EDIT DISTANCE

THEORY & APPLICATIONS OF NLP
WOA7013

SPELLING ERRORS DETECTION & CORRECTION

non-word error detection: detecting spelling errors which result in non-words (e.g. *tomrrow* for *tomorrow*).

isolated-word error correction: correcting spelling errors which result in non-words, e.g. correcting *tomrrow* to *tomorrow*, but looking only at the word in isolation.

context-dependent error detection and correction: Using the context to help detect and correct spelling errors even if they accidentally result in an actual word (real-word errors). This REAL-WORD ERRORS can happen from typographical errors (insertion, deletion, transposition) which accidently produce a real word (e.g. there for three), or because the writer substituted the wrong spelling of a homophones (e.g. piece for peace).

Kukich (1992)

INTRODUCTION

How similar are 2 strings?

- Spell Checker/Correction
  - The user typed “markt”. Which is closest?
    - *mark*
    - *make*
    - *market*
- Computational Biology
  - Align 2 sequences of nucleotides

    AGGCTATCACCTGACC

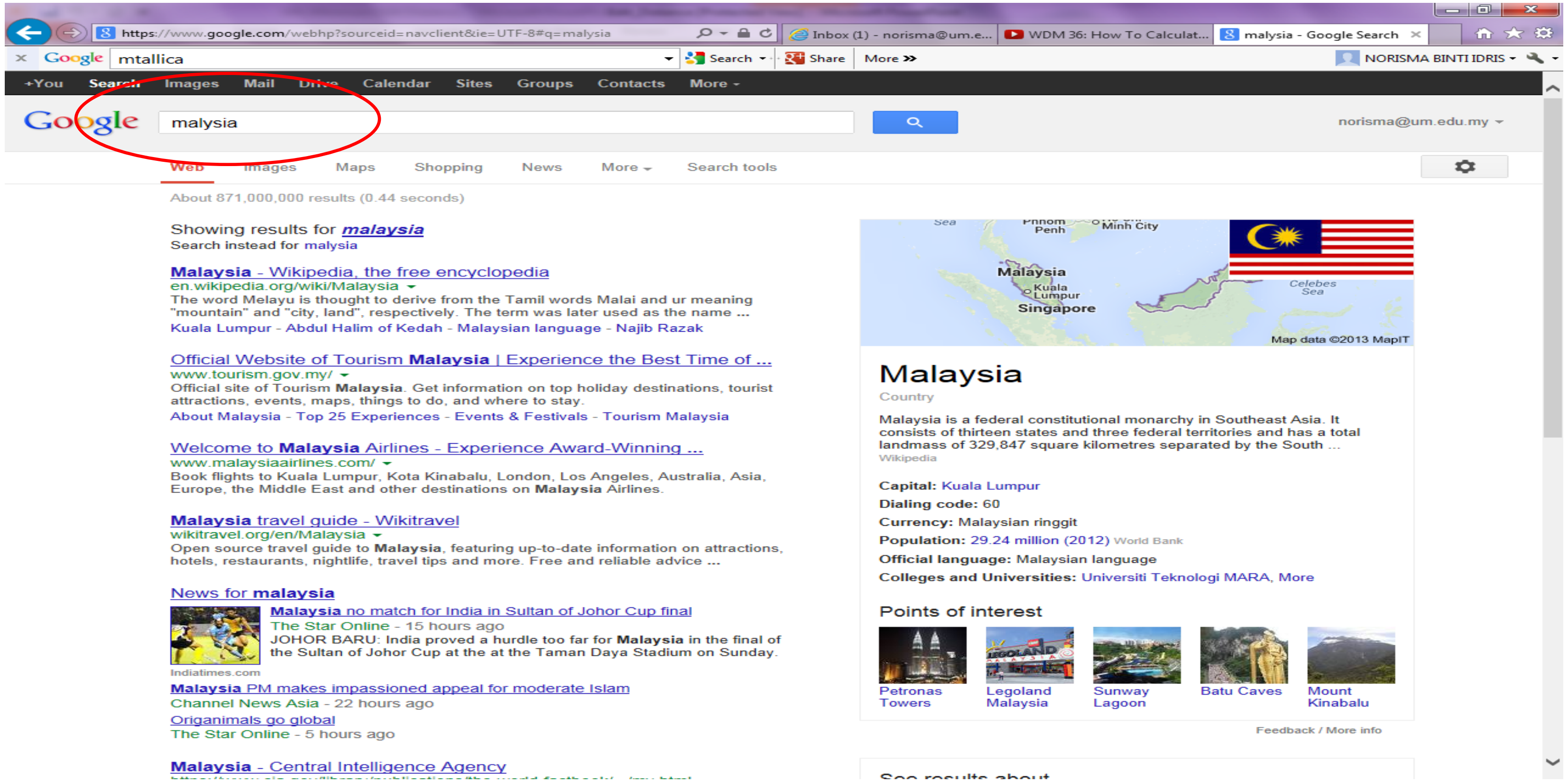
    TAGCTATCAGCACCGC
  - Resulting alignment

