Computational Linguistics 1

CMSC/LING 723, LBSC 744

Kristy Hollingshead Seitz
Institute for Advanced Computer Studies
University of Maryland

Lecture 4: 13 September 2011

Agenda

- HW1 – due next Tuesday
- Questions?
- Morphology
  - Corrections from previous lecture
- Computational morphology
  - Continuation from previous lecture
- Phonology
- Computational phonology

Topology of Morphologies

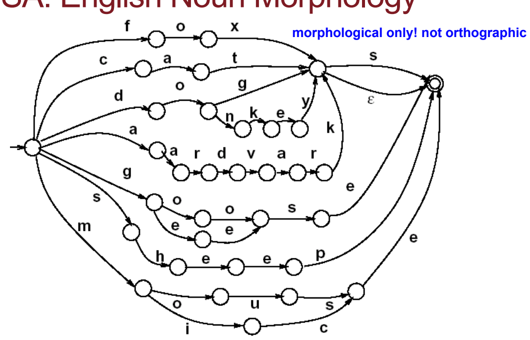
- Concatenative vs. non-concatenative
- Derivational vs. inflectional
- Regular vs. irregular

Inflection vs. Derivation vs. Compounding

- Inflection yields new forms of the same word
  - tense, number, mood, voice marking in verbs
  - case, number, gender marking in nominals
  - comparison of adjectives (e.g., big bigger biggest)
- Derivation yields different words
  - Derived nominals
  - Denominal adjectives
  - Denominal verbs
  - (adjectives & verbs derived from nouns)
- Compounding forms new words out of 2+ other words
  - Noun-noun compounding
  - Incorporation

FSA: English Noun Morphology

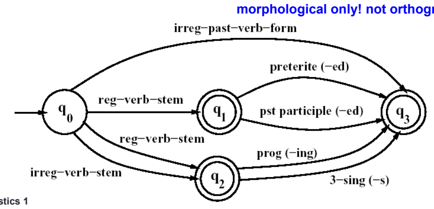
morphological only! not orthographic



FSA: English Verb Morphology

reg-verb-stem	irreg-verb-stem	irreg-past-verb	past	past-part	pres-part	3sg
walk	cut	caught	-ed	-ed	-ing	-s
fry	speak	ate				
talk	spoken	eaten				
impeach	sing					
	sang					

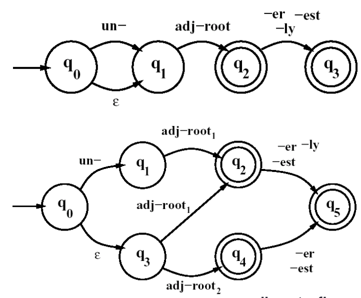
morphological only! not orthographic



FSA: English Adjectival Morphology

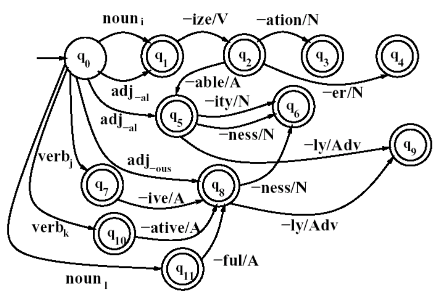
- Examples:
  - big, bigger, biggest
  - smaller, smaller, smallest
  - happy, happier, happiest, happily
  - unhappy, unhappier, unhappiest, unhappily
- Morphemes:
  - Roots: big, small, happy, etc.
  - Affixes: un-, -er, -est, -ly

FSA: English Adjectival Morphology



adj-root_1: {happy, real, …}
adj-root_2: {big, small, …}

FSA: Derivational Morphology

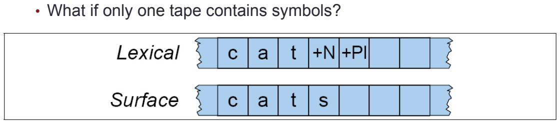


Agenda

- HW1 – due next Tuesday
- Questions?
- Morphology
  - Corrections from previous lecture
- Computational morphology
  - Finite-state methods: FSAs, FSTs
- Phonology
- Computational phonology

Morphological Parsing with FSTs

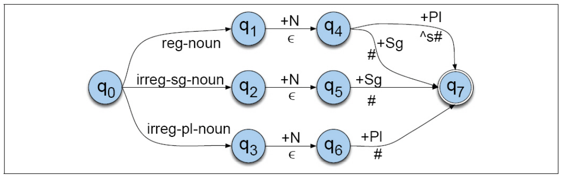
- Limitation of FSA:
  - Accepts or rejects an input… but doesn't actually provide an analysis
- Use FSTs instead!
  - One tape contains the input, the other tape as the analysis
  - What if both tapes contain symbols?
  - What if only one tape contains symbols?



