

# CSE 842

## Natural Language Processing

### Lecture 2: Morphology

1/14/2009

CSE842, Spring 2009, MSU

1

## What is morphology?

- The study of how words are composed of **morphemes** (the smallest meaning-bearing units of a language)
- Two broad classes of morphemes:
  - Stems: “main” morpheme of the word, supplying meaning
  - Affixes: Bits and pieces that combine with stems to modify their meanings and grammatical functions (prefixes, suffixes, circumfixes, infixes)
    - Unlike
    - Trying
- Multiple affixes
  - Unreadable

1/14/2009

CSE842, Spring 2009, MSU

2

## Ways to form words

- Inflection: new forms of the same word (usually in the same class)
  - Tense, number, mood, voice marking in verbs
  - Number, gender marking in nominals
  - Comparison of adjectives
- Derivation: yield different words in different class
  - Deverbal nominals
  - Denominal adjectives and verbs
- Compounding: new words out of two or more other words
  - Noun-noun compounding (e.g., doghouse)
- Cliticization: combine a word with a *clitic* (which acts syntactically like a word but in a reduced form, e.g., I've)

1/14/2009

CSE842, Spring 2009, MSU

3

## English Inflectional Morphology

- Word stem combines with grammatical morpheme
  - Usually produces word of same class
  - Usually serves a grammatical role that the stem could not (e.g. agreement)
    - like → likes or liked
    - bird → birds
- Nouns have a simple inflectional morphology: markers for plural and markers for possessives
- Verbs are slightly more complex:

1/14/2009

CSE842, Spring 2009, MSU

4

## Nominal Inflection

- Nominal morphology
  - Plural forms
    - s or es
    - Irregular forms, e.g., Goose/Geese, Mouse/Mice
  - Possessives
    - children's

1/14/2009

CSE842, Spring 2009, MSU

5

## Verbal Inflection

- Main verbs (**walk, like**) are relatively regular
  - -s, ing, ed
  - And productive: **Emailed, instant-messaged, faxed**
  - But **eat/ate/eaten, catch/caught/caught**
- Primary (**be, have, do**) and modal verbs (**can, will, must**) are often irregular and not productive
  - Be: am/is/are/were/was/been/being
- Irregular verbs few (~250) but frequently occurring
- English verbal inflection is much simpler than e.g. Latin

1/14/2009

CSE842, Spring 2009, MSU

6

## Regulars and Irregular Verbs

Morphological Form Classes	Regularly Inflected Verbs		
Stem	merge	try	map
-s form	merges	tries	maps
-ing participle	merging	trying	mapping
Past form or -ed participle	merged	tried	mapped

Morphological Form Classes	Irregularly Inflected Verbs		
Stem	eat	catch	cut
-s form	eats	catches	cuts
-ing participle	eating	catching	cutting
Past form	ate	caught	cut
-ed participle	eaten	caught	cut

1/14/2009

CSE842, Spring 2009, MSU

7

## English Derivational Morphology

- Word **stem** combines with grammatical **morpheme**
  - Usually produces word of **different class**
  - More complicated than inflectional
- Example: **nominalization**
  - -ize verbs → -ation nouns
  - generalize, realize → generalization, realization
- Example: verbs, nouns → adjectives
  - embrace, pity → embraceable, pitiable
  - care, wit → careless, witless

1/14/2009

CSE842, Spring 2009, MSU

8

- Example: adjective → adverb
  - happy → happily
- More complicated to model than inflection
  - Less productive: \*science-less, \*concern-less, \*go-able, \*sleep-able
  - Meanings of derived terms harder to predict by rule

1/14/2009

CSE842, Spring 2009, MSU

9

## Morphological Parsing

- Taking a surface input and identifying its components and underlying structure
- Morphological parsing: parsing a word into stem and affixes and identifying the parts and their relationships
  - Stem and **features**:
    - goose → goose +N +SG or goose + V
    - geese → goose +N +PL
    - geese → goose +V +3SG

1/14/2009

CSE842, Spring 2009, MSU

10

## Why parse words?

- For spell-checking
  - Is **munchable** a legal word?
- To identify a word's part-of-speech (pos)
  - For **sentence parsing**, for **machine translation**, ...
- To identify a word's stem
  - For **information retrieval**
- Why not just list all word forms in a lexicon?

