CS-AD 220 – Spring 2016

Natural Language Processing

Session 8: 23-Feb-16

Prof. Nizar Habash

Some slides are adapted from Jurafsky and Martin's course slides on Speech and Language Processing

NYUAD CS-AD 220 – Spring 2016
Natural Language Processing

Assignment #2
Finite State Machines
Assigned Feb 18, 2016
Due Mar 10, 2016 (11:59pm)

I. Grading & Submission

This assignment is about the development of finite state machines using the OpenFST and Thrax toolkits. The assignment accounts for 15% of the full grade. It consists of three exercises. The first is a simple "machine translation" system for animal sounds to help with learning the tools. The second is about modeling how numbers are read in English and French. And the third is about Spanish verb conjugation. The answers should be placed in a zipped folder with separate sub-directories for each exercise.

The assignment is due on March 10 before midnight (11:59pm). For late submissions, 10% will be deducted from the homework grade for any portion of each late day. The student should upload the answers in a single zipped to NYU Classes (Assignment #2).

Assignment #2 posted on NYU Classes

Moving Legislative Day Class

- Spring Break is March 18 – 25, 2016
- Sat March 26, 2016 is a Legislative *Thursday*
- Move to

Sat April 2, 2016 at 10am

Same Classroom C2-E049

Stemming and Lemmatization

- Morphological variation
  - Inflectional: score, scored, scores, scoring
    be, is, was, were, am , being, been
  - Derivational: play, playful
- **Stemming**
  - Reduce morphological variation in a text by mapping the words to their common stem.
    - Tend to be a shallow algorithm using cascade rewrite rules.
      - score, scored, scores, scoring → scor
      - be, is, was, were, am , being, been → be, i, wa, wer, am*
      - play, playful → play

**Depending on the algorithm.*

Stemming and Lemmatization

- Morphological variation
  - Inflectional: score, scored, scores, scoring
    be, is, was, were, am , being, been
  - Derivational: play, playful
- **Lemmatization**
  - Reduce morphological variation in a text by mapping the words to their lemma (aka dictionary form, citation form).
    - Tend to be more complex than stemming; it models morphology properly
      - score, scored, scores, scoring → score
      - be, is, was, were, am , being, been → be
      - play, playful → play , playful ($play_{NOUN}$+ful_{ADJ})

The Porter Stemmer (Porter, 1980)

- A simple rule-based algorithm for stemming
- An example of a HEURISTIC method
- Based on rules like:
  - ATIONAL -> ATE (e.g., *relational* -> *relate*)
- The algorithm consists of seven sets of rules, applied in order
- The Porter Stemmer home page (with the original paper and code): http://www.tartarus.org/~martin/PorterStemmer/

The Porter Stemmer: definitions

- Definitions:
  - CONSONANT: a letter other than A, E, I, O, U, and Y preceded by consonant
  - VOWEL: any other letter
- With this definition, all words are of the form:

  $C?(VC)^mV?$

  C = string of one or more consonants (con+)

  V = string of one or more vowels

  m = the measure of a string
- E.g.,
  - Tree C V , m=0
  - Oats V C , m=1
  - Troubles C V C V C, m=2

The Porter Stemmer: rule format

- The rules are of the form:

  (condition) S1 → S2

  Where S1 and S2 are suffixes
- Conditions:

m (in $C?(VC)^mV?$)	The measure of the stem
*<letter> , e.g. *S	The stem ends with <letter>, e.g. ends with S
v	The stem contains a vowel
*d	The stem ends with a double consonant
*o	The stem ends in CVC (second C not W, X, or Y)

The Porter Stemmer:
Step 1 (only one rule is applied; longest match)

- SSES → SS
  - *caresses -> caress*
- IES → I
  - *ponies -> poni*
  - *ties -> ti*
- SS → SS
  - *caress -> caress*
- S → ϵ
  - *cats -> cat*

