Ling/CSE 472: Introduction to Computational Linguistics

4/13/17

Computational phonology

Announcement: Carriage returns

Overview

- Term projects: timeline
- Representing sounds
- Computational phonology: tasks
- FSTs for phonological rules
- Rule ordering and two-level phonology
- Optimality Theory: OT
- Machine learning of phonological rules
- Next time: TTS

Term projects: Timeline

- 4/21: Plan for final project
- 5/5: Revised plan for final project
- 5/29: Write-up outline + stage 1 results
    - This is Memorial Day, please plan ahead
- 6/1, 6/2: Presentations
- 6/7: Final project (executable + writeup)

Term projects: Specifications

- Evaluated in terms of precision and recall
- Comparison to baseline
- Two or more stage experiment where stage 2 tries to improve on stage 1 by changing the methodology in some way and measuring against the same gold standard (comparative evaluation)
- Must deal with natural language
- The write up is ***very important***

Due 4/21

- Decide if you'll work alone or with a partner
- Specify:
    - Task to be attempted
    - Data to be used
    - Means of measuring P & R
    - Any additional metrics
    - Who will do what (partner projects)

Before 4/18

- Be in contact with us (David, Emily) about your ideas, so we can help make sure they are feasible

- Use Canvas to discuss

- Explore what data are available:

  - https://vervet.ling.washington.edu/db/livesearch-corpus-form.php

  - Anything from LDC we can get, if we don’t have it already, but leave time for this

Overview

- Term projects: timeline
- Computational phonology: tasks
- Representing sounds
- FSTs for phonological rules
- Rule ordering and two-level phonology
- Optimality Theory: OT
- Machine learning of phonological rules
- Next time: TTS

Computational phonology: Representing sounds

- Orthographic systems are not always transparent representations of pronunciation.
    - Examples?
    - Why/when would we need to know how a word is pronounced?

Phonetics

- The study of the speech sounds of the world's languages

- Speech sounds can be described by their place and manner of articulation, plus some other features (oral/nasal, length, released/unreleased). [articulatory phonetics]

- Also: acoustic phonetics and perceptual phonetics

Phonetics

- Alphabetic writing systems represent the speech sounds used to make up words, but imperfectly:
    - Predictable phonological processes not represented (examples?)
    - Historical muddling of systems is common (examples?)
- IPA: An evolving standard with the goal of transcribing the sounds of all human languages.
- ARPAbet: A phonetic alphabet designed for American English using only ASCII symbols.

Phonological rules

- Much of the distribution of actual speech sounds in any given language is predictable.

- Particular phones can be grouped into equivalence classes (allophones) that appear in phonologically describable environments.

- Phonological and morphophonological rules relate underlying representations to surface forms.

- Computational phonology: What kinds of rules are required to model NL phonological systems, and how can they be implemented (with finite-state technology or otherwise)?

Computational phonology: Tasks

- Given an underlying form, what is its pronunciation?
- Given a surface form (pronunciation), what is the underlying form?
- Given an underlying (or surface) form, where are the syllable boundaries?
- Given a database of underlying and surface forms, what are the rules that relate them?
- Given a transcribed or written but otherwise unannotated corpus, what are the morphemes in it (and which ones are different forms of the same morpheme)?

