Language and Statistics II

Lecture 7:  

Noah Smith
Finite-State Technology

• Formally well-understood
  – Regular languages, rational relations
  – Generalizes n-grams, HMMs

• Many applications in NL technologies
  – Speech recognition
  – Lexical, morphological processing
  – Information extraction
  – Translation (!)
  – Parsing (!)

• Several toolkits

• Often determinizable: very fast
Finite-State Automata (Recognizers)

- Automaton that recognizes **regular** language
- Implementation of a **regular** expression
- Regular languages are closed under numerous operations
  - Concatenation, union, intersection, Kleene *, difference, reverse, complement, ...
- Correspond to regular grammars (type 3 in Chomsky hierarchy)
- Pumping lemma: necessary condition for a language to be regular
FSM as a Dictionary

- Example: 850 words in “Basic English”
- Each word is an FSM
Ten-Word Dictionary
Remove $\varepsilon$-transitions
Determinize
Minimize
## Full 850-Word Dictionary

<table>
<thead>
<tr>
<th>Operation</th>
<th>states</th>
<th>final states</th>
<th>arcs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Union</td>
<td>5303</td>
<td>850</td>
<td>5302</td>
</tr>
<tr>
<td>Remove ε-transitions</td>
<td>4454</td>
<td>850</td>
<td>4453</td>
</tr>
<tr>
<td>Determinize</td>
<td>2609</td>
<td>848</td>
<td>2608</td>
</tr>
<tr>
<td>Minimize</td>
<td>744</td>
<td>42</td>
<td>1535</td>
</tr>
</tbody>
</table>
Algorithms

- Removing $\varepsilon$-transitions
- Determinization
- Minimization
Generalizations

• FS Recognizer is a function from $\Sigma^* \rightarrow \{0,1\}$
  – Meaning: $\text{fsa}(s) = 1 \iff s$ is in the language

• Other rational relations …
  – FS Transducer: $\Sigma^* \rightarrow \Delta^*$
  – Weighted FSA: $\Sigma^* \rightarrow \mathbb{R}$
  – Weighted FST: $\Sigma^* \rightarrow \Delta^* \times \mathbb{R}$

• WFSAs and WFSTs can be considered probabilistic (but don’t have to be)
WFSTs

FSTs

WFSAs

Mapping to \{0, 1\}

Output = Input
Finite-State Transducers

• Input alphabet $\Sigma$
• Output alphabet $\Delta$
• Set of states $Q$
• Initial state $q_0$
• Final states $F \subseteq Q$
• Arcs, $Q \times \Sigma \times Q \times \Delta^*$

**Sequential**: arcs are functions from $Q$ to $\Sigma \times Q \times \Delta^*$.

**$p$-subsequential**: deterministic, except each final state has at most $p$ output strings after each final state.
Finite-State Transducers

• Biggest application: morphology
  – Xerox tools: 20+ languages

• Example …

This represents one path.
Ambiguity and Optionality

• leaves ➔
  \{leaf +N +pl, leave +V +pres +3p +sing\}

• advice +maker ➔
  \{advisor, adviser\}

• inter+ nation +al +ize +ation ➔
  \{internationalization, internationalisation\}
Also, Phonology

- Mapping between pronunciation (phonemes or phonetic symbols) and lexical entries (morpheme sequences or orthography).
- Optionality even more necessary here!
FST Composition

\{vouloir \rightarrow veux, vouloir \rightarrow veut, vouloir \rightarrow voulons, vouloir \rightarrow voulez, vouloir \rightarrow veulent, \ldots \} 

\{veux \rightarrow vœ, veut \rightarrow vœ, voulons \rightarrow vulō, voulez \rightarrow vule, veulent \rightarrow vœl, \ldots \}
FST Composition

• Formally, \((x, z) \in f \circ g\) iff there exists \(y\) such that \((x, y) \in f\) and \((y, z) \in g\).

• Set and relation:
  \((x, z) \in f \circ g\) iff \(x \in f\) and \((x, z) \in g\)

• Relation and set:
  \((x, z) \in f \circ g\) iff \((x, z) \in f\) and \(z \in g\)

• Set and set (intersection):
  \(x \in f \circ g\) iff \(x \in f\) and \(x \in g\)

Basically, treat sets as identity relations.
Why?

• String into transducer (to compute $f(s)$): string is a set of size one
• Feed set of strings to transducer in parallel!
• Filter a relation by the outputs (compose with a filter set)
• Building a morphological lexicon: define lexicon by a FSA (set), rules by a bunch of transducers (relation)
FST Projection

• Can strip off input or output symbols … get a FSA. ( Might want to determinize.)
Weighted FSAs

• Instead of

  *Is it grammatical (possible)?*

  we might ask,

  *How grammatical (likely) is it?*

• Examples:
  – N-gram models
  – HMMs
  – Acoustic lattices
  – Perfect hash
Weighted Finite-State Acceptors

- Alphabet $\Sigma$
- Set of states $Q$
- Initial weight function, $\pi : Q \rightarrow \mathbb{R}$
- Final weight function, $\xi : Q \rightarrow \mathbb{R}$
- Arcs in $Q \times \Sigma \times Q \times \mathbb{R}$
Unigram model as a WFSA

One state.
One arc for every word.
Bigram model as a WFSA

|\Sigma| + 2 states.
One arc for every bigram.
Bigram HMM as a WFSA

- Alphabet $\Sigma$ (HMM’s alphabet)
- Set of states $Q$ (HMM’s states)
- Initial weight function, $\pi : Q \rightarrow \mathbb{R}$
  
  \[ (0 \text{ for start state, } -\infty \text{ for others}) \]
- Final weight function, $\xi : Q \rightarrow \mathbb{R}$
- Arcs in $Q \times \Sigma \times Q \times \mathbb{R}$
  \[ \log \gamma(s_0 | q) \]
  \[ \log \gamma(q' | q) + \log \eta(s | q) \]

Can you tell how to build a WFSA from a trigram HMM?
WFSA as a Log-Linear Model

• WFSAs assign weights to paths through the network.
• Can exponentiate, normalize, and interpret this as a joint $p(\text{path, output})$ or conditional $p(\text{path | output})$.
• Feature schemata:
  – Initial state is $q$
  – Stop state is $q$
  – Number of times arc $(q, q', s)$ was crossed
Weighted FSTs

• Weighted relation on $\Sigma^* \times \Delta^*$
• Like FSTs, closed under composition
• Examples:
  – Spelling correction
  – Morphological disambiguation
  – Edit distance
  – Machine translation
  – Speech recognition
Toolkits (links on course page)

• FSM libraries (AT&T)
  – Free binaries
  – Implements pretty much everything you need to build weighted and unweighted FS recognizers and transducers … except training!

• Xerox FS toolkit
  – Web demo; software can be purchased
  – No weights

• RWTH FSA toolkit
  – Newer, open-source
  – Not sure what’s implemented
Interesting analogy

- (weighted) Regexps
- Regexp compiler
- (W)FSTs
- Determinization, minimization, …
- Composition
- Inversion

- Source code
- Compiler
- Object code
- Optimization
- Composition
- Inversion
Summary

- FSAs, FSTs, WFSAs, WFSTs as formal systems, with reference to some key operations, algorithms, and toolkits.
- Next time(s): examples of WFSTs, and learning WFSTs from data
  - Speech and MT
  - Semirings
  - Parameter estimation
  - Grammatical inference