Natural Language Processing

2: Tokenization and Segmentation

Quiz

(1) Q1: What is language?

(3) Q2: Describe two parts of the BPE (Byte Pair Encoding) algorithm

Objectives

to be able to answer questions

(1) Tokenization

(2) Segmentation

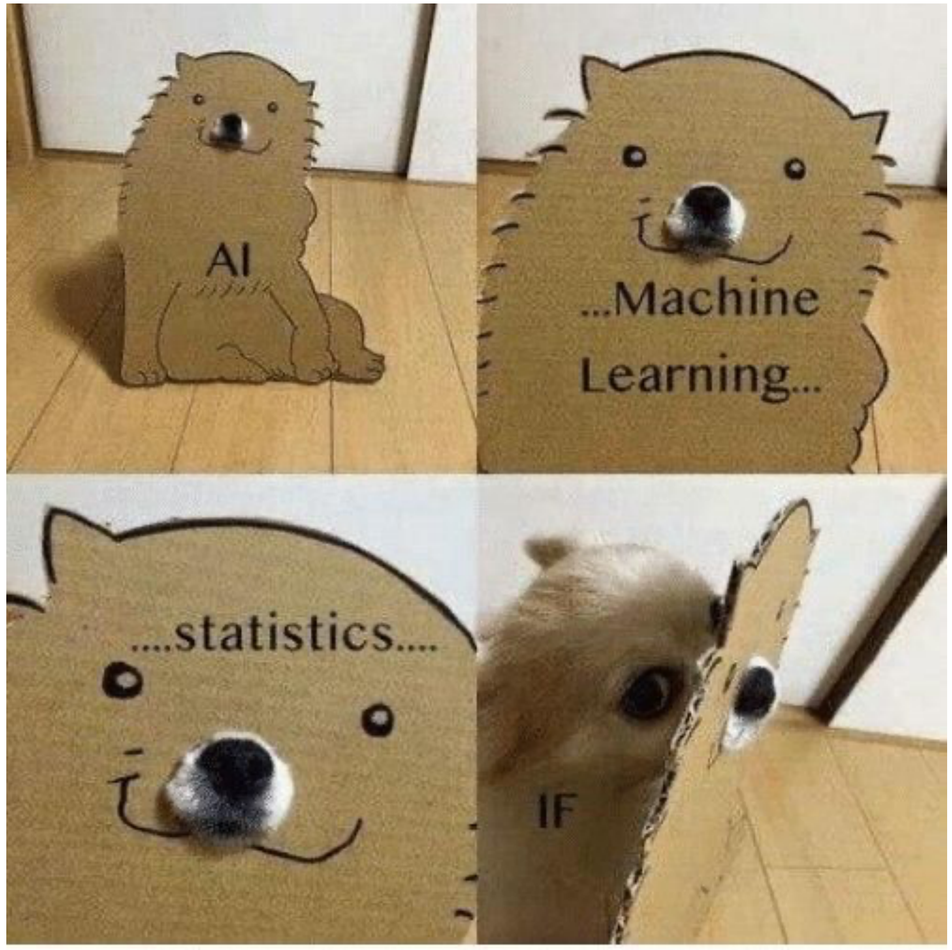
(3) Edit Distance

Objectives

(1) To be able to distinguish between a corpus / dataset / text collection

(2) To understand Sentence Segmentation

(3) To understand how Edit Distance method work



IF IF IF IF IF IF IF IF IF WE!

Words and Corpora

Corpora

corpora = corpuses

Words don't appear out of nowhere!

A text is produced by

(1) a specific writer(s),

(2) at a specific time,

(3) in a specific variety,

(4) of a specific language,

(5) for a specific function.

Corpora vary along dimension

- Language: 7097 languages in the world
- Variety: like African American Language varieties.
  - AAE Twitter posts might include forms like "iont" (I don't)
- Code switching, e.g., Spanish/English, Hindi/English:

  S/E: Por primera vez veo a @username actually being hateful! It was beautiful:)
  [For the first time I get to see @username actually being hateful! it was beautiful:)]

  H/E: dost tha or ra- hega ... dont wory ... but dherya rakhe
  [“he was and will remain a friend ... don’t worry ... but have faith”]

- Genre: newswire, fiction, scientific articles, Wikipedia
- Author Demographics: writer's age, gender, ethnicity

Corpus datasheets

Motivation:

(1) Why was the corpus collected?
(2) By whom?
(3) Who funded it?

Situation: In what situation was the text written?

Collection process: If it is a subsample how was it sampled? Was there consent? Pre-processing

Annotation process, language variety, demographics, etc.

