Neuro-symbolic phonology

Michael Hammond
U. of Arizona

April. 20, 2026

Overview

The plan today:

Phonology

Neural approaches

Neuro-symbolic computation

Testing

Rules

Conclusion

References

What is phonology?

The majority of human languages are spoken languages where words and sentences are made up of speech sounds. These sounds are subject to at least these important constraints:

- The set of sounds for any language is finite.
- The sounds themselves are meaningless.
- The set of sounds a language has is drawn from a larger (finite) set of possible sounds that all languages draw from.
- The set of sounds any language has is not random.
- The cooccurrence of sounds in words and phrases in any language is also not random.

Distribution of sounds: phonotactics

The sounds of a language are not randomly distributed in words or phrases. For example:

- English, Welsh, and Persian all have [s] and [t]. English and Welsh allow [st] at the beginning of words, but Persian does not.
- English, Welsh, and Persian all have [b] and [r], but Welsh and Persian allow [br] at the end of a word, e.g. in Persian [abr] ‘cloud’ and Welsh [ɬujbr] ‘path’, and English does not.
- English, Welsh, and Persian all have [k] and [n], but only Welsh allows [kn] at the beginning of a word, e.g. in [knauɥ] ‘nuts’

Calculating phonotactics

We can calculate the average predictability of segments based on previous segments in a language using average entropy.

$$H(X) = -\frac{1}{n}\sum_{i=2}^{n} p(x_i|x_{i-1}) \log_2 p(x_i|x_{i-1})$$

This is a measure of how much new information is provided by each subsequent segment.

Average entropy for some languages

How predictable is a segment given the preceding segment?

1 = not predictable at all
0 = completely predictable

Persian	.23	Choctaw	.27
Arabic	.26	Tibetan	.26
Navaho	.26	Welsh	.25
Amharic	.27	Polish	.26
English	.23		

These languages have very different inventories and phonotactics, but these numbers are strikingly similar.

(Data from `https://github.com/CUNY-CL/wikipron/`, prolog code in `ent.pl`)

Why should we care about phonology?

- Science. Language is a perhaps unique human artifact and we understand people better if we understand language. Phonology is part of language.
- Language description. Phonological systems vary widely and a central part of how languages differ. Describing a language for practical purposes (teaching, documentation, revitalization, etc.) entails also describing its phonology.
- Language technology. Phonological description is an often invisible yet integral part of how speech technology works. Mapping to or from a string of sounds (or letters) typically involves knowing what's a legal string of sounds.

Traditional theories of phonology

There have been many approaches to treating phonological generalizations. Here are a few:

- Categorical rules (Chomsky and Halle, 1968)
- Variable rules (Labov, 1969; Sankoff and Labov, 1979)
- Finite-state approaches (Karttunen, 1983)
- Strictly-ranked weighted constraints (Prince and Smolensky, 1993; McCarthy and Prince, 1993)
- Weighted constraints that are not strictly ranked (Hayes and Wilson, 2008)

What about a neural approach?

- Traditional analyses are usually hand-crafted by linguists.
- Traditional analyses allow us to posit specific general effects encoded by individual rules or constraints.
- Rule-based formalisms are built on models of human cognition that were in vogue in the middle of the last century.
- However, when we try to use traditional analyses for practical purposes, they do not work as well as neural approaches.

How neural nets work

Assume we want to have a neural net model the relationship between input and output forms.

1. Get a set of input–output pairs.
2. Convert those to numbers, e.g. convert each sound to a different vector and represent each form as a matrix of numbers.
3. Build a huge nested/sequential regression model that converts from one matrix of numbers to another.
4. Run the input–output pairs through the network again and again adjusting parameters for each batch of data until correct outputs are generated for each input.

Schematic neural net

Oversimplifying, we have lots and lots of little linear regression equations:

$$\begin{bmatrix} 3 \\ 7 \end{bmatrix} \rightarrow \begin{array}{l} a_1 3 + b_1 = y_1 \\ a_2 7 + b_2 = y_2 \end{array} \rightarrow a_3 y_1 + a_4 y_2 + b_3 = y_3 \rightarrow \ldots$$

The point is to learn the right values for a_1, b_1, a_2, b_2, a_3, a_4, and b_3, so we can map to the right output. Modern neural nets can have *millions* of such values to learn.

