Text Similarity

Dr. Demetrios Glinos
University of Central Florida

CAP6640 – Computer Understanding of Natural Language

Today

- Text Similarity
- Minimum Edit Distance
- Computing Minimum Edit Distance
- Backtrace for Computing Alignments
- Weighted Minimum Edit Distance
- Minimum Edit Distance in Computational Biology

Text Similarity

- The similarity of text and strings has many applications in NLP and computational biology
  - information extraction, machine translation, speech recognition,
- Example: Spelling correction
  - Suppose the input is "graffe"
  - Which term is the best match?
    - graf
    - graft
    - grail
    - giraffe

Plagiarism Detection

- Finding similar text in two documents
- May include modifications

Section 504
of the Rehabilitation Act, and state and local requirements regarding students with disabilities. In compliance with federal and state
regulations, reasonable accommodations are provided to qualified students with disabilities. For more information about accommodations for students with disabilities, click or contact us at . here
dsa@apus.edu Our Admissions staff is available Monday ฏ Friday from 8am ฏ 5:00 pm, EST to answer your questions and provide assistance
with the admissions process. You can reach us by phone at 1-877-777-9081, email us at , or click on the "chat now" icon on any page on this site to correspond instantly during business hoursHave questions? The University System fosters an
environment that
promotes a life of learning for its constituents and uses feedback from its participants and supporters to improve the quality of its teaching, learning, and support. Learning: The University System holds a strong commitment to high standards in all aspects of its educational activities, learning outcomes, and support services. It seeks to continuously
strengthen
the overall effectiveness of its operations. Quality: The University System conducts its operations and makes its public representations in an ethical manner. It assesses its operations in an open
and collaborative manner and practices fairness, honesty, and objectivity in dealing with its constituencies.Integrity: The Major Field Tests are designed to measure the basic student learning outcomes in
these fields
of study. Also, graduate
degree-seeking students in Business Administration (MBA) are required to take the Major Field Test. The Major Field Tests are designed to measure basic student learning outcomes in these fields of
study. Major Field Tests Read about the that have been utilized at APUS to engage faculty and administrators in the assessment process. strategies to see a graphical representation of the Learning
Outcomes Assessment Program at
APUS. Click here Faculty
members - for assistance in developing your course objectives. The Bachelor's degree program is open to students who possess a high school diploma or its equivalent. This minimum 120- to 124-credit-hour
program is the standard four-year
undergraduate degree that most employers want and prepares students for graduate study.Bachelor's Degree AMU offers certificate programs at the undergraduate level to applicants who have a high school
diploma or its
equivalent. Certificate programs allow students to focus on a particular topic of interest