The Porter Stemmer: Step 2a (past tense, progressive)

m	Stem measure $C?(VC)^mV?$
*<letter>	Stem ends with <letter>
v	Stem contains a vowel
*d	Stem ends with a double consonant
*o	Stem ends in C_1VC_2 (C_2 not W, X, or Y)

- (m>0) EED → EE
  - Condition verified: *agreed -> agree*
  - Condition not verified: *feed -> feed*
- (*V*) ED → ϵ
  - Condition verified: *plastered -> plaster*
  - Condition not verified: *bled -> bled*
- (*V*) ING → ϵ
  - Condition verified: *motoring -> motor*
  - Condition not verified: *sing -> sing*

The Porter Stemmer: Step 2b (cleanup)

m	Stem measure $C?(VC)^mV?$
*<letter>	Stem ends with <letter>
v	Stem contains a vowel
*d	Stem ends with a double consonant
*o	Stem ends in C_1VC_2 (C_2 not W, X, or Y)

- (These rules are ran if second or third rule in 2a apply)
- AT→ ATE
  - *conflat(ed) -> conflate*
- BL → BLE
  - *Troubl(ing) -> trouble*
- (*d & not (*L or *S or *Z)) → single letter
  - Condition verified*: hopp(ing) -> hop, tann(ed) -> tan*
  - Condition not verified*: fall(ing) -> fall*
- (m=1 & *o) → E
  - Condition verified*: fil(ing) -> file*
  - Condition not verified*: fail(ing) -> fail*

The Porter Stemmer: Steps 3 and 4

m	Stem measure $C?(VC)^mV?$
*<letter>	Stem ends with <letter>
v	Stem contains a vowel
*d	Stem ends with a double consonant
*o	Stem ends in C_1VC_2 (C_2 not W, X, or Y)

- Step 3: Y Elimination (*V*) Y → I
  - Condition verified: *happy -> happi*
  - Condition not verified: *sky -> sky*
- Step 4: Derivational Morphology, I
  - (m>0) ATIONAL → ATE
    - *Relational -> relate*
  - (m>0) IZATION → IZE
    - *generalization-> generalize*
  - (m>0) BILITI → BLE
    - *sensibiliti -> sensible*

The Porter Stemmer: Steps 5 and 6

m	Stem measure $C?(VC)^mV?$
*<letter>	Stem ends with <letter>
v	Stem contains a vowel
*d	Stem ends with a double consonant
*o	Stem ends in C_1VC_2 (C_2 not W, X, or Y)

- Step 5: Derivational Morphology, II
  - (m>0) ICATE → IC
    - *triplicate -> triplic*
  - (m>0) FUL → ϵ
    - *hopeful -> hope*
  - (m>0) NESS → ϵ
    - *goodness -> good*
- Step 6: Derivational Morphology, III
  - (m>1) ANCE → ϵ
    - *allowance-> allow*
  - (m>1) ENT → ϵ
    - *dependent-> depend*
  - (m>1) IVE → ϵ
    - *effective -> effect*

The Porter Stemmer: Step 7 (cleanup)

m	Stem measure $C?(VC)^mV?$
*<letter>	Stem ends with <letter>
v	Stem contains a vowel
*d	Stem ends with a double consonant
*o	Stem ends in C_1VC_2 (C_2 not W, X, or Y)

- Step 7a
  - (m>1) E → ϵ
    - *Probate* → *probat*
    - *rate* → *rate*
  - (m=1 & not *o) E → ϵ
    - *cease* → *ceas*
- Step 7b
  - (m>1 & *d & *L) → single letter
    - Condition verified*: controll -> control*
    - Condition not verified*: roll -> roll*

Examples

- *computers*
  - Step 1, Rule 4: -> *computer*
  - Step 6, Rule 4: -> *compute*
- *singing*
  - Step 2a, Rule 3: -> *sing*
- *controlling*
  - Step 2a, Rule 3: -> *controll*
  - Step 7b : -> *control*
- *generalizations*
  - Step 1, Rule 4: -> *generalization*
  - Step 4, Rule 11: -> *generalize*
  - Step 6, last rule: -> *general*