Terminology

- Transducer alphabet (pairs of symbols):
  - a:b = *a* on the upper tape, *b* on the lower tape
  - a:ε = *a* on the upper tape, nothing on the lower tape
  - If a:a, write a for shorthand
- Special symbols
  - # = word boundary
  - ^ = morpheme boundary
  - (For now, think of these as mapping to ε)

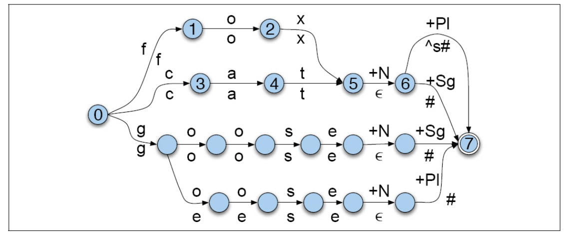
FST for English Nouns



- What's the problem here?

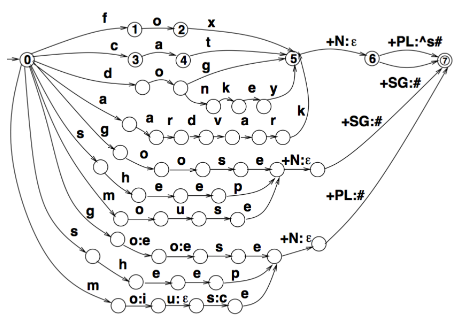
from Jurafsy & Martin

FST for English Nouns

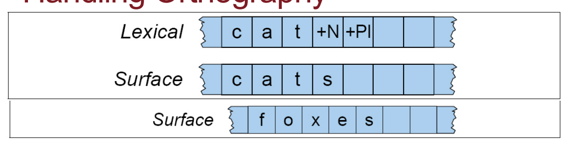


from Jurafsy & Martin

FST Expanded for More Nouns



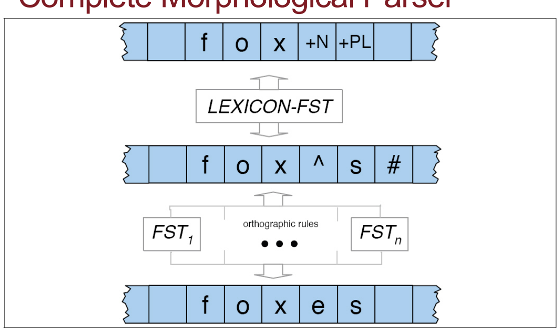
Handling Orthography



Name	Description of Rule	Example
Consonant doubling	1-letter consonant doubled before *-ing/-ed*	beg/begging
E deletion	silent e dropped before *-ing* and *-ed*	make/making
E insertion	e added after *-s,-z,-x,-ch, -sh* before *-s*	watch/watches
Y replacement	*-y* changes to *-ie* before *-s, -i* before *-ed*	try/tries
K insertion	verbs ending with *vowel + -c* add *-k*	panic/panicked

