Announcements

- My office hours today:
  ~ 2pm
  ~ Room 6395 in CS building
- Will send out reading for today's lecture later today
  ~ in the future I will try to send this out earlier so you can read before the lecture

NLP Problem: Tokenization

- Important Terminology:
  - Type vs Token

- Input: raw text (stripped of markup)
- Output: sequence of word and punctuation tokens
- Dumb tokenizer: contiguous sequences of letters are tokens, contiguous sequences of digits are tokens, and contiguous punctuation symbols are tokens.
- How many lines of Perl?
NLP Problem: Tokenization

Dr. Smith said tokenization of English is “harder than you’ve thought.” When in New York, he paid $12.00 a day for lunch and wondered what it would be like to work for AT&T or Google, Inc.

- Commas, quotation marks, periods, apostrophes, capitalization - these are all ambiguous.
- Related but easier: sentence segmentation.
- Also easier to evaluate.

Orthography versus Morphology

- In NLP, these sort of run together, especially in English.
- We don’t usually care how words sound, we care how they are written.
  (Unless we’re dealing with speech data, or we think pronunciation might help us better model spelling errors, etc.)
- We’ll take a practical view; sometimes we’re solving orthography problems and sometimes morphology problems.

Morphemes

- A morpheme is a minimal, meaning-bearing unit of language.
- Too small (in English): ‘p’
- Too big: ‘processing’
- In some languages (Chinese), words and morphemes are basically the same.
- In some languages (Czech, Turkish), most words are made up of several morphemes.
- English is in the middle.
Inflection

- Change a word grammatically, usually to make it agree.
- Pluralizing a noun:
  - cat becomes cats
  - finch becomes finches
  - mouse becomes mice (irregular)
- Third person singular of a verb:
  - (catch) the cat catches a mouse
  - (kill) the cat kills mice
  - (have) the cat has a snack (irregular)
- Other tenses:
  - (kill) the cat is killing the mouse
  - (kill) the cat killed the mouse
  - (catch) the cat caught the mouse (irregular)

Irregularity is Common

- The verbs be, have, do:
  - be have do
  - am have do
  - are have do
  - is has does
  - was had did
  - were had did
  - been had done
  - being having doing

Question

- Does anyone have any good examples of inflectional morphology in another language?
Derivational Morphology

- Nominalization
  - digitize → digitization
  - code → coder

- Creation of adjectives
  - computation → computational
  - clue → clueless

- These changes are not productive; you can’t just use them on any verb (or noun).

What's A Word?

- Are ‘cat’ and ‘cats’ the same word?
- Are ‘computer’ and ‘computerize’?
- Is ‘haven’t’ the same as ‘have not’?
- What about ‘caught’?
- Can we enumerate all of the words in a language?
  - That set may be effectively infinite, especially if we want to handle newly created words.

NLP Problem: Stemming

- Input: word
- Output: stem
  - This is a “poor person's” morphology. Good enough for web search and some other things.
  - Porter stemmer (available online) is the most widely used.
    - Just a set of rules that transform suffixes:
      - -sses → -ss
      - -ies → i
      - -ss → s
      - etc.
    - Some counter-intuitive mappings (organization → organ; noise/noisy, Illustrated → illustr)
NLP Problem: Morphological Parsing

- Input: a word
- Output: morphological analysis of the word

Examples in English:
- geese → goose +N +Pl
- gooses → goose +V +3P +Sg

Key idea: a lexicon is a database of knowledge about some aspect of a language. Today, morphological lexicons.

Want to encode this knowledge intelligently, leveraging regularity where we can.

Turkish Example

uygarla ştıramadıklarımızdan mı şınızına

“(behaving) as if you are among those whom we were not able to civilize”

uygar "civilized"
+lag "become"
+vr "cause to"
+ama "not able"
+dik past participle
+lzr plural
+emiz first person plural possessive (“our”) +dan second person plural (“y’all”) +miz past +sniz ablative case (“from/among”) +casına finite verb → adverb (“as if”)
Ambiguity Rears Its Head

- dog → \{dog +N +Sg, dog +V\}
- leaves → \{leaf +N +Pl, leave +V +3P +Sg\}
- Our model should represent all of the alternatives.
- One day, we might want it to be able to disambiguate.
- Not today.

Simple Idea #1

- Great big table listing all of the words and their analyses.
- How fast to look up a word?
- How efficient to store?

