

Edit Distance Learning Circle

Trey Jones, Search Platform
January 2020



WIKIMEDIA
FOUNDATION

What is edit distance?

- It's a string similarity metric
- Based on number of “edits” to transform one string to another
- Basic edit distance (**Levenshtein**) allows insertion or deletion ($ab \leftrightarrow abc$), and substitution ($abc \leftrightarrow adc$)
- Swaps ($abcd \leftrightarrow acbd$) are a common variant (**Damerau–Levenshtein**)

pax

—keep the p

—insert an e (edit #1)

—keep the a

—change x to c (edit #2)

—insert another e (edit #3)

peace



Dynamic programming

The usual implementation of the edit distance algorithm uses **dynamic programming**, which breaks a problem down into smaller overlapping problems, often recursively. The **edit distance** algorithm, fortunately, is iterative, not recursive.



Why dynamic programming?

A: Over 8 million! (For the math nerds: the number of paths through such a grid is called the *Delannoy number*.)

		e	f	f	i	c	i	e	n	c	y
	1	1	1	1	1	1	1	1	1	1	1
d	1	3	5	7	9	11	13	15	17	19	21
y	1	5	13	25	41	61	85	113	145	181	221
n	1	7	25	63	129	231	377	575	833	1,159	1,561
a	1	9	41	129	321	681	1,289	2,241	3,649	5,641	8,361
m	1	11	61	231	681	1,683	3,653	7,183	13,073	22,363	36,365
i	1	13	85	377	1,289	3,653	8,989	19,825	40,081	75,517	134,245
c	1	15	113	575	2,241	7,183	19,825	48,639	108,545	224,143	433,905
i	1	17	145	833	3,649	13,073	40,081	108,545	265,729	598,417	1,256,465
t	1	19	181	1,159	5,641	22,363	75,517	224,143	598,417	1,462,563	3,317,445
y	1	21	221	1,561	8,361	36,365	134,245	433,905	1,256,465	3,317,445	8,097,453

Why dynamic programming?

There are already 63 possible edit paths just to convert *eff* to *dyn*.

		e	f	f	i	c	i	e	n	c	y
	1	1	1	1	1	1	1	1	1	1	1
d	1	3	5	7	9	11	13	15	17	19	21
y	1	5	13	25	41	61	85	113	145	181	221
n	1	7	25	63	129	231	377	575	833	1,159	1,561
a	1	9	41	129	321	681	1,289	2,241	3,649	5,641	8,361
m	1	11	61	231	681	1,683	3,653	7,183	13,073	22,363	36,365
i	1	13	85	377	1,289	3,653	8,989	19,825	40,081	75,517	134,245
c	1	15	113	575	2,241	7,183	19,825	48,639	108,545	224,143	433,905
i	1	17	145	833	3,649	13,073	40,081	108,545	265,729	598,417	1,256,465
t	1	19	181	1,159	5,641	22,363	75,517	224,143	598,417	1,462,563	3,317,445
y	1	21	221	1,561	8,361	36,365	134,245	433,905	1,256,465	3,317,445	8,097,453

Why dynamic programming?

However—if we are limited to insertion, deletion, and substitution—there are only three cells we can come from directly. If we've already found the optimal path to each of those three...

		e	f	f	i	c	i	e	n	c	y
	1	1	1	1	1	1	1	1	1	1	1
d	1	3	5	7	9	11	13	15	17	19	21
y	1	5	13	25	41	61	85	113	145	181	221
n	1	7	25	63	129	231	377	575	833	1,159	1,561
a	1	9	41	129	321	681	1,289	2,241	3,649	5,641	8,361
m	1	11	61	231	681	1,683	3,653	7,183	13,073	22,363	36,365
i	1	13	85	377	1,289	3,653	8,989	19,825	40,081	75,517	134,245
c	1	15	113	575	2,241	7,183	19,825	48,639	108,545	224,143	433,905
i	1	17	145	833	3,649	13,073	40,081	108,545	265,729	598,417	1,256,465
t	1	19	181	1,159	5,641	22,363	75,517	224,143	598,417	1,462,563	3,317,445
y	1	21	221	1,561	8,361	36,365	134,245	433,905	1,256,465	3,317,445	8,097,453

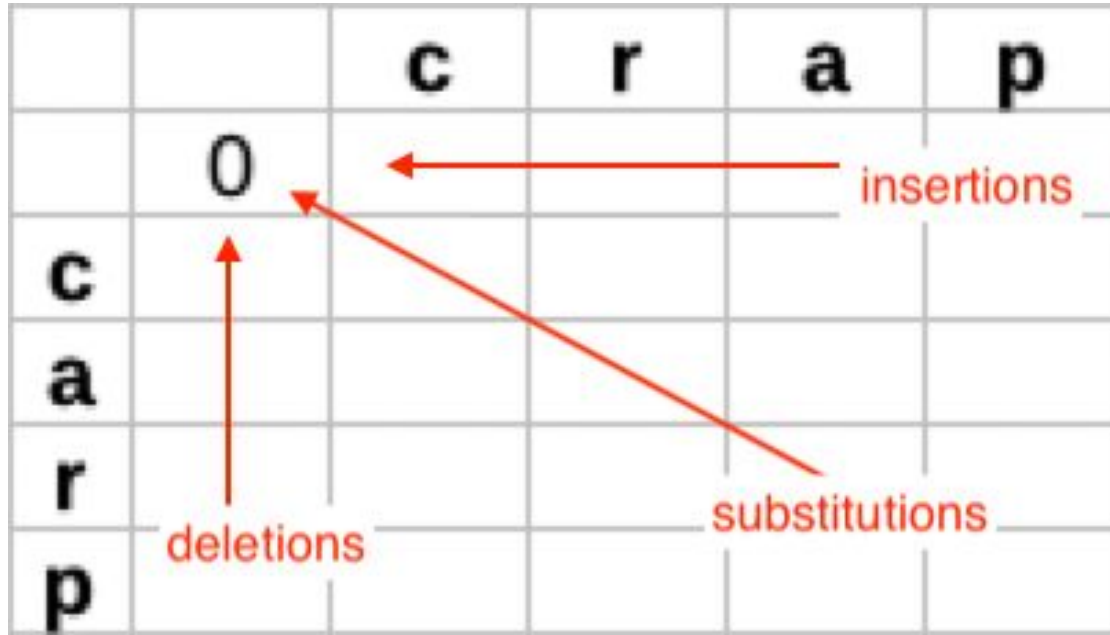
Worked edit distance example

Below we can work through an example of editing *carp* into *crap*. The upper right corner is initialized to 0 because it takes zero edits to convert the empty string to the empty string.

		c	r	a	p
	0				
c					
a					
r					
p					

Worked edit distance example

“Insertions” and “deletions” are really the same thing, but by convention we’ll say insertions are horizontal. Also note that the *arrows point to where you came from*, not to where you are going, so you can backtrack at the end. (It’s a little confusing.)