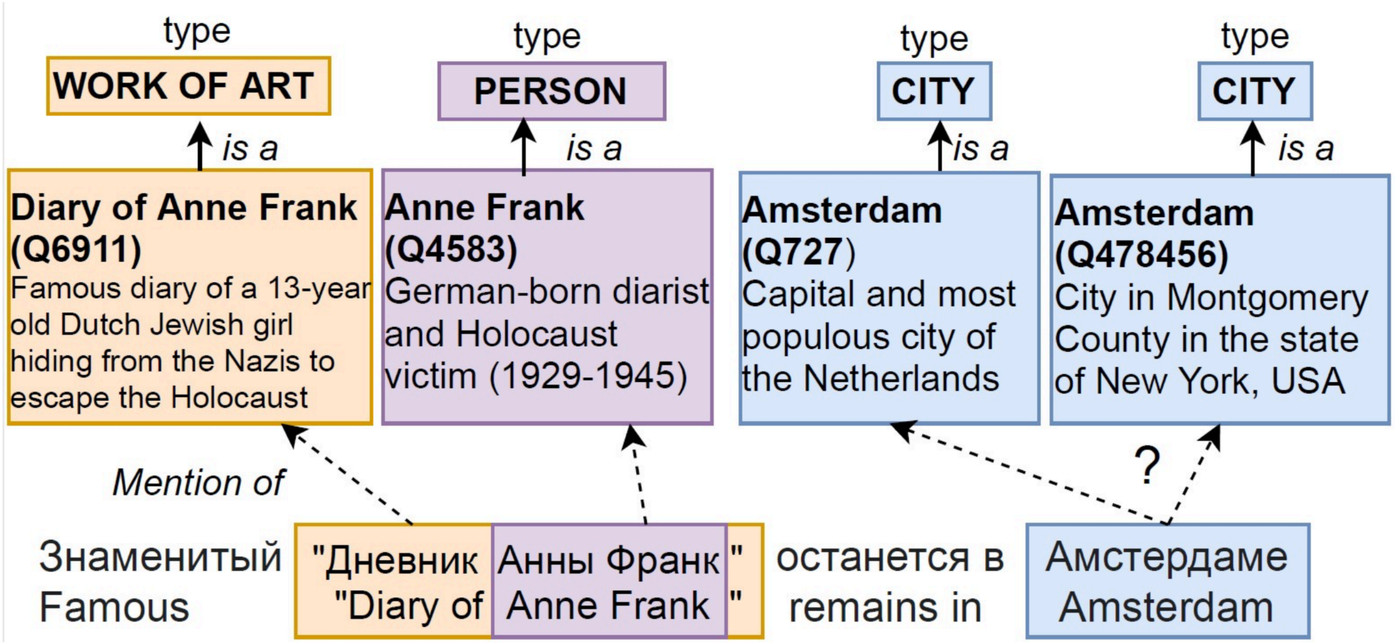
Example: NEREL

(1) Nested named entities

(2) Events

(3) Relations

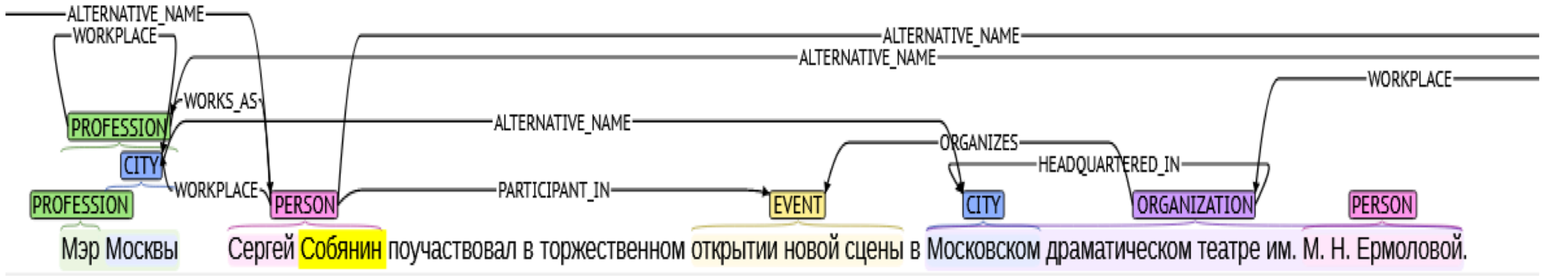
(4) Links to Wikidata



(6) 29 entity and 49 relation types

(7) 56K+ entities

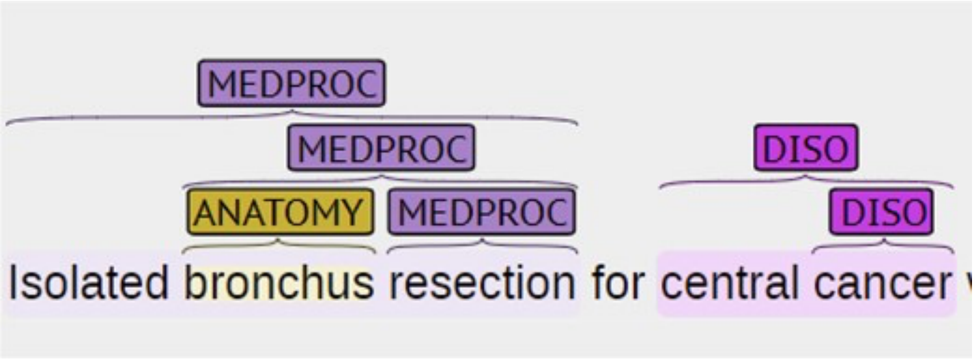
(8) 39K+ relations

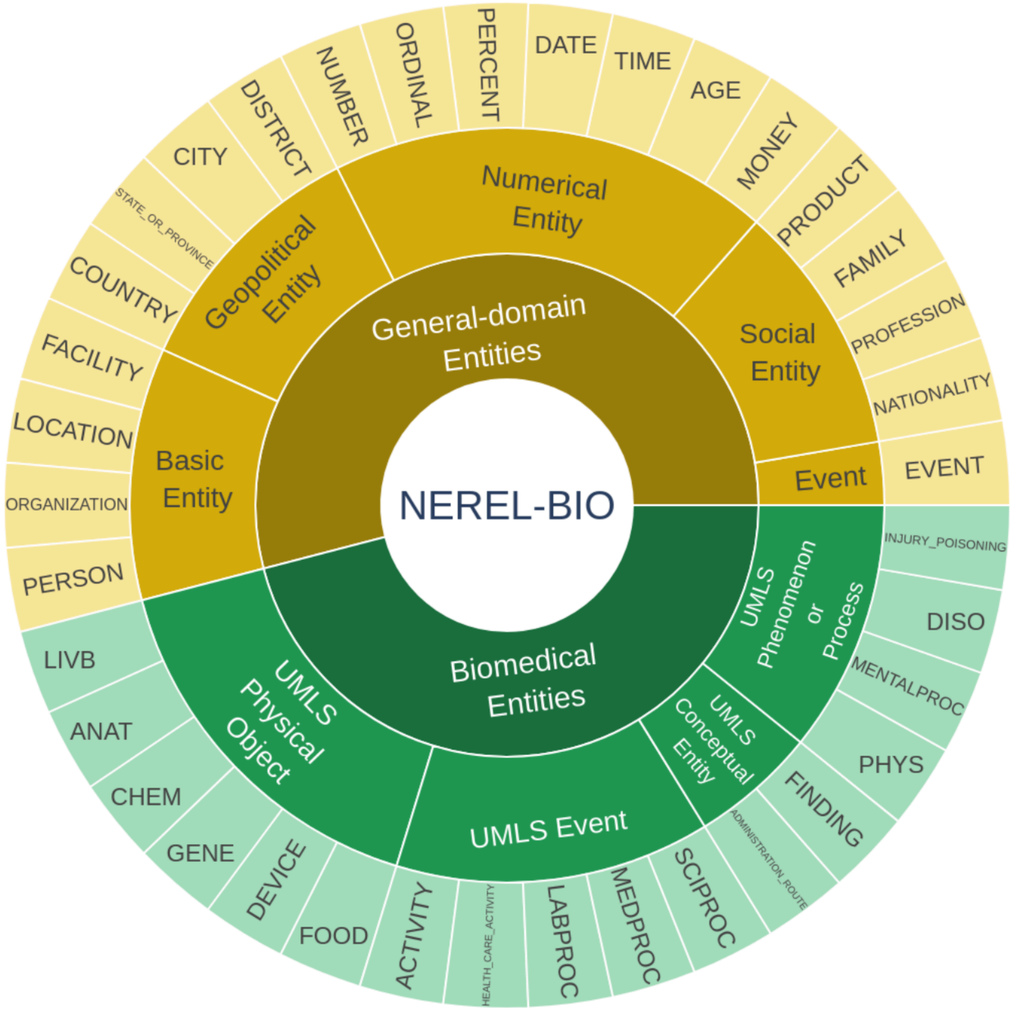


https://github.com/nerel-ds/NEREL/tree/master

NEREL-BIO

Biomedical Corpus for Nested Named





https://github.com/nerel-ds/NEREL-BIO

Loukachevitch N., Manandhar S., Baral E., Rozhkov I., Braslavski P., Ivanov V., Batura T., Tutubalina E. NEREL-BIO: a dataset of biomedical abstracts annotated with nested named entities. Bioinformatics. 2023. Volume 39, Issue 4, btad161. https://doi.org/10.1093/bioinformatics/btad161

Datasets vs. Corpora vs. Text Collections

(1) Discussion: What is the difference?

Text Normalization

Every NLP task requires text normalization:

1. Tokenizing (segmenting) words
2. Normalizing words
3. Segmenting sentences

Space-based tokenization

(1) A very simple way to tokenize
    (1) For languages that use space characters between words
    (2) Segment off a token between instances of spaces
(2) Unix tools for space-based tokenization
    (1) The "tr" command
    (2) Inspired by Ken Church's UNIX for Poets
    (3) Given a text file, output the word tokens and their frequencies

Simple Tokenization in UNIX

(1)(Inspired by Ken Church's UNIX for Poets.)

(2)Given a text file, output the word tokens and their frequencies

```
tr -sc 'A-Za-z' '\n' < shakes.txt | sort | uniq –c
```

```
1945 A
  72 AARON
  19 ABBESS
   5 ABBOT
 ... ...
```

```
25 Aaron
 6 Abate
 1 Abates
 5 Abbess
 6 Abbey
 3 Abbot
... …
```

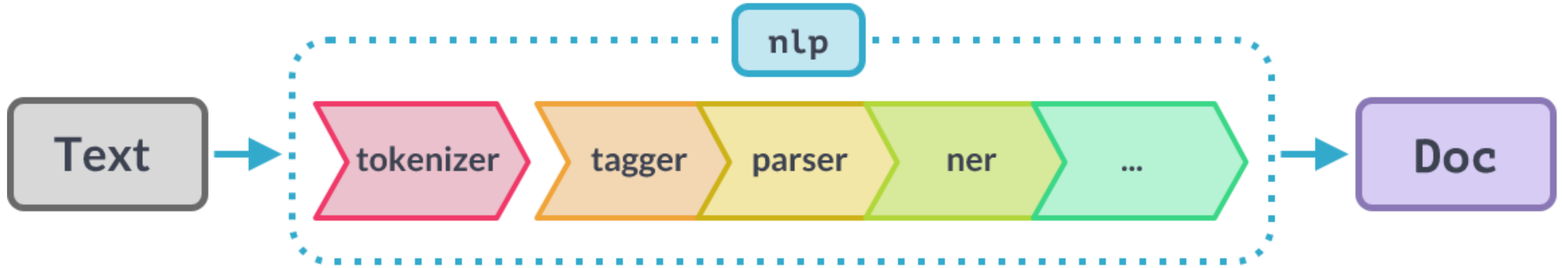
Issues in Tokenization

(1)Can't just blindly remove punctuation:
    (1)m.p.h., Ph.D., AT&T, cap'n
    (2)prices ($45.55)
    (3)dates (01/02/06)
