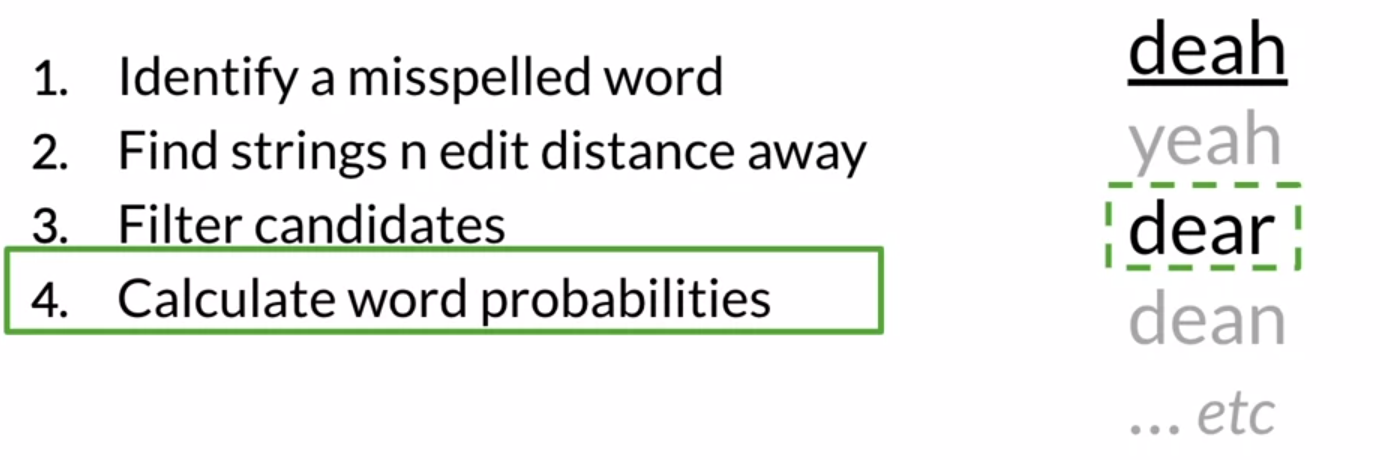
Auto correct 🡪 changes **misspelled word** ke kata yang benar.

Dahlah tau, contohnya

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

Ini Step step Autocorrect



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.

1. Cari string yang totalnya N edit distance

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

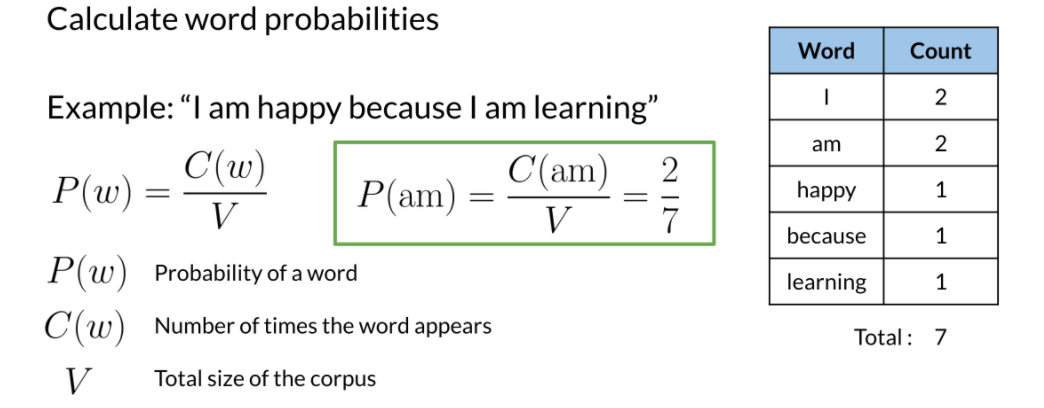
**Operasi** – operasi :

* 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”,…**

1. 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**

1. Calculate word probabilities



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 reprsentasinya 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 pahamin 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

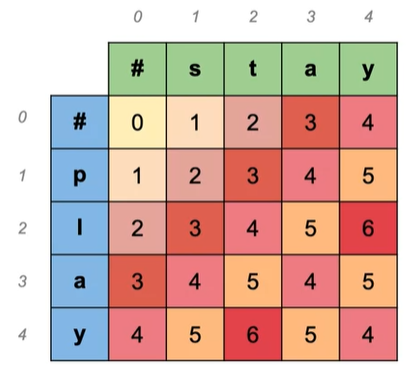


Nilai 2 : nilai[I, j-1] + insert cost



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 :  


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