Learning allomorphs as a basis for morphophonemic learning

I. BACKGROUND: LEARNING MODELS IN LINGUISTICS

1. The research program of generative grammar

- Chomsky (1965:24-26) and similar aspirational work, on human children:

   ![Diagram](image)

   - Representative learning data → Learning device → Grammar → Elicitation and experimental testing

2. What is in the “learning device”??

   - Learning mechanisms of some sort
   - Linguistic theory; often construed as principles of Universal Grammar (UG)

3. The role of learning models

   - Chomsky describes people, but we can try to model people by devising systems that perform the same way — linguistics as reverse engineering.¹
   - What would success (in the distant future) look like?
     - Machine-learned grammars will respond to elicitation identically to native speakers, including ambivalence.
     - They will perform identically in wug-tests and other experimental tasks.
     - Algorithms will occasionally learn nonviridical patterns (mismatch to ambient data), just as people do.
   - I personally feel that a system as good as this is likely to be a reasonable analogue to the true system inside people’s heads.

4. Phonology seems to be a good area for trying this sort of work²

   - In Optimality Theory, we know how to rank constraints correctly, given a suitable set of inputs, winners, and losers (Tesar and Smolensky 1993, 2000)
   - Free variation doesn’t faze us, since several frameworks and ranking/weighting algorithms can accurately learn quantitative patterns presented to the learner (Boersma and Hayes 2000, Goldwater and Johnson 2003, Boersma and Pater 2016, etc.)

---

¹ Metaphor from Dupoux (2018).
• We can learn phonotactics from surface forms (Chomsky and Halle 1965: *brick, brick, bnick*); Hayes and Wilson (2008).

5. What has not yet happened (to my knowledge)

• A linguistic-based solver-system that can:
  ➢ input **paradigm data** — like a classical phonology problem set, where phonological processes cause the morphemes to show up in different forms in different parts of the paradigm (cf. (9) below).
  ➢ find the **underlying forms** of each morpheme present
  ➢ **discover and order rules** (Chomsky and Halle 1968), or **rank a constraint set** in Optimality Theory (Prince and Smolensky 1993) so as to derive the surface forms

6. Solving problem sets is an idealization/research way-station

• A large body of research tells us that the problem set answer falls far short of what real learners know about phonological patterning — there is just a lot more detail.
• I have ideas on how to scale up the work here to handle detail, but will not go into this here.

II. A INCREMENTALIST APPROACH TO SOLVING PARADIGM PROBLEMS

7. What would be a sensible thing to try first?

• How do human analysts best solve these problems?
• I have a guess, and it appears in my textbook, *Introductory Phonology* (2008:Ch. 8): a “5-step program”.

8. Toy illustration: Final consonants and clusters in Catalan

• Here is my pedagogical scheme as applied to five stems taken from Kenstowicz and Kisseberth’s (1979:328) beautiful Catalan problem, (9) below.
• Quick answer:
  ➢ Stems are /ultim/, /plen/, /klar/, /profund/, /fort/
  ➢ Fem. sg. and fem. plur. endings are /-ə/ and /-es/
  ➢ Phonology: /n/ and /r/ are deleted finally, after which /t,d/ delete / C ___ ], exposing new [n]’s and [r]’s to word-final position.

---

3 Computer scientists, racing ahead of us, can already solve paradigm problems, most effectively with neural networks; see Cotterell et al. (2017). Finding systems that perform as well or better as these will be an important test of our theoretical ideas, perhaps leading us to revise them.
9. **Step 1: divide the word into its morphemes**

| ultim | ‘last’ | profund | ‘deep’ |
| ultim | ‘last f.sg.’ | profund | ‘deep f.sg.’ |
| ultim | ‘last f.pl.’ | profund | ‘deep f.pl.’ |

| ple | ‘full’ | fort | ‘strong’ |
| plen | ‘full f.sg.’ | fort | ‘strong f.sg.’ |
| plen | ‘full f.pl.’ | fort | ‘strong f.pl.’ |

| kla | ‘plain’ |
| klar | ‘plain f.sg.’ |
| klar | ‘plain f.pl.’ |

10. **Step 2: Consulting the divisions made, list all allomorphs of each morpheme**

| ‘last’: | [ultim] | f.sg.: | [-ə] |
| ‘full’: | [plen] ~ [ple] | f.pl.: | [-es] |
| ‘plain’: | [klar] ~ [kla] |
| ‘deep’: | [profund] ~ [profun] |
| ‘strong’: | [fort] ~ [for] |