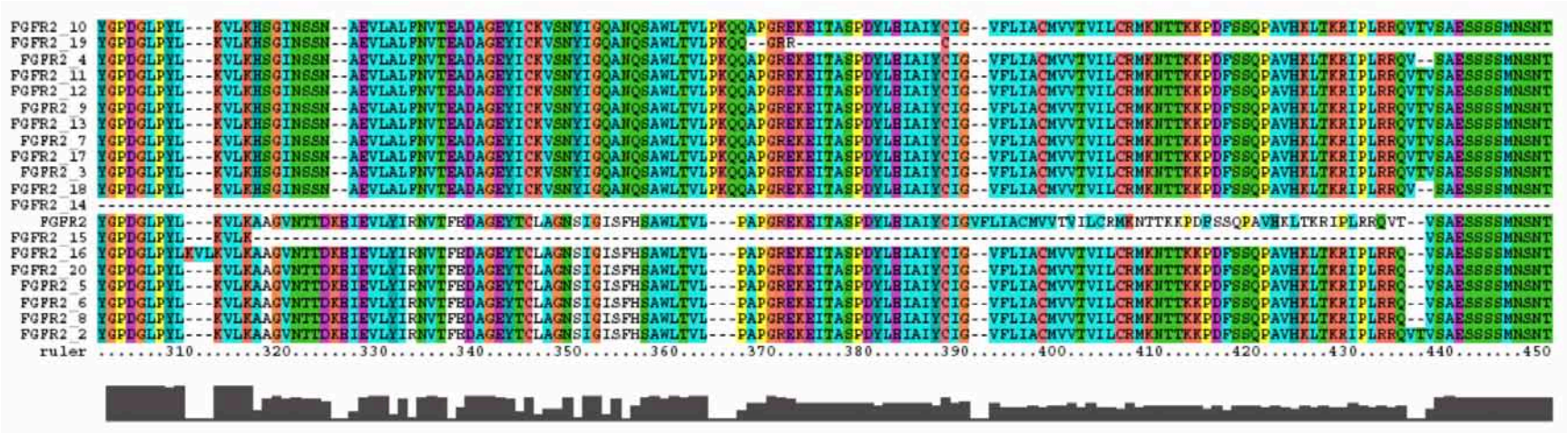
Welcome to the Learning Outcomes Assessment Website at the . The ongoing process of learning outcomes assessment involves all members of the APUS
community. Continuous assessment activities allow us to identify what our students are learning and enables us to improve the student learning environment through our courses and academic programs. American Public University System
Dr. Jennifer Stephens, Dean of Assessment
Learning Outcomes Assessment Test for APUS Students
The American Public University System uses a variety of assessment activities designed to document student learning. Degree-seeking students are required to participate in the learning outcomes assessment activities at APUS and will be given adequate notice of course and non-course related assessment activities they are responsible for completing. Assessments are used for the purpose of continuously improving on our academic programs and courses. The students' survey results are confidential and are used only to inform the University System of performance in a given area. APUS is committed to protecting the privacy of its students. This includes ensuring the confidentiality of student work submitted for assessment as well as the feedback resulting from assessment activities.
National testing and other assessment are not calculated into the student's grade point average or considered part of the student's academic performance. Completion of the assessment is required prior to the student's conferral date; however, those who have not completed the required LOA exam will still maintain their conferral date. In the event that the student does not complete the assessment, APUS will not issue final transcripts or the diploma until the assessment has been completed.
Currently, national testing required for degree-seeking undergraduate students and MBA candidates include:
(MAPP) tests are for all undergraduate degree-seeking students. The MAPP is a general education assessment designed to evaluate and improve the quality of teaching and learning by measuring student achievement in critical thinking, reading, writing, and mathematics. Measure of Academic Proficiency & Progress
(MFTs) are in subject-specific areas for undergraduate degree-seeking students in Business Administration, Criminal Justice, History, English, Political Science, Psychology, and Sociology. The Major Field Tests are designed to measure the basic student learning outcomes in these fields of study. Also, graduate degree-seeking students in Business Administration (MBA) are required to take the Major Field Test. The Major Field Tests are designed to measure basic student learning outcomes in these fields of study. Major Field Tests
Read about the that have been utilized at APUS to engage faculty and administrators in the assessment process. strategies to see a graphical representation of the Learning Outcomes Assessment Program at APUS. Click here
Faculty members - for assistance in developing your course objectives. Click here

Computational Biology

- Alignment of nucleotide sequences in DNA strands

```
AGAGTGCTGCCGCC
AGATGTACTGCGCC
```

```
AGA-GTGCTGCCGCC
||| || |||| |||
AGATGTACTGC-GCC
```



Today

- Text Similarity
- Minimum Edit Distance
- Computing Minimum Edit Distance
- Backtrace for Computing Alignments
- Weighted Minimum Edit Distance
- Minimum Edit Distance in Computational Biology

Minimum Edit Distance Defined

- We use the notion of "distance" as a measure of similarity for text and strings generally
- Common measure: minimum edit distance
  - the minimum number of editing operations
    - insertions
    - deletions
    - substitutions
  - needed to transform either string into the other