Worked edit distance example

For the top row—which represents converting the empty string to *crap*—insertion is the only option. Cost is 1 per letter.

		c	r	a	p
	0	← 1	← 2	← 3	← 4
c			inserts (1)		
a					
r					
p					

Worked edit distance example

Similarly, in the first column of each row, deletion is the only option.

		c	r	a	p
	0	1	2	3	4
c	1				
a	delete c (1)				
r					
p					

Worked edit distance example

For the other cells, insertion, deletion, and substitution (each at a cost of 1)—or keeping a letter unchanged for free—are all options. We choose the lowest total cost—in this case keeping the *c*—and put it in the cell.

		keep c (0+0) c	r	a	p
	0	1	2	3	4
c	1	0			
a					
r					
p					

Annotations in the table:
- A green arrow points from the cell (row 1, col 1) to the cell (row 2, col 2).
- A red arrow points from the cell (row 2, col 1) to the cell (row 2, col 2).
- A red arrow points from the cell (row 2, col 2) to the cell (row 1, col 3).
- The text "delete c (1+1)" is written in red next to the cell (row 2, col 3).
- The text "insert c (1+1)" is written in red next to the cell (row 3, col 1).

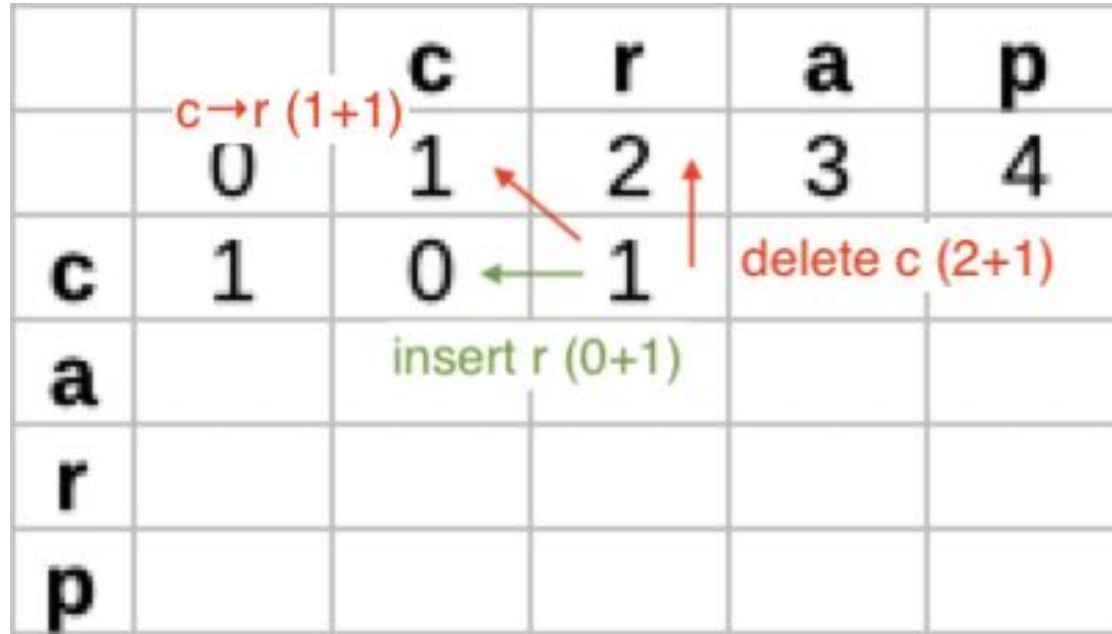
Worked edit distance example

This cell represents editing *c* to *cr*. The cheapest overall path is keep the *c* and insert an *r*.

		c	r	a	p
	0	1	2	3	4
c	1	0	1		
a					
r					
p					

Annotations:

- $c \rightarrow r (1+1)$ (red text above the cell (0,2))
- delete c (2+1) (red text to the right of the cell (1,3))
- insert r (0+1) (green text below the cell (1,2))



Worked edit distance example

For the rest of the row, insertions are the cheapest option. The end of the row represents editing *c* to *crap* by keeping the *c* and inserting *r-a-p*.

		c	r	a	p
	0	1	2	3	4
c	1	0	1	2	3
a					
r					
p					

inserts are the best choice

Worked edit distance example

New row, new deletion of the first letter is the only option—now representing editing *ca* to the empty string.

		c	r	a	p
	0	1	2	3	4
c	1	0	1	2	3
a	2				
r	delete a (1+1)				
p					

Worked edit distance example

In the next cell (ca to c), the best next step is to delete the a . Overall, this is keeping the c and deleting the a .

		c	r	a	p
	$a \rightarrow c \ (1+1)$	1	2	3	4
c	1	0	1	2	3
a	2	1			
r	$insert\ c\ (2+1)$				
p					

Diagram illustrating the edit distance calculation between the strings ca and c . The table shows the edit distance values for each character combination. Red arrows indicate transitions: from (c, a) to (c, c) (labeled $a \rightarrow c \ (1+1)$) and from (a, c) to (a, a) (labeled $delete\ a \ (0+1)$). A green arrow points from the cell (a, c) to the cell (c, c) , indicating the optimal path.

Worked edit distance example

Our best move into the next cell is to come from the cell with 0 edits and change *a* to *r*.

		c	r	a	p
	0	1	2	3	4
c	1	0	1	2	3
a	2	1	1		
r					
p					

a → *r* (0+1)

delete *a* (1+1)

insert *r* (1+1)

Worked edit distance example


Often, if both letters are the same, transitioning along the diagonal for free is the optimal path into a cell. (But not always!)

		c	r	a	p
	0	1	1 ^{keep a (1+0)}	3	4
c	1	0	1	2	3
a	2	1	1	1 ^{delete a (2+1)}	
r				1 ^{insert a (1+1)}	
p					

Worked edit distance example

Again, at the end of an early row, inserting extra letters is often the best option.

		c	r	a	p
	0	1	2	3	4
c	1	0	1	2	3
a	2	1	1	1	2
r					
p					



The diagram shows three arrows originating from the cell containing '1' at the intersection of row 'a' and column 'a':

- A red arrow pointing diagonally up and right to the cell containing '2' at the intersection of row 'c' and column 'a'.
- A red arrow pointing diagonally up and right to the cell containing '3' at the intersection of row 'c' and column 'p'.
- A green arrow pointing horizontally left to the cell containing '1' at the intersection of row 'a' and column 'a'.