    -AGGCTATCACCTGACC--

    TAG-CTATCAC--GACCGC

REAL APPLICATION

Searching keywords via the net: usually by “malysia”, we mean “malaysia”



MINIMUM EDIT DISTANCE

Edit distance, also known as **Levenshtein distance** or **Evolutionary distance.**

The min edit distance between 2 strings is the min number of editing operations needed to transform one string into another.

- *Given 2 strings, s_1 and s_2 , the minimum number of operations/edits to change from one string to another.*

Operations/edits are typically character-level:

- Deletion (d)
- Insertion (i)
- Substitution (s)

It is the most common measure to expose the dissimilarity between two strings (Levenshtein 1966; Navarro & Raffinot 2002).

For example:

- To find the distance btw 2 words:

Source string	I	N	T	E	*	N	T	I	O	N
Target string	*	E	X	E	C	U	T	I	O	N
	d	s	s		i	s				

The operations involved

The transformation list from intention to execution:

intention	
ntention	- delete i
etention	- substitute *n* by *e*
exention	- substitute *t* by *x*
exenution	- insert u
execution	- substitute *n* by *c*

We can assign a particular cost/weight to each of these operations.

- If each operation has cost of 1
  - Distance between these is 5
- If substitutions cost 2
  - Distance between them is 8

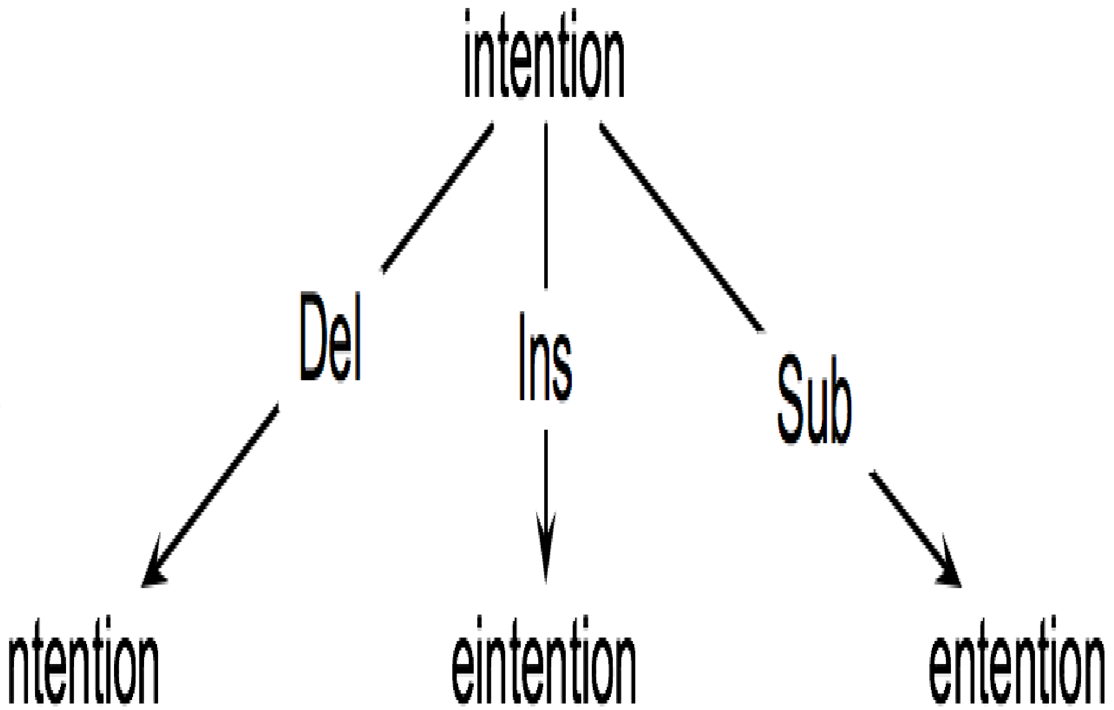
Determine the minimum edit distance between these 2 strings:

