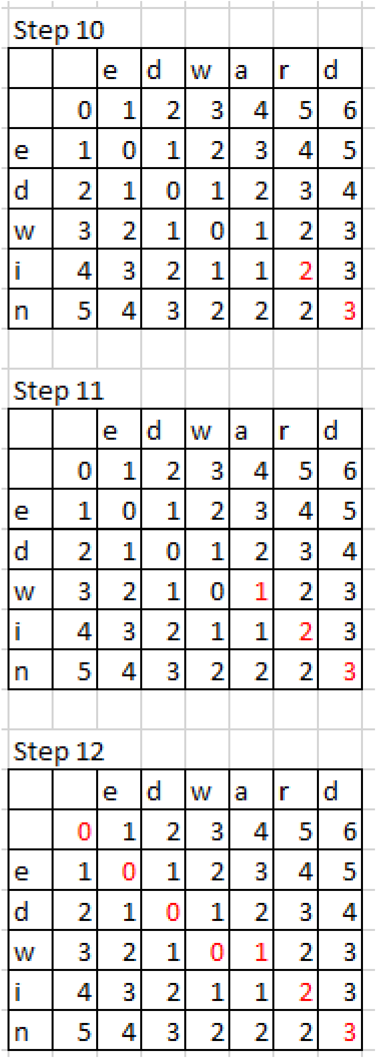


Step 8

		e	d	w	a	r	d
	0	1	2	3	4	5	6
e	1	0	1	2	3	4	5
d	2	1	0	1	2	3	4
w	3	2	1	0	1	2	3
i	4	3	2	1	1	2	3
n	5	4	3	2	2	2	3

Step 9

		e	d	w	a	r	d
	0	1	2	3	4	5	6
e	1	0	1	2	3	4	5
d	2	1	0	1	2	3	4
w	3	2	1	0	1	2	3
i	4	3	2	1	1	2	3
n	5	4	3	2	2	2	3



Step 10

		e	d	w	a	r	d
	0	1	2	3	4	5	6
e	1	0	1	2	3	4	5
d	2	1	0	1	2	3	4
w	3	2	1	0	1	2	3
i	4	3	2	1	1	2	3
n	5	4	3	2	2	2	3

Step 11

		e	d	w	a	r	d
	0	1	2	3	4	5	6
e	1	0	1	2	3	4	5
d	2	1	0	1	2	3	4
w	3	2	1	0	1	2	3
i	4	3	2	1	1	2	3
n	5	4	3	2	2	2	3

Step 12

		e	d	w	a	r	d
	0	1	2	3	4	5	6
e	1	0	1	2	3	4	5
d	2	1	0	1	2	3	4
w	3	2	1	0	1	2	3
i	4	3	2	1	1	2	3
n	5	4	3	2	2	2	3

Step 7: Repeat Step 3 for the following line

Step 8: Repeat Step 3 for the rest of box

Step 9: Levenshtein Distance is 3 (highlighted in red) which is the last one of this matrix

Step 10: After that, e can either go to left (deletion) or diagonal (substitution) because both value are same which is 2. I go for the diagonal this time.

Step 11: Again, we can either go to left (deletion) or diagonal (substitution) because both value are same which is 1. I go for the diagonal this time.

Step 12: This time, the minimum value is 0 which is left (deletion)

From the above formula, we transform “edward” to “edwin” by the following operation

1. Substitute “n” by “d”
2. Substitute “i” by “r”
3. Delete “a”

```
import editdistance

data = ['edward', 'Edward']

for record in data:
    dist = editdistance.eval(record, 'edwin')
    print('Edit Distance for %s and %s is %d' % (record, 'edwin', 
dist))
```

Output

```
Edit Distance for edward and edwin is 3
Edit Distance for Edward and edwin is 4
```

Longest Common Subsequence Distance

It is LCS Distance in short. It is quite similar to Levenshtein Distance except LCS does not include substitution operation.

Therefore, the value is 5 when transforming “edward” to “edwin”. Operations are deleting “a”, “r”, “d” and inserting “i” and “n”.

Take Away

To access project template, you can visit this github repo.

- Edit distance is a **simple and effective way to measure the transposition** between two word.
- It is **case sensitive**.
- Edit distance can be **applied to the correction of spelling error or OCR error**.
- In my case, **tolerance error is 2**. If the true label is “edward”, I will still accept the OCR result if it is “edweed”.

About Me

I am Data Scientist in Bay Area. Focusing on state-of-the-art in Data Science, Artificial Intelligence , especially in NLP and platform related. You can reach me from Medium Blog, LinkedIn or Github.

Reference

Christopher D. M., Prabhakar R., Hinrich S. Introduction to Information Retrieval.

Machine Learning Python NLP Towards Data Science Programming