Distance Metrics

- Levenshtein
  - +1 for each insertion, deletion, or substitution
  - also a variant in which substitutions cost +2
- Other metrics
  - Damerau-Levenshtein
    - also allows transposition of adjacent characters
  - Hamming distance
    - only substitutions allowed
    - strings must be same length
  - Jaro
    - only transpositions allowed

Example: Minimum Edit Distance

```
A =    I N T E * N T I O N
       | | | | | | | | | |
B =    * E X E C U T I O N
       d s s   i s
```

source: Fig. 3.23 (annotated)

- Levenshtein distance = 5
- Levenshtein variant distance = 8
- NOTE: In general, there may be multiple optimal solutions
  e.g., first 3 actions could also be s-d-s or s-s-d, with same distance

Other uses of Edit Distance in NLP

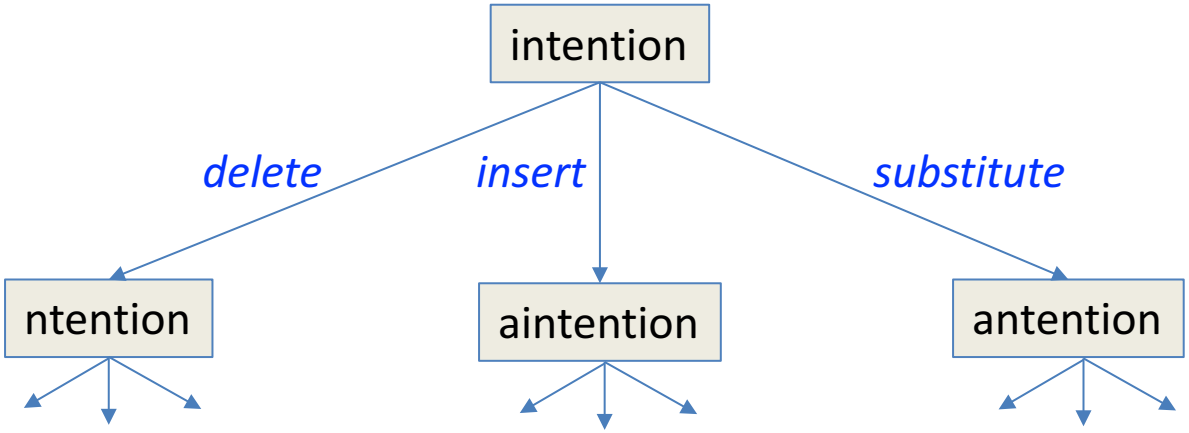
- Machine translation and speech recognition

Spokesmen	confirm		senior	government	advisor	was	fired	
Spokesmen	said	the	senior		advisor	was	fired	today
	SUB	INS		DEL				I

- Named entity recognition and entity coreference
  - IBM Inc. announced today ...
  - IBM profits were reported ...
  - UCF President John Hitt announced today ...
  - for University of Central Florida President John Hitt ...

Finding the Min Edit Distance

- This is a classic cost-sensitive search problem
  - Initial state: one of the strings
  - Successor function: effects of insertions, deletions, and substitutions
  - Goal test: the other string
  - Path cost: Levenstein (for example)



Q: How large is this search tree?

Feasibility of Search

- The space of all edit sequences is huge!
  - Consider only the substitutions at the current step
    - For each character in the current string, there are 25 possible substitutions
    - for a string of n characters, this is $O(25^n)$ just for substitutions!
  - Plus for insertions: $O(26^{n+1})$
  - And for deletions: $O(n)$
- Many distinct paths end up at the same state
- We cannot afford to navigate naïvely
  - we don't need to keep track of all the different paths to a state
  - just the shortest one

Today

- Text Similarity
- Minimum Edit Distance
- Computing Minimum Edit Distance
- Backtrace for Computing Alignments
- Weighted Minimum Edit Distance
- Minimum Edit Distance in Computational Biology

Notation

- Given two strings
  - X of length n
  - Y of length m
- Then,
  - 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
  - D(n, m) is therefore the edit distance between X and Y