11. **Step 3: Consulting the allomorph list, find the segmental alternations**

n ~ Ø, r ~ Ø, d ~ Ø, t ~ Ø

12. **Step 4: Consider general hypotheses about underlying forms and reconstruct the derivations needed**

- A good guess: *in Catalan nouns the feminine (prevocalic) forms always provide the right basis for the UR* (more on this below).
- So we can run with this idea and set up “proto-derivations”:

| /plen/ | /plen-ə/ | /klar/ | /klar-ə/ | /profund/ | /profund-ə/ |

hypothesized underlying forms

| pleØ | klaØ | profunØ | what the phonology must do |
| [plen] | [plenə] | [kla] | [klarə] | [profun] | [profundə] |

surface forms

- This set-up requires us to discover deletion at Step 5
- A full implementation of the approach would also try the other option, /ple/, /kla/, /profun/ with consonant insertion — and would fail at Step 5.

13. **Step 5: Find a phonological grammar that will do what needs to be done**

- For the above, in rule-based phonology, this would be:
14. Can Steps 1-5, designed for humans, be made the basis of a learning algorithm?

- Step 1, “break up the words”, turns out to be hardest, and is the focus here.
- Steps 2-3, “find the alternations”, seems to be easier; see below.
- Step 4, “guess the UR’s” is trivial for Albrightianists, hard for others; see below.
- Step 5, “find the grammar that can achieve these mappings”, is easy if we are given OT constraints in advance, harder if we need to discover them.

- So let’s outline the five steps and then solve some phonology problems …

15. What I am trying to avoid by using an incremental strategy

- Explosion of hypothesis space: the set of possible UR’s coupled with grammars is incomparably vast for any decent-size problem.
- My own system does have a big search-space bottleneck — Step 1 — but I think it’s not too big.

16. The success of the strategy is a contingent matter

- Is it really possible to segment the words into morphemes without knowing the phonology?

17. A sobering example

- Thought experiment; you hear
  

- Is this /mapar/, with Final /r/ Deletion and suffix /-u/?
Could be, since Catalan works like this.

- Is this /mapa/, /tapan/ with suffix allomorphs /-u/ post-C, /-ru/ post-V?
  - Could be, since Japanese works like this.
- We need to try out both cases, figuring out what principles could distinguish the two.

III. PUTTING IN THE HYPHENS — DEFINING THE TASK

18. The task

- Given a glossed paradigm, as above, discover an appropriate division into morphemes, without yet knowing the phonology.

19. Making it harder for realism’s sake: discontinuous allomorphs

- Let include cases where the surface forms are discontinuous, due to
  - infixation, as in Tagalog [bago] ~ [b-um-ago].
  - metathesis of segments belonging to separate morphemes [pama], /naj-pama/ → [na-p-j-ama], as in Yagua (Powlison 1962)
- So we can’t just put in hyphens; we must instead coindex every segment with the morphemic gloss of the morpheme it belongs to.

20. Sample of input data: part of my fictional “Suffix Fricatives” language

kuŋanpa turtle-NOM ruxiŋpa dove-NOM tufærpa fox-NOM
kuŋanta turtle-DAT ruxiŋta dove-DAT tufærta fox-DAT
kuŋanka turtle-ACC ruxiŋka dove-ACC tufærka fox-ACC

piθoφa dog-NOM ḋẹxeφa wolf-NOM
piθoθa dog-DAT ḋẹxeθa wolf-DAT
piθoxa dog-ACC ḋẹnexa wolf-ACC

- Pretty clear to any phonologist that we have:
  - stems: /kuŋan/, /ruxiŋ/, /tufær/ , /piθo/, /ḍẹxe/
  - suffixes: /-pa/, /-ta/, /-ka/
  - /ptk/ spirantize intervocally to [θx].