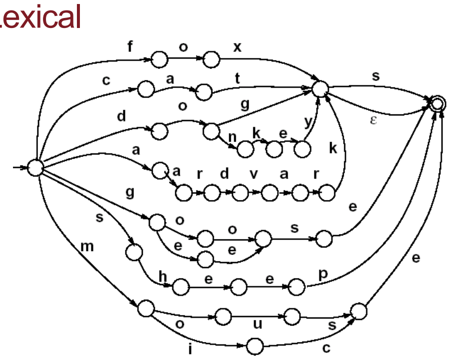
from Jurafsy & Martin

Complete Morphological Parser

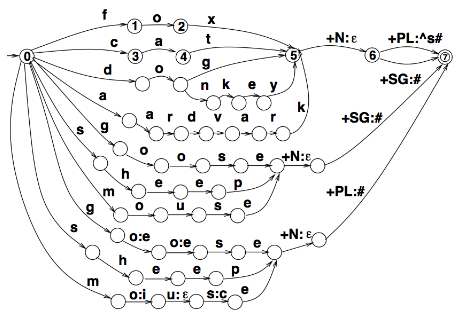


from Jurafsy & Martin

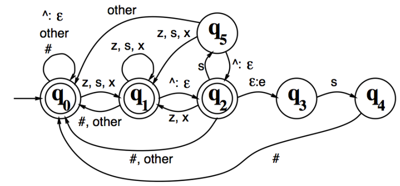
Lexical



Lexical to Morphemes



Morphemes to Orthographic Form



F O X ˆ S #

Practical NLP Applications

- In practice, it is almost never necessary to write FSTs by hand…
- Typically, one writes rules:
  - Chomsky and Halle Notation: a → b / c__d = rewrite a as b when occurs between c and d
  - E-Insertion rule

ε → e / {x, s, z} ^ __ s #

- Rule → FST compiler handles the rest…

from Jimmy Lin

Morphological Dictionaries

- Most commonly, simply build a dictionary from a closed vocabulary
- Compile dictionary into a transducer
- Exceptions for very productive morphological systems, e.g., Turkish, which result in too large a lexicon
- Having an explicit off-line dictionary allows for optimizations (structure sharing)
- Similar issues in phonology (coming up next!)

FSTs and Ambiguity

- unionizable
  - **union** +ize +able
  - un+ **ion** +ize +able
- assess
  - **assess** +V
  - **ass** +N +essN

Agenda

- HW1 – due next Tuesday
- Questions?
- Morphology
  - Corrections from previous lecture
- Computational morphology
  - Finite-state methods: FSAs, FSTs
  - One final question: is morphology finite?
- Phonology
- Computational phonology

Agenda

- HW1 – due next Tuesday
- Questions?
- Morphology
  - Corrections from previous lecture
- Computational morphology
  - Finite-state methods: FSAs, FSTs
- Phonology
- Computational phonology

Phonology

- Phonology is the study of *sound alternations* in language
- Computational phonology is computational models of those alternations
- Putting morphemes together to create words typically involves some amount of phonological alternation (sometimes quite a lot)
  - So computational morphology invariably also involves computational phonology too
- Most morphological analyzers deal with text
  - So what counts as computational phonology is really "computational orthography"

Orthography vs Phonology

- Some languages/writing systems have a very close relation between spelling and pronunciation
  - e.g., Spanish, Serbocroatian, Finnish, Turkish
  - In these languages,
    modeling spelling alternations ~ modeling phonological alternations
- In other languages, the spelling is relatively far removed from the pronunciation
  - English, French, Gaelic
  - In English, many of the alternations one must unravel in a morphological analyzer are spelling alternations

Orthography vs Phonology

- This can both help and hurt...
- Phonological alternations can be obscured by the spelling:
  - Newton Newtonian
  - maniac maniacal
  - electric electricity
- Or the spelling alternations may have no counterpart in the phonology:
  - innovate innovation
  - picnic picnicking
  - happy happiest
  - gooey gooiest

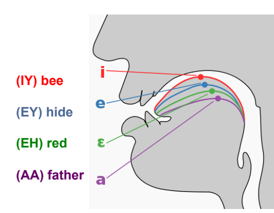
Phonemes

Phoneme	Example	Translation
AA	odd	AA D
AE	at	AE T
AH	hut	HH AH T
AO	ought	AO T
AW	cow	K AW
AY	hide	HH AY D
B	be	B IY
CH	cheese	CH IY Z
D	dee	D IY
DH	thee	DH IY
EH	Ed	EH D
ER	hurt	HH ER T
EY	ate	EY T
F	fee	F IY
G	green	G R IY N
HH	he	HH IY
IH	it	IH T
IY	eat	IY T
JH	gee	JH IY
K	key	K IY
L	lee	L IY
M	me	M IY
N	knee	N IY
NG	ping	P IH NG
OW	oat	OW T
OY	toy	T OY
P	pee	P IY
R	read	R IY D
S	sea	S IY
SH	she	SH IY
T	tea	T IY
TH	theta	TH EY T AH
UH	hood	HH UH D
UW	two	T UW
V	vee	V IY
W	we	W IY
Y	yield	Y IY L D
Z	zee	Z IY
ZH	seizure	S IY ZH ER