Dynamic Programming for Min Edit Distance

- Dynamic programming
  - tabular method for solving problems by combining solutions to subproblems
  - we can use this to compute D(n, m)

- Bottom-up approach
  - compute D(i, j) for small i, j
  - compute larger D(i, j) values based on previously computed smaller values
  - we will ***recursively*** 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
```

- Recurence relation

Levenstein variant metric

```
For each i = 1..N
    For each j = 1..M
```

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

- Termination

```
return D(N,M)   ← the min edit distance
```

The Edit Distance Table

- After the initialization step:

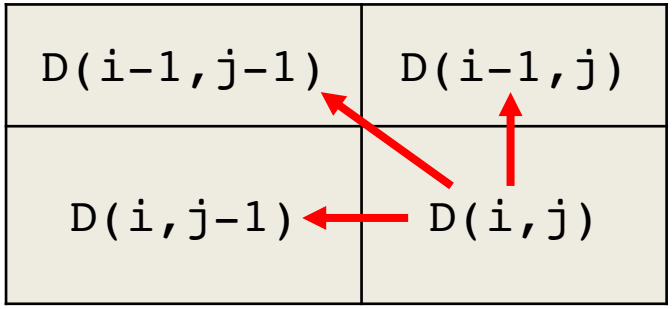
(0,0)

Y = target ⟶

⟵ X = source

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

Building the Table Row by Row



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

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

$$D(1,1) = 2 = \min \begin{cases} D(0,1) + 1 = 1 + 1 = 2 \\ D(1,0) + 1 = 1 + 1 = 2 \\ D(0,0) + 2 = 0 + 2 = 2, \text{ since } I \neq E \end{cases}$$

The Completed Table

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

- D(N, M) = 8 is the min edit distance, using the alternate Levenstein metric

Another Example

- The strings do not need to be the same length

	#	I	N	F	L	E	C	T	I	O	N
#	0	1	2	3	4	5	6	7	8	9	10
I	1	0	1	2	3	4	5	6	7	8	9
N	2	1	0	1	2	3	4	5	6	7	8
T	3	2	1	2	3	4	5	4	5	6	7
E	4	3	2	3	4	3	4	5	6	7	8
N	5	4	3	4	5	4	5	6	7	8	7
T	6	5	4	5	6	5	6	5	6	7	8
I	7	6	5	6	7	6	7	6	5	6	7
O	8	7	6	7	8	7	8	7	6	5	6
N	9	8	7	8	9	8	9	8	7	6	5

Today

- Text Similarity
- Minimum Edit Distance
- Computing Minimum Edit Distance
- Backtrace for Computing Alignments
- Weighted Minimum Edit Distance
- Minimum Edit Distance in Computational Biology

Computing Alignments

- Alignments are particularly useful
  - computing word error rate in speech recognition
  - aligning texts in machine translation systems
  - plagiarism detection
  - DNA sequencing

- We can find an optimal alignment by using our edit table and working backwards from the end

- But we need to know which neighboring value produced the value that we find in the current cell.

- We do this by keeping a "backtrace" of pointers as we build the edit table

Min Edit Distance Algorithm with Backtrace

- Initialization

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

- Recurence relation

```
For each i = 1..N
    For each j = 1..M
```

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

$$ptr(i,j) = \begin{cases} \text{LEFT} & \leftarrow \textit{insertion} \\ \text{UP} & \leftarrow \textit{deletion} \\ \text{DIAG} & \leftarrow \textit{substitution or same} \end{cases}$$

- Termination