SPE/FST rules

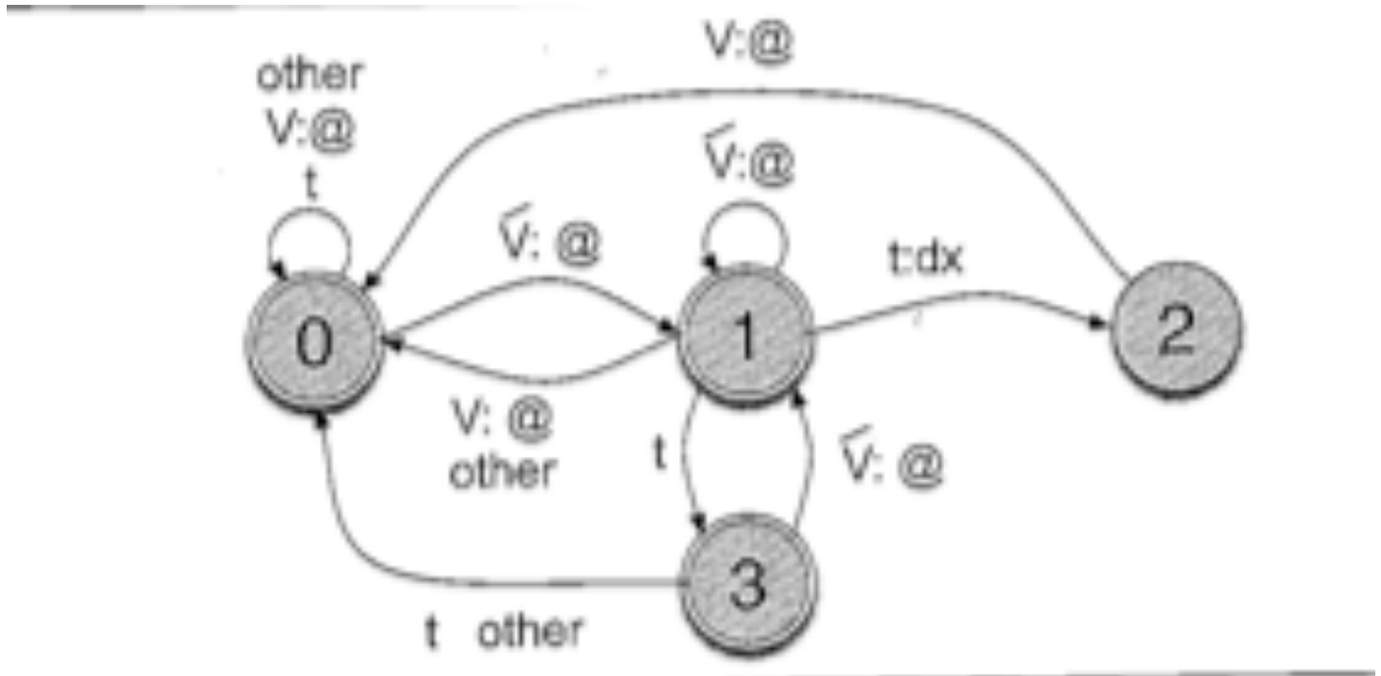
- Flapping rule: $/\mathrm{t}/ \rightarrow [\mathrm{dx}] \; / \; \acute{\mathrm{V}} \_\_ \; \mathrm{V}$

- “accepts any string in which flaps occur in the correct places, and rejects strings in which flapping should occur but doesn’t, or in which flapping occurs in the wrong environment”

- What strings should we use to test these claims?

Flapping rule as FST

- Fig 11.1, pg 362 of J&M. “other” is any feasible pair not used elsewhere in the transducer; “@” is any symbol not used elsewhere.



Rule ordering

- Rules can feed or bleed each other, but creating or destroying the next rule's environment.

- A long standing issue in phonology is whether rule systems require extrinsic ordering, or whether all ordering is intrinsic

- Example: faks+z ('foxes')

$$\epsilon \rightarrow [\text{barred i}] \; / \; [+\text{sibilant}] \; \hat{} \; \_\_ \; \text{z} \; \#$$

$$/\text{z}/ \rightarrow [\text{s}] \; / \; [-\text{voice}] \; \hat{} \; \_\_ \; \text{z} \; \#$$

More elaborate rule ordering: Yawelmani Yokuts

- Vowel harmony: suffix vowels agree in backness and roundness with the preceding stem vowel, if the vowels are of the same height.
- Lowering: Long high vowels become low.
- Shortening: Long vowels in closed syllables become short.
- Order: Harmony, Lowering, Shortening:
  - /ʔu:t’+it/ → [ʔo:t’ut]
  - /sudu:k+hin/ → [sudokhun]
- How do we know what the underlying forms are?
- How do these examples show that that’s the order?

Modeling rule ordering

- Cascaded or composed FSTs
- But: Most phonological rules are independent of each other.
- Koskenniemi's two-level rules run in parallel and finesse the issue of ordering by potentially referring to both underlying and surface forms.
- Example: Fig. 11.6

More on two level rules

- Two level rules can refer to upper or lower tape (or both) for both left and right context.

- Different types of two level rules differentiated by when they apply: a is realized as b whenever it appears in the context c d, only in that context, always and only, or never.

- XFST allows both approaches. Composing FSTs out of notionally ordered rules can be easier for linguists to maintain.