    (4)URLs (http://www.stanford.edu)
    (5)hashtags (#nlproc)
    (6)email addresses (someone@cs.colorado.edu)

(2)Clitic: a word that doesn't stand on its own
    (1)"are" in we're, French "je" in j'ai, "le" in l'honneur

(3)When should multiword expressions (MWE) be words?
    (1)New York, rock 'n' roll

Tokenization in Spacy




```
import spacy
nlp = spacy.load("en_core_web_md")
doc = nlp("We are learning NLP using spaCy.")
print ([token.text for token in doc])
```

✓ 2.6s

```
['We', 'are', 'learning', 'NLP', 'using', 'spaCy', '.']
```

https://medium.com/analytics-vidhya/nlp-with-spacy-tutorial-part-2-tokenization-and-sentence-segmentation-352df790a214

https://machinelearningknowledge.ai/complete-guide-to-spacy-tokenizer-with-examples/

Tokenization in languages without spaces

Many languages (like Chinese, Japanese, Thai) don't use spaces to separate words!

How do we decide where the token boundaries should be?

Word tokenization in Chinese

Chinese words are composed of characters called "**hanzi**" (or sometimes just "**zi**")

Each one represents a meaning unit called a morpheme.

Each word has on average 2.4 of them.

But deciding what counts as a word is complex and not agreed upon.

李叶真的跟她的妈妈不一样，她看起来又隘又屌，还经常生病。她总是喜欢生气，生气的时候总是媸。如果李叶的妈妈听到她媸，就会很生气。所有的人都不喜欢这个孩子，他们从来没有让过这样的孩子。为了不让李叶媸，她的阿姨总是很听李叶的话。李叶资弦什么，她的阿姨就朐她什么。李叶觉得在这个家里只有她的阿姨关心她。

How to do word tokenization in Chinese?

the table down there

姚明进入总决赛 “Yao Ming reaches the finals”

3 words?
姚明 进入 总决赛
YaoMing reaches finals

5 words?
姚 明 进入 总 决赛
Yao Ming reaches overall finals

7 characters? (don't use words at all):
姚 明 进 入 总 决 赛
Yao Ming enter enter overall decision game

Word tokenization / segmentation

In Chinese it's common to just treat each character (zi) as a token.

(1)So the **segmentation** step is very simple

In other languages (like Thai and Japanese), more complex word segmentation is required.