1/14/2009

CSE842, Spring 2009, MSU

11

## What do we need to build a morphological parser?

- Lexicon: stems and affixes (w/ corresponding pos)
- Morphotactics of the language: model of how morphemes can be affixed to a stem
- Orthographic rules: spelling modifications that occur when affixation occurs
  - y → ie (e.g., city → cities)

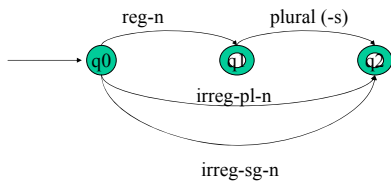
1/14/2009

CSE842, Spring 2009, MSU

12

## Morphotactic Models

- English nominal inflection



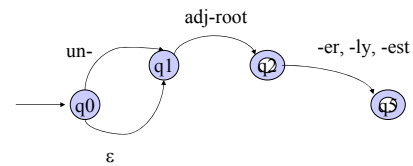
- Inputs: cats, goose, geese

1/14/2009

CSE842, Spring 2009, MSU

13

- Derivational morphology: adjective fragment



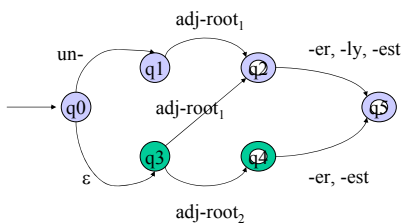
What kind of adjectives will this FSA recognize/generate?

1/14/2009

CSE842, Spring 2009, MSU

14

- Derivational morphology: adjective fragment



- Adj-root<sub>1</sub>: clear, real
- Adj-root<sub>2</sub>: big, red

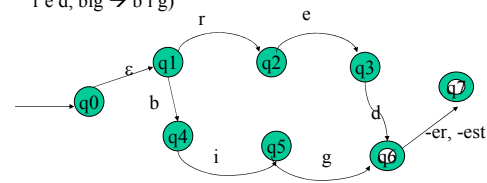
1/14/2009

CSE842, Spring 2009, MSU

15

## Using FSAs to Represent the Lexicon and Do Morphological Recognition

- Lexicon: We can expand each non-terminal in our NFSA into each stem in its class (e.g. adj\_root<sub>2</sub> = {big, red}) and expand each such stem to the letters it includes (e.g. red → r e d, big → b i g)



1/14/2009

CSE842, Spring 2009, MSU

16

## Limitations

- To cover all of e.g. English will require very large FSAs with consequent search problems
  - Adding new items to the lexicon means recomputing the FSA
  - Non-determinism
- FSAs can only tell us whether a word is in the language or not – what if we want to know more?
  - What is the stem?
  - What are the affixes and what sort are they?
  - We used this information to build our FSA: can we get it back?

1/14/2009

CSE842, Spring 2009, MSU

17

## Parsing with Finite State Transducers

- cats → cat +N +PL
- Kimmo Koskeniemi's **two-level morphology**
  - Words represented as correspondences between **lexical** level (the morphemes) and **surface** level (the orthographic word)
  - Morphological parsing: building **mappings** between the lexical and surface levels

	c	a	t	+N	+PL	→ Lexical
	c	a	t	s		→ Surface

1/14/2009

CSE842, Spring 2009, MSU

18

## Finite State Transducers

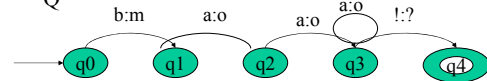
- FSTs map between one set of symbols and another using an FSA whose alphabet  $\Sigma$  is composed of pairs of symbols from **input** and **output** alphabets
- In general, FSTs can be used for
  - Translator (**Hello:Ciao**)
  - Parser/generator (**Hello:How may I help you?**)
  - To map between the lexical and surface levels of Kimmo's 2-level morphology

1/14/2009

CSE842, Spring 2009, MSU

19

- FST is a 5-tuple consisting of
  - $Q$ : set of states  $\{q_0, q_1, q_2, q_3, q_4\}$
  - $\Sigma$ : an alphabet of complex symbols, each an i/o pair s.t.  $i \in I$  (an input alphabet) and  $o \in O$  (an output alphabet) and  $\Sigma$  is in  $I \times O$
  - $q_0$ : a start state
  - $F$ : a set of final states in  $Q$   $\{q_4\}$
  - $\delta(q, i: o)$ : a transition function mapping  $Q \times \Sigma$  to  $Q$

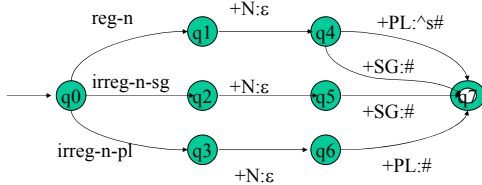


1/14/2009

CSE842, Spring 2009, MSU

20

## FST for English Nominal Inflection



1/14/2009

CSE842, Spring 2009, MSU

21

## FST for a 2-level Lexicon

- E.g.

