October 7, 2004

Chapter 3.1-3.2

Finite State Morphological Parsing

Overview

- Announcements: Assignment 1
- Morphology primer
- Using FSAs to recognize morphologically complex words
- FSTs (definition, cascading, composition)
- FSTs for morphological parsing

Morphology: A Primer (1/2)

- Words consist of stems and affixes.
- Affixes may be prefixes, suffixes, circumfixes, or infixes.
- (Also: root and pattern morphology)
- Phonological processes can sometimes apply to combinations of morphemes.

Morphology: A Primer (2/2)

- Languages vary in the richness of their morphological systems.
- Languages also vary in the extent to which phonological processes apply at (and sometimes blur) morpheme boundaries.
- English has relatively little inflectional morphology, but fairly rich (if not perfectly productive) derivational morphology.
- Turkish has more than 200 billion word forms.
- Bottom line: We'll want to be able to parse words morphologically, for reasons of linguistic interest as well as practical considerations.

Using FSAs to recognize morphological complex words

- Effectively concatenate FSAs for roots/stems with FSAs for affixes.
- Develop with word classes as stand ins for lists of words.
- Allow optional multiple affixes (end states with arcs leading out).
- Spelling change rules later (FSTs).

Using FSAs to recognize inflected words

- Why are there two end states in Figure 3.2 (p.67)?
- Why are there two arcs for -ed in Figure 3.3 (p.67)?
- Why are there two arcs labeled reg-verb-stem in Fig 3.3?

Using FSAs to recognize derivational morphology

- What sequence of states would the FSA in Fig 3.4 (p.68) follow to recognize *uncoolest*?
- *unbiggest*? (overgeneration)
- What sequence of states would the FSA in Fig 3.5 (p.69) follow as it tries to recognize *unbiggest*?
- Why do *-er* and *-est* show up on two different arcs in Fig 3.5?
- What sequences of states would the FSA in Fig 3.6 (p.70) follow to recognize *talkativeness*?

And a few more notes...

- The FSA in Fig 3.7 (p.70) is ‘compiled’ from the one in Fig 3.2 and a small lexicon.
- What does ‘compiled’ mean in this context?
- This FSA is also ‘minimized’. What does that mean?
- How far should morphological parsing go?

Finite-State Transducers

- Like FSAs, but with two tapes.
- Define *regular relations* – mappings between sets of strings.
- Can be used to recognize, generate, translate or relate sets.

Finite-State Transducers: Mealy machines

- Q: a finite set of states $q_0, q_1, \ldots, q_N$
- Σ: a finite alphabet of complex symbols $i : o$ such that $i \in I$ and $o \in O$. $\Sigma \subseteq I \times O$. I and O may each include ϵ.
- q_0: the start state
- F: the set of final states, $F \subseteq$ Q.
- $\delta(q, i : o)$: the transition matrix.

FSTs: Properties

- A set is *closed* under an operation if applying that operation to two elements of the set produces an element of the set.
- Regular relations are closed under union, but not necessarily difference, complementation or intersection.
- They are closed under inversion and composition.

FSTs: Non-linguistic example

- Father-of relation: $\{<$ Suku, Vijay $>, <$ Jim, Emily $>, <$ Vijay, Toby $>\}$
- Parent-of relation: $\{<$ Suku, Vijay $>, <$ Jim, Emily $>, <$ Vijay, Toby $><$ Vanaja, Vijay $>, <$ Sheila, Emily $>, <$ Emily, Toby $>\}$
- Grandfather-of relation: $\{<$ Suku, Toby $>, <$ Jim, Toby $>\}$
- Muthachan-of relation: $\{<$ Suku, Toby $>\}$

Composing FSTs (1/2)

- If the lower tape of one machine and the upper tape of the other can be aligned, then go straight from the upper tape of T_1 to the lower tape of T_2.
- Composing is equivalent to cascading (different from compiling).

Composing FSTs (2/2)

- Given automata T_1 and T_2 with state sets Q_1 and Q_2 and transition functions δ_1 and δ_2
- Create set of possible states: $\{(x, y) \mid x \in Q_1, y \in Q_2\}$
- Define new transition function:

  $\delta_3((x_a, y_a), i : o) = (x_b, y_b)$ if

  $\exists c$ such that $\delta_1(x_a, i : c) = x_b$

  and $\delta_2(y_a, c : o) = y_b$

An FST to parse English nouns

- T_{num} (Fig 3.9; p.74) parses the same set of nouns that the FSA in 3.2 recognizes.
- Why does it have more states?
- What are its input and output alphabets?
- The lexicon given for 3.9 has a funny spelling for only two words. Why?

An FST to parse English nouns

- Fig 3.10 (p.75) gives another FST T_{stems} which represents the lexicon for T_{num}.
- Why can't the lexicon just be listed?
- Why is there an arc labeled @:@ from q_1 to q_1?
- If 3.10 and 3.9 are cascaded, which applies first?

An FST to parse English nouns

- Fig 3.11 (p.76) gives T_{lex}, the result of compiling T_{stems} and T_{num}.
- What sequence of states does T_{lex} go through in parsing the input *goose* and what output does it give?
- What about for *geese*?

Overview

- Morphology primer
- Using FSAs to recognize morphologically complex words
- FSTs (definition, cascading, composition)
- FSTs for morphological parsing
- Next time: Spelling change rules, human morphological processing