(2)The standard algorithms are **neural sequence models** trained by supervised machine learning.

Byte Pair Encoding

Another option for text tokenization

Instead of

(1)white-space segmentation

(2)single-character segmentation

Use the data to tell us how to tokenize.

Subword tokenization (because tokens can be parts of words as well as whole words)

Subword tokenization

(1) All have 2 parts:

    (1) A token **learner**

    (2) A token **segmenter**

Word Normalization and other issues

Word Normalization

(1)Putting words/tokens in a standard format

(1)U.S.A. or USA

(2)uhhuh or uh-huh

(3)Fed or fed

(4)am, is, be, are

Case folding

(1) Applications like IR: reduce all letters to lower case

  (1) Since users tend to use lower case

  (2) Possible exception: upper case in mid-sentence?

    (1) e.g., ***General Motors***

    (2) ***Fed*** vs. ***fed***

    (3) ***SAIL*** vs. ***sail***

(2) For sentiment analysis, MT, Information extraction

  (1) Case is helpful (***US*** versus ***us*** is important)

Lemmatization

Represent all words as their lemma, their shared root

= dictionary headword form:

(1)am, are, is → be

(2)car, cars, car's, cars' → car

(3)Spanish quiero ('I want'), quieres ('you want')

→ querer 'want'

(1)He is reading detective stories

→ He be read detective story

Lemmatization is done by Morphological Parsing

(1)Morphemes:

(1)The small meaningful units that make up words

(2)Stems: The core meaning-bearing units

(3)Affixes: Parts that adhere to stems, often with grammatical functions

(2)Morphological Parsers:

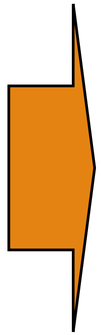
(1)Parse *cats* into two morphemes *cat* and *s*

(2)Parse Spanish *amaren* ('if in the future they would love') into morpheme *amar* 'to love', and the morphological features *3PL* and *future subjunctive*.

Stemming

Reduce terms to stems, chopping off affixes crudely

This was not the map we found in Billy Bones's chest, but an accurate copy, complete in all things-names and heights and soundings-with the single exception of the red crosses and the written notes.



Thi wa not the map we found in Billi Bone s chest but an accur copi complet in all thing name and height and sound with the singl except of the red cross and the written note
.

Dealing with complex morphology is necessary for many languages

(1) e.g., the Turkish word:

(2) Uygarlastiramadiklarimizdanmissinizcasina

(3) `(behaving) as if you are among those whom we could not civilize'

(4) Uygar `civilized' + las `become'

+ tir `cause' + ama `not able'

+ dik `past' + lar 'plural'

+ imiz 'p1pl' + dan 'abl'

+ mis 'past' + siniz '2pl' + casina 'as if'

Sentence Segmentation

Sentence Segmentation

!, ? mostly unambiguous but **period** "." is very ambiguous

(1) Sentence boundary

(2) Abbreviations like Inc. or Dr.

(3) Numbers like .02% or 4.3

Common algorithm: Tokenize first: use rules or ML to classify a period as either (a) part of the word or (b) a sentence-boundary.

(4) An abbreviation dictionary can help

Sentence segmentation can then often be done by rules based on this tokenization.

Simple rule-based

fast and dirty

(1) Use certain characters for boundaries

(2) . ? !

(3) + extra features

  (1) case distinctions

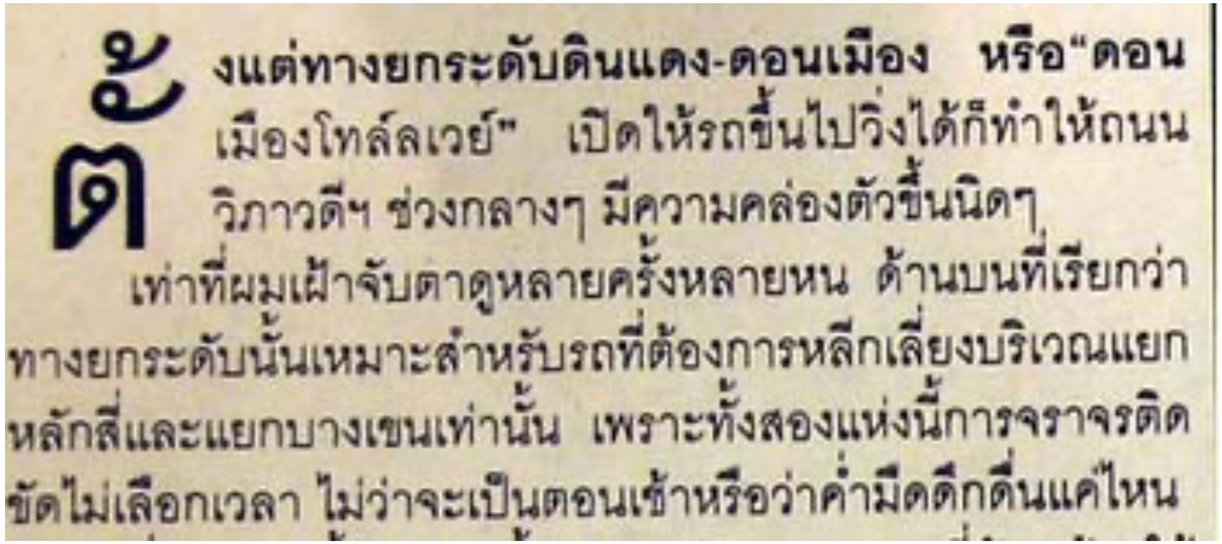
  (2) part of speech

  (3) word length

  (4) lexical endings

  (5) prefixes and suffixes

  (6) abbreviation classes


ตั้งแต่ทางยกระดับดินแดง-ดอนเมือง หรือ"ดอนเมืองโทล์ลเวย์" เปิดให้รถขึ้นไปวิ่งได้ก็ทำให้ถนนวิภาวดีฯ ช่วงกลางๆ มีความคล่องตัวขึ้นนิดๆ เท่าที่ผมเฝ้าจับตาดูหลายครั้งหลายหน ด้านบนที่เรียกว่าทางยกระดับนั้นเหมาะสำหรับรถที่ต้องการหลีกเลี่ยงบริเวณแยกหลักสี่และแยกบางเขนเท่านั้น เพราะทั้งสองแห่งนี้การจราจรติดขัดไม่เลือกเวลา ไม่ว่าจะเป็นตอนเช้าหรือว่าค่ำมืดดึกดื่นแค่ไหน

Issues

how many sentences?