Worked edit distance example

New row! In the first cell, deletion is the only options. In the second, the best is to delete from the cell above. In the third cell, *r* maps to *r*, so a 0-cost diagonal transition is the best.

		c	r	a	p
	0	1	2	3	4
c	1	0	1	2	3
a	2	1	1	1	2
r	3	2	1		
p					

Worked edit distance example


All that work for nothing! Not really—but all available paths from *car* to *cra* have the same total cost.

		c	r	a	p
	0	1	2	3	4
c	1	0	1	2	3
a	2	1	1	1	2
r	3	2	1	2	
p					

Worked edit distance example

Finish the row with a diagonal transition.

		c	r	a	p
	0	1	2	3	4
c	1	0	1	2	3
a	2	1	1	1	2
r	3	2	1	2	2
p					



Worked edit distance example

The final row is similar to previous ones. The last cell is a 0-cost diagonal transition (keep p).

		c	r	a	p
	0	1	2	3	4
c	1	0	1	2	3
a	2	1	1	1	2
r	3	2	1	2	2
p	4	3	2	2	2

Worked edit distance example

Below are the overlapping optimal edit paths. When converting *ar* to *ra* you can change *a* to *r* and *r* to *a*, **or** delete and re-insert one around the other. Two substitutions is also the cost of *carp* → *chip*! That's why we want discounted swaps!

		c	r	a	p
	0	1	2	3	4
c	1	0	1	2	3
a	2	1	1	1	2
r	3	2	1	2	2
p	4	3	2	2	2

A bigger example

Below is an worked out edit distance table (using only insertions, deletions, and substitutions).

		e	f	f	i	c	i	e	n	c	y
	0	1	2	3	4	5	6	7	8	9	10
d	1	1	2	3	4	5	6	7	8	9	10
y	2	2	2	3	4	5	6	7	8	9	9
n	3	3	3	3	4	5	6	7	7	8	9
a	4	4	4	4	4	5	6	7	8	8	9
m	5	5	5	5	5	5	6	7	8	9	9
i	6	6	6	6	5	6	5	6	7	8	9
c	7	7	7	7	6	5	6	6	7	7	8
i	8	8	8	8	7	6	5	6	7	8	8
t	9	9	9	9	8	7	6	6	7	8	9
y	10	10	10	10	9	8	7	7	7	8	8

A bigger example—with arrows!

This table has arrows to indicate the optimal edits to each cell. When there are no “good” edits, lots of sub-paths have the same cost (e.g., *eff* to *dynamic*). You can work backward from any cell to find the optimal edit path(s).

		e	f	f	i	c	i	e	n	c	y
	0	1	2	3	4	5	6	7	8	9	10
d	1	1	2	3	4	5	6	7	8	9	10
y	2	2	2	3	4	5	6	7	8	9	9
n	3	3	3	3	4	5	6	7	7	8	9
a	4	4	4	4	4	5	6	7	8	8	9
m	5	5	5	5	5	5	6	7	8	9	9
i	6	6	6	6	5	6	5	6	7	8	9
c	7	7	7	7	6	5	6	6	7	7	8
i	8	8	8	8	7	6	5	6	7	8	8
t	9	9	9	9	8	7	6	6	7	8	9
y	10	10	10	10	9	8	7	7	7	8	8

A bigger example—with arrows!

The various overlapping optimal paths to the final cell (there are 31 total) are highlighted in green below.

		e	f	f	i	c	i	e	n	c	y
	0	1	2	3	4	5	6	7	8	9	10
d	1	1	2	3	4	5	6	7	8	9	10
y	2	2	2	3	4	5	6	7	8	9	9
n	3	3	3	3	4	5	6	7	7	8	9
a	4	4	4	4	4	5	6	7	8	8	9
m	5	5	5	5	5	5	6	7	8	9	9
i	6	6	6	6	5	6	5	6	7	8	9
c	7	7	7	7	6	5	6	6	7	7	8
i	8	8	8	8	7	6	5	6	7	8	8
t	9	9	9	9	8	7	6	6	7	8	9
y	10	10	10	10	9	8	7	7	7	8	8

A bigger example—with arrows!

All of these paths have the same edit distance: 8.

		e	f	f	i	c	i	e	n	c	y
	0	1	2	3	4	5	6	7	8	9	10
d	1	1	2	3	4	5	6	7	8	9	10
y	2	2	2	3	4	5	6	7	8	9	9
n	3	3	3	3	4	5	6	7	7	8	9
a	4	4	4	4	4	5	6	7	8	8	9
m	5	5	5	5	5	5	6	7	8	9	9
i	6	6	6	6	5	6	5	6	7	8	9
c	7	7	7	7	6	5	6	6	7	7	8
i	8	8	8	8	7	6	5	6	7	8	8
t	9	9	9	9	8	7	6	6	7	8	9
y	10	10	10	10	9	8	7	7	7	8	8

A direct path

In this case, there is a simple straightforward path along the diagonal (shown in blue and purple). The letters *i* and *y* happen to line up (free transitions in purple), and the other 8 letters are substituted.

		e	f	f	i	c	i	e	n	c	y
	0	1	2	3	4	5	6	7	8	9	10
d	1	1	2	3	4	5	6	7	8	9	10
y	2	2	2	3	4	5	6	7	8	9	9
n	3	3	3	3	4	5	6	7	7	8	9
a	4	4	4	4	4	5	6	7	8	8	9
m	5	5	5	5	5	5	6	7	8	9	9
i	6	6	6	6	5	6	5	6	7	8	9
c	7	7	7	7	6	5	6	6	7	7	8
i	8	8	8	8	7	6	5	6	7	8	8
t	9	9	9	9	8	7	6	6	7	8	9
y	10	10	10	10	9	8	7	7	7	8	8

An indirect path

Other paths take advantage of both words containing *ici*, meaning three transitions can take place at zero cost (in purple), making up for the insertions and deletions needed to get to the overlapping *ici* path.

		e	f	f	i	c	i	e	n	c	y
	0	1	2	3	4	5	6	7	8	9	10
d	1	1	2	3	4	5	6	7	8	9	10
y	2	2	2	3	4	5	6	7	8	9	9
n	3	3	3	3	4	5	6	7	7	8	9
a	4	4	4	4	4	5	6	7	8	8	9
m	5	5	5	5	5	5	6	7	8	9	9
i	6	6	6	6	5	6	5	6	7	8	9
c	7	7	7	7	6	5	6	6	7	7	8
i	8	8	8	8	7	6	5	6	7	8	8
t	9	9	9	9	8	7	6	6	7	8	9
y	10	10	10	10	9	8	7	7	7	8	8

Another indirect path

From the start of each word to *ici*, and from *ici* to the final *y*, there are several possible paths—contributing the majority of the 31 possible paths.

		e	f	f	i	c	i	e	n	c	y
	0	1	2	3	4	5	6	7	8	9	10
d	1	1	2	3	4	5	6	7	8	9	10
y	2	2	2	3	4	5	6	7	8	9	9
n	3	3	3	3	4	5	6	7	7	8	9
a	4	4	4	4	4	5	6	7	8	8	9
m	5	5	5	5	5	5	6	7	8	9	9
i	6	6	6	6	5	6	5	6	7	8	9
c	7	7	7	7	6	5	6	6	7	7	8
i	8	8	8	8	7	6	5	6	7	8	8
t	9	9	9	9	8	7	6	6	7	8	9
y	10	10	10	10	9	8	7	7	7	8	8

Swaps

- A common edit distance variant is **Damerau–Levenshtein**, which gives a discount for swaps (e.g., 1.5 for a swap vs 2 for two substitutions).