from the CMU Pronouncing Dictionary

Phonetic Classes

- Stops/Plosives
  - Voiced: b, d, g
  - Unvoiced: p, t, k
- Fricatives
  - Voiced: v, dh, z, zh
  - Unvoiced: f, th, s, sh
- Nasals
  - m, n, ng
- Liquids
  - l, r
- Vowels, Dipthongs
  - high/low, back, round



from Wikipedia

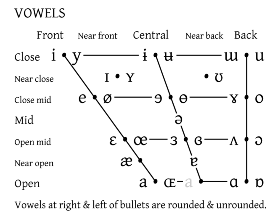
Phonetic Features

- **Height** (vertical dimension)
  - Height of tongue (high/low)
  - Relative frequency of the first formant (inverse)
  - Openness of jaw (close/open)
- **Backness** (horizontal dimension)
  - Position of the tongue during articulation
- **Roundedness** (lip position)

Encoding Vowel Features

- Typically binary features
- Encode place and manner of articulation

Feature	High	Low	Back	Round
IY ("bee")	+	-	-	-
UW ("two")	+	-	+	+
AY ("hide")	+	-	-	-
EH ("red")	-	+	-	-
AA ("odd")	-	+	-	-
AO ("hot")	-	+	+	+



from Wikipedia

Encoding Articulatory Classes

Feature	Values
voicing	+voice, -voice, silence
front-back	front, back, nil, silence
rounding	+round, -round, nil, silence
manner	stop, vowel, lateral, nasal, fricative, silence
cplace	labial, coronal, palatal, velar
vplace	glottal, high, mid, low, silence

- Can help to explain some orthographic phenomena
  - e.g., "inconceivable", "imperfect"

from Jurafsky & Martin

Interaction of Phonetic Features

- Rate of speech
  - How quickly can you move your articulators?
  - Speech is efficient
- Chomsky-Halle phonological re-write rules
  - Phonemes in a "phonetic environment", e.g., rule for flapping (t|d → dx)
- Statistical analysis
  - Phonetic reduction processes in fast speech
  - A word with higher conditional probability more likely to have reduced vowels or deleted consonants
  - Sociolinguistic factors: dialect, register, style
  - Coarticulation
- All important for speech recognition or synthesis

Agenda

- HW1 – due next Tuesday
- Questions?
- Morphology
  - Corrections from previous lecture
- Computational morphology
  - Finite-state methods: FSAs, FSTs
- Phonology
- Computational phonology

Computational Phonology

- Explicit rules to model alternations
- Constraint-based approach
  - Generate all variants, filter using surface constraints to disallow illegal variants
  - e.g., generate both "inperfect" and "imperfect" then filter (disallow) coronal-labial sequence of inperfect
- Optimality Theory (in SaLP)
  - GEN generates all possible forms.
  - Use a set of rank-ordered (supposedly universal) violable constraints to assign violations to each form
  - Of the set of forms and the worst violation assigned to each of them, choose the form with the *least* ranked of these violations

Ordered Rules vs Optimality Theory

- It has been argued that there is no computational difference between traditional ordered rules and Optimality Theory
- Traditional ordered rules can be implemented using composed transducers...
- ...so can we implement Optimality Theory using composed transducers?
- OT can be implemented using constraints *leniently composed* together

History of Computational Phonology

- The theory of phonology, based on Chomsky-Halle's rewrite rules, had the problem that unconstrained rewrite rules were too powerful
  - But in fact, the “context sensitive” rewrite rules, as they are invariably used in phonology, were really much weaker, and in fact are equivalent to regular relations
  - The main constraint is that such rules cannot apply arbitrarily to their own output
- So, if rewrite rules are implementable as FSTs, can one build a compiler that takes a set of these rules and produces an FST?
  - Kaplan & Kay, 1970s-1994

Two-Level Morphology [Koskenniemi]