Problems

- *elephants -> eleph*
  - Step 1, Rule 4: -> *elephant*
  - Step 6, Rule 7: -> *eleph*
- **Errors of Commission:**
  - organization, organ → organ
  - generalization, generic → gener
  - numerical, numerous → numer
- **Errors of Omission:**
  - explain → explain **BUT** explanation → explan
  - Analysis → analysi **BUT** analyses → analys
  - noise → nois **BUT** noisy → noisi

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

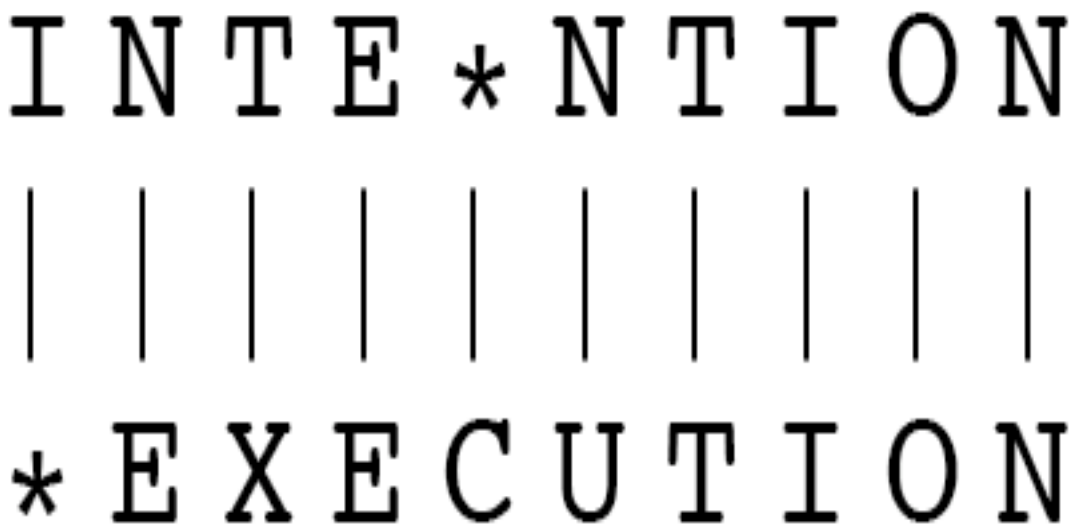
  - Resulting alignment:

    –AGG**CTATCAC**C**TGACCTC**C**AGG**C**CGA--TGCCC**---
    TAG–CTATCAC--GACCG**C--GG**T**CGA**TT**TGCCC**GAC

- Also for Machine Translation, Speech Recognition, etc.
- The **minimum edit distance** between two strings is the minimum number of editing operations needed to transform one string into the other
  - Insertion, Deletion, Substitution

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
  - Distance between them is 8

Other uses of Edit Distance in NLP

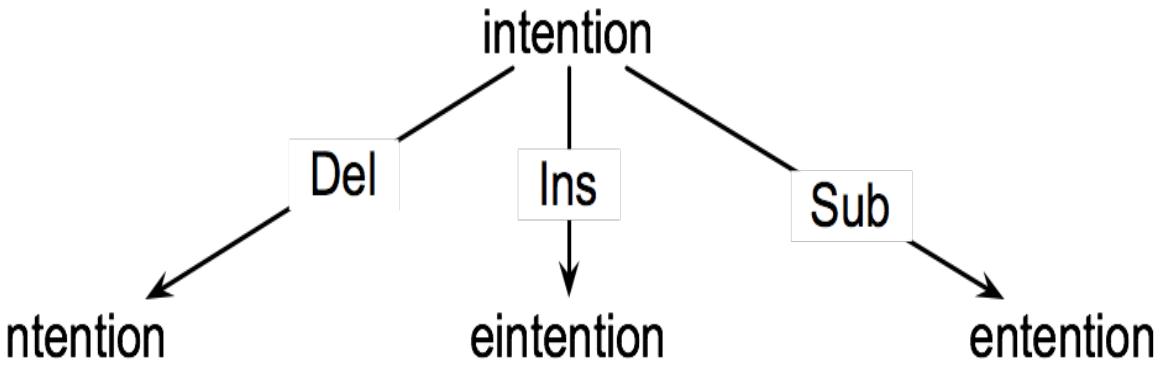
- Evaluating Machine Translation and Speech Recognition