A Damerau-Levenshtein distance matrix comparing the string 'c' (row) with 'rap' (column). The matrix shows the edit distance between prefixes of the two strings. Red arrows indicate the path of minimum cost, and blue arrows highlight a swap operation between 'r' and 'a'.

		c	r	a	p	
		0	1	2	3	4
c		1	0	1	2	3
a		2	1	1	1	2
r		3	2	1	1.5	2
p		4	3	2	2	1.5

Optimizations

- If you don't need to recover the optimal edit path(s), you only need two rows of the table at a time. (Three rows if you allow swaps.)
- If you have a max allowable edit distance, you can terminate early!
 - You can have a max “raw” edit distance—say, 2 edits—or you can have a max proportional edit distance—say, 40% of the length of the string.
 - Proportional edit distance can be compared to the longer string (permissive), the shorter string (conservative), or a specific one of the strings (directional).

Early termination

If the limit is 2 edits, then by the third row (the n in *dynamicity*), it's already too late; the edit distance isn't going to decrease in later rows.*

* Unless the swap cost is *less* than the insertion cost.

		e	f	f	i	c	i	e	n	c	y
	0	1	2	3	4	5	6	7	8	9	10
d	1	1	2	3	4	5	6	7	8	9	10
y	2	2	2	3	4	5	6	7	8	9	9
n	3	3	3	3	4	5	6	7	7	8	9
a	4	4	4	4	4	5	6	7	8	8	9
m	5	5	5	5	5	5	6	7	8	9	9
i	6	6	6	6	5	6	5	6	7	8	9
c	7	7	7	7	6	5	6	6	7	7	8
i	8	8	8	8	7	6	5	6	7	8	8
t	9	9	9	9	8	7	6	6	7	8	9
y	10	10	10	10	9	8	7	7	7	8	8

Other variants: custom weights

On other projects, I've implemented a weighted edit distance that had a lookup table for specific edits. For example, changing one vowel to another might only cost 0.25, or swapping *s/z* or *t/d* or *m/n* might be 0.50.

These kinds of custom weights tend to be very application-specific. For matching personal names in the U.S., you could adapt something based on a phonetic algorithm like **Soundex**.

Other variants: multi-character edits

- Russian **Щ** was historically pronounced /ʃtʃ/ (written “shch” in English), so that’s how it is transliterated in Latin-based European languages
- European languages don’t have a consistent way of representing those sounds
- Composer Родион **Щедрин**’s surname has been transliterated as **Shchedrin** (English), **Ščedrin** (Czech), **Schtschedrin** (German), **Chtchedrine** (French), **Szczedrin** (Polish), **Sxedrín** (Catalan), **Sjtjedrin** (Danish), **Scsedrin** (Hungarian), **Sjtsjedrin** (Dutch), **Şcedrin** (Romanian), and **Štšedrin** (Finnish).

Other variants: multi-character edits

So, when working with Russian names in transliteration, we might want to say that *shch*, *šč* (and maybe *sc*), *schtsch*, *chtch*, and *szcz* all have a low edit distance from each other, like 0.5.

Notes: Implementation is a pain! Early termination is all but impossible (*sc* ↔ *schtsch* would look like an edit distance of 4 until the row with the last *h*, when it drops to 0.5). You need a lot more of the edit distance table because you might need to look back 6 rows for *schtsch*—it's probably easier to keep the whole thing.

Enter Glent, Method 1

- Glent is a spelling correction/suggestion system for on-wiki search.
- Method 0 looks at same-user corrections to make suggestions.
- Method 1 looks across users for similar searches to make suggestions.
- It's fairly likely that similar searches (say, with an edit distance of ≤ 2) in Method 0 are related.
- In Method 1, there are no guarantees of relatedness, so weird things can happen...

Glent, Method 1—Anti-patterns

- Search syntax should be excluded
 - *focus* → *-ous*
 - *king* → *king**
- Edit limits should be per-token[†]
 - *cf gene* → *a gene*
 - *hot instagram* → *her instagram*
 - *1949 uk* → *1999 us*
 - *the 43* → *the (*
- Basic tokenization would help
 - *redmax* → *red an*
 - *the 43* → *the (*
 - *hassan* → *has an*
 - *c3 c4 plants* → *c3 plants (2 spaces)*
 - *coburg oregon* → *coburg, oregon*
 - *cf gene* → *a gene*
 - *hot instagram* → *her instagram*
 - *red herring* → *red hering*
 - *greek goddesses* → *greek godness*

[†] While looking at these, I realized I'd never used edit distance on multi-word strings!

Glent, Method 1—Anti-patterns

- First letter of a word is more important
 - *billy smith* → *kelly smith*
 - *cia assassinations* → *mi6 assassinations*
 - *cf gene* → *a gene*
 - *6th gen ipad* → *4th gen ipad*
- Changing token counts is bad
 - *abbys* → *a b s*
 - *redmax* → *red an*
- Numbers matter more
 - *6th gen ipad* → *4th gen ipad*
 - *2018 skating* → *2010 skating*
 - *1949 uk* → *1999 us*

Glent, Method 1—Good Patterns

- **Duplicate letters should get a discount**
 - *agripinna* → *agri**pp**ina*
 - *aggripina* → *agri**pp**ina*
- **Swaps should get a discount**
 - *queit* → *qui**e**t*
- **Space-only differences should get a discount**
 - *mack th eknife* → *mack the **k**nife*

Note: These particular results happened to be chosen as the best more or less by accident. We want to make them more likely.

- *queit* could just as easily matched *quilt*, *qubit*, or *quart*.
- *macktheknife* is too many edits (3) from *mack the knife*
- *aggripinna* is also too many edits (3) from *agrippina*



Token-Aware Edit Distance Goals

Initial Improvement Goals

- Discount for swaps (< 2.0)
- Discount for duplicate letters ($\ll 1$)
- Discount if strings differ only by spaces (< 1)
- Enforce proportional edit limits per-token
- Penalty for changing digits
- Penalty for changing the first letter of a token
- Penalty for changing the number of tokens
- Customizable weights and limits

Token-Aware Edit Distance Goals

The real improvements are the ones we made along the way...

