The Role of Computation in Phonological Typology and Learning

Jane Chandlee
Haverford College
Dartmouth College
January 31, 2017
Computational Linguists pursue a variety of research goals:

- Algorithms and methods for handling natural language data.
  - Siri, Google Translate, Amazon Echo, etc.
- Using the study of computation to understand what language *is*.
  - Computational theory of language
# Levels of language

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Phonetics</strong></td>
<td>Production and perception of speech sounds</td>
</tr>
<tr>
<td><strong>Phonology</strong></td>
<td>Sound patterns</td>
</tr>
<tr>
<td><strong>Morphology</strong></td>
<td>Word formation processes</td>
</tr>
<tr>
<td><strong>Syntax</strong></td>
<td>Sentence structure</td>
</tr>
<tr>
<td><strong>Semantics</strong></td>
<td>Meaning</td>
</tr>
<tr>
<td><strong>Pragmatics</strong></td>
<td>Social/cultural conventions</td>
</tr>
</tbody>
</table>
Levels of language

<table>
<thead>
<tr>
<th>Levels</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phonetics</td>
<td>Production and perception of speech sounds</td>
</tr>
<tr>
<td><strong>Phonology</strong></td>
<td><strong>Sound patterns</strong></td>
</tr>
<tr>
<td>Morphology</td>
<td>Word formation processes</td>
</tr>
<tr>
<td>Syntax</td>
<td>Sentence structure</td>
</tr>
<tr>
<td>Semantics</td>
<td>Meaning</td>
</tr>
<tr>
<td>Pragmatics</td>
<td>Social/cultural conventions</td>
</tr>
</tbody>
</table>
Central question: what is the nature of the computations involved in phonological systems?

Main result: phonology is quite restrictive in its computational complexity, and this restrictiveness gives us insight into both cognition and language learning.
## Phonological patterns

<table>
<thead>
<tr>
<th>Phonotactics</th>
<th>Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>German: [zaːk] (*zaːg), ‘say’</td>
<td>/zaːɡ/ $\mapsto$ [zaːk]</td>
</tr>
<tr>
<td>English: [ɡɪəɪps] (*ɡɪəɪpz), ‘grapes’</td>
<td>/ɡɪəɪpz/ $\mapsto$ [ɡɪəɪps]</td>
</tr>
</tbody>
</table>
### Phonotactics

| Attested                  | Don’t end a word with sound $x$
|                          | Don’t start a word with sound $x$
|                          | Don’t allow sequences of sound $x$ followed by sound $y$
|                          | etc.                          |
| Unattested                | Don’t have an even/odd number of sound $x$ in a word
|                          | If a word starts with sound $x$ it can’t end with sound $y$
|                          | A word can have up to 3 sound $x$’s, but no more
|                          | etc.                          |
## Phonotactics

| Attested                          | Don’t end a word with sound $x$
|                                 | Don’t start a word with sound $x$
|                                 | Don’t allow sequences of sound $x$ followed by sound $y$
| etc.                             | |
| Unattested                        | Don’t have an even/odd number of sound $x$ in a word
|                                 | If a word starts with sound $x$ it can’t end with sound $y$
|                                 | A word can have up to 3 sound $x$’s, but no more
| etc.                             | |
## Phonotactics

| Attested | Don’t end a word with sound $x$
|          | Don’t start a word with sound $x$
<table>
<thead>
<tr>
<th></th>
<th>Don’t allow sequences of sound $x$ followed by sound $y$ etc.</th>
</tr>
</thead>
</table>
| Unattested | Don’t have an even/odd number of sound $x$ in a word
|          | If a word starts with sound $x$ it can’t end with sound $y$
|          | A word can have up to 3 sound $x$’s, but no more etc. |

Goal: Explain this boundary in terms of computational complexity.
Phonotactics as formal languages

- A formal language is a set of strings built from an alphabet, or set of symbols, $\Sigma$

(1) English: $\Sigma = \{ p, t, k, b, d, g, m, n,  noreferrer\eta, s, z, f, z, \ldots \}$

- A phonotactic constraint can be modeled with the set of strings that do not violate it.

(2) $\{ g\epsilonips, \æp\|z, \fizg, \æp.nkats, \ips, \ldots \}$
Classifying formal languages

Recursively-enumerable

| Context-sensitive
| Context-free
| Regular

| SF
| TSL  | LTT  |
|     | LT   |      |
|     |      | PT   |
| SL   |      | SP   |

(Chomsky, 1956; Rogers and Pullum, 2011; Rogers et al., 2013)
Hypothesis: phonotactics are regular

(Chomsky, 1956; Rogers and Pullum, 2011; Rogers et al., 2013)
Hypothesis: phonotactics are regular

Don’t end in [pz].
\[ \Sigma = \{p, z, s, a\} \]

Don’t have an odd number of [a]’s.
\[ \Sigma = \{p, z, s, a\} \]
Hypothesis: phonotactics are regular