21. Correct intended algorithm-output for Suffix Fricatives

k4u4ŋa4n4p1a1 turtle4 NOM1 r5u5x5i5p1a1 dove5 NOM1 t6u6φæx6p1a1 fox6 NOM1
k4u4ŋa4n4t2a2 turtle4 DAT2 r5u5x5i5t2a2 dove5 DAT2 t6u6φæx6t2a2 fox6 DAT2
k4u4ŋa4n4k3a3 turtle4 ACC3 r5u5x5i5k3a3 dove5 ACC3 t6u6φæx6k3a3 fox6 ACC3

p7iθo7φ1a1 dog7 NOM1 ḋẹxe8x8x8φ1a1 wolf8 NOM1
p7iθo7θ2a2 dog7 DAT2 ḋẹxe8x8x8θ2a2 wolf8 DAT2
p7iθo7θ3a3 dog7 ACC3 ḋẹxe8x8x8x3a3 wolf8 ACC3
IV. THE PROPOSED APPROACH

22. Overall plan

- Let’s use GEN + EVAL as in OT.
  - I.e., we lay out choices, then give formal criteria for picking a winner.
  - Such architectures are not just for linguistics, but are common in cognitive science as learning models.4

23. GEN and its size

- Our GEN = all possible coindexations of segments with the morphemes of their word.
  
  For: [sui] ‘pig1-VOC.2’
  GEN is: \{ s1u1i1, s1u1i2, s1u2i1, s1u2i2, s2u1i1, s2u1i2, s2u2i1, s2u2i2 \} 

- (21) shows the correct coindexation for the Suffix Fricative language; I calculate there are 634 octillion others.
- In general, the GEN needed is really big, an issue below.

24. Choice of constraint-based model

- I use Harmonic Grammar (Smolensky 1986, Pater 2009, Potts et al. 2010, etc.), so constraints are weighted, not ranked.
- For the computations I use the maxent variant of Harmonic Grammar (Smolensky 1986, Della Pietra et al. 1997, Goldwater and Johnson 2003), for reasons to be made clear.

25. How we extract a prediction from the model

- The intended division of the words into morphemes will be the highest-probability candidate.
- In maxent, there is a tweaking procedure (“temperature”; Smolensky 1986:224-226) for weights that can make the winner-probability as close to 1 as we desire
- So for practical purposes coming in first by any margin is the equivalent of total victory.

26. Constraints

- The learning system will include a variety of constraints embodying what my experience suggests are properties shown by correct morpheme divisions.
- We will run through these constraints in what follows.

---

4 See e.g. Samut (2010), entitled “Learning as search”, in a computer science encyclopedia.
27. The more important basis for constraints: similarity of allomorphs

- Here is the right division for the Suffix Fricative language in (20):

A. kuŋan-pa turtle₄ NOM₁  ruxin-ŋa dove₅ NOM₁  tufær-pa fox₆ NOM₁
  kuŋan-ta turtle₄ DAT₂  ruxin-ŋa dove₅ DAT₂  tufær-ta fox₆ DAT₂
  kuŋan-ka turtle₄ ACC₃  ruxin-ŋa dove₅ ACC₃  tufær-ka fox₆ ACC₃

B. piθo-φa dog₇ NOM₁  ηexe-φa wolf₈ NOM₁
  piθo-θa dog₇ DAT₂  ηexe-θa wolf₈ DAT₂
  piθo-xa dog₇ ACC₃  ηexe-xa wolf₈ ACC₃

- The right division yields this allomorph list:
  *-pa ~ -φa NOM
  *-ta ~ -θa DAT
  *-ka ~ -xa ACC


- Here is a sample wrong division:

kuŋan-pa turtle₄ NOM₁  ruxin-ŋa dove₅ NOM₁  tufær-pa fox₆ NOM₁
  kuŋan-ta turtle₄ DAT₂  ruxin-ŋa dove₅ DAT₂  tufær-ta fox₆ DAT₂
  kuŋan-ka turtle₄ ACC₃  ruxin-ŋa dove₅ ACC₃  tufær-ka fox₆ ACC₃