Input to output

$$\text{hat} \rightarrow \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow \boxed{\text{net}} \rightarrow \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow \text{шляпа}$$

Why and why not neural nets

- Why?
  - It's provable that certain kinds of neural nets are "Turing-complete", that they can model any relationship we might want (Pérez et al., 2019).
  - With sufficient appropriate data, a suitable network will learn whatever generalizations exist in the data.
- Why not?
  - Do we have sufficient training data?
  - The network can't tell us what the generalizations are.
  - A neural net can learn anything, including things that aren't languages.
- So do we want accuracy/precision or do we want interpretability?

Frequency effects for LLMs

Neural nets are susceptible to unfortunate frequency effects as a function of training (McCoy et al., 2024):

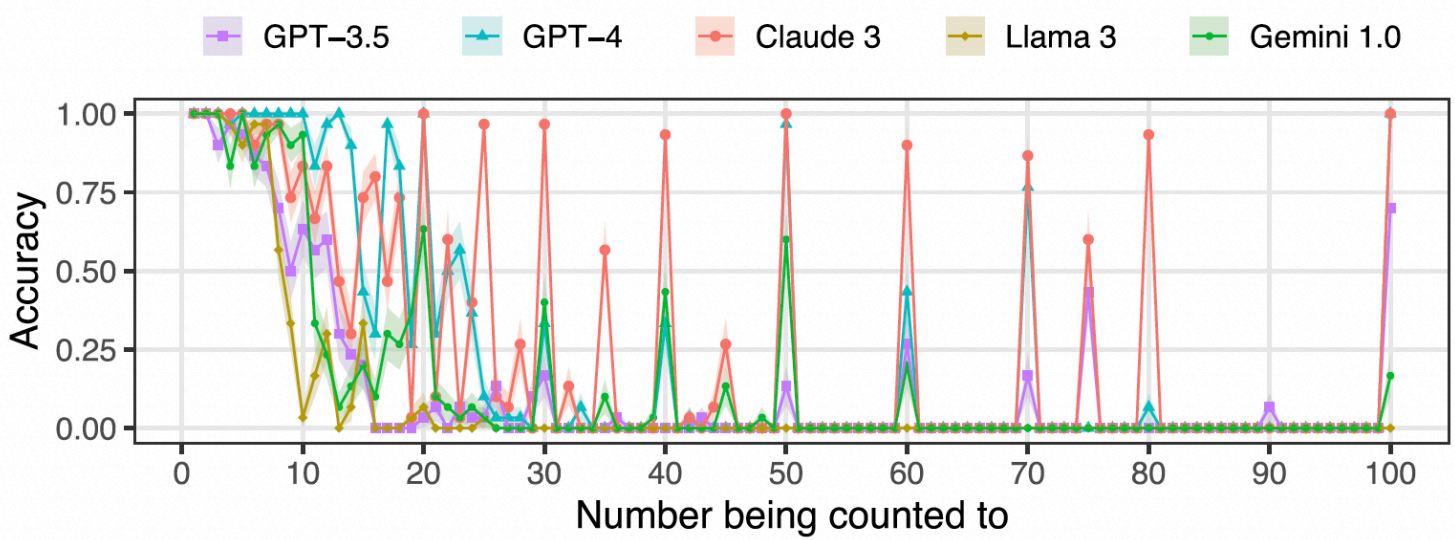


Fig. 5. Model performance on counting a list of words, as a function of the length of the list. The intervals around the lines show one SE.

Understanding twice

- From the sentence processing literature, there is evidence that sentences are processed twice (Bever and Townsend, 2001).
- There is an initial statistical phase, where processing seems to be a direct function of the frequency of words and word sequences.
- There is then a second phase where traditional syntactic and semantic rules/constraints come into play.
- For example:

Who do you think Mary said that Bill saw Fred?

Neuro-symbolic computation

Why not both?

- The general idea behind neuro-symbolic computation is to combine symbolic and neural approaches (Valiant, 2008; Marcus, 2020).
- There have been various related ideas in linguistics, e.g. Prince and Smolensky (1993), Townsend and Bever (2001), Smolensky and Goldrick (2016).
- Optimality Theory was, in fact, a very restricted integration of neural/connectionist approaches and symbolic approaches. We can take OT with gradient constraints as moving OT even more toward a neural system, e.g. Legendre et al. (1990), Smolensky (2006), Hayes and Wilson (2008), etc.
- I'll take a different approach here (Manhaeve et al., 2018; Winters et al., 2021; Huang et al., 2021).