Simple Idea #2

- Pack words together using prefixes, and build a trie (a special kind of finite-state automaton).
What We Want

- Regularity should be encoded directly in the lexicon.
- This will make it easier to build, modify, and understand our model.
- But we like being able to go “down to the letters” - that’s good for efficiency on a computer.
- And we like finite-state machines.

Super-Fast FSA Review

- Q: a finite set of states
- q₀ ∈ Q: a special start state
- F ⊆ Q: a set of final states
- Σ: a finite alphabet (with an epsilon symbol)
- Arcs:
  - Encodes a set of strings that can be recognized by following paths from q₀ to some state in F.

http://xkcd.com/851_make_it_better
Idea #3

- Use a trie for each morpheme category.
- Substitute each arc in the rule FSA with the appropriate trie.
- Result: great big letter-based FSA.
- What we get is a recognizer that tells us which words are in the language.
- Pro: derived using the rules (probably easier to build)
- Con: Gives us yes/no answers, not analysis!
Finite-State Transducers

- A two-tape automaton.
- Each arc now has two symbol strings (possibly from two alphabets).
- Instead of storing a language (like FSAs), stores a relation between sets of strings.
- Four views:
  - FSTs are recognizers that accept \( (x, y) \) iff \( x \mathcal{R} y \).
  - FSTs are generators that create \( (x, y) \) such that \( x \mathcal{R} y \).
  - FSTs are translators that read \( x \) and generate \( y \) such that \( x \mathcal{R} y \).
  - FSTs are set relaters that efficiently compute relations between sets.

Finite State Transducers

- \( Q \): a finite set of states
  - \( q_0 \in Q \): a special start state
  - \( F \subseteq Q \): a set of final states
- \( \Sigma \) and \( \Delta \): two finite alphabets
- Arcs:
  \[ q_i \xrightarrow{s: t} q_j \]
  \[ s \in \Sigma^* \text{ and } t \in \Delta^* \]

Some Formal Properties

- FSTs are closed under union and concatenation.
- Not closed under difference, complementation, or intersection.
- Closed under inversion: switch the input and output strings on all arcs.
- Closed under composition: \( A \circ B(\ldots) = B(A(\ldots)) \)
Some Formal Properties

• Recall that you can always determinize an FSA.
• Important for speed.
• Can’t determinize FSTs in general.
• Subclass called sequential transducers can be determinized.
  • Outputs and transitions deterministic given input, state
  • No \( \epsilon \) input strings
• Other interesting classes that permit some ambiguity.
• This is mainly about speed.

Morphological Parsing with FSTs

• Two tapes:
  • Lexical level: cat +N +Pl
  • Surface level: cat s

J&M pp. 54-55, 61-62

• Note “same symbol” shorthand.
• \(^\wedge\) denotes a morpheme boundary.
• \(^#\) denotes a word boundary.
Not Quite There ...

- This FST’s relation holds for \((\text{fox} + \text{N} + \text{Pl}, \text{fox}^\#)\).
- But “fox^#” is not English!
- This is just an intermediate form.
- To get all the way from “foxes” to “fox +N +Pl” (or vice versa), we need to think about spelling.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description of Rule</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consonant doubling</td>
<td>Silent e dropped before -ing or -ed</td>
<td>make/boxing</td>
</tr>
<tr>
<td>E-deletion</td>
<td>e added after -ing, -ed or -ad</td>
<td>watching</td>
</tr>
<tr>
<td>Y-replacement</td>
<td>y changes to u before -s or -ed</td>
<td>try/yries</td>
</tr>
<tr>
<td>E-insertion</td>
<td>v replaced with vowel + a or add -al</td>
<td>panic/paralyzed</td>
</tr>
</tbody>
</table>

J&M p. 63

The E Insertion Rule as a FST

![Diagram of the E Insertion Rule as a FST](image)

Putting It All Together

![Diagram illustrating the process of parsing and generating](image)

J&M p. 63
The Real Magic of FSTs

Another Advantage

- FSTs encode relations. So it's perfectly okay to have nondeterminism like ambiguity:
  (foxes, fox +N +Pl) as in "Englishmen hunt foxes."
  (foxes, fox +V +3P , +Sg) as in "He foxes me every time!"
- And also optionality:
  (tomatos, tomato +N +Pl)
  (tomatoes, tomato +N +Pl)

Final Note

- There are tools that you can use to help you build FSTs in more intuitive ways.
- XFST (from Xerox)
- FSM libraries (from AT&T)
- OpenFST (from Google - open source!)
- Important statistical generalization we may come back to: weighted FSTs.