  *piθoφ-a dog₇ NOM₁  *ηexeφ-a wolf₈ NOM₁
  *piθoθ-a dog₇ DAT₂  *ηexeθ-a wolf₈ DAT₂
  *piθox-a dog₇ ACC₃  *ηexex-a wolf₈ ACC₃

- The wrong division yields this allomorph list:
  *-pa ~ -a NOM
  *-ta ~ -a DAT
  *-ka ~ -a ACC

  kuŋan ‘turtle’  ruxin ‘dove’  tufær ‘fox’

  *piθoφ ~ piθoθ ~ piθox ‘dog’
  *ηexeφ ~ nexeθ ~ nexex ‘wolf’

- These allomorphs are far less mutually similar than the allomorphs of the correct answer.

- N.B. Pressure toward allomorph-similarity is known elsewhere in phonology (historical change, Kiparsky (1982); elicitation from children, Jo (2017); artificial grammar learning studies (Wilson 2006; White 2013, 2014); infixation locus (Zuraw 2007)).
28. **SIMILARITY** stated intuitively

- “Penalize candidates to the extent that the allomorphs they imply fail to be mutually similar.”

29. **Making this concrete**

- Violations are based on the **summed dissimilarity of all allomorph pairs in the data** (all possible pairs of allomorphs of the same morpheme, summed across morphemes).
- English *visit* has three (in *visit, visitor, visitation*), so three comparisons would be made:

```
[vɪzət]  [vɪzəɾ]  [vɪzətʰ]
```

- The calculation is based on **type frequency**, not token frequency (each allomorph counted just once, no matter how many words it appears in).
  - I’ve tried token frequency and it doesn’t work as well.

30. **Calculating (dis)similarity in phonology**

- This is a long-studied problem and I’m following pretty standard methods.
- Let’s break it down.

31. **Breakdown**

- The **similarity of two segments** is best modeled as a weighted sum of the features on which they differ (Wilson and Obdeyn 2009).
- The **weights of the features** can be obtained (White 2012) by setting them to fit a maxent model to the error rates in a segment-confusion experiment.
  - I used the confusion matrices from Cutler et al. (2004).
  - You also need a value for dissimilarity of segments to null ("MAX/DEP"), as in [klar] ~ [kla], which I am obtaining by fitting to all of my learning simulations at once.
- The **similarity of two strings of segments** is the sum of the segmental differences under the best segment-by-segment alignment (Bailey and Hahn 2001, Albright and Hayes 2003).

32. **Alignment of two strings**

- Here is good alignment and a bad one for two of the *visit* allomorphs above:

```
a. v i z i t
   | | | | |
v i z i f

b. *v i z i t ∅
   | | | | |
v i z i f
```
33. **Summarizing the hierarchy of similarity-computation, bottom to top**

Feature weights → Segment dissimilarity → String dissimilarity →
Aggregate dissimilarity of a candidate morpheme-division

34. **First refinement to SIMILARITY: the stem/affix distinction**

- Languages widely rank Faithfulness constraints for stems higher than for affixes (McCarthy and Prince (1995), Casali (1997), Walker (2011:§2.5))
- So we might expect it to be useful to penalize stem-variation more harshly than affix-variation.

  \[\text{SIMILARITY}_{\text{STEM}}\]

  “Penalize candidates to the extent that the stem allomorphs they imply fail to be mutually similar.”

  \[\text{SIMILARITY}_{\text{AFFIX}}\]

  “Penalize candidates to the extent that the affix allomorphs imply fail to be mutually similar.”

35. **Second refinement to SIMILARITY: abstract away from size of data sample**

- Stem similarity is normalized by dividing by the total number of stems.
- Affix similarity is normalized by dividing by the total number of affixes.