Combining symbolic and neural approaches

- One can apply rules/constraints at the output of a neural net.
- The rules affect how the net learns.
- If the rules are correct, neural nets trained like this may learn and/or perform better.
- Incorrect rules can have the opposite effect.
- The associated rules are directly interpretable (though the rest of the net is not).
- The net and rules can be applied separately so we can see their different effects.

We can have our cake and eat it.

Neural nets and phonology

- Use data from English, Persian, and Welsh.
- Build a neural net that predicts the next segment from previous segments.
- Add rules in a specific format that reflect phonological generalizations and that can operate "cooperatively" with a neural net. For example:
  - a rule against geminate consonants, an incorrect rule for all three languages.
  - a rule against word-initial geminates, a correct rule for all three languages.
  - a rule against any specific segment, e.g. [k], which is incorrect for all three languages.

Testing

We test our system in three ways:

- We examine *loss* over training, a measure of how well the system is learning.
- We generate random sequences, a display of what the system is learning.
- We calculate the predicted probability of items not seen in training, a measure of whether the system is learning generalizations or memorizing training items (overfitting).

Neural architecture

We use a simple older architecture:

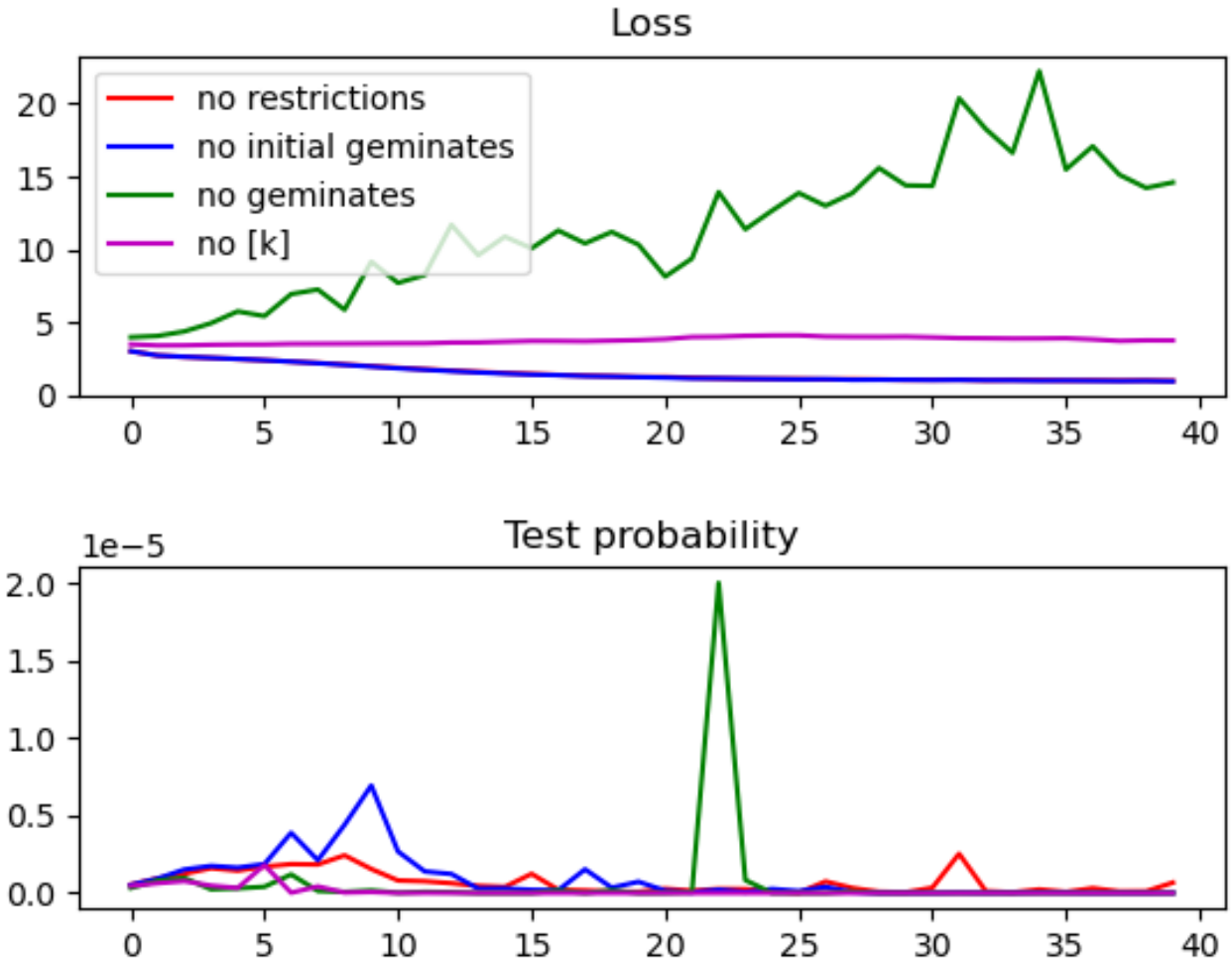
- LSTM nodes (long short-term memory). These generate outputs at each step that depend on all previous input-outputs.
- Three layers.
- 128 nodes on each layer.
- Approximately 300,000 trainable parameters, depending on number of distinct characters
- 2000 training items, 100 test items
- 40 epochs