Don’t end in [pz].
$\Sigma = \{p, z, s, a\}$

Don’t have an odd number of [a]’s.
$\Sigma = \{p, z, s, a\}$
Hypothesis: phonotactics are regular

Don’t end in \([pz]\).
\[\Sigma = \{p, z, s, a\}\]

Don’t have an odd number of \([a]\)’s.
\[\Sigma = \{p, z, s, a\}\]

\[\begin{array}{c}
\lambda \\
p \\
pz
\end{array}\]
\[\begin{array}{c}
\lambda \\
a
\end{array}\]

\[\begin{array}{c}
\lambda \\
z, s, p
\end{array}\]

\[\begin{array}{c}
z \\
a \\
p \\
z
\end{array}\]
Hypothesis: phonotactics are regular

Don’t end in [pz].
\[ \Sigma = \{p, z, s, a\} \]

Don’t have an odd number of [a]’s.
\[ \Sigma = \{p, z, s, a\} \]
Hypothesis: phonotactics are regular

Don’t end in [pz].
\[ \Sigma = \{p, z, s, a\} \]

Don’t have an odd number of [a]’s.
\[ \Sigma = \{p, z, s, a\} \]
Hypothesis: phonotactics are regular

Don’t end in [pz].
\[ \Sigma = \{p, z, s, a\} \]

Don’t have an odd number of [a]’s.
\[ \Sigma = \{p, z, s, a\} \]
Hypothesis: phonotactics are regular

Don’t end in [pz].
\[ \Sigma = \{p, z, s, a\} \]

Don’t have an odd number of [a]’s.
\[ \Sigma = \{p, z, s, a\} \]
Hypothesis: phonotactics are regular

Don’t end in [pz].
\[ \Sigma = \{p, z, s, a\} \]

Don’t have an odd number of [a]’s.
\[ \Sigma = \{p, z, s, a\} \]
Hypothesis: phonotactics are regular

Don’t end in [pz].
\[ \Sigma = \{p, z, s, a\} \]

Don’t have an odd number of [a]’s.
\[ \Sigma = \{p, z, s, a\} \]
Hypothesis: phonotactics are regular

Don’t end in [pz].
\[ \Sigma = \{p, z, s, a\} \]

Don’t have an odd number of [a]’s.
\[ \Sigma = \{p, z, s, a\} \]
Hypothesis: phonotactics are regular

Don’t end in [pz].
\[ \Sigma = \{p, z, s, a\} \]

Don’t have an odd number of [a]’s.
\[ \Sigma = \{p, z, s, a\} \]
Hypothesis: phonotactics are regular

Don’t end in [pz].
\[ \Sigma = \{ p, z, s, a \} \]

Don’t have an odd number of [a]’s.
\[ \Sigma = \{ p, z, s, a \} \]
Hypothesis: phonotactics are regular

Don’t end in [pz].
\[ \Sigma = \{ p, z, s, a \} \]

Don’t have an odd number of [a]’s.
\[ \Sigma = \{ p, z, s, a \} \]
Hypothesis: phonotactics are regular

Don’t end in [pz].
\[ \Sigma = \{p, z, s, a\} \]

Don’t have an odd number of [a]’s.
\[ \Sigma = \{p, z, s, a\} \]
Hypothesis: phonotactics are regular

Don’t end in \([pz]\).
\(\Sigma = \{p, z, s, a\}\)

Don’t have an odd number of \([a]\)’s.
\(\Sigma = \{p, z, s, a\}\)
✓ Hypothesis: phonotactics are regular

However,...
Hypothesis: phonotactics are *subregular*

(Chomsky, 1956; Heinz, 2007; Rogers and Pullum, 2011; Rogers et al., 2013)
Strictly Local FSAs

Don’t end in \([pz]\).
\[ \Sigma = \{p, z, s, a\} \]

States represent last segment(s) seen.

Don’t have an odd number of \([a]\)’s.
\[ \Sigma = \{p, z, s, a\} \]

States represent even/odd \([a]\)’s.
Phonological processes

- Assumption: the English plural suffix is /z/, but in some cases it is pronounced [s].
  
  \[
  \begin{array}{ll}
  bags & bægz \\
  chips & tʃɪps \\
  \end{array}
  \]

- To avoid sequences of [pz], we have a process that changes /z/ in this context to [s].
  
  \[
  tʃɪpz \rightarrow tʃɪps
  \]
Phonological processes as functions

- A processes can be represented with a function that maps tʃɪpz to tʃɪps.
- A function is a set of string pairs:
  
  \[
  \{(tʃɪpz, tʃɪps), (bægz, bægz), \ldots \}
  \]

- I’ll call these phonological maps (see also Tesar (2012)).
Complexity of phonological maps