36. **CONTIGUITY**

- **Penalize a morpheme whose segments are not contiguous.**
- Invented (for phonology) by McCarthy and Prince (1995)
- In practice, this penalizes:
  - **Real cases** (infixation, metathesis across morpheme boundaries, Semitic morphology). So the weight of CONTIGUITY cannot be infinite.
  - **Stupid candidates** we want to rule out (quite common). Consider an example modeled on real-life Lomongo, with intervocalic /b/ Deletion. Correct parse:

    ‘duck’
    
    molon-e baram-e NOM.
    molon-o baram-o ACC.
    pa-molon pa-aram GEN.
    ti-molon ti-aram DAT.

    Allomorphs of ‘goose’ are [baram] ~ [aram].
Wrong parse: ‘duck’ ‘goose’
molon-e b-aram-e NOM.
molon-o b-aram-o ACC.
pa-molon pa-aram GEN.
ti-molon ti-aram DAT.

— Wrong parse: ‘goose’ is [aram], [-e] and [-o] have circumfix allomorphs [b- -e] and [b- -o]; CONTIGUITY discourages them.

37. **VARIEGATION**

- This constraint is the least like ordinary phonology.\(^5\)
- Consider the **Prefix-Temptation** language: /kimen/, /kurat/, /petep/, /loran/, with prefixes /ni-/, /bi-/, /ri-/ undergoing vowel harmony (vowel changes to [u] next to stem [u]).\(^6\)
- Here is the bad analysis we want to avoid:

\[
\begin{align*}
n-ikimen & \text{ ‘sing 1p.’} \\
b-ikimen & \text{ ‘sing 2p.’} \\
r-ikimen & \text{ ‘sing 3p.’} \\
n-ipetep & \text{ ‘sit 1p.’} \\
b-ipetep & \text{ ‘sit 2p.’} \\
r-ipetep & \text{ ‘sit 3p.’}
\end{align*}
\]

\[
\begin{align*}
n-ukurat & \text{ ‘swim-1p.’} \\
b-ukurat & \text{ ‘swim-2p.’} \\
r-ukurat & \text{ ‘swim-3p.’} \\
n-uloran & \text{ ‘think-1p.’} \\
b-uloran & \text{ ‘think-2p.’} \\
r-uloran & \text{ ‘think-3p.’}
\end{align*}
\]

- This analysis is *perfect* with respect to SIMILARITY but is probably wrong.
- Such analyses can be discouraged by requiring the stem inventory to be Variegated.

38. **VARIEGATION (intuitively)**

- “Disfavor candidate parses to the extent that the stem inventory is dominated by a single frequent initial or final segment.”
- (37) is bad because every verb stem begins with either [i] or [u], an unlikely situation.
- I skip formalization for lack of time.

39. **Summary: the full constraint set**

\[
\begin{align*}
\text{SIMILARITY} & \text{STEM} \\
\text{SIMILARITY} & \text{AFFIX} \\
\text{CONTIGUITY} \\
\text{VARIEGATION}
\end{align*}
\]

\(^5\) As Paul Smolensky has pointed out to me, it is loosely related to the idea of the Rich Base in Optimality Theory (Prince and Smolensky 1993).

\(^6\) Maltese Arabic, below, seems to be close to a real-life example where this danger arises.
40. Assigning weights to the constraints

- I gathered a big set of candidate parses for all my simulation-languages (see below), with correct and incorrect candidates.
  - Some incorrect candidates constructed by me, based on the point of the example.
  - Some from failed learning runs, or from the “beam” of a beam search (see below).
- I calculated a set of weights by which the correct morpheme division always receives the best harmony score.
  - Don’t try to intuit the weights, since the violation scales are not comparable.

<table>
<thead>
<tr>
<th>Stem Similarity</th>
<th>12.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Affix Similarity</td>
<td>4.9</td>
</tr>
<tr>
<td>Variegation</td>
<td>240.0</td>
</tr>
<tr>
<td>Contiguity</td>
<td>10.8</td>
</tr>
</tbody>
</table>

- This relies on a huge virtue of Maxent: there is a provably reliable method for finding the weights that best fit the data. You can even perform the computation in Excel, as I did here.