(python program does this: `neuro2.py`)

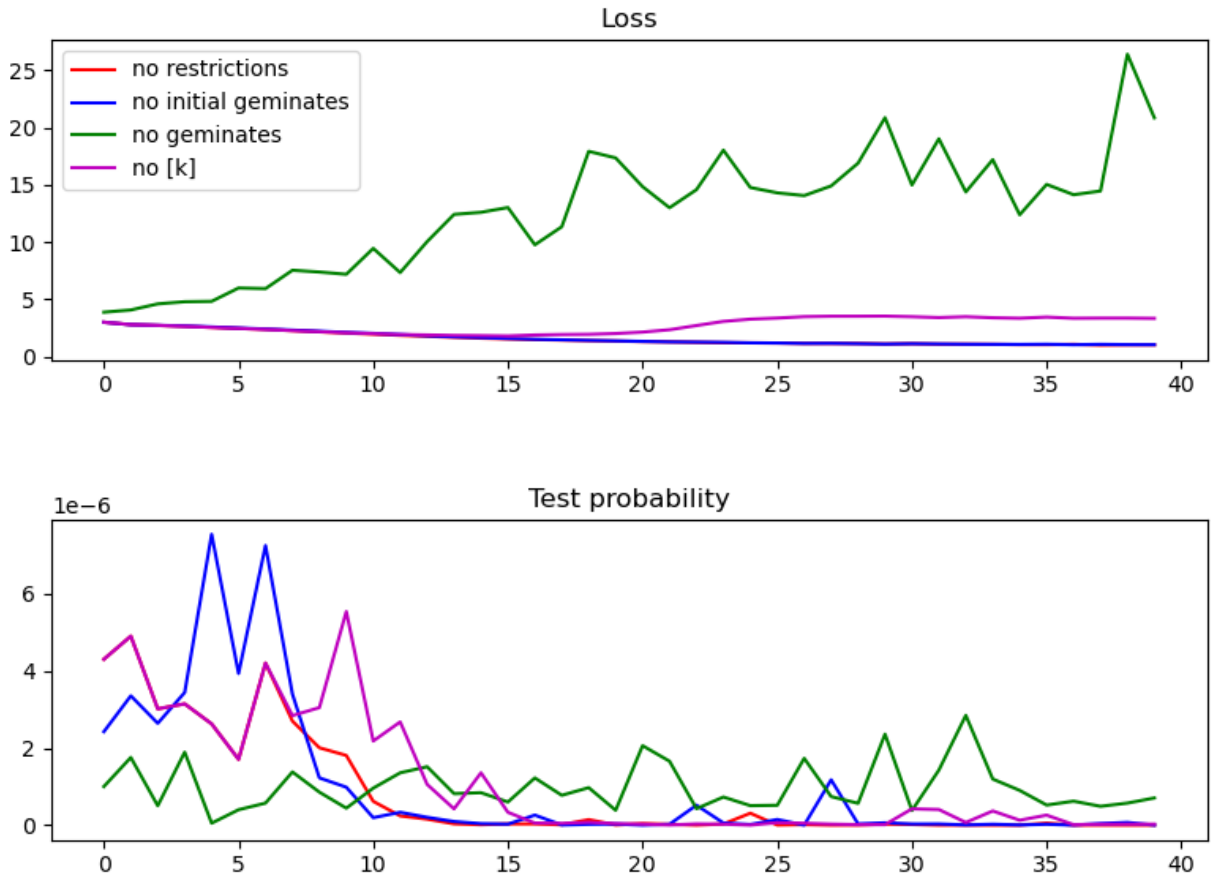
Random strings from epoch 40 for English