1. don and dog = ?
2. cat and act = ?
3. cat and dog = ?

HOW TO FIND A MIN EDIT DISTANCE

Searching for a path (sequence of edit) from the start string to the final string:

- **Initial state**: the word we're transforming
- **Operators**: *D, I, S*
- **Goal state**: the word we're trying to get to
- **Path cost**: what we want to minimize: the number of edits



MIN EDIT AS SEARCH

The space of all edit sequences is huge.

- Lots of possible sequences wind up at the same state
  - We don’t have to keep track all of them
  - Just the shortest path to each of those revisited states.

DEFINING MIN EDIT DISTANCE

For two strings

- X of length n
- Y of length m

We define D(i,j)

- the edit distance between X[1..i] and Y[1..j]
  - i.e., the first i characters of X and the first j characters of Y
- The edit distance between X and Y is thus D(n,m)

The base cases for edit distance function:

1. $D(0,0) = 0$

$s_1 = 0$ character and $s_2 = 0$. Thus, no operations involved

2. $D(1,0) = 1$

$s_1 = 1$ character and $s_2 = 0$. Thus, only 1 operation involved

3. $D(i,0) = i$

$s_1 = i$ character and $s_2 = 0$. Thus, i operations involved

4. $D(0,j) = j$

$s_1 = 0$ character and $s_2 = j$. Thus, j operations involved

COMPUTING MIN EDIT DISTANCE

Dynamic programming: A tabular computation of D(n,m)

Solving problems by combining solutions to subproblems.

Bottom-up

- We compute $D(i,j)$ for small i,j
- And compute larger $D(i,j)$ based on previously computed smaller values
- i.e., compute D(i,j) for all i (0 < i < n) and j (0 < j < m)

MIN EDIT DISTANCE ALGORITHM

Initialization

$D(i, 0) = i$

$D(0, j) = j$

Recurrence Relation

$For\ each\ i = 1 \ldots m$

$For\ each\ j = 1 \ldots n$

$$D(i,j) = min \begin{cases} D(i-1,j)+1 & \text{deletion} \\ D(i,j-1)+1 & \text{insertion} \\ D(i-1,j-1)+\begin{cases} 2; & if\ X(i) \neq Y(j) \\ 0; & if\ X(i) = Y(j) \end{cases} & \text{substitution} \end{cases}$$

Termination

$D(m, n)$ is the distance

THE EDIT DISTANCE TABLE

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1	2	3	4	5	6	7	6	7	8
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1									
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases}$$

N	9	8	9	10	11	12	11	10	9	**8**
O	8	7	8	9	10	11	10	9	8	9
I	7	6	7	8	9	10	9	8	9	10
T	6	5	6	7	8	9	8	9	10	11
N	5	4	5	6	7	8	9	10	11	10
E	4	3	4	5	6	7	8	9	10	9
T	3	4	5	6	7	8	7	8	9	8
N	2	3	4	5	6	7	8	7	8	7
I	1	2	3	4	5	6	7	6	7	8
#	**0**	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases}$$