41. Searching for the best candidate

- Now we have GEN (23) and EVAL (constraints and weights) — but how to find the winning candidate?
- Recall:
  - Candidates look like (21).
  - There are a huge number of them.
- Even worse, the search space is crammed with local maxima — the bane of learning-by-search.
- A few things have proven useful are given in (42).

42. Things that have helped

- Start by finding nonalternating segments and fixing their affiliations permanently.
- Then start the core search with random guesses.
- New candidates are found by trying out edits on the old ones.
- Use beam search — ten best candidates are kept in contention at once.
- Alternatingly search in various ways:
  - Search small: change affiliation of one segment at a time, or flip the affiliations of two segments.
  - Search big: re-conceive the search space as the list of allomorphs. Change an allomorph, and implement the change throughout the data.

V. THE SIMULATIONS I TRIED AND THEIR RESULTS

43. Choice of data sets

- 10 are made-up languages, meant to pose some particular challenge to the system
  - the variegation language of (37).
an infixation language

- Pseudo-Japanese, to test the Catalan/Japanese minimal pair described in (17).
- 10 are problem sets from Kenstowicz and Kisseberth (1979): Bizcayan, Chamorro, Catalan, Polish, Lamba, Maori, Maltese, Lomongo, Okpe, Modern Hebrew

- These often include dramatic phonological phenomena of the type beloved to phonologists, e.g. in Okpe:

\[
\begin{align*}
[z\u] & \quad \text{‘fan’} \\
/e-z\u\-\u/ & \rightarrow [ez\u\u] \quad \text{‘fan-inf.’}
\end{align*}
\]

with opaque harmony of [ATR], [round] and [nasal].

44. Sample parsing result: Catalan

- Data: 27 adjectival stems, in masc. sg. (-Ø), masc. pl. (-s), f. sg. (-\u), f.pl. (-es)).
- Sample issue:
  - We want [kl\u] to be parsed [klar] + [\u], not *[kla] + [r\u] (like Japanese).

45. Tableau

- Standard maxent calculations leading to probability values are given in bold.
- Candidates are parses of all 104 words in the problem, not just this stem.
- Just two candidates are presented; many others exist (they are bad, and lose)
- The bad candidate shown treats [r] as part of suffix, in Japanese fashion, in the three stems of the klar- pattern.

<table>
<thead>
<tr>
<th>SIMILARITYSTEM</th>
<th>SIMILARITYAFFIX</th>
<th>VARIEGATION</th>
<th>CONTIGUITY</th>
<th>Harmony</th>
<th>eHarmony</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>weights:</td>
<td>12.2</td>
<td>4.9</td>
<td>240.0</td>
<td>10.8</td>
<td>347.8</td>
<td>8.66 * 10^{-152}</td>
</tr>
<tr>
<td>Correct parse</td>
<td>19.4</td>
<td>15.1</td>
<td>0.1481</td>
<td>0</td>
<td>435.3</td>
<td>8.91 * 10^{-190}</td>
</tr>
<tr>
<td>Parse treating stem-final [r] as if it were suffix</td>
<td>14.4</td>
<td>45.4</td>
<td>0.1481</td>
<td>0</td>
<td>347.8</td>
<td>8.66 * 10^{-152}</td>
</tr>
</tbody>
</table>

- Essential violations:
  - Correct analysis suffers from stem allomorphy: [kla] ~ [klar] etc., so worse on SIMILARITYSTEM
  - Bad analysis suffers from affix allomorphy: [-\u] ~ [-r\u] and [-es] ~ [-res], so worse on SIMILARITYAFFIX
- Intuitively, the stem violations are less salient because they occur only in a small subset of the total stem count (cf. (35)).
46. How well did the system do in general?

- All 10 made-up languages, plus 8 real languages: the outcome of the search is the correct answer.
- As we hoped, Catalan and Pseudo-Japanese (17) are each given their own correct analysis.
- For Okpe and Hebrew, search fails, landing on a bad parse. But this parse is far less harmonic than the linguist’s parse, which I conjecture to be the true-but-unfindable optimum.