- no restrictions:
  ægəɹɪd kɜˠsəl dʌntəˠbɑðə dɪsmaʊndiː ɪnɹoʊdz ɪmbæksfəˠ meɪv kɪdmɛ
- no initial geminates:
  k əʊθiːəni ɑntɹæntɪ̯ts tɹɛtəʊ miːt kɪəɹɪk tɹækɹɒndi hɪɹɛdɪk sme
- no geminates:
  əb tmmlɪə ənɪd ɪtɪbf ɪɹɪəj məd deɪnɪ ɪvtən bænɪf ɪntɪɜːɹk seɪ
- no [k]:
  eːlə maɪliəs ʊdssɪsɪfaɪ dæsʧɪn pæsɪnmoʊvəˠ mɛfiəl fɪmɪɹiəl sɛn

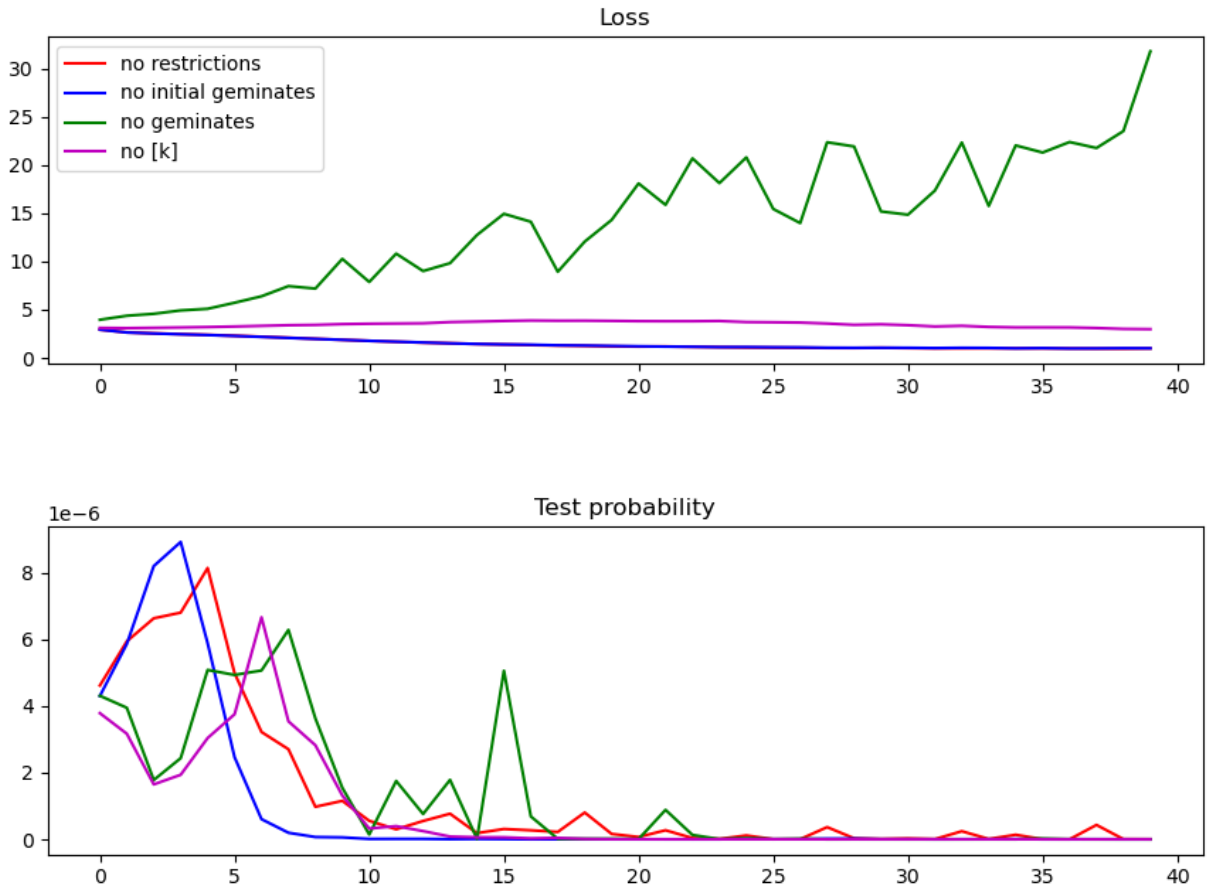
English results



Persian results



Welsh results



Neuro-symbolic loss patterns

- Loss falls steadily except in the no-geminates case.
- Loss falls less in the no-[k] case.