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

- Named Entity Extraction and Entity Coreference
  - IBM Inc. announced today
  - IBM profits
  - Stanford President John Hennessy announced yesterday
  - for Stanford University President John Hennessy

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



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 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*)

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…M

  For each j = 1…N

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

- Termination:

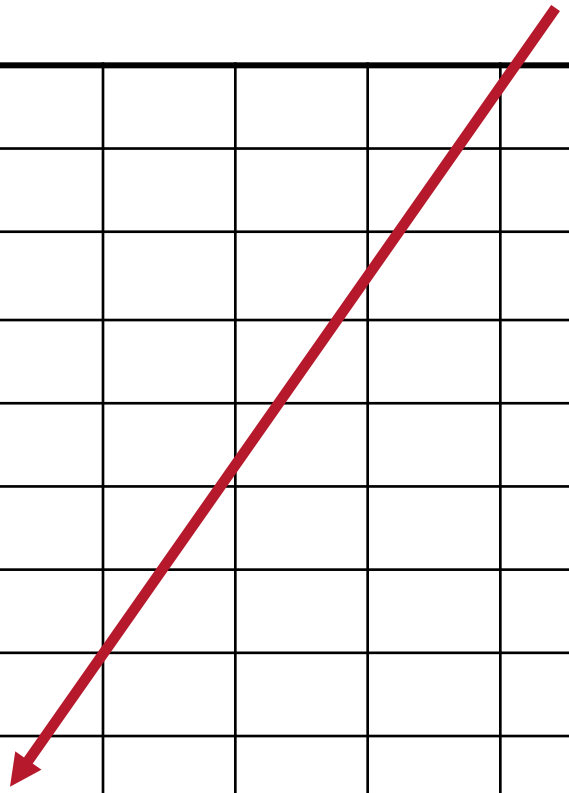
  D(N,M) is distance

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

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

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

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

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

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

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

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

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	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

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

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:

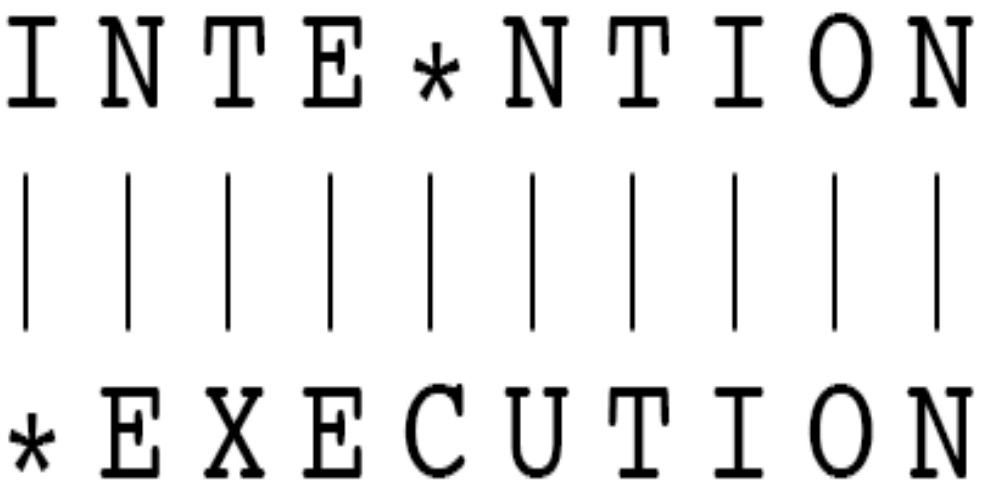
```
For each  i = 1…M
      For each  j = 1…N
                          D(i-1,j) + 1                    deletion
          D(i,j)= min     D(i,j-1) + 1                    insertion
                          D(i-1,j-1) +  2; if X(i) ≠ Y(j)
                                        0; if X(i) = Y(j)   substitution

                          LEFT                            insertion
          ptr(i,j)=       DOWN                            deletion
                          DIAG                            substitution
```