VI. THE REMAINING STEPS TO PHONOLOGY

47. Back to Step 2: list all allomorphs of each morpheme

- These are trivially read off the indexed representations used here, e.g. for (21) above:
  
k₄u₄ŋ₄a₄n₄p₄a₄ turtle₄ NOM₁
  k₄u₄ŋ₄a₄n₄t₄a₂ turtle₄ DAT₂
  k₄u₄ŋ₄a₄n₄k₃a₃ turtle₄ ACC₃
  r₅u₅s₅i₅ŋ₅p₅a₁ dove₅ NOM₁
  r₅u₅s₅i₅ŋ₅t₅a₂ dove₅ DAT₂
  r₅u₅s₅i₅ŋ₅k₃a₃ dove₅ ACC₃
  t₆u₆f₆x₆ŋ₆p₆a₁ fox₆ NOM₁
  t₆u₆f₆x₆ŋ₆t₆a₂ fox₆ DAT₂
  t₆u₆f₆x₆ŋ₆k₃a₃ fox₆ ACC₃
  p₇i₇θ₇o₇a₁ dog₇ NOM₁
  p₇i₇θ₇o₇t₂a₂ dog₇ DAT₂
  p₇i₇θ₇o₇x₃a₃ dog₇ ACC₃
  p₇i₇θ₇o₇θ₇o₇x₇ŋ₇x₇w₇δ₇a₁ wolf₇ NOM₁
  p₇i₇θ₇o₇θ₇o₇θ₇o₇x₇x₇w₇x₇δ₇a₂ wolf₇ DAT₂
  p₇i₇θ₇o₇θ₇o₇x₇x₇x₇x₇w₇x₇x₇δ₇a₃ wolf₇ ACC₃

- For the morpheme glossed as NOM., index 1, we get the allomorph set \{[-pa], [-φa]\}

48. Step 3: Consulting the allomorph list, find the segmental alternations

- We use the standard procedure of string-alignment-by-similarity; already used in calculating SIMILARITY violations ((32)).

\[\begin{array}{lcl}
a. & v & u \quad \text{not:} \\
| & | & |

b. & v & u \quad \text{∅} \\
| & | & |

\end{array}\]

\[\begin{array}{lcl}
v & o & dz \\
\text{∅} & v & o \quad dz \\
\end{array}\]

Therefore:

\[[u] \sim [o] \quad \text{is an attested alternation.}\]
\[[tɕ] \sim [dz] \quad \text{is an attested alternation.}\]
\[[u] \sim [v] \quad \text{is not an attested alternation.}\]

49. Step 4: Consider general hypotheses about underlying forms and reconstruct the derivations needed

- For the moment, let’s all be Albrightianists.
- There is one privileged slot in the paradigm from which the UR of a stem is taken; the child makes this choice early and sticks with it for life.
- See Albright (2002a,b; 2005; 2008a,b; 2012) for supporting evidence.
• There are so few phonologically-distinct paradigm slots that it is quick and easy for an algorithmic learner to try them all.
• Catalan: we try the word final allomorph and the prevocalic (feminine) allomorph; as shown earlier only the latter will work since it encodes the essential underlying distinctions in stem-final consonants.
• If at some point I want to try more ambitious, trans-Albrightian, UR’s, then we have a hidden-structure problem, and I hope to try out Jarosz’s (forthcoming) promising new approach.

50. Step 5: Find a phonological grammar that will do what needs to be done

• Let’s do it with OT.

51. We need a GEN

• Hooray, we know all the segmental alternations (Step 3), and we also have our candidate underlying forms.
• No surface allomorph can exist that is not derived from a UR by attested alternations.
• So, if we simply apply every alternation in every possible location, we will have a GEN that is in a sense “complete” and sufficient for learning.\(^7\)

52. GEN example: German phonology

• Classical data: stems contrasting in final obstruent voicing, neutralized to voiceless in final position


• We segment morphemes, find allomorphs, and find one alternation, the [t] ~ [d] of ‘wheel’.
• Activate GEN: using the extracted alternation [t] ~ [d], we replace the [t] of [rat] ‘advice’ with [d], obtaining a new allomorph *[rad]
  ➢ This allomorph is completely impossible, yet it this turns out to be good, because when we use it as a candidate, it tells us how to rank phonological constraints (Pater et al. 2012).