Neuro-symbolic probability patterns

- Probabilities bounce around a fair amount.
- Probabilities peak at around 6-8 epochs. This suggests that there is subsequent overfitting.

How the rules work

- The neural part of the system outputs a column of values for each segment in the output.
- Each value represents the likelihood that the segment in the corresponding position is identified as some particular segment.
- Imagine we have an output with three segments, $s_1 s_2 s_3$, and that the inventory has only 5 segments: [p, t, k, a, u]. We might then have an output like this:

	s_1	s_2	s_3
p	2.1	-1.9	1.7
t	-5.3	-3.2	1.5
k	1.1	1.2	6.2
a	-0.2	4.3	-2.2
u	-3.1	2.2	-1.1

How the winner is chosen

For each column, the segment with the highest value is the output segment, marked in red here. Here we would then have [pak].

	s_1	s_2	s_3
p	2.1	-1.9	1.7
t	-5.3	-3.2	1.5
k	1.1	1.2	6.2
a	-0.2	4.3	-2.2
u	-3.1	2.2	-1.1

Applying a rule

- The key innovation here is the form of the rules/constraints.
- Imagine we have a rule prohibiting word-final [k]. We go through the output and identify where a violating element occurs and reduce its value.
- In our tests, we subtract the absolute value of a cell plus the absolute value of the minimum for that column ($6.2 - \text{abs}(6.2 + \text{abs}(-2.2)) = -2.2$).

	s_1	s_2	s_3	
p	2.1	-1.9	1.7	
t	-5.3	-3.2	1.5	
k	1.1	1.2	**6.2**	$\rightarrow -2.2$
a	-0.2	4.3	-2.2	
u	-3.1	2.2	-1.1	

- We would now have [pap].

Theory of rules (so far)

- This manipulation applies to the output of the neural net *before* loss is calculated in training.
- This means that training is influenced by any rules present.
- This also means that we can separate the application of the net from the application of the rules. As in the Bever and Townsend proposal, the net comes before the rules.
- It's an open question what kinds of rules we want to allow in terms of structural properties.
- It's also an open question what kinds of quantitative adjustments we should make when a configuration is identified.

Preliminary rule semantics

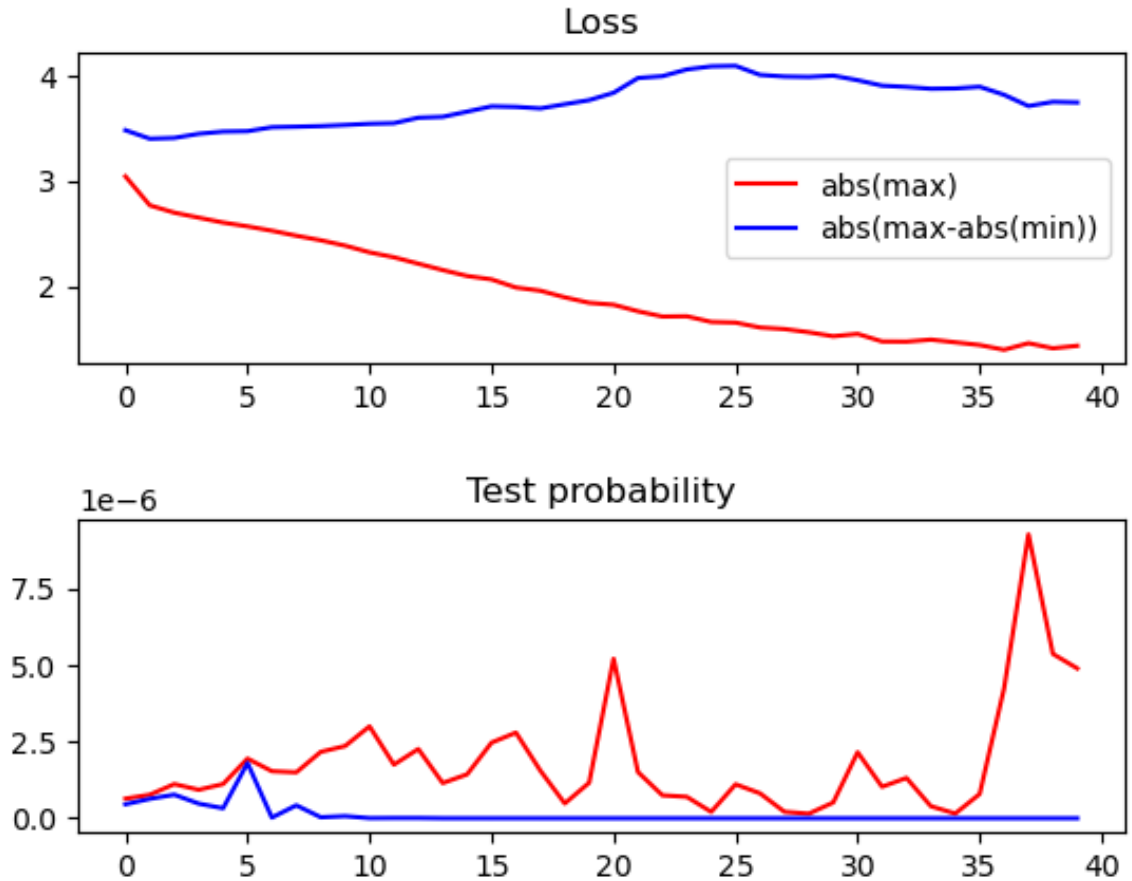
- We can specify a sequence of positions/columns.
- We can specify what the current segmental assignment of any column is.
- We can specify identity across multiple columns.
- We specify which column changes.