**Regular relations** (Johnson, 1972; Kaplan and Kay, 1994)

↓

**Subsequential functions** (Mohri, 1997)

↓

**Strictly local functions** (Chandlee, 2014)
Strictly Local function

(4) Korean (Lee and Pater, 2008)
/papmul/ \(\mapsto\) [pammul] ‘rice water’

\[ \begin{align*}
V : V & \quad \rightarrow \quad m : m \\
V : pV & \quad \rightarrow \quad p : \lambda \\
p : p & \quad \rightarrow \quad p : p
\end{align*} \]
Strictly Local function

(4) Korean (Lee and Pater, 2008)
/papmul/ $\mapsto$ [pammul] ‘rice water’
Strictly Local function

(4) Korean (Lee and Pater, 2008)
/papmul/ \rightarrow [pammul] ‘rice water’
Strictly Local function

(4) Korean (Lee and Pater, 2008)
\(/\text{pampul}/ \rightarrow [\text{pammul}] \quad \text{‘rice water’}\)
Strictly Local function

(4) Korean (Lee and Pater, 2008)
/papmul/ $\mapsto$ [pammul] ‘rice water’
Complexity of phonological maps

- Local phonological processes are Strictly Local functions (Chandlee, 2014)
(5) Kikongo (Meinof, 1932; Odden, 1994; Rose and Walker, 2004)

/tunik-idi/ $\mapsto$ [tunik-ini] ‘we ground’

- SL version of this phonotactic constraint: don’t have [d] after [niki]
Long-distance phonotactics are TSL

(Heinz et al., 2011; McMullin, 2016)
First designate a subset of the alphabet, called the tier:
\( T = \{ n, d \} \)

Ignoring non-tier symbols, the constraint is:
‘Don’t have [d] after [n].’
Tier-based Strictly Local FSA
Tier-based Strictly Local FSA
Tier-based Strictly Local FSA
Tier-based Strictly Local FSA
Tier-based Strictly Local FSA
Tier-based Strictly Local FSA

diagram of a finite state automaton with states labeled by symbols and transitions labeled by arrows.

Symbols:
- \( \lambda \)
- \( n \)
- \( d \)
- ?

Transitions:
- \( \lambda \) to \( n \) labeled with \( d \)
- \( n \) to \( n \) labeled with \( ? \)
- \( d \) to \( ? \) labeled with \( d \)

Text below the diagram:

tu nik iid i
Tier-based Strictly Local FSA
Long-distance phonotactics are TSL (and therefore subregular)

(Heinz et al., 2011; McMullin, 2016)
Long-distance processes

What about long-distance maps?
Hierarchy of maps

Regular relations

Subsequential functions

Tier-based Strictly Local functions

Strictly Local functions
Tier-based Strictly Local functions

(6) /tunikidi/ $\mapsto$ [tunikini]
Tier-based Strictly Local functions

(6) /tunikidi/ $\mapsto$ [tunikini]
Tier-based Strictly Local functions

(6) /tunikidi/ $\mapsto$ [tunikini]

\[ \lambda \]

Diagram: 
- `n` with `n:n` edges
- `d` with `d:d` and `?:?,d:d` edges
- `?:?,n:n,d:n` edges

Text: 
- `x` with `t u n i k i d i` and `t u n`
Tier-based Strictly Local functions

(6) /tunikidi/ \mapsto [tunikini]
Tier-based Strictly Local functions

(6) /tunikidi/ $\mapsto$ [tunikini]
Tier-based Strictly Local functions

(6) /tunikidi/ \mapsto [tunikini]
Tier-based Strictly Local functions

(6) \[ /\text{tunikididi} / \mapsto [\text{tunikini}] \]
Tier-based Strictly Local functions

(6) \(/\text{tunikidi}/ \mapsto [\text{tunikini}]\)
Complexity of phonological maps

- Long-distance phonological processes are conjectured to be Tier-based Strictly Local functions (Chandlee et al., 2017)

```
\[
\text{Regular relations} \\
\downarrow \\
\text{Subsequential functions} \\
\downarrow \\
\text{Tier-based Strictly Local functions} \\
\downarrow \\
\text{Strictly Local functions}
\]```
Main result

- Both types of phonological patterns (phonotactics and processes) belong to subregular classes of formal languages and functions.
  - SL or TSL
- These classes provide a better fit to the typology than the regular languages and relations.
Implications for phonological learning

- The regular relations are not learnable from positive data...
- but the SL languages and functions are (Chandlee et al., 2014; Jardine et al., 2014)!
Implications for cognition

- What kind of information must we keep track of when performing phonological computations?
- Subregular analyses suggest it’s very limited.
Future work and open questions

- Fill out the hierarchy of subregular functions.
Subregular hierarchy of maps

- Regular relations
  - Subsequential functions
    - Tier-based Strictly Local functions
      - LTT?
      - LT?
      - PT?
    - Strictly Local functions
      - SP?
Future work and open questions

- Fill out the hierarchy of subregular functions.
- Identify logical characterizations of the various classes.
- Test whether subregular classes of FSTs improve efficiency of various NLP/HLT algorithms:
  - grapheme-to-phoneme conversion
  - pronunciation variation
  - etc.