- Customizable normalization
(longest, shortest, first)
- Ability to disable per-token limits
- Penalty for changing a space to not-a-space
- Customizable tokenization
 - Redefinable “space”
 - Custom locale for lowercasing
 - Custom tokenization regex
 - Hook for external tokenizer

Token-Aware Edit Distance Goals

Optimizations

- Allow for early termination when...
 - ...over the raw or proportional edit limit for a given row
 - ...the token count delta is too much
 - ...the character set overlap is too small (inspired by Jaccard similarity)
 - Check is $O(m+n)$ instead of $O(m \cdot n)$ for the full edit distance

What's the implementation?

Bookkeeping—lots and lots of bookkeeping.

- In addition to tracking the optimal edit distance from the start to each cell, we also need to track:
 - The token boundaries with each string
 - The normalized length of the current token pair
 - The number of edits within the current token pair



Norm length of token pair—MAX

		c	b	a		d	d	e	e
	0	0	0	0	0	0	0	0	0
a	0	1	2	3	0	1	1.05	2.05	2.1
b	0	2	2	3	0	2	2	2.05	2.1
c	0	3	3	3	0	3	3	3	3
d	0	4	4	4	0	4	4	4	4
	0	0	0	0	0	0	0	0	0
e	0	1	2	3	0	1	1.05	2.05	2.1
e	0	1.05	2	3	0	1.05	1.05	2.05	2.1
f	0	2.05	2.05	3	0	2.05	2.05	2.05	2.1

Norm length of token pair—MIN

		c	b	a		d	d	e	e
	0	0	0	0	0	0	0	0	0
a	0	1	1	1	0	1	1	1	1
b	0	1	2	2	0	1	1.05	2	2
c	0	1	2	3	0	1	1.05	2.05	2.1
d	0	1	2	3	0	1	1.05	2.05	2.1
	0	0	0	0	0	0	0	0	0
e	0	1	1	1	0	1	1	1	1
e	0	1	1.05	1.05	0	1	1.05	1.05	1.05
f	0	1	2	2.05	0	1	1.05	2.05	2.05

Implementation by example

Let's compare *Le Mode!* (with several spaces) and *a_la_mood*, with an edit limit of 5.0, a proportional limit of 60%, normalization type “MAX”, and the default tokenizer.

- Lowercase and Tokenize: [le, mode] vs [a, la, mood]
 - Count tokens and add a penalty for the delta: $0.25 * |3 - 2|$
- Create strings to compute edit distance: “le mode” vs “a la mood”
- Compute spaceless versions: “lemode” != “alamood”

Implementation by example

We now have “**le mode**” vs “**a la mood**”

- Compute normalized lengths: 7 vs 8.05 (double “oo” only counts as 1.05)
 - $MAX = 8.05$; $60\% * 8.05$ is 4.83, so the edit limit for this pair is 4.83
- The token count penalty (0.25) is not enough by itself to bail early
- The character sets in each string are: [, d, e, l, m, o] vs [, a, d, l, m, o]
 - *e* and *a* are distinct, which implies a minimum of one substitution, which is not enough to bail early



Implementation by example

le mode vs a la mood

	ø	a	l	a	m	o	o	d				
ø	← 0.00	← 1.25	← 2.25	← 3.50	← 4.50	← 5.50	← 6.75	← 7.75	← 7.80	← 8.80		
tc	0.00	1.25	0.00	1.25	2.25	0.00	1.25	2.25	2.30	3.30		
tnl	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		
l	↑ 1.25	↖ 1.25	↑ 3.50	↖ 2.25	← 3.25	← 4.25	← 5.50	← 6.50	← 6.55	← 7.55	min: 1.25	+ 0.25
tc	1.25	1.25	0.00	0.00	1.00	0.00	1.25	2.25	2.30	3.30		
tnl	0.00	1.00	0.00	1.00	2.00	0.00	1.00	2.00	2.05	3.05		
e	↑ 2.25	↑ 2.25	↑ 4.50	↑ 3.25	↖ 3.25	← 4.25	↘ 5.50	↘ 6.50	← 6.55	↘ 7.55	min: 2.25	+ 0.25
tc	2.25	2.25	0.00	1.00	1.00	0.00	1.25	2.25	2.30	3.30		
tnl	0.00	2.00	0.00	2.00	2.00	0.00	2.00	2.00	2.05	3.05		
	↑ 3.25	← 4.50	↘ 5.50	↑ 4.25	↑ 4.25	↖ 3.25	← 4.50	← 5.50	← 5.55	← 6.55	min: 3.25	+ 0.25
tc	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		
tnl	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		
m	↑ 4.50	↖ 4.50	↖ 6.25	↑ 5.50	↘ 5.50	↑ 4.50	↖ 3.25	← 4.25	← 4.30	← 5.30	min: 3.25	+ 0.25
tc	1.25	1.25	0.00	1.25	1.25	0.00	0.00	1.00	1.05	2.05		
tnl	0.00	1.00	0.00	1.00	2.00	0.00	1.00	2.00	2.05	3.05		
o	↑ 5.50	↑ 5.50	↑ 7.25	↑ 6.50	↘ 6.50	↑ 5.50	↑ 4.25	↖ 3.25	← 3.30	← 4.30	min: 3.25	+ 0.25
tc	2.25	2.25	0.00	2.25	2.25	0.00	1.00	0.00	0.05	1.05		
tnl	0.00	2.00	0.00	2.00	2.00	0.00	2.00	2.00	2.05	3.05		
d	↑ 6.50	↑ 6.50	↑ 8.25	↑ 7.50	↘ 7.50	↑ 6.50	↑ 5.25	↑ 4.25	↖ 4.25	↖ 3.30	min: 3.30	+ 0.25
tc	3.25	3.25	0.00	3.25	3.25	0.00	2.00	1.00	1.00	0.05		
tnl	0.00	3.00	0.00	3.00	3.00	0.00	3.00	3.00	3.00	3.05		
e	↑ 7.50	↑ 7.50	↑ 9.25	↑ 8.50	↘ 8.50	↑ 7.50	↑ 6.25	↑ 5.25	↘ 5.25	↑ 4.30	min: 4.30	+ 0.25
tc	4.25	4.25	0.00	4.25	4.25	0.00	3.00	2.00	2.00	1.05		
tnl	0.00	4.00	0.00	4.00	4.00	0.00	4.00	4.00	4.00	4.00		