Rule	Positions	Identity	Changes
no k	1	1 = k	1
no geminates	1 2	1 = 2	2
no initial geminate	1 2 3	1 = #, 2 = 3	2

This is surely insufficient.

Size of adjustment

If the adjustment is too small, an “incorrect” rule can be overcome.



(English, no [k])

When to add rules

- Here, we add rules at the start of training, so the net reacts and learns differently with rules present.
- We could add rules later, only *after* the net is trained.
- If rules are added after training, we can have additional subsequent training or not.

Something you can play with

Code to try out rules you write on any of the wikipron languages (or a language you put in that format):

- `neuro3.py`: the program, tweak this to specify where your training data are, number of epochs, etc.
- `rules.txt`: where you put your rules
- `examplerules.txt`: some examples of rules in the appropriate format
- `parserules.py`: used by `neuro3.py` to parse the `rules.txt` file

(rule format is restricted)

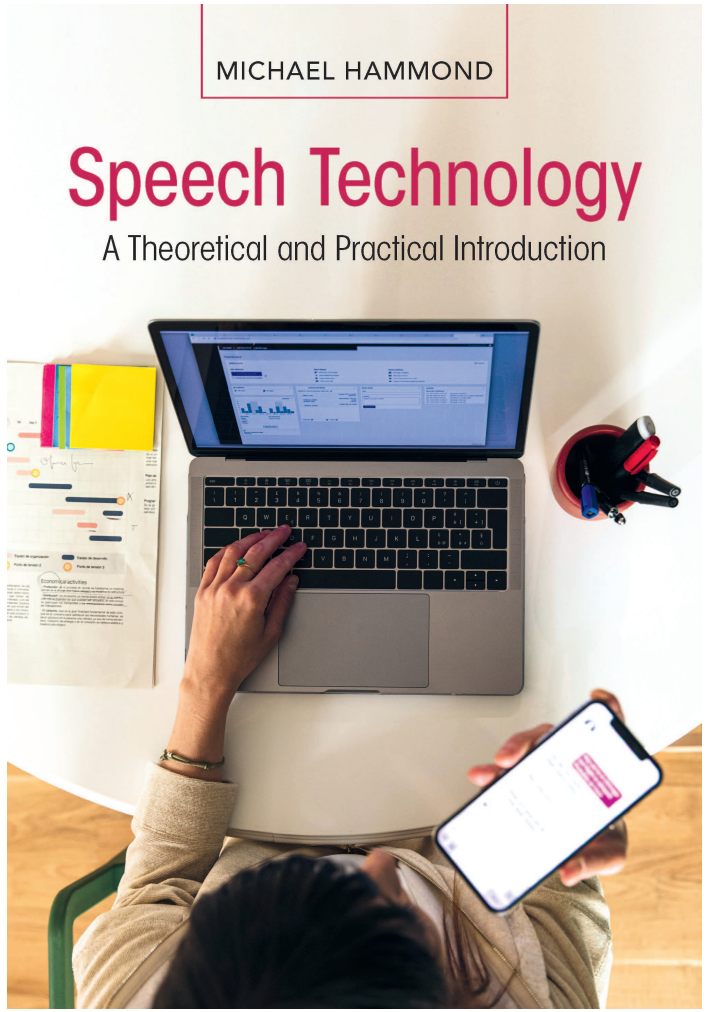
Conclusions

What have we done?