```
return D(N,M)  ← the min edit distance
```

Using the Backtrace

Edit Distance Table:

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

Bactrace Table:

	#	E	X	E	C	U	T	I	O	N
#		LT	LT	LT	LT	LT	LT	LT	LT	LT
I	UP	DI	DI	DI	DI	DI	DI	DI	LT	LT
N	UP	DI	DI	DI	DI	DI	DI	UP	DI	DI
T	UP	DI	DI	DI	DI	DI	DI	LT	DI	UP
E	UP	DI	LT	DI	LT	LT	LT	DI	DI	UP
N	UP	UP	DI	DI	DI	DI	DI	DI	DI	DI
T	UP	UP	DI	DI	DI	DI	DI	LT	LT	LT
I	UP	UP	DI	DI	DI	DI	UP	DI	LT	LT
O	UP	UP	DI	DI	DI	DI	UP	UP	DI	LT
N	UP	UP	DI	DI	DI	DI	UP	UP	UP	DI

Alignment:

Source:	I	N	T	E	–	N	T	I	O	N
Target:	–	E	X	E	C	U	T	I	O	N
Action:	d	s	s		i	s				

Levenshtein distance = 8

We can compute the alignment from the backtrace table

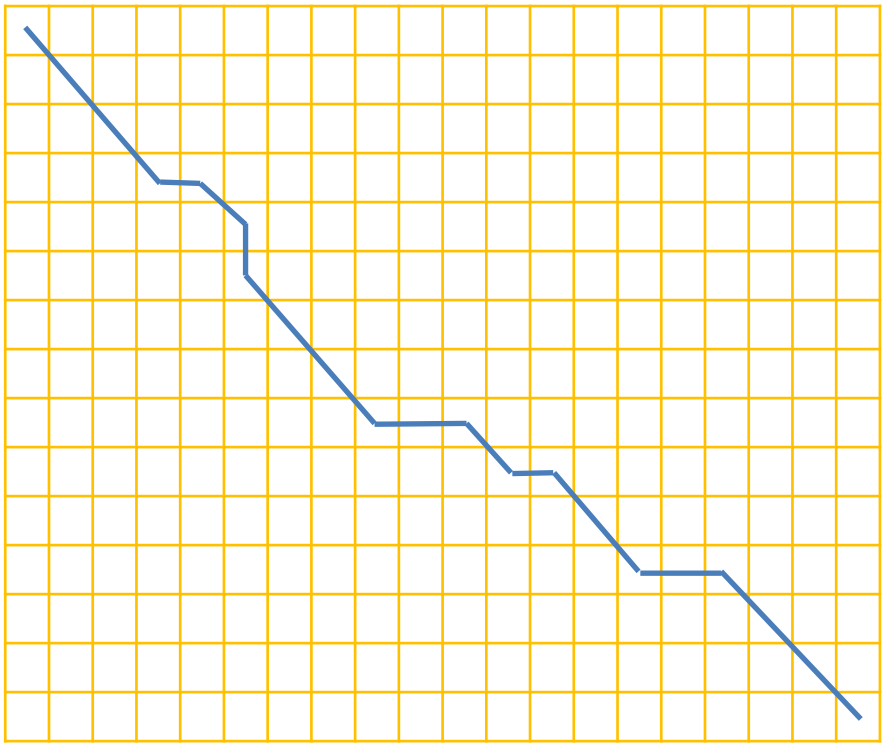
Global Alignments

Every non-decreasing path

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

corresponds to an alignment of the two sequences

An optimal alignment is composed of optimal subalignments



Performance

- Time: O(nm)
- Space: O(nm)
- Backtrace: O(n + m)

Today

- Text Similarity
- Minimum Edit Distance
- Computing Minimum Edit Distance
- Backtrace for Computing Alignments
- Weighted Minimum Edit Distance
- Minimum Edit Distance in Computational Biology

Weighted Edit Distance

- Weighting can be useful
  - spelling 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

- Off-diagonal entries are errors

Weighted Min Edit Distance Algorithm

- 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
```

- Recurence relation

```
For each i = 1..N
    For each j = 1..M
```

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

- Termination

```
return D(N,M)   ← the weighted min edit distance
```

Today

- Text Similarity
- Minimum Edit Distance
- Computing Minimum Edit Distance
- Backtrace for Computing Alignments
- Weighted Minimum Edit Distance
- Minimum Edit Distance in Computational Biology