(8) *There was nothing so VERY remarkable in that; nor did Alice think it so VERY much out of the way to hear the Rabbit say to itself, 'Oh dear! Oh dear! I shall be late!' (when she thought it over afterwards, it occurred to her that she ought to have wondered at this, but at the time it all seemed quite natural); but when the Rabbit actually TOOK A WATCH OUT OF ITS WAISTCOAT-POCKET, and looked at it, and then hurried on, Alice started to her feet, for it flashed across her mind that she had never before seen a rabbit with either a waistcoat-pocket, or a watch to take out of it, and burning with curiosity, she ran across the field after it, and fortunately was just in time to see it pop down a large rabbit-hole under the hedge.*

from Lewis Carroll's Alice in Wonderland

Issues

embedded sentences

(9) *The holes certainly were rough - "Just right for a lot of vagabonds like us," said Bigwig - but the exhausted and those who wander in strange country are not particular about their quarters.*

taken from (Adams, 1972)

Simple rule-based

fast and dirty

(1) Use certain characters for boundaries

(2) . ? !

(3) A simple rule applied on the Wall Street Journal Corpus with 16466 periods as sentence boundaries

```
IF (right context = period + space + capital letter
                    OR period + quote + space + capital letter
                    OR period + space + quote + capital letter)
THEN sentence boundary
```

(7) This simple rule would detect only 14562 (88.4%)

(8) while producing **2900 false positives**, placing a boundary where one does not exist

Simple rule-based

fast and dirty

(1) Use certain characters for boundaries

(2) . ? !

(3) An updated rule applied on the WSJ Corpus with 16466 periods as sentence boundaries

```
IF ((right context = period + space + capital letter
                     OR period + quote + space + capital letter
                     OR period + space + quote + capital letter)
    AND (left context != abbreviation))
THEN sentence boundary
```

(7) while producing 283 false positives,

(8) it also introduces **713 false negatives**

ML-based solutions

since (guess, which year?)

(1) How would you apprach the sentence segmentation from the ML perspective?

Break, 5 min.

Edit Distance

Minimum Edit Distance

How similar are two strings?

Spell correction

- The user typed “graffe”

Which is closest?

- graf
- graft
- grail
- giraffe

- Computational Biology
  - Align two sequences of nucleotides

```
AGGCTATCACCTGACCTCCAGGCCGATGCCC
TAGCTATCACGACCGCGGTCGATTTGCCCGAC
```

  - Resulting alignment:

```
—AGGCTATCACCTGACCTCCAGGCCGA--TGCCC---
TAG-CTATCAC--GACCGC--GGTCGATTTGCCCGAC
```

- Also for Machine Translation, Information Extraction, Speech Recognition

Edit Distance

The minimum edit distance between two strings

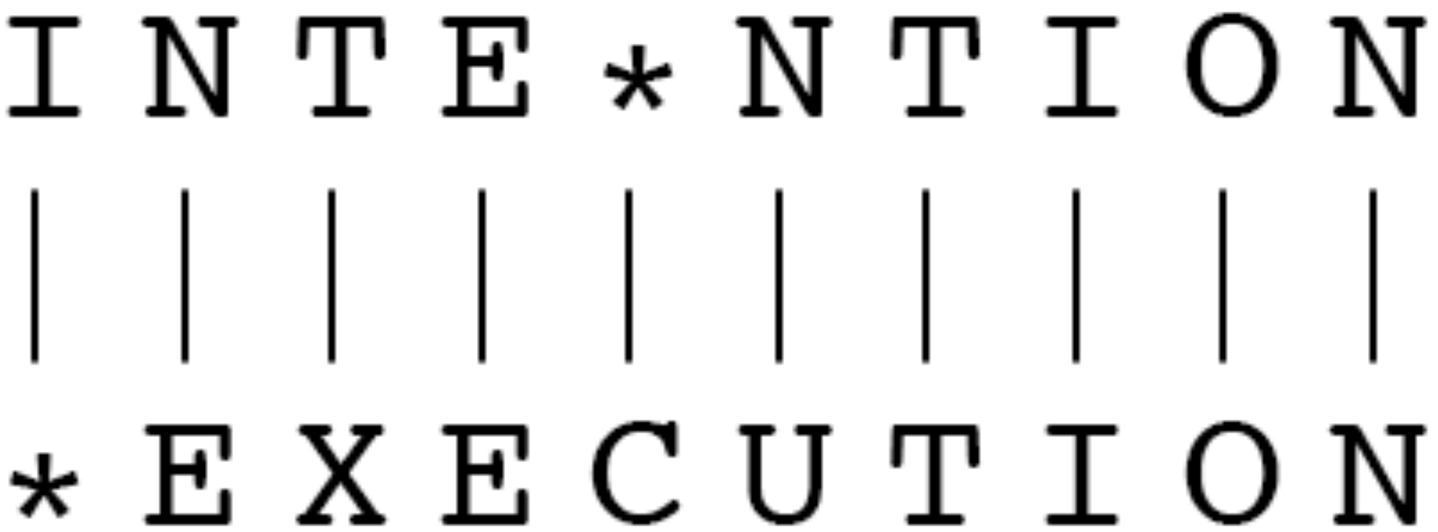
Is the minimum number of editing operations

- Insertion
- Deletion
- Substitution

Needed to transform one into the other

Minimum Edit Distance

Two strings and their **alignment**:



Minimum Edit Distance

```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N

d s s   i s
```

If each operation has cost of 1

- Distance between these is 5

If substitutions cost 2 (Levenshtein)

- Distance between them is 8

Alignment in Computational Biology

Given a sequence of bases

AGGCTATCACCTGACCTCCAGGCCGATGCCC
TAGCTATCACGACCGCGGTCGATTTGCCCGAC

An alignment:

–AGG**CTATCAC**C**T****GACC**T**C**C**A****GG**C**CGA****––TGCCC**–––
T**AG****–CTATCAC****––GACC**G**C****––GG**T**CGA**TT**TGCCC**G**AC**

Given two sequences, align each letter to a letter or gap

Other uses of Edit Distance in NLP

Evaluating Machine Translation and speech recognition

```
R Spokesman confirms    senior government adviser was appointed
H Spokesman said    the senior          adviser was appointed
              S     I             D                       I
```