MinEdit with Backtrace

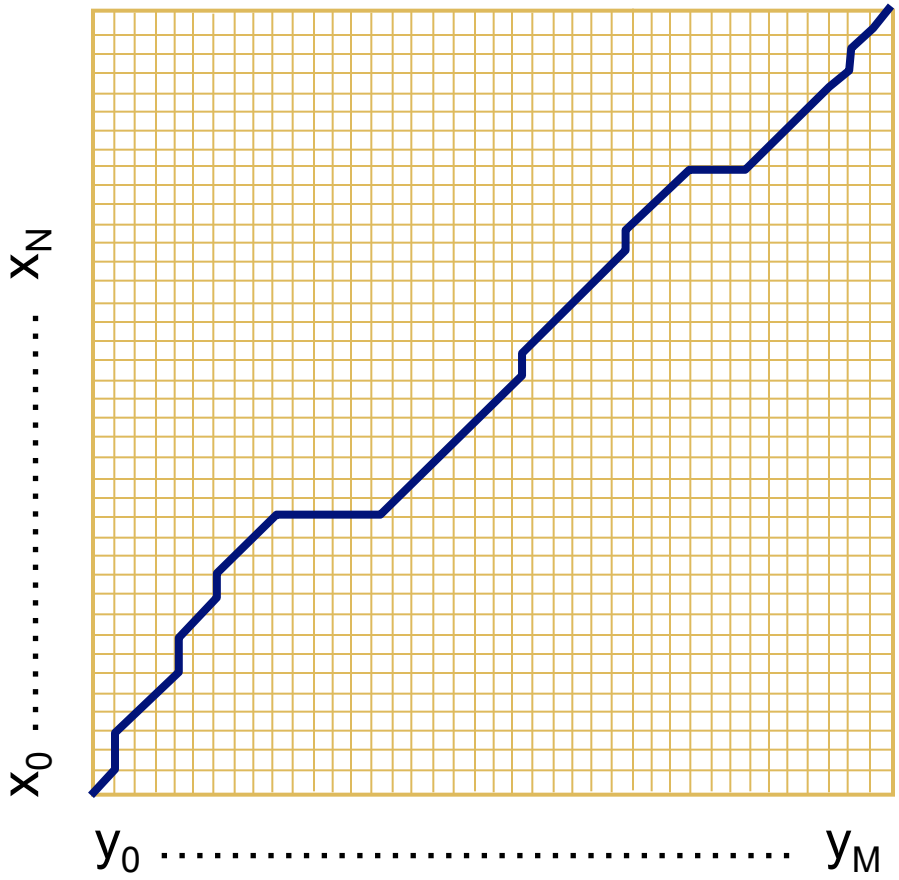
n	9	↓ 8	↙←↓ 9	↙←↓ 10	↙←↓ 11	↙←↓ 12	↓ 11	↓ 10	↓ 9	↙ **8**
o	8	↓ 7	↙←↓ 8	↙←↓ 9	↙←↓ 10	↙←↓ 11	↓ 10	↓ 9	↙ **8**	← 9
i	7	↓ 6	↙←↓ 7	↙←↓ 8	↙←↓ 9	↙←↓ 10	↓ 9	↙ **8**	← 9	← 10
t	6	↓ 5	↙←↓ 6	↙←↓ 7	↙←↓ 8	↙←↓ 9	↙ **8**	← 9	← 10	←↓ 11
n	5	↓ 4	↙←↓ 5	↙←↓ 6	↙←↓ 7	↙←↓ **8**	↙←↓ 9	↙←↓ 10	↙←↓ 11	↙↓ 10
e	4	↙ 3	← 4	↙← **5**	← **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**

