Morphology

Computational Linguistics I: Jordan Boyd-Graber

University of Maryland

September 16, 2013
Outline

1. Why Morphology
2. Finite State Automaton
3. Finite State Transducer
Why morphology

Morpheme
Smallest unit of language that carries meaning

- “books”: two morphemes (“book” and “s”), one syllable
- “unladylike”: three morphemes, four syllables
- To do an analysis of language, we must do an analysis of the most fundamental unit of language!
- This subfield of linguistics is called morphology
Definitions

Derivational
You have a **new** word **derived** from an existing word that alters the **meaning**

- Nominalization: computerization, appointee, killer
- Adjectivization: computational, clueless, embraceable

Inflectional
You have a **variation** of a word that expresses **grammatical** contrast

- tense, number, person
- word class doesn’t change
- “The pizza guy comes at noon” (from “come”)

Definitions

- Root: common to a set of derived or inflected forms
- Stem: root or roots of a word together with derivational affixes
- Affix: bound morpheme that comes after or within a root or stem
- Clitic: a morpheme that functions like a word but doesn’t appear on its own (e.g., the ’ve in “I’ve”)
Examples

- Rechts+schutz+ver+sicher+ungs+gesell+schaft+en: Legal protection insurance policy (German)
- uygar+la¸s+tır+ama+dık+ları+dan+mış+sını+casına: Behaving as if you are among those whom we could not cause to become civilized (Turkish)
- “tú amaste” “ellos aman” “yo amaría” (Spanish)
- “I eat”, “he eats”, “they’re eating”, “I ate” (English)
- “wo ai”, “ni ai”, “ni.men ai” (Chinese)
Comparative Morphology

- Chinese is very easy
- English is fairly simple and regular
  - Few irregular verbs, but they’re frequent
  - Derivational morphology is very productive (e.g., “faxed”, “Skyped”, “Brittaed”)
Outline

1. Why Morphology
2. Finite State Automaton
3. Finite State Transducer
A Simple Problem

- We want to know whether a word is in a language or not
- For English, it’s possible to get by just with making a list
- Much harder for other languages
- Even for English, you miss out on derivations and inflections
A Simple Problem

- We want to know whether a word is in a language or not
- For English, it’s possible to get by just with making a list
- Much harder for other languages
- Even for English, you miss out on derivations and inflections
- Turn to a tool called Finite State Automaton (FSA)
We define a language to be a set of strings over some alphabet $\Sigma$.

- A set of states $Q$
- A designated start state $q_0$
- A set of accepting final states $F \subset Q$
- Edges: given current state $q_i$ and input $x \in \Sigma$, gives new state $q_j$

FSA over alphabet $\{a, b\}$
We define a language to be a set of strings over some alphabet $\Sigma$.

- A set of states $Q$.
- A designated start state $q_0$.
- A set of accepting final states $F \subset Q$.
- Edges: given current state $q_i$ and input $x \in \Sigma$, gives new state $q_j$.

Important tip: every state should have an edge for every element in the alphabet.
Examples

All binary strings

START

q1

q2

1,0,2

2

1
Examples

All non-zero binary strings of even length

START

- $q_1$
  - $0$
- $q_2$
  - $1,0$
- $q_3$
  - $0,1$
Examples

All non-zero binary strings of odd length

START

q1

q2

q3

0

1

1,0

1,0

1,0
Examples

Suppose we wanted to accept the language of questioning cows

- every string must begin with a “m”
- every string must end with a question mark “?”
- there can only be “o” in between
Inquisitive cow

START

q1

m

q2

o,

m,

q5

m,o,

q3

o

q4

?
What can you do with FSAs

- Equivalence to regular expressions
- Intersection: given two languages \((L_1, L_2)\), give \(L_1 \cap L_2\)
- Difference: given two languages \((L_1, L_2)\), give \(L_1 - L_2\)
- Complementation: given a language \(L_1\), give \(\Sigma^* - L - 1\)
- Reversal: given a language \(L_1\), give \(\{x : x^R \in L_1\}\)
- Concatenation: Given two languages \((L_1, L_2)\), give \(\{x : x = y + z, y \in L_1, z \in L_2\}\)
- Closure: infinite repetition
Uhh ... what about morphology?

- We’ve been talking about toy languages, but it works for real languages too
- Why do you want to recognize languages?
  - Spell checkers
  - Language identification
  - Speech synthesis
- Suppose you have an FSA for English stems (one for nouns, verbs, adjectives, etc.)
- Now suppose that you have an FSA that can generate inflectional forms
- Combine them with union / concatenation!
Nouns and their plurals

START

q1

d

c

q2

q3

q4

q5

q6

o

g

a

t
Nouns and their plurals
Non-deterministic FSA

- Allow empty input
- Allows multiple “universes” for strings to follow
- If any accepts, then it is part of the language
- Book uses $\epsilon$, I’ll use a blank edge
Non-deterministic composition
Non-deterministic composition
Non-deterministic composition
Non-deterministic composition
Outline

1. Why Morphology
2. Finite State Automaton
3. Finite State Transducer
FSA to FST

- FSA gives a binary input: is this a string or not
- What if we want to, for example, inflect words to reflect morphological variation? (Or vice-versa, given an inflected form, get back the stem.)
  - Useful for searching (“foxes” and “fox” are related)
  - Useful for generation: I want to say “go”, but what’s the third-person past tense?
- The answer is a finite state transducer
In addition to everything that you had from an FSA, now each transition also has an output (possibly empty)

Think of this as “translating” an input string to an output
Turning the inquisitive cow into emphatic sheep
Emphatic sheep strings start with “b” have any number of “a” and end with “!”
FSTs for Morphological Parsing

- Subject of first “real” homework
- Take input like “cat+N+Pl”
- Produce output like “cats”
- Read chapter 3.5 very carefully
- Read assignment carefully
In class . . .

- A quiz from this lecture! (won’t be as easy as last time)
- Answer your questions on the assignment
- Example problem converting between character sets