- We've reviewed the general goals of phonology.
- We've outlined a simple approach to integrating symbolic analyses with neural approaches in modern phonology: neuro-symbolic phonology.
- This approach is compatible with other neuro-symbolic approaches.
- This approach results in a system where neural outputs and rule-based outputs can be separated.
- There is a lot more to do here, but we hope we've outlined a fruitful general approach.

(Slides and code available at `https://hammondm.github.io`)

shameless promotion



(CUP, 2026)

Selected references I

Ambridge, B. and Blything, L. (2024). Large language models are better than theoretical linguists at theoretical linguistics. *Theoretical Linguistics*, 50:33–48.

Bever, T. G. and Townsend, D. J. (2001). Some sentences on our consciousness of sentences. *Language, brain, and cognitive development: Essays in honor of Jacques Mehler*, pages 143–155.

Chomsky, N. and Halle, M. (1968). *The Sound Pattern of English*. Harper & Row, New York.

Hayes, B. and Wilson, C. (2008). A maximum entropy model of phonotactics and phonotactic learning. *Linguistic Inquiry*, 39:379–440.

Huang, J., Li, Z., Chen, B., Samel, K., Naik, M., Song, L., and Si, X. (2021). Scallop: From probabilistic deductive databases to scalable differentiable reasoning. *Advances in Neural Information Processing Systems*, 34:25134–25145.

Karttunen, L. (1983). KIMMO: a general morphological processor. *Texas Linguistic Forum*, 22:163–186.

Labov, W. (1969). Contraction, deletion, and inherent variability of the English copula. *Language*, 45:715–762.

Legendre, G., Miyata, Y., and Smolensky, P. (1990). Can connectionism contribute to syntax? harmonic grammar, with an application. *CLS*, 26:237–252.

Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., and De Raedt, L. (2018). Deepproblog: Neural probabilistic logic programming. *Advances in neural information processing systems*, 31.

Marcus, G. (2020). The next decade in AI: four steps towards robust artificial intelligence. *arXiv preprint arXiv:2002.06177*.

McCarthy, J. and Prince, A. (1993). Prosodic morphology: Constraint interaction and satisfaction. University of Massachusetts, Amherst.

McCoy, R. T., Yao, S., Friedman, D., Hardy, M. D., and Griffiths, T. L. (2024). Embers of autoregression show how large language models are shaped by the problem they are trained to solve. *Proceedings of the National Academy of Sciences*, 121.

Moran, S. and McCloy, D., editors (2019). *PHOIBLE 2.0*. Max Planck Institute for the Science of Human History, Jena.

Pater, J. (2019). Generative linguistics and neural networks at 60: foundation, friction, and fusion. *Language*, 95:41–74.

Pérez, J., Marinković, J., and Barceló, P. (2019). On the turing completeness of modern neural network architectures. *arXiv preprint arXiv:1901.03429*.

Prince, A. and Smolensky, P. (1993). Optimality Theory. University of Massachusetts, Amherst and University of Colorado at Boulder.

Rahbar, E. R. (2012). *Aspects of Persian phonology and morpho-phonology*. PhD thesis, University of Toronto.

Sankoff, D. and Labov, W. (1979). On the uses of variable rules. *Language in Society*, 8:189–222.

Smolensky, P. (2006). Harmony in linguistic cognition. *Cognitive Science*, 30:779–801.

Smolensky, P. and Goldrick, M. (2016). Gradient symbolic representations in grammar: The case of French liaison. ROA #1286.

Selected references II

Toosarvandani, M. D. (2004). Vowel length in modern Farsi. *Journal of the Royal Asiatic Society*, 14:241–251.

Townsend, D. J. and Bever, T. G. (2001). *Sentence comprehension: The integration of habits and rules*. MIT Press.

Valiant, L. G. (2008). Knowledge infusion: In pursuit of robustness in artificial intelligence. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (2008)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik.

Winters, T., Marra, G., Manhaeve, R., and De Raedt, L. (2021). Deepstochlog: Neural stochastic logic programming. *arXiv preprint arXiv:2106.12574*.