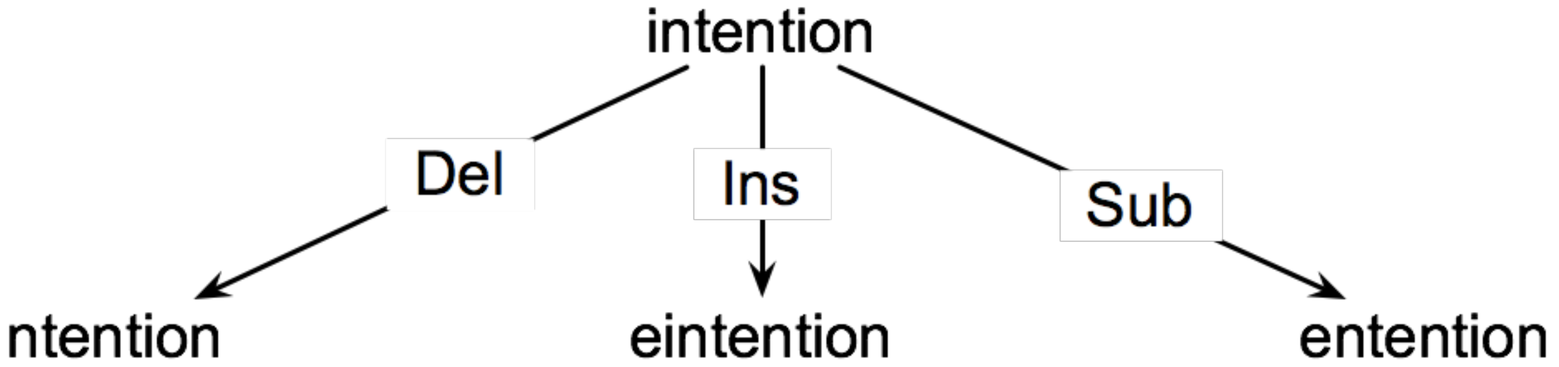
Named Entity Extraction and Entity Coreference

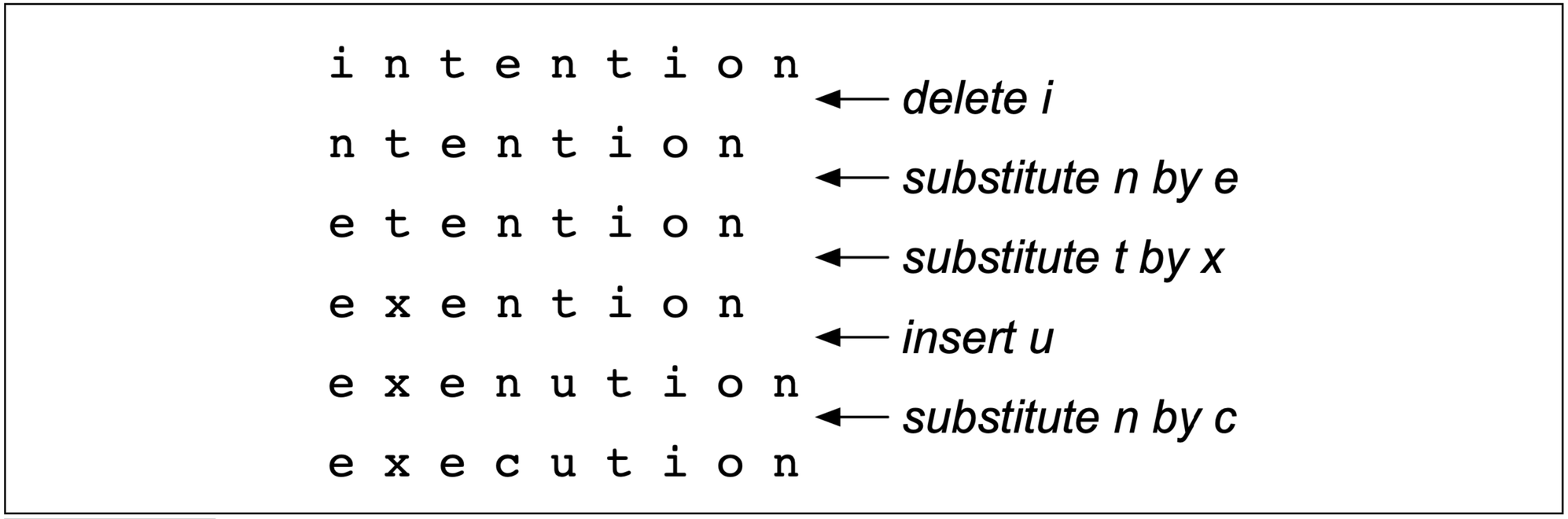
- IBM Inc. announced today
- IBM profits
- Stanford Professor Jennifer Eberhardt announced yesterday
- for Professor Eberhardt…

How to find the Min Edit Distance?

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

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




i n t e n t i o n
n t e n t i o n
e t e n t i o n
e x e n t i o n
e x e n u t i o n
e x e c u t i o n
delete i
substitute n by e
substitute t by x
insert u
substitute n by c

Minimum Edit as Search

But the space of all edit sequences is huge!

- We can’t afford to navigate naïvely
- Lots of distinct paths wind up at the same state.
  - We don’t have to keep track of all of them
  - Just the shortest path to each of those revisted 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*)

Computing Minimum Edit Distance

Dynamic Programming for Minimum 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)

Defining Min Edit Distance (Levenshtein)

Initialization

$D(i,0) = i$

$D(0,j) = j$

Recurrence Relation:

For each $i = 1 \ldots M$

For each $j = 1 \ldots N$

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

Termination:

$D(N,M)$ is distance

function MIN-EDIT-DISTANCE(*source*, *target*) **returns** *min-distance*

$n \leftarrow$ LENGTH(*source*)
$m \leftarrow$ LENGTH(*target*)
Create a distance matrix *D[n+1,m+1]*

Initialization: the zeroth row and column is the distance from the empty string
$D[0,0] = 0$
for each row i **from** 1 **to** n **do**
    $D[i,0] \leftarrow D[i\text{-}1,0]$ + *del-cost*($source[i]$)
for each column j **from** 1 **to** m **do**
    $D[0,j] \leftarrow D[0,j\text{-}1]$ + *ins-cost*($target[j]$)