Implementation by example

le mode vs a la mood

	ø	a	l	a	m	o	o	d			
ø	← 0.00	← 1.25	← 2.25	← 3.50	← 4.50	← 5.50	← 6.75	← 7.75	← 7.80	← 8.80	
tc	0.00	1.25	0.00	1.25	2.25	0.00	1.25	2.25	2.30	3.30	
tnl	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
l	↑ 1.25	↖ 1.25	↑ 3.50	↖ 2.25	← 3.25	← 4.25	← 5.50	← 6.50	← 6.55	← 7.55	min: 1.25 + 0.25
tc	1.25	1.25	0.00	0.00	1.00	0.00	1.25	2.25	2.30	3.30	
tnl	0.00	1.00	0.00	1.00	2.00	0.00	1.00	2.00	2.05	3.05	
e	↑ 2.25	↑ 2.25	↑ 4.50	↑ 3.25	↖ 3.25	← 4.25	↘ 5.50	↘ 6.50	← 6.55	↘ 7.55	min: 2.25 + 0.25
tc	2.25	2.25	0.00	1.00	1.00	0.00	1.25	2.25	2.30	3.30	
tnl	0.00	2.00	0.00	2.00	2.00	0.00	2.00	2.00	2.05	3.05	
	↑ 3.25	← 4.50	↘ 5.50	↑ 4.25	↑ 4.25	↖ 3.25	← 4.50	← 5.50	← 5.55	← 6.55	min: 3.25 + 0.25
tc	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
tnl	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
m	↑ 4.50	↖ 4.50	↖ 6.25	↑ 5.50	↘ 5.50	↑ 4.50	↖ 3.25	← 4.25	← 4.30	← 5.30	min: 3.25 + 0.25
tc	1.25	1.25	0.00	1.25	1.25	0.00	0.00	1.00	1.05	2.05	
tnl	0.00	1.00	0.00	1.00	2.00	0.00	1.00	2.00	2.05	3.05	
o	↑ 5.50	↑ 5.50	↑ 7.25	↑ 6.50	↘ 6.50	↑ 5.50	↑ 4.25	↖ 3.25	← 3.30	← 4.30	min: 3.25 + 0.25
tc	2.25	2.25	0.00	2.25	2.25	0.00	1.00	0.00	0.05	1.05	
tnl	0.00	2.00	0.00	2.00	2.00	0.00	2.00	2.00	2.05	3.05	
d	↑ 6.50	↑ 6.50	↑ 8.25	↑ 7.50	↘ 7.50	↑ 6.50	↑ 5.25	↑ 4.25	↖ 4.25	↖ 3.30	min: 3.30 + 0.25
tc	3.25	3.25	0.00	3.25	3.25	0.00	2.00	1.00	1.00	0.05	
tnl	0.00	3.00	0.00	3.00	3.00	0.00	3.00	3.00	3.00	3.05	
e	↑ 7.50	↑ 7.50	↑ 9.25	↑ 8.50	↘ 8.50	↑ 7.50	↑ 6.25	↑ 5.25	↘ 5.25	↑ 4.30	min: 4.30 + 0.25
tc	4.25	4.25	0.00	4.25	4.25	0.00	3.00	2.00	2.00	1.05	
tnl	0.00	4.00	0.00	4.00	4.00	0.00	4.00	4.00	4.00	4.00	

Implementation by example

le mode vs a la mood

	ø	a	l	a	m	o	o	d			
ø	← 0.00	← 1.25	← 2.25	← 3.50	← 4.50	← 5.50	← 6.75	← 7.75	← 7.80	← 8.80	
tc	0.00	1.25	0.00	1.25	2.25	0.00	1.25	2.25	2.30	3.30	
tnl	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
l	↑ 1.25	↖ 1.25	↑ 3.50	↖ 2.25	← 3.25	← 4.25	← 5.50	← 6.50	← 6.55	← 7.55	min: 1.25 + 0.25
tc	1.25	1.25	0.00	0.00	1.00	0.00	1.25	2.25	2.30	3.30	
tnl	0.00	1.00	0.00	1.00	2.00	0.00	1.00	2.00	2.05	3.05	
e	↑ 2.25	↑ 2.25	↑ 4.50	↑ 3.25	↖ 3.25	← 4.25	↘ 5.50	↘ 6.50	← 6.55	↘ 7.55	min: 2.25 + 0.25
tc	2.25	2.25	0.00	1.00	1.00	0.00	1.25	2.25	2.30	3.30	
tnl	0.00	2.00	0.00	2.00	2.00	0.00	2.00	2.00	2.05	3.05	
	↑ 3.25	← 4.50	↘ 5.50	↑ 4.25	↑ 4.25	↖ 3.25	← 4.50	← 5.50	← 5.55	← 6.55	min: 3.25 + 0.25
tc	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
tnl	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
m	↑ 4.50	↖ 4.50	↖ 6.25	↑ 5.50	↘ 5.50	↑ 4.50	↖ 3.25	← 4.25	← 4.30	← 5.30	min: 3.25 + 0.25
tc	1.25	1.25	0.00	1.25	1.25	0.00	0.00	1.00	1.05	2.05	
tnl	0.00	1.00	0.00	1.00	2.00	0.00	1.00	2.00	2.05	3.05	
o	↑ 5.50	↑ 5.50	↑ 7.25	↑ 6.50	↘ 6.50	↑ 5.50	↑ 4.25	↖ 3.25	← 3.30	← 4.30	min: 3.25 + 0.25
tc	2.25	2.25	0.00	2.25	2.25	0.00	1.00	0.00	0.05	1.05	
tnl	0.00	2.00	0.00	2.00	2.00	0.00	2.00	2.00	2.05	3.05	
d	↑ 6.50	↑ 6.50	↑ 8.25	↑ 7.50	↘ 7.50	↑ 6.50	↑ 5.25	↑ 4.25	↖ 4.25	↖ 3.30	min: 3.30 + 0.25
tc	3.25	3.25	0.00	3.25	3.25	0.00	2.00	1.00	1.00	0.05	
tnl	0.00	3.00	0.00	3.00	3.00	0.00	3.00	3.00	3.00	3.05	
e	↑ 7.50	↑ 7.50	↑ 9.25	↑ 8.50	↘ 8.50	↑ 7.50	↑ 6.25	↑ 5.25	↖ 5.25	↑ 4.30	min: 4.30 + 0.25
tc	4.25	4.25	0.00	4.25	4.25	0.00	3.00	2.00	2.00	1.05	
tnl	0.00	4.00	0.00	4.00	4.00	0.00	4.00	4.00	4.00	4.00	