<table>
<thead>
<tr>
<th>/rat-əs/ ‘advice-gen.’</th>
<th>IDENT(VOICE)</th>
<th>*INTERVOCALIC VOICELESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>✗ rat-əs</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>✗ rad-əs</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>

➢ This shows that German is a final devoicing language like Dutch, not an intervocalic-voicing language like Korean.

---

\(^7\) This must be adjusted for epenthesis, which must be given a modest context to keep it from applying everywhere, or indeed an infinite number of times.
53. **Constraints**

- For now, I’m just being an unreconstructed classical OT person, typing in a rather thorough set of universal constraints (Markedness, Faithfulness).
- I am optimistic that these constraints can ultimately be learned from the data.

54. **For each choice of UR’s, do this:**

- Concatenate the aligned allomorphs (either attested or GENerated) appropriately to form words, combining them in all possible ways to create a candidate set for each input.
- So far, I’ve found that this fits on a spreadsheet, not more than a few thousand rows…

55. **The last step**

- Add to your spreadsheet the constraints and their violations, and perform OT constraint ranking with any one of the many reliable ranking or weighting algorithms (see (4) above).
- I use the venerable Recursive Constraint Demotion algorithm (Tesar and Smolensky 1993 et seq.).
- If ranking converges, then the UR’s you are testing are sufficient.

56. **Sample from a machine-generated tableau for Catalan**

- Choice of Albrightian UR for the stems: either of the Feminine forms (/ ___ -ə, -es).
- Candidates from GEN:
  - Catalan has [r] ~ [Ø] alternations like [klar-ə] ~ [kla], so GEN freely substitutes Ø for /r/.
  - Catalan has [k] ~ [ɣ] alternations elsewhere in the system, so GEN freely substitutes [ɣ] for /k/.

<table>
<thead>
<tr>
<th>/klar/</th>
<th>ID(voice)</th>
<th>*CODA R</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>ə̞ kla</td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>klar</td>
<td></td>
<td>✔️ !</td>
<td></td>
</tr>
<tr>
<td>ɣla</td>
<td></td>
<td>✔️ !</td>
<td>*</td>
</tr>
<tr>
<td>ɣlar</td>
<td></td>
<td>✔️ !</td>
<td>*</td>
</tr>
</tbody>
</table>

57. **Phonology problems solved so far in this way**

- Bizcayan, Catalan, Chamorro, Lamba, Lomongo, Okpe, Polish, also Maori (not KK).
- Not Maltese, nor my concocted pseudo-Yagua (cf. KK 73-74), since both of these have metathesis and my GEN so far is can only concatenate morphemes.

7. **CONCLUSIONS**

58. **Incrementalism**

- This is the key idea.
• Heuristics matching optimal behavior in phonologists keep the search space size under control, as we proceed through the Five Steps toward an answer.

59. An topic left unexplored here: non-phonological allomorphy

• Learning allomorphs is necessary no matter what, to learn irregular forms and non-phonological allomorphy — both quite abundant in languages.
  ➢ I.e. even a full-scale effort to learn underlying forms and phonology together (dominant approach in the current literature) would still need to do something else to discover non-phonologically-derived allomorphs.
• My system can learn a system of lexical allomorphs found in Persian verbs.

60. Future work

• Remove the idealization given above in (6): scale up to include the highly detailed environments, used by human learners.
• With this done, go more empirical, with wug-testing:
  ➢ Test the predictions of the learned grammars against intuitions of adult native speakers
  ➢ Test the predictions of grammars learned on child-size dataset on the productions and intuitions of children.

——— Finis ————
References


Albright, Adam and Bruce Hayes (2003). Rules vs. analogy in English past tenses: A computational/experimental study. Cognition 90. 119-161


Ernestus, Miriam and Harald Baayen (2003). Predicting the unpredictable: Interpreting neutralized segments in Dutch. Language 79, 5-38