Result of Backtrace

- Two strings and their **alignment**:



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 sub-alignments

Slide adapted from Serafim Batzoglou

Performance

- Time:

  O(nm)

- Space:

  O(nm)

- Backtrace

  O(n+m)

Weighted Edit Distance

- Why would we add weights to the computation?
  - Spell Correction: some letters are more likely to be mistyped 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

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-1,j)   + del[x(i)]
D(i,j)= min      ⎨ D(i,j-1)   + ins[y(j)]
                 ⎩ D(i-1,j-1) + sub[x(i),y(j)]
```

- Termination:

```
D(N,M) is distance
```

A Note on Arabic Morphology

- Form
  - Concatenative: prefix, suffix, circumfix
  - Templatic: root+pattern
- Function
  - Derivational
    - Creating new words
    - *Mostly templatic*
  - Inflectional
    - Modifying features of words
      - Tense, number, person, mood, aspect
    - *Mostly concatenative*

Arabic Derivational Morphology

- Templatic Morphology
  - Root: ك ت ب
    k=1 t=2 b=3
  - Pattern: ma12ū3 *passive participle*; 1ā2i3 *active participle*
  - Lexeme: مكتوب maktūb *written*; كاتب kātib *writer*

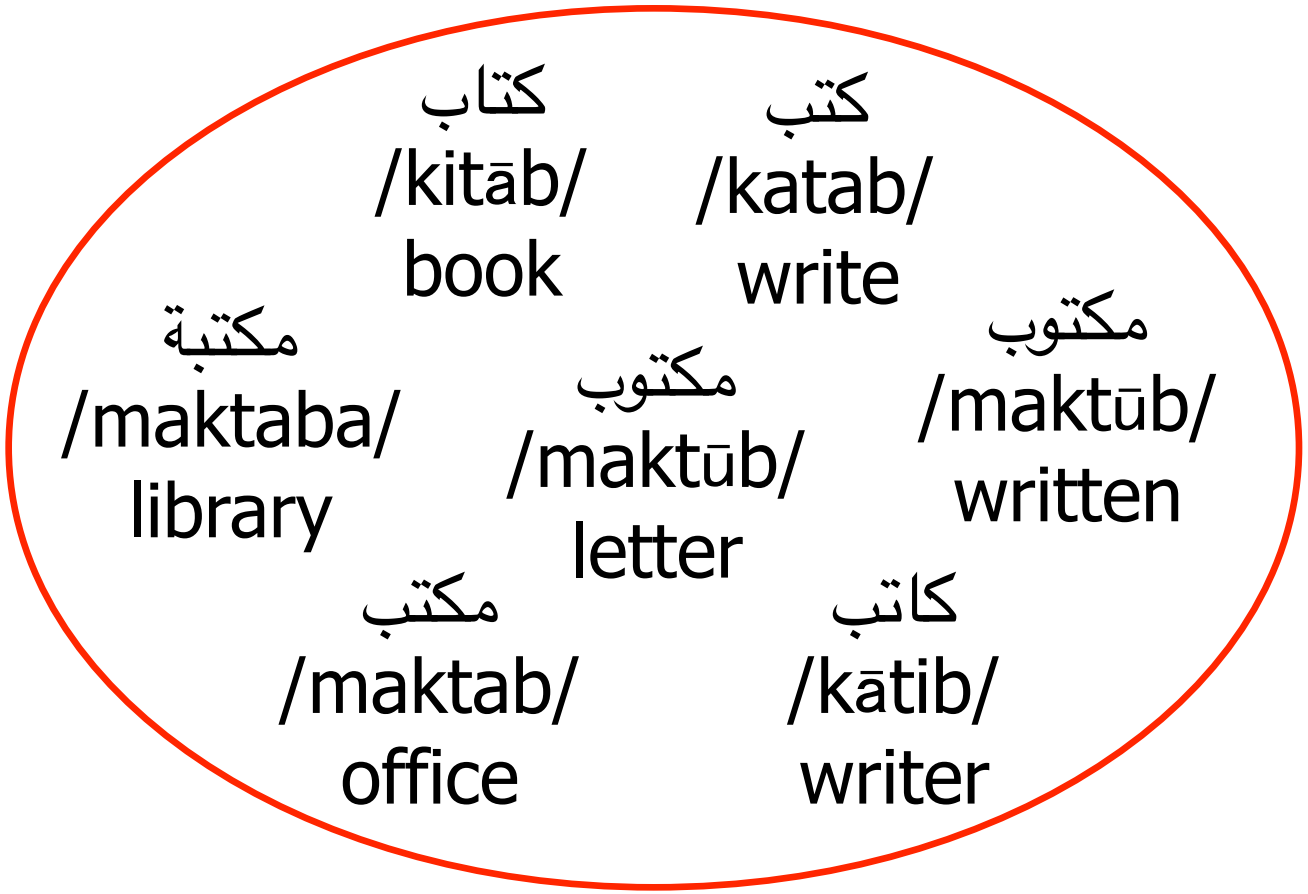
Lexeme.Meaning =
*(Root.Meaning+Pattern.Meaning)*Idiosyncrasy.Random*

Arabic Derivational Morphology

Root Meaning

KTB

ك ت ب



Arabic Root Polysemy

LHM-1 لحم

"meat"

/laħm/ لحم

Meat

/laħħām/لحام

Butcher

LHM-2 لحم

"battle"