Sequence Alignment

- Alignment of nucleotide sequences in DNA strands

```
AGAGTGCTGCCGCC
AGATGTACTGCGCC
```

```
AGA-GTGCTGCCGCC
|||  ||  ||||  |||
AGATGTACTGC-GCC
```

- Reasons for alignment
  - splicing fragments to sequence DNA
  - comparing individuals to look for mutations
  - comparing genes or regions from different species
    - to find important regions
    - to determine function
    - to uncover evolutionary forces

Alignment Terminology

- In NLP
  - we generally talk about distance (minimized)
  - and weights
- In Computational Biology
  - we generally talk about similarity (maximized)
  - and scores

Needleman-Wunsch Algorithm

- Initialization

```
D(0,0) = 0
D(i,0) = -i * g     1 ≤ i ≤ N
D(0,j) = -j * g     1 ≤ j ≤ M
```

NOTE: also use pointer table to keep track of predecessors

- Recurence relation

```
For each i = 1..N
    For each j = 1..M
```

$$D(i,j) = \max \begin{cases} D(i-1,j) & + \text{ gap} \\ D(i,j-1) & + \text{ gap} \\ D(i-1,j-1) & + \text{ match}(x_i, y_j) \end{cases}$$

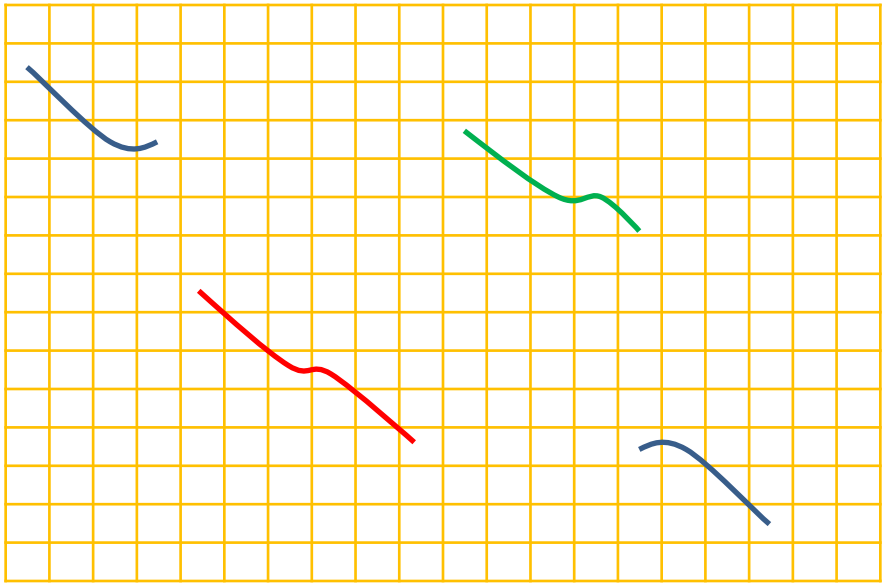
- Termination

```
return D(N,M)   ← distance
```

Local Alignment

- Needleman-Wunsch, like our Min Edit, computes a ***global*** alignment
- Often, we wish to find a longest ***local*** alignment
  - matching a 1,000-base DNA strand in a 1-million base gene
  - matching a text fragment within a larger document
- Local alignment problem
  - find the most similar substrings
- Example

```
x = aaaaacccccgggggtta
y = ttcccgggaaccaatc
```



Smith-Waterman Algorithm

- Finds the longest similar substrings
- Modifies Needleman-Wunsch not to allow negative values
- Initialization