Another approach: Optimality Theory (OT)

- Grammar consists of GEN and EVAL
- GEN takes an underlying form and produces all possible surface forms.
- EVAL consists of a set of ranked constraints and an algorithm for choosing the best candidate.
- The best candidate is the one who's highest constraint violation is lower than any of the others. In the case of a tie, the next constraint violations are considered.
- Constraints are meant to be universal, ranking language-specific ⇒ 'factorial typology'

Example tableau

/?ilk-hin/	*Complex	FaithC	FaithV
?ilk.hin	*!		
?il.khin	*!		
?il.hin		*!	
?i.lik.hin			*
?ak.pid		*!	

Implementing OT

- Explicit interpretation of constraints
- GEN: a regular relation (FST)
- EVAL: Cascade the constraints, but with ‘lenient composition’, defined in terms of priority union (Karttunen 1998)
  - .P. (priority union): Q .P. R = Q | [~[Q.u] .o. R]
  - ‘Take all mappings from Q and those from R that don’t conflict.’
  - .0. (lenient composition): R .0. C = [R .o. C] .P. R
  - ‘Compose GEN with a constraint, but for inputs that have no perfect output, pass them through unchanged.’

Counting violations

- OT is finite-state under one important condition: There is a finite upper bound on the number of violations to be considered.

- The winning candidate is the one with the least violations of the highest-ranked constraint.

- Lenient composition alone isn’t enough to capture this.

- Instead: separate constraints for each number of violations

- Need to decide ahead of time how many to put in

Counting violations

- “It is curious that violation counting should emerge as the crucial issue that potentially pushes optimality theory out of the finite-state domain thus making it formally more powerful than rewrite systems and two-level models. It has never been presented as an argument against the older models that they do not allow unlimited counting. Is the additional power an asset or an embarrassment?” (Karttunen 1998, p.11)

Learning Rankings

- Tesar & Smolensky (1993, 1998): Error-Driven Constraint Demotion, learns ordinal rankings.

- Boersma (1997, 1998, 2000): Gradual Learning Algorithm learns stochastic rankings, can handle optionality and variation, as well as noisy training data.

Learning Rules

- Machine learning systems automatically induce a model for some domain, given some data and potentially other information.

- Supervised algorithms are given correct answers for some of the data and use the answers to induce generalizations to apply to further data.

- Unsupervised algorithms works only from data, plus potentially some learning biases.

Learning rules

- Ex: Gildea & Jurafsky (1996) specialize a learning algorithm for a subtype of FSTs to learn two-level phonological transducers from a corpus of input/output pairs.

- Learning biases: Faithfulness and Community

Reading questions

- What is the difference between a cascade and a pipeline?

- Starting on Page 363, reference is made to running multiple replacement rules in parallel. I'm having difficulty conceptualizing a way one would implement multiple such rules that didn't just involve applying one after the other. What exactly does it mean to run these things in parallel? How does the two-level model enable this? And do the rules need to be compatible in some way for this to work?

- How exactly are the rule operators in two-level morphology (pg 363) useful, i.e. how does one go from such rules to an FST? Also, it seems like not much definite information is gained from something like a:b=>c__d. Is such an expression supposed to be paired with something else, like more rules or a probability?

Reading questions

- Are there examples in which processing input in parallel vs. in a cascade with respect to a set of topological rules results in different outputs? Or are they always the same?

- The point about putting a syllabification transducer before a morphological parsing transducer so that syllabification can be influenced by morphological structure was interesting to me. What applications could this setup be used in?

Reading questions

- My question is about Optimality Theory. GEN produces all the surface forms, EVAL applies constraints to GEN in order of constraint rank. What does it mean by "ranked order" of constraints? How is the order generated?

- I am a bit confused on the concept of 'lenient composition' and why it would be beneficial for it to retain all candidates if no output met the constraint (above Fig. 11.10).

Reading questions

- Do markedness constraints play a role in computational optimality theory? The book doesn't discuss much about the GEN function and I'm wondering if markedness constraints would be included in the GEN function, perhaps they would just be more simple because they do not involve the output generated from EVAL.

- The book only seems to list reasons why Stochastic Optimality theory (particularly with Gausian distrobutions) is better than Non-Stochastic Optimality Theory. Is there any reason to ever implement Non-Stochastic Optimality Theory over SOT?

- I am wondering what are the common practices to avoid the program converged to a non-optimal state, when we are doing unsupervised learning?