/malħama/ ملحمة

Fierce battle

Massacre

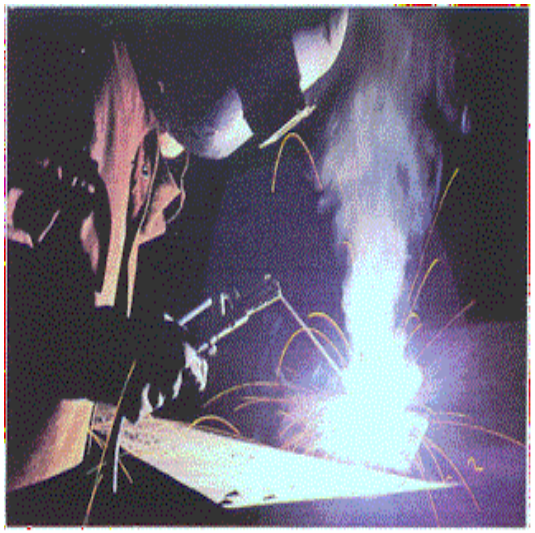
Epic

LHM-3 لحم

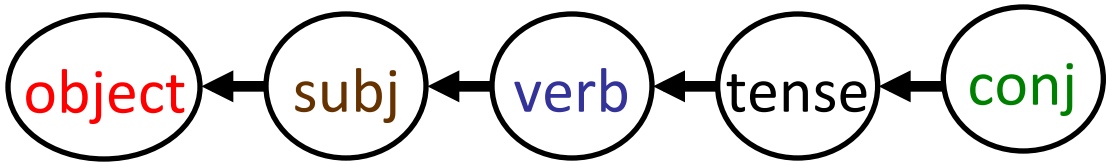
"soldering"

/laħam/ لحم

Weld, solder, stick, cling



Arabic Inflectional Morphology
Standard Arabic Verbs



وسنقولها
/wasanaqūluhā/
و+ س+ ن+ قول + ها
wa+sa+na+qūl+u+hā
and+will+we+say+it
And we will say it

فقلناها
/faqulnāhā/
ف+ قل+ نا+ ها
fa+qul+na+hā
so+said+we+it
So we said it.

- Morphotactics
- Subject conjugation (suffix or circumfix)

Inflectional Morphology

katab 'to write'

- **Perfect** verb subject conjugation (*suffixes only*)

	Singular	Dual	Plural
1	katabtu كتبت	katabnā كتبنا	
2	katabta كتبت	katabtumā كتبتما	katabtum كتبتم
3	kataba كتبَ	katabā كتبا	katabtū كتبوا

- **Imperfect** verb subject conjugation (*prefix+suffix*)

	Singular	Dual	Plural
1	aktubu اكتبُ	naktubu نكتبُ	
2	taktubu تكتبُ	taktubān تكتبان	taktubūn تكتبون
3	yaktubu يكتبُ	yaktubān يكتبان	yaktubūn يكتبون

Feminine form and other verb moods not shown

Inflectional Morphology

Terminology

Word	A space/punctuation delimited string	lilmaktabapi
Lexeme	The **set** of all inflectionally related words	maktabap, lilmaktabapi, Almaktabapu, walimaktabatihA, etc.
Lemma	An ad hoc word form used to represent the lexeme	maktabap
Features	The space of variation of words in a lexeme	Clitics: li_prep, Al_det, Gen:f, num:s, stt:d, cas:g
Root جذر	The root morpheme of the Lexeme	k-t-b