Implementation by example

le mode vs a la mood

	ø	a	l	a	m	o	o	d			
ø	← 0.00	← 1.25	← 2.25	← 3.50	← 4.50	← 5.50	← 6.75	← 7.75	← 7.80	← 8.80	
tc	0.00	1.25	0.00	1.25	2.25	0.00	1.25	2.25	2.30	3.30	
tnl	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
l	↑ 1.25	↖ 1.25	↑ 3.50	↖ 2.25	← 3.25	← 4.25	← 5.50	← 6.50	← 6.55	← 7.55	min: 1.25 + 0.25
tc	1.25	1.25	0.00	0.00	1.00	0.00	1.25	2.25	2.30	3.30	
tnl	0.00	1.00	0.00	1.00	2.00	0.00	1.00	2.00	2.05	3.05	
e	↑ 2.25	↑ 2.25	↑ 4.50	↑ 3.25	↖ 3.25	← 4.25	↘ 5.50	↘ 6.50	← 6.55	↘ 7.55	min: 2.25 + 0.25
tc	2.25	2.25	0.00	1.00	1.00	0.00	1.25	2.25	2.30	3.30	
tnl	0.00	2.00	0.00	2.00	2.00	0.00	2.00	2.00	2.05	3.05	
	↑ 3.25	← 4.50	↘ 5.50	↑ 4.25	↑ 4.25	↖ 3.25	← 4.50	← 5.50	← 5.55	← 6.55	min: 3.25 + 0.25
tc	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
tnl	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
m	↑ 4.50	↖ 4.50	↖ 6.25	↑ 5.50	↘ 5.50	↑ 4.50	↖ 3.25	← 4.25	← 4.30	← 5.30	min: 3.25 + 0.25
tc	1.25	1.25	0.00	1.25	1.25	0.00	0.00	1.00	1.05	2.05	
tnl	0.00	1.00	0.00	1.00	2.00	0.00	1.00	2.00	2.05	3.05	
o	↑ 5.50	↑ 5.50	↑ 7.25	↑ 6.50	↘ 6.50	↑ 5.50	↑ 4.25	↖ 3.25	← 3.30	← 4.30	min: 3.25 + 0.25
tc	2.25	2.25	0.00	2.25	2.25	0.00	1.00	0.00	0.05	1.05	
tnl	0.00	2.00	0.00	2.00	2.00	0.00	2.00	2.00	2.05	3.05	
d	↑ 6.50	↑ 6.50	↑ 8.25	↑ 7.50	↘ 7.50	↑ 6.50	↑ 5.25	↑ 4.25	↖ 4.25	↖ 3.30	min: 3.30 + 0.25
tc	3.25	3.25	0.00	3.25	3.25	0.00	2.00	1.00	1.00	0.05	
tnl	0.00	3.00	0.00	3.00	3.00	0.00	3.00	3.00	3.00	3.05	
e	↑ 7.50	↑ 7.50	↑ 9.25	↑ 8.50	↘ 8.50	↑ 7.50	↑ 6.25	↑ 5.25	↖ 5.25	↑ 4.30	min: 4.30 + 0.25
tc	4.25	4.25	0.00	4.25	4.25	0.00	3.00	2.00	2.00	1.05	
tnl	0.00	4.00	0.00	4.00	4.00	0.00	4.00	4.00	4.00	4.00	

Implementation by example

le mode vs a la mood

∅	← 0.00	← 1.25	← 2.25	← 3.50	← 4.50	← 5.50	← 6.75	← 7.75	← 7.80	← 8.00
tc	0.00	1.25	0.00	1.25	2.25	0.00	1.25	2.25	2.30	3.00
tnl	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
l	↑ 1.25	↖ 1.25	↑ 3.50	↖ 2.25	← 3.25	← 4.25	← 5.50	← 6.50	← 6.55	← 7.00
tc	1.25	1.25	0.00	0.00	1.00	0.00	1.25	2.25	2.30	3.00
tnl	0.00	1.00	0.00	1.00	2.00	0.00	1.00	2.00	2.05	3.00
e	↑ 2.25	↑ 2.25	↑ 4.50	↑ 3.25	↖ 3.25	← 4.25	↘ 5.50	↘ 6.50	← 6.55	↘ 7.00
tc	2.25	2.25	0.00	1.00	1.00	0.00	1.25	2.25	2.30	3.00
tnl	0.00	2.00	0.00	2.00	2.00	0.00	2.00	2.00	2.05	3.00
	↑ 3.25	← 4.50	↘ 5.50	↑ 4.25	↑ 4.25	↖ 3.25	← 4.50	← 5.50	← 5.55	← 6.00
tc	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
tnl	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
m	↑ 4.50	↖ 4.50	↖ 6.25	↑ 5.50	↘ 5.50	↑ 4.50	↖ 3.25	← 4.25	← 4.30	← 5.00
tc	1.25	1.25	0.00	1.25	1.25	0.00	0.00	1.00	1.05	2.00
tnl	0.00	1.00	0.00	1.00	2.00	0.00	1.00	2.00	2.05	3.00
o	↑ 5.50	↑ 5.50	↑ 7.25	↑ 6.50	↘ 6.50	↑ 5.50	↑ 4.25	↖ 3.25	← 3.30	← 4.00

Implementation by example

le mode vs a la mood

∅	← 0.00	← 1.25	← 2.25	← 3.50	← 4.50	← 5.50	← 6.75	← 7.75	← 7.80	← 8.00
tc	0.00	1.25	0.00	1.25	2.25	0.00	1.25	2.25	2.30	3.00
tnl	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
l	↑ 1.25	↖ 1.25	↑ 3.50	↖ 2.25	← 3.25	← 4.25	← 5.50	← 6.50	← 6.55	← 7.00
tc	1.25	1.25	0.00	0.00	1.00	0.00	1.25	2.25	2.30	3.00
tnl	0.00	1.00	0.00	1.00	2.00	0.00	1.00	2.00	2.05	3.00
e	↑ 2.25	↑ 2.25	↑ 4.50	↑ 3.25	↖ 3.25	← 4.25	↘ 5.50	↘ 6.50	← 6.55	↘ 7.00
tc	2.25	2.25	0.00	1.00	1.00	0.00	1.25	2.25	2.30	3.00
tnl	0.00	2.00	0.00	2.00	2.00	0.00	2.00	2.00	2.05	3.00
	↑ 3.25	← 4.50	↘ 5.50	↑ 4.25	↑ 4.25	↖ 3.25	← 4.50	← 5.50	← 5.55	← 6.00
tc	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
tnl	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
m	↑ 4.50	↖ 4.50	↖ 6.25	↑ 5.50	↘ 5.50	↑ 4.50	↖ 3.25	← 4.25	← 4.30	← 5.00
tc	1.25	1.25	0.00	1.25	1.25	0.00	0.00	1.00	1.05	2.00
tnl	0.00	1.00	0.00	1.00	2.00	0.00	1.00	2.00	2.05	3.00
o	↑ 5.50	↑ 5.50	↑ 7.25	↑ 6.50	↘ 6.50	↑ 5.50	↑ 4.25	↖ 3.25	← 3.30	← 4.00