- Rather than trying to compile rules into transducers and compose them serially, instead have a set of very compact transducers
  - Each transducer relates the surface and lexical forms
- The rules would be interpreted in parallel (formally equivalent to intersection)
- More than just a computational model: it was a theory of phonology
  - Essentially claimed that there was never any need to create intermediate levels between underlying (abstract) forms and surface forms
- Koskenniemi developed a set of transducers by hand for the entire morphology of Finnish

Two-Level Rules

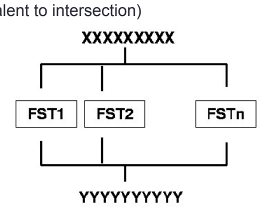
- Basic formalism:
  CorrespondencePair **op** LeftContext — RightContext
  - Exclusion rule a:b /⇐ LC __ RC
  - Context restriction rule a:b ⇒ LC __ RC
  - Surface coercion rule a:b ⇐ LC __ RC
  - Composite rule a:b ⇔ LC __ RC
- Interpretation:
  - **Exclusion rule**: *a* cannot be realized as *b* in the stated context.
  - **Context restriction rule**: *a* can only be realized as *b* in the stated context (and nowhere else)
  - **Surface coercion rule**: *a* must be realized as *b* in the stated context
  - **Composite rule**: *a* is realized as *b* obligatorily and only in the stated context

Systems Based on Two-Level Rules

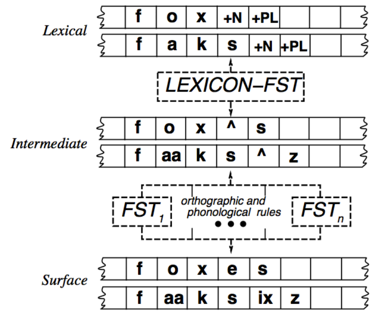
- Many morphological analyzers have been built using the Koskenniemi approach
- But many systems are not purely two-level: many systems are based on cascaded two-level rules
- Two-level rules are not strictly *necessary*; sometimes they make the description more convenient, but never required
- Systems of two-level rules and systems of cascaded rules are formally equivalent

Two-Level Rules as FSTs

- Recall that:
- Rather than trying to compile rules into transducers and compose them serially, instead have a set of very compact transducers
  - Each transducer relates the surface and lexical forms
- The rules would be interpreted in parallel (formally equivalent to intersection)



Lexical to Surface Form FSTs



Initialize FST

- From a simple dictionary:

FOX	F AA1 K S
FOXES	F AA1 K S AH0 Z
CAT	K AE1 T
CATS	K AE1 T S
DOG	D AO1 G
DOGS	D AA1 G Z
DONKEY	D AA1 NG K IY0
DONKEYS	D AA1 NG K IY0 Z
etc.	

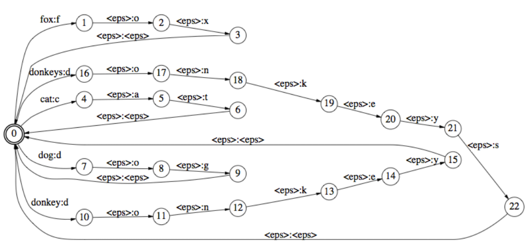
- Compile into a transducer offline & optimize

From Dictionary to Transducer (Format)

0	1	fox	f
1	2	<eps>	o
2	3	<eps>	x
3	0	<eps>	<eps>
0	4	cat	c
4	5	<eps>	a
5	6	<eps>	t
6	0	<eps>	<eps>
0	7	dog	d
7	8	<eps>	o
8	9	<eps>	g
9	0	<eps>	<eps>
0	10	donkey	d
10	11	<eps>	o
11	12	<eps>	n
12	13	<eps>	k
13	14	<eps>	e
14	15	<eps>	y
15	0	<eps>	<eps>
0	16	donkeys	d
16	17	<eps>	o
17	18	<eps>	n
18	19	<eps>	k
19	20	<eps>	e
20	21	<eps>	y
21	22	<eps>	s
22	0	<eps>	<eps>
0			

(spelling dictionary FST)

Create FST



- Optimize: determinize
- Test! Accept/reject, generate.

Agenda

- Morphology
  - Corrections from previous lecture
- Computational morphology
  - Finite-state methods: FSAs, FSTs
- Phonology
- Computational phonology
- Next time: language modeling, probabilistic models
- Homework due next Tuesday