Stem جذع	The core root+pattern substring; it does not include any affixes	maktab
Segmentation	A shallow separation of affixes	li+l+maktab+ap+i
Tokenization	Segmentation + morpheme recovery	li+Al+maktab+ap+i

Inflectional Features

Feature Name			(Some Important) Feature Values	
PER	Person	الشخص	1st, 2nd, 3rd, na	متكلم، مخاطب، غائب، غ/م
ASP	Aspect	الزمن	perfect, imperfect, command, na	ماضي، مضارع، أمر، غ/م
VOX	Voice	البناء	active, passive, na	للمعلوم، للمجهول، غ/م
MOD	Mood	الصيغة	indicative, subjunctive, jussive, na	مرفوع، منصوب، مجزوم، غ/م
GEN	Gender	الجنس	feminine, masculine, na	مؤنث، مذكر، غ/م
NUM	Number	العدد	singular, dual, plural, na	مفرد، مثنى، جمع، غ/م
STT	State	التعريف	indefinite, definite, construct, na	نكرة، معرفة، مضاف، غ/م
CAS	Case	الحالة	nominative, accusative, genitive, na	مرفوع، منصوب، مجرور، غ/م

Cliticization Features

Feature Name			(Some Important) Feature Values	
PRC3	Proclitic 3	سابقة 3	>a_ques, 0	أداة استفهام، 0
PRC2	Proclitic 2	سابقة 2	fa_conj, wa_conj, 0	حروف عطف، 0
PRC1	Proclitic 1	سابقة 1	bi_prep, li_prep, sa_fut, 0	حروف جر، سين الاستقبال، 0
PRC0	Proclitic 0	سابقة 0	Al_det, mA_neg, 0	ال التعريف، أداة نفي، 0
ENC0	Enclitic	لاحقة 0	3ms_dobj, 3ms_poss, …, 0	ضمير مفعول به مباشر مفرد مذكر للغائب، ضمير ملكية مفرد مذكر للغائب، ... ، 0

Elixir FM (Smrž 2007)

- http://quest.ms.mff.cuni.cz/cgi-bin/elixir/index.fcgi?mode=resolve

Next Time

- Read J+M Chap 4 (intro up to 4.5)

- Assignment #2 due March 10 before midnight
  - Start early!