# *Recurrence relation:*
for each row i **from** 1 **to** n **do**
    for each column j **from** 1 **to** m **do**
        $D[i,j] \leftarrow$ MIN($D[i-1,j]$ + *del-cost*($source[i]$),
                $D[i-1,j-1]$ + *sub-cost*($source[i]$, $target[j]$),
                $D[i,j-1]$ + *ins-cost*($target[j]$))
Termination
return D[n,m]

The Edit Distance Table

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

The Edit Distance Table

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

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

Edit Distance

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

N	9									
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

The Edit Distance Table

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

Backtrace for Computing Alignments

Computing alignments

Edit distance isn't sufficient

- We often need to **align** each character of the two strings to each other

We do this by keeping a "backtrace"

Every time we enter a cell, remember where we came from

When we reach the end,

- Trace back the path from the upper right corner to read off the alignment

Edit Distance

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

N	9									
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

MinEdit with Backtrace

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**	

Adding Backtrace to Minimum Edit Distance

Base conditions:

D(i,0) = i D(0,j) = j

Termination:

D(N,M) is distance

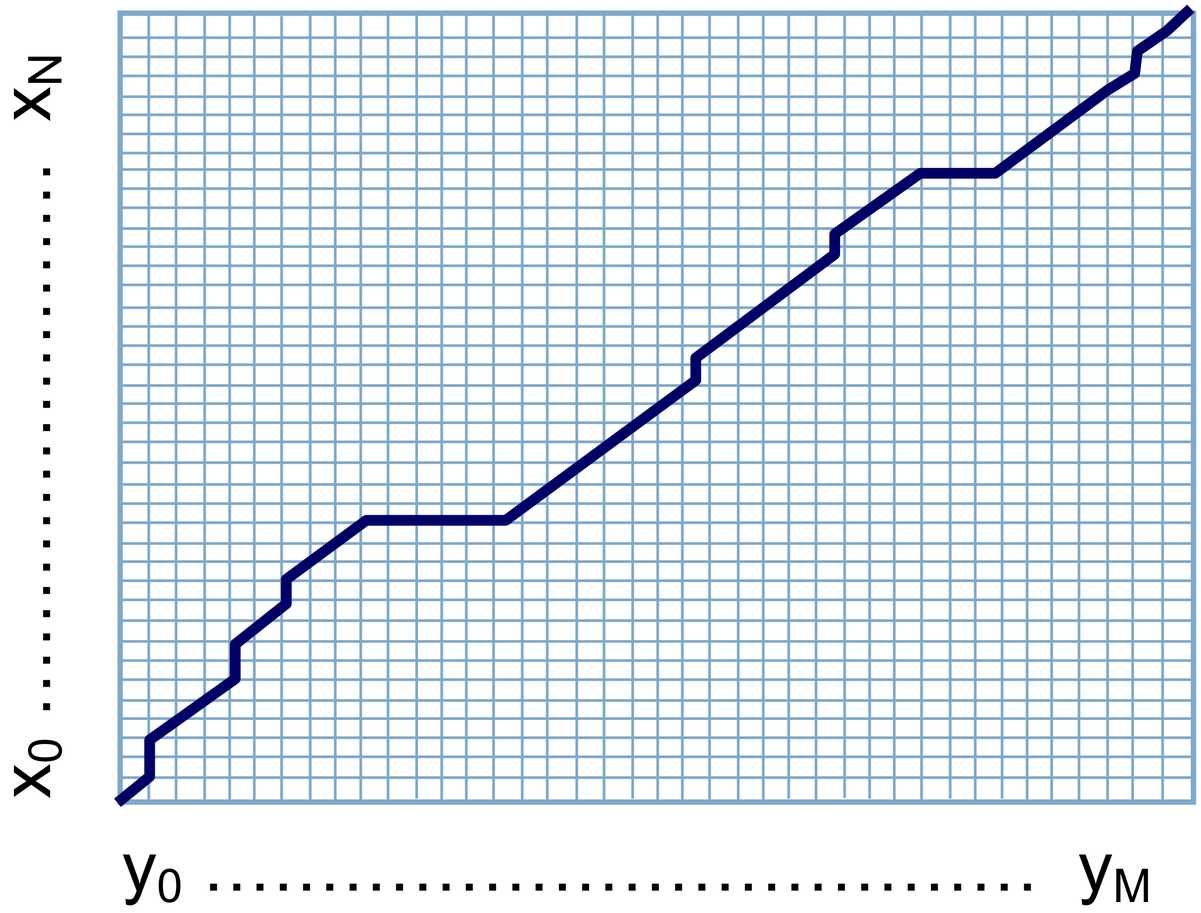
Recurrence Relation:

For each i = 1…M
  For each j = 1…N

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

$$ptr(i,j) = \begin{cases} \text{LEFT} & \text{insertion} \\ \text{DOWN} & \text{deletion} \\ \text{DIAG} & \text{substitution} \end{cases}$$

The Distance Matrix



Every non-decreasing path

from (0,0) to (M, N)

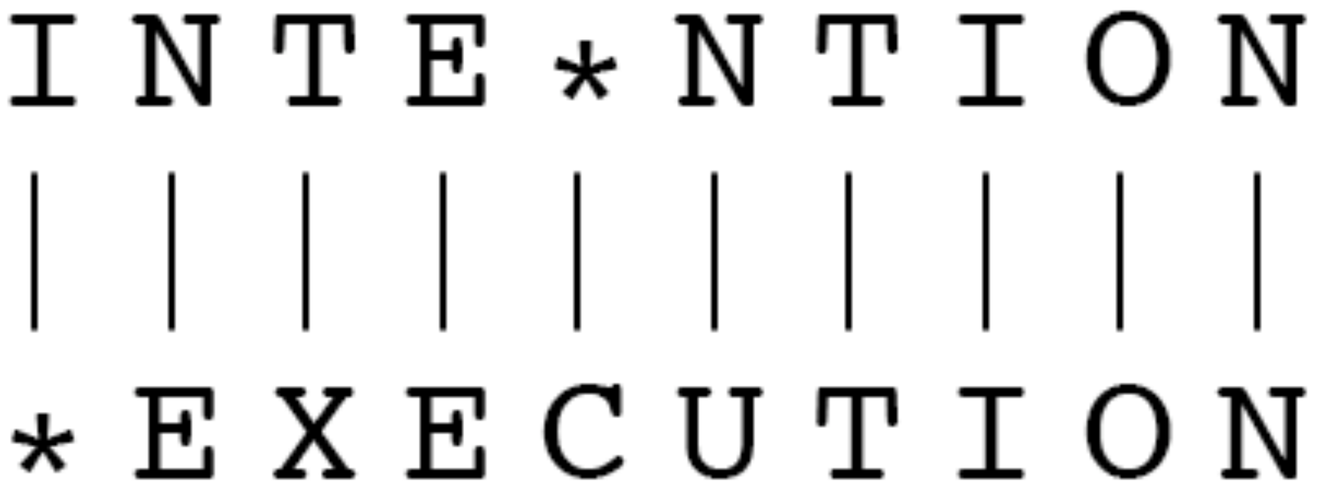
corresponds to
an alignment
of the two sequences