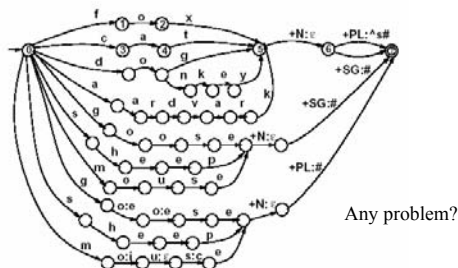
Reg-n	Irreg-pl-n	Irreg-sg-n
c a t	g o : e o : e s e	g o o s e

1/14/2009

CSE842, Spring 2009, MSU

22

## A fleshed-out English Nominal Inflection FST



1/14/2009

CSE842, Spring 2009, MSU

23

## Orthographic Rules and FSTs

- Define additional FSTs to implement rules such as **consonant doubling** (**beg** → **begging**), **'e' deletion** (**make** → **making**), **'e' insertion** (**watch** → **watches**), etc.

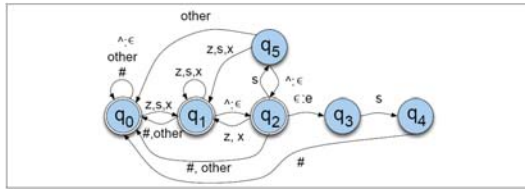
Lexical	f	o	x	+N	+PL	
Intermediate	f	o	x	^	s	#
Surface	f	o	x	e	s	

1/14/2009

CSE842, Spring 2009, MSU

24

## FSA for Orthographic Rules: E-Insertion

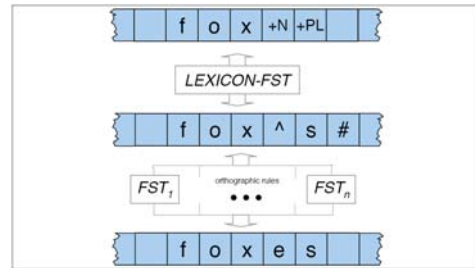


1/14/2009

CSE842, Spring 2009, MSU

25

## Overall Architecture



1/14/2009

CSE842, Spring 2009, MSU

26

## Stemming

- FSTs provide a useful tool for implementing a standard model of morphological analysis
  - Key is to provide an FST for each of multiple levels of representation and then to combine those FSTs using a variety of operators (cf AT&T FSM Toolkit)
- Other (older) approaches are still widely used, e.g. the rule-based Porter Stemmer (mainly for IR)
  - Only acquire the stems, not to use any morphological structure
  - Without access to a lexicon
  - Contain rules: e.g., ATIONAL -> ATE (relational -> relate)

1/14/2009

CSE842, Spring 2009, MSU

27

## Problems with Finite State Machines

- In the case of reject, they offer no advice as to why a string was rejected
- In the case of global ambiguity (two or more accept paths), FSAs simply provide all or one.
- Need augmentation

1/14/2009

CSE842, Spring 2009, MSU

28

## Minimum Edit Distance

- Is the minimum number of editing operations needed to transform one into the other
  - Insertion
  - Deletion
  - Substitution
- Many applications in string comparison/alignment, e.g., spell checking, WER in SR, machine translation, bioinformatics, etc.

1/14/2009

CSE842, Spring 2009, MSU

29

## 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

1/14/2009

CSE842, Spring 2009, MSU

30

## Minimum Edit Distance

One possible path

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

1/14/2009

CSE842, Spring 2009, MSU

31

## Defining Min Edit Distance

- For two strings  $S_1$  of len  $n$ ,  $S_2$  of len  $m$ 
  - distance( $i,j$ ) or  $D(i,j)$ 
    - means the edit distance of  $S_1[1..i]$  and  $S_2[1..j]$
    - i.e., the minimum number of edit operations need to transform the first  $i$  characters of  $S_1$  into the first  $j$  characters of  $S_2$
    - The edit distance of  $S_1, S_2$  is  $D(n,m)$
- We compute  $D(n,m)$  by computing  $D(i,j)$  for all  $i$  ( $0 \leq i \leq n$ ) and  $j$  ( $0 \leq j \leq m$ )

1/14/2009

CSE842, Spring 2009, MSU

32

## Defining Min Edit Distance

- Base conditions:

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

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

- Recurrence Relation:

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

1/14/2009

CSE842, Spring 2009, MSU

33

1/14/2009

33

## Dynamic Programming

- A tabular computation of  $D(n,m)$
- Bottom-up
  - We compute  $D(i,j)$  for small  $i,j$
  - And compute increase  $D(i,j)$  based on previously computed smaller values

1/14/2009

CSE842, Spring 2009, MSU

34

1/14/2009

34

## 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	

1/14/2009

CSE842, Spring 2009, MSU

35

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	

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

1/14/2009

CSE842, Spring 2009, MSU

36

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	

## Suppose we want the alignment too

- We can keep a “backtrace”
- Every time we enter a cell, remember where we came from
- Then when we reach the end, we can trace back from the upper right corner to get an alignment

## 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 MinEdit

- Base conditions:

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

- Recurrence Relation:

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

## 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	

## Performance

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