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

NOTE: also use pointer table to keep track of predecessors

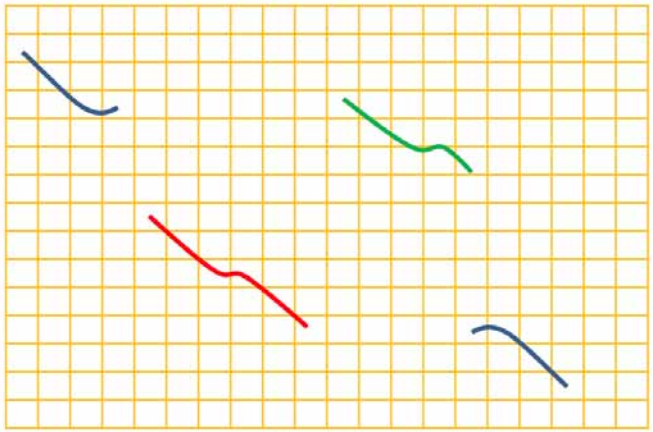
- Iteration

$$D(i,j) = \max \begin{cases} 0 \\ D(i-1,j) + gap \\ D(i,j-1) + gap \\ D(i-1,j-1) + match(x_i, y_j) \end{cases}$$

- Typical values: gap = -1, match(x_i, y_j) = +2 if $x_i = y_j$ and -1 otherwise

Using Smith-Waterman

- Maximally similar local alignment(s) will produce the largest value in the distance table
- Find the alignment(s) by backtracing from all occurrences of this value until encounter a zero value
- Tune the search by adjusting the gap and match/mismatch values
- Can also insert custom code to "jump" gaps of zero values to find longer similiar strings



Example: Smith-Waterman

String X = "AAAACCCCCGGGGTTA"
String Y = "TTCCCGGGAACCAATC"

Edit Distance Table:

	#	T	T	C	C	C	G	G	G	A	A	C	C	A	A	T	C
#	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0	2	2	1	0	2	2	1	0
A	0	0	0	0	0	0	0	0	0	2	4	3	2	2	4	3	2
A	0	0	0	0	0	0	0	0	0	2	4	3	2	4	4	3	2
A	0	0	0	0	0	0	0	0	0	2	4	3	2	4	6	5	4
C	0	0	0	2	2	2	1	0	0	1	3	6	5	4	5	5	7
C	0	0	0	2	4	4	3	2	1	0	2	5	8	7	6	5	7
C	0	0	0	2	4	6	5	4	3	2	1	4	7	7	6	5	7
C	0	0	0	2	4	6	5	4	3	2	1	3	6	6	6	5	7
C	0	0	0	2	4	6	5	4	3	2	1	3	5	5	5	5	7
G	0	0	0	1	3	5	8	7	6	5	4	3	4	4	4	4	6
G	0	0	0	0	2	4	7	10	9	8	7	6	5	4	3	3	5
G	0	0	0	0	1	3	6	9	12	11	10	9	8	7	6	5	4
G	0	0	0	0	0	2	5	8	11	11	10	9	8	7	6	5	4
T	0	2	2	1	0	1	4	7	10	10	10	9	8	7	6	8	7
T	0	2	4	3	2	1	3	6	9	9	9	9	8	7	6	8	7
A	0	1	3	3	2	1	2	5	8	11	11	10	9	10	9	8	7

Bactrace Table:

	#	T	T	C	C	C	G	G	G	A	A	C	C	A	A	T	C
#																	
A										DI	DI	LT		DI	DI	LT	
A										DI	DI	LT	LT	DI	DI	LT	LT
A										DI	DI	LT	LT	DI	DI	LT	LT
A										DI	DI	LT	LT	DI	DI	LT	LT
C				DI	DI	DI	LT			UP	UP	DI	LT	LT	UP	DI	DI
C				DI	DI	DI	LT	LT	LT		UP	UP	DI	LT	LT	LT	DI
C				DI	DI	DI	LT	LT	LT	LT	UP	UP	UP	DI	LT	LT	DI
C				DI	DI	DI	LT	LT	LT	LT	LT	UP	UP	UP	DI	LT	DI
C				DI	DI	DI	LT	LT	LT	LT	LT	DI	UP	UP	UP	DI	DI
G				UP	UP	UP	DI	LT	LT	LT	LT	LT	UP	UP	UP	UP	UP
G					UP	UP	UP	DI	LT	LT	LT	LT	LT	LT	UP	UP	UP
G					UP	UP	UP	UP	DI	LT	LT	LT	LT	LT	LT	LT	UP
G						UP	UP	UP	UP	DI	LT	LT	LT	LT	LT	LT	LT
T		DI	DI	LT		UP	UP	UP	UP	UP	DI	LT	LT	LT	LT	DI	LT
T		DI	DI	LT	LT	LT	UP	UP	UP	UP	UP	DI	LT	LT	LT	DI	LT
A		UP	UP	DI	LT	LT	UP	UP	UP	DI	DI	LT	LT	DI	LT	LT	LT