Implementation by example

le mode vs a la mood

$t-5$	0.00	2.00	0.00	2.00	2.00	0.00	2.00	2.00	2.05	3.05
↑	3.25	← 4.50	↓ 5.50	↑ 4.25	↑ 4.25	↖ 3.25	← 4.50	← 5.50	← 5.55	← 6.55
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
↑	4.50	↖ 4.50	↖ 6.25	↑ 5.50	↔ 5.50	↑ 4.50	↖ 3.25	← 4.25	← 4.30	← 5.30
	1.25	1.25	0.00	1.25	1.25	0.00	0.00	1.00	1.05	2.05
	0.00	1.00	0.00	1.00	2.00	0.00	1.00	2.00	2.05	3.05
↑	5.50	↑ 5.50	↑ 7.25	↑ 6.50	↔ 6.50	↑ 5.50	↑ 4.25	↖ 3.25	← 3.30	← 4.30
	2.25	2.25	0.00	2.25	2.25	0.00	1.00	0.00	0.05	1.05
	0.00	2.00	0.00	2.00	2.00	0.00	2.00	2.00	2.05	3.05
↑	6.50	↑ 6.50	↑ 8.25	↑ 7.50	↔ 7.50	↑ 6.50	↑ 5.25	↑ 4.25	↖ 4.25	↖ 3.30
	3.25	3.25	0.00	3.25	3.25	0.00	2.00	1.00	1.00	0.05
	0.00	3.00	0.00	3.00	3.00	0.00	3.00	3.00	3.00	3.05
↑	7.50	↑ 7.50	↑ 9.25	↑ 8.50	↔ 8.50	↑ 7.50	↑ 6.25	↑ 5.25	↔ 5.25	↑ 4.30
	4.25	4.25	0.00	4.25	4.25	0.00	3.00	2.00	2.00	1.05
	0.00	4.00	0.00	4.00	4.00	0.00	4.00	4.00	4.00	4.00

Implementation by example

le mode vs a la mood

	ø	a	l	a	m	o	o	d			
ø	← 0.00	← 1.25	← 2.25	← 3.50	← 4.50	← 5.50	← 6.75	← 7.75	← 7.80	← 8.80	
tc	0.00	1.25	0.00	1.25	2.25	0.00	1.25	2.25	2.30	3.30	
tnl	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
l	↑ 1.25	↖ 1.25	↑ 3.50	↖ 2.25	← 3.25	← 4.25	← 5.50	← 6.50	← 6.55	← 7.55	min: 1.25 + 0.25
tc	1.25	1.25	0.00	0.00	1.00	0.00	1.25	2.25	2.30	3.30	
tnl	0.00	1.00	0.00	1.00	2.00	0.00	1.00	2.00	2.05	3.05	
e	↑ 2.25	↑ 2.25	↑ 4.50	↑ 3.25	↖ 3.25	← 4.25	↘ 5.50	↘ 6.50	← 6.55	↘ 7.55	min: 2.25 + 0.25
tc	2.25	2.25	0.00	1.00	1.00	0.00	1.25	2.25	2.30	3.30	
tnl	0.00	2.00	0.00	2.00	2.00	0.00	2.00	2.00	2.05	3.05	
	↑ 3.25	← 4.50	↘ 5.50	↑ 4.25	↑ 4.25	↖ 3.25	← 4.50	← 5.50	← 5.55	← 6.55	min: 3.25 + 0.25
tc	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
tnl	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
m	↑ 4.50	↖ 4.50	↖ 6.25	↑ 5.50	↘ 5.50	↑ 4.50	↖ 3.25	← 4.25	← 4.30	← 5.30	min: 3.25 + 0.25
tc	1.25	1.25	0.00	1.25	1.25	0.00	0.00	1.00	1.05	2.05	
tnl	0.00	1.00	0.00	1.00	2.00	0.00	1.00	2.00	2.05	3.05	
o	↑ 5.50	↑ 5.50	↑ 7.25	↑ 6.50	↘ 6.50	↑ 5.50	↑ 4.25	↖ 3.25	← 3.30	← 4.30	min: 3.25 + 0.25
tc	2.25	2.25	0.00	2.25	2.25	0.00	1.00	0.00	0.05	1.05	
tnl	0.00	2.00	0.00	2.00	2.00	0.00	2.00	2.00	2.05	3.05	
d	↑ 6.50	↑ 6.50	↑ 8.25	↑ 7.50	↘ 7.50	↑ 6.50	↑ 5.25	↑ 4.25	↖ 4.25	↖ 3.30	min: 3.30 + 0.25
tc	3.25	3.25	0.00	3.25	3.25	0.00	2.00	1.00	1.00	0.05	
tnl	0.00	3.00	0.00	3.00	3.00	0.00	3.00	3.00	3.00	3.05	
e	↑ 7.50	↑ 7.50	↑ 9.25	↑ 8.50	↘ 8.50	↑ 7.50	↑ 6.25	↑ 5.25	↖ 5.25	↑ 4.30	min: 4.30 + 0.25
tc	4.25	4.25	0.00	4.25	4.25	0.00	3.00	2.00	2.00	1.05	
tnl	0.00	4.00	0.00	4.00	4.00	0.00	4.00	4.00	4.00	4.00	

Links of potential interest

- ❑ Edit distance
 - ❑ Levenshtein_distance
 - ❑ Damerau–Levenshtein distance
- ❑ Dynamic programming
- ❑ Soundex
- ❑ Delannoy number
 - ❑ 63 paths through a 3x3 grid
- ❑ The Cyrillic letter И
- ❑ Jaccard similarity
- ❑ Blog post on name matching (including И)
- ❑ Initial write up on Glent Method 1, including patterns and anti-patterns
- ❑ Glent Phab Epic (with links to M0, M1, M2)



Thank you for coming to my

TAED Talk

Appendices



WIKIMEDIA
FOUNDATION

Delannoy numbers formula

$$D(m, n) = \sum_{k=0}^{\min(m, n)} \binom{m}{k} \binom{n}{k} 2^k$$



How many edit paths

We can use the same dynamic programming approach to compute the total number of optimal paths through an edit distance table.

		e	f	f	i	c	i	e	n	c	y
	0	1	2	3	4	5	6	7	8	9	10
d	1	1	2	3	4	5	6	7	8	9	10
y	2	2	2	3	4	5	6	7	8	9	9
n	3	3	3	3	4	5	6	7	7	8	9
a	4	4	4	4	4	5	6	7	8	8	9
m	5	5	5	5	5	5	6	7	8	9	9
i	6	6	6	6	5	6	5	6	7	8	9
c	7	7	7	7	6	5	6	6	7	7	8
i	8	8	8	8	7	6	5	6	7	8	8
t	9	9	9	9	8	7	6	6	7	8	9
y	10	10	10	10	9	8	7	7	7	8	8

