

Auto correct → changes **misspelled word** ke kata yang benar.

Dahlah tau, contohnya

**Ak** bsa kok bnt km → Dengan autocorrect → Aku bisa kok bantu kamu.

Ini Step step Autocorrect

1. Identify a misspelled word
2. Find strings n edit distance away
3. Filter candidates
4. Calculate word probabilities

deah  
yeah  
[dear]  
dean  
... etc

Identifikasi misspelled wordnya, setelah itu cari strings yang total N edit distance, ambil dah kata kata itu. Filter dengan melakukan perhitungan **Kesamaan** antar kandidat kata tadi, ambil yang probabilitasnya tinggi.

So kita breakdown satu Saturday

1. Identifikasi Misspelled words

So gimana caranya identifikasi misspelled ?

Salah satunya :

Kita kan punya **vocabulary**, kalau kata tersebut tidak ada di **vocabulary** ya kemungkinan itu misspelled words. Pendekatan ini ga mempertimbangkan **konteks** dari pada kalimat ya.

2. Cari string yang totalnya N edit distance

**Edit** : sebuah operasi yang dilakukan pada sebuah string

**Operasi** – operasi :

- Insert. Awal katanya **“to”**, dengan insert → **“top”, “two”, ...**
- **Delete**, Hapus satu **huruf**, awal katanya **“hat”**, dengan delete → **“ha”, “at”, “at”**
- **Switch**, **ubah 2 kata** yang bersebelahan.
- Replace, ubah 1 huruf di dalam 1 kata, awal katanya **“jaw”**, dengan replace → **“jar”, “paw”, ...**

3. Filter candidates

Dari seluruh kata yang di bentuk di step 2, y akita pengen keep **actual words** yang make sense. I,e ada di **Vocabulary**

4. Calculate word probabilities

## Calculate word probabilities

Example: "I am happy because I am learning"

$$P(w) = \frac{C(w)}{V}$$

$$P(\text{am}) = \frac{C(\text{am})}{V} = \frac{2}{7}$$

$P(w)$  Probability of a word

$C(w)$  Number of times the word appears

$V$  Total size of the corpus

Word	Count
I	2
am	2
happy	1
because	1
learning	1

Total: 7

Corpus itu kek total kata di **vocabulary** 😊, biasa suka ngebuat bingung

But untuk evaluasi **similarity** antara 2 strings? **Tentu ga pake lagi cost similarity / Euclidean distance**, ya karena disini representasinya berupa **Kata** bukan **vector**, so kita gunain **MINIMUM EDIT DISTANCE**

**Minimum Edit Distance** → **Minimum Number of edits** yang dibutuhkan untuk melakukan transformasi

**Benefitnya :**

1. Evaluasi Kesamaan antar 2 kata
2. Dapat mencari total edit minimum antara 2 kata

**Oke kita ke contoh : ini gunain dynamic programming yak**

Karena kita ingin mehingkung **similarity** antar 2 kata, so kita buat dulu tabelnya :

Source : Kata 1 : play

Cost : Insert = 1

Target : Kata 2 : stay

Delete = 1 , replace = 2 , MAX EDIT COST = 4

	#	S	T	A	Y
#	0	1	2	3	4
P	1	2	3	4	
L	2	3			
A	3				
Y	4				

Ini sebenarnya harus pahami videonya dlu

# = Empty string, -> = diubah

# -> # = Cost Editnya 0 , iyalah

# -> S = Cost Editnya 1, insert. Dari Empty String jadi S

P -> # = Cost Editnya 1, delete. Dari P jadi Empty String

Sekarang P ingin di ubah ke S, ada beberapa kemungkinan yang bisa dilakukan

Insert + delete = P+S → S , Cost 2

Delete + insert = # → # + S, Cost 2

Replace = # → S , Cost 2

Dari ketiga hal diatas ambil yang minimumnya, 2 disini

P → ST

Insert + Insert + delete = PS → PST → ST , cost 3

Delete + insert + insert = P → S → ST, cost 3

Replace + insert = S → ST, cost 3

P → STA

Insert + insert + insert + delete = PS → PST → PSTA → STA, cost 4

Delete + insert + insert + insert = P → S → ST → STA, cost 4

Replace + insert + insert = S → ST → STA , cost 4

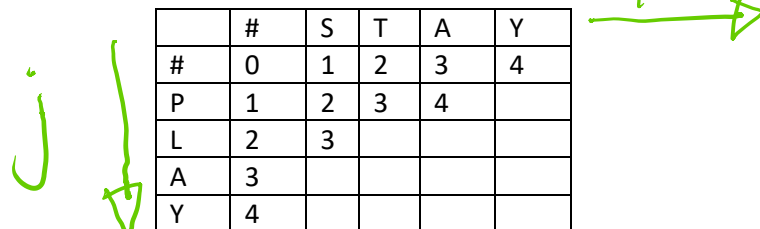
PL → S

Insert + delete + delete = PLS → PS → S , cost 3

Delete + Delete + insert = P → L → S, cost 3

Replace + delete = PS → S, cost 3

Panjang gw males, disini kita dapet rumus boy Gw kasih tanda yak



	#	S	T	A	Y
#	0	1	2	3	4
P	1	2	3	4	
L	2	3			
A	3				
Y	4				

So untuk ngisi tabel [i,j] kita butuh 3 nilai

Nilai Atas

Nilai sebelah kiri

Nilai sebelah kiri atas ( warna kuning )

Source itu yang #Stay

Target itu yang #Target

Untuk menentukan nilai ke [i,j]

Kita bisa nulis →

Nilai 1 : nilai[i - 1, j] + delete cost

Nilai 2 : nilai[i, j-1] + insert cost

Nilai 3 : nilai[i-1,j-1] + rep cost {0, kalau misalnya source[i] == target[j],

Rep\_cost ,kalau misalnya source[i] != target[j] }

Hasil akhirnya kek gini :

	0	1	2	3	4	
	#	s	t	a	y	
0	#	0	1	2	3	4
1	p	1	2	3	4	5
2	l	2	3	4	5	6
3	a	3	4	5	4	5
4	y	4	5	6	5	4

Oh iya, ini dynamic programming namanya, menyelesaikan sub problem terkecil dahulu yang dikembangkan untuk menyelesaikan permasalahan yang besar.