An optimal alignment is composed of optimal subalignments

SLIDE ADAPTED FROM SERAFIM BATZOGLOU WITH PERMISSION

Result of Backtrace

Two strings and their **alignment**:



Performance

Time:

O(nm)

Space:

O(nm)

Backtrace

O(n+m)

Weighted Minimum Edit Distance

Weighted Edit Distance

Why would we add weights to the computation?

- Spell Correction: some letters are more likely to be mistyped than others
- Biology: certain kinds of deletions or insertions are more likely than others

Confusion matrix for spelling errors

sub[X, Y] = Substitution of X (incorrect) for Y (correct)

X	Y (correct)																								
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	0	7	1	342	0	0	2	118	0	1	0	0	3	76	0	0	1	35	9	9	0	1	0	5	0
b	0	0	9	9	2	2	3	1	0	0	0	5	11	5	0	10	0	0	2	1	0	0	8	0	0	0
c	6	5	0	16	0	9	5	0	0	0	1	0	7	9	1	10	2	5	39	40	1	3	7	1	1	0
d	1	10	13	0	12	0	5	5	0	0	2	3	7	3	0	1	0	43	30	22	0	0	4	0	2	0
e	388	0	3	11	0	2	2	0	89	0	0	3	0	5	93	0	0	14	12	6	15	0	1	0	18	0
f	0	15	0	3	1	0	5	2	0	0	0	3	4	1	0	0	0	6	4	12	0	0	2	0	0	0
g	4	1	11	11	9	2	0	0	0	1	1	3	0	0	2	1	3	5	13	21	0	0	1	0	3	0
h	1	8	0	3	0	0	0	0	0	0	2	0	12	14	2	3	0	3	1	11	0	0	2	0	0	0
i	103	0	0	0	146	0	1	0	0	0	0	6	0	0	49	0	0	0	2	1	47	0	2	1	15	0
j	0	1	1	9	0	0	1	0	0	0	0	2	1	0	0	0	0	0	5	0	0	0	0	0	0	0
k	1	2	8	4	1	1	2	5	0	0	0	0	5	0	2	0	0	0	6	0	0	0	4	0	0	3
l	2	10	1	4	0	4	5	6	13	0	1	0	0	14	2	5	0	11	10	2	0	0	0	0	0	0
m	1	3	7	8	0	2	0	6	0	0	4	4	0	180	0	6	0	0	9	15	13	3	2	2	3	0
n	2	7	6	5	3	0	1	19	1	0	4	35	78	0	0	7	0	28	5	7	0	0	1	2	0	2
o	91	1	1	3	116	0	0	0	25	0	2	0	0	0	0	14	0	2	4	14	39	0	0	0	18	0
p	0	11	1	2	0	6	5	0	2	9	0	2	7	6	15	0	0	1	3	6	0	4	1	0	0	0
q	0	0	1	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	0	14	0	30	12	2	2	8	2	0	5	8	4	20	1	14	0	0	12	22	4	0	0	1	0	0
s	11	8	27	33	35	4	0	1	0	1	0	27	0	6	1	7	0	14	0	15	0	0	5	3	20	1
t	3	4	9	42	7	5	19	5	0	1	0	14	9	5	5	6	0	11	37	0	0	2	19	0	7	6
u	20	0	0	0	44	0	0	0	64	0	0	0	0	2	43	0	0	4	0	0	0	0	2	0	8	0
v	0	0	7	0	0	3	0	0	0	0	0	1	0	0	1	0	0	0	8	3	0	0	0	0	0	0
w	2	2	1	0	1	0	0	2	0	0	1	0	0	0	0	7	0	6	3	3	1	0	0	0	0	0
x	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0
y	0	0	2	0	15	0	1	7	15	0	0	0	2	0	6	1	0	7	36	8	5	0	0	1	0	0
z	0	0	0	7	0	0	0	0	0	0	0	7	5	0	0	0	0	2	21	3	0	0	0	0	3	0


~ `
! 1
@ 2
# 3
$ 4
% 5
^ 6
& 7
* 8
(9
) 0
— -
+ =
Tab
Q
W
E
R
T
Y
U
I
O
P
{ [
}]
Caps Lock
A
S
D
F
G
H
J
K
L
: ;
" '
Enter
Shift
Z
X
C
V
B
N
M
< ,
> .
? /
Shift
Ctrl
Alt
Alt

Weighted Min Edit Distance

Initialization:

```
D(0,0) = 0
D(i,0) = D(i-1,0) + del[x(i)];      1 < i ≤ N
D(0,j) = D(0,j-1) + ins[y(j)];      1 < j ≤ M
```

Recurrence Relation:

$$D(i,j) = \min \begin{cases} D(i-1,j) + del[x(i)] \\ D(i,j-1) + ins[y(j)] \\ D(i-1,j-1) + sub[x(i),y(j)] \end{cases}$$

Termination:

D(N,M) is distance

Where did the name, dynamic programming, come from?

…The 1950s were not good years for mathematical research. [the] Secretary of Defense …had a pathological fear and hatred of the word, research…

I decided therefore to use the word, "**programming**".

I wanted to get across the idea that this was dynamic, this was multistage… I thought, let's … take a word that has an absolutely precise meaning, namely **dynamic**… it's impossible to use the word, **dynamic**, in a pejorative sense. Try thinking of some combination that will possibly give it a pejorative meaning. It's impossible.

Thus, I thought dynamic programming was a good name. It was something not even a Congressman could object to."

Richard Bellman, "Eye of the Hurricane: an autobiography" 1984.

Summary