Example 2: Smith-Waterman

String X = "AAAACCCCTGCGGTTA"
String Y = "TTCCACGGGAACCAATC"

Maximal value = 9

Maxima:

[11, 17]

[13, 9]

[16, 10]

[16, 11]

Edit Distance Table:

	#	T	T	C	C	A	C	G	G	G	A	A	C	C	A	A	T	C
#	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	2	1	0	0	0	2	2	1	0	2	2	1	0
A	0	0	0	0	0	2	1	0	0	0	2	4	3	2	2	4	3	2
A	0	0	0	0	0	2	1	0	0	0	2	4	3	2	4	4	3	2
A	0	0	0	0	0	2	1	0	0	0	2	4	3	2	4	6	5	4
C	0	0	0	2	2	1	4	3	2	1	1	3	6	5	4	5	5	7
C	0	0	0	2	4	3	3	3	2	1	0	2	5	8	7	6	5	7
C	0	0	0	2	4	3	5	4	3	2	1	1	4	7	7	6	5	7
C	0	0	0	2	4	3	5	4	3	2	1	0	3	6	6	6	5	7
T	0	2	2	1	3	3	4	4	3	2	1	0	2	5	5	5	8	7
G	0	1	1	1	2	2	3	6	6	5	4	3	2	4	4	4	7	7
C	0	0	0	3	3	2	4	5	5	5	4	3	5	4	3	3	6	9
G	0	0	0	2	2	2	3	6	7	7	6	5	4	4	3	2	5	8
G	0	0	0	1	1	1	2	5	8	9	8	7	6	5	4	3	4	7
T	0	2	2	1	0	0	1	4	7	8	8	7	6	5	4	3	5	6
T	0	2	4	3	2	1	0	3	6	7	7	7	6	5	4	3	5	5
A	0	1	3	3	2	4	3	2	5	6	9	9	8	7	7	6	5	4

Example 2: Smith-Waterman

String X = "AAAACCCCTGCGGTTA"
String Y = "TTCCACGGGAACCAATC"

```
Alignment 0:

   Source:   A  A  C  C  C  C  T  G  C
   Target:   A  A  C  C  A  A  T  -  C
   Action:                s  s     d
```

String X = "AAAACCCCTGCGGTTA"
String Y = "TTCCACGGGAACCAATC"

```
Alignment 1:

   Source:   C  C  -  C  T  G  C  G  G
   Target:   C  C  A  C  -  G  -  G  G
   Action:         i     d     d
```

String X = "AAAACCCCTGCGGTTA"
String Y = "TTCCACGGGAACCAATC"

```
Alignment 2:

   Source:   C  C  -  C  T  G  C  G  G  T  T  A
   Target:   C  C  A  C  -  G  -  G  G  -  -  A
   Action:         i     d     d        d  d
```

String X = "AAAACCCCTGCGGTTA"
String Y = "TTCCACGGGAACCAATC"

```
Alignment 3:

   Source:   C  C  -  C  T  G  C  G  G  T  T  A
   Target:   C  C  A  C  -  G  -  G  G  A  -  A
   Action:         i     d     d        s  d
```