



University of
Massachusetts
Amherst

Vanilla Sequence-to-Sequence Neural Nets cannot Model Reduplication

Item Type	Working Paper
Authors	Prickett, Brandon
DOI	10.7275/R5N877Z9
Download date	2026-05-12 06:02:43
Link to Item	https://hdl.handle.net/20.500.14394/30636

Vanilla Sequence-to-Sequence Neural Nets cannot Model Reduplication¹

Brandon Prickett

University of Massachusetts Amherst

December, 2017

1. Introduction

Connectionist models of language have been critiqued for lacking “*variables* that stand for sets of individuals, regardless of their featural decomposition and over which quantified generalizations can be made” (Pinker and Prince 1988:176; emphasis theirs). A number of processes observed in natural language seem to require the use of variables (Marcus 2001). One of these is reduplication, a common morphological process that involves copying all or part of a word’s stem, as in (1):

(1) *Reduplication in Karao (from Štekaurer et al. 2012):*

man ba kal	→	man ba kal
‘fight each other (two people)’		‘fight each other (more than two people)’

Typically, symbolic accounts of reduplication use templates that cause copying to occur (McCarthy 1981) or compare the stem and reduplicant directly (McCarthy and Prince 1995). Crucially, these accounts are able to learn reduplication in a generalizable way. For example, if trained on the mappings [ba]→[baba] and [ga]→[gaga], symbolic theories would predict the production of [dada] from the input [da].

Because most connectionist models can’t make this kind of stem-reduplicant comparison, it has been suggested that learning a generalizable pattern of this type in a connectionist framework is impossible (Tupper and Shahriari 2016). While this finding has been supported by a number of simulations using simple recurrent neural networks and multilayer feed forward networks (for example, see Marcus et al. 1999 and Tupper and Shahriari 2016, respectively), it is not obvious that it generalizes to more state-of-the-art neural network architectures. Sequence-to-Sequence neural networks with LSTM (henceforth *Seq2Seq*; Sutskever et al. 2014) are a recent innovation that have been shown to correlate well with human behavior on a number of morphological tasks (Kirov 2017). This paper reports the results of testing one such model (Rahman 2016), to see if it can learn a reduplication rule that generalizes to novel stems.

2. Methods

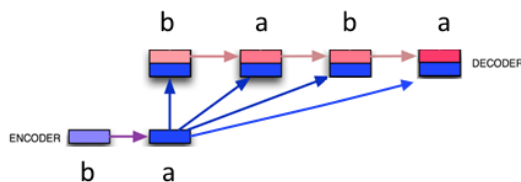
Seq2Seq models were originally created for machine translation tasks (Sutskever et al. 2014). They are made up of two separate recurrent neural networks: the encoder and the decoder. The encoder steps through an input string one element at a time, and transforms the sequence into an output that it gives to the decoder. The decoder then unpacks the encoder’s output one element at a time, creating the model’s output string. This is illustrated in (2).

For the simulations reported here, I used the Seq2Seq Python package (Rahman 2016), which automatically creates a Seq2Seq neural network with LSTM (“Long short-term memory”; Hochreiter and Schmidhuber 1997). LSTM is a mechanism used in many recurrent neural networks to help the model more easily keep track of long-distance dependencies. The presence

¹ Thanks to Joe Pater, Gaja Jarosz, and the members of UMass’s Sound Workshop for helpful discussion on this topic. If you have questions or comments about this manuscript, you can contact me at blprickett@outlook.com.

of LSTM is another important difference between this model and those that have been used to model reduplication in the past.²

(2) *An illustration of a Seq2Seq Model (adapted from Nag 2016):*



The Seq2Seq package also allows users to specify the number of hidden layers, the number of nodes in each layer, and the input/output lengths. The results presented in §3 are from a model with three hidden layers each in the encoder and decoder, ten nodes in each hidden layer, and four nodes in the input and output layers. The length for inputs was set to 2 segments and the length of outputs was set to 4. I experimented with a variety of settings for these parameters and none seemed to have an effect on the model’s ability to properly learn reduplication.

The toy languages used in these simulations had eight different consonants and a single vowel. To represent these segments, I used four standard phonological features: [±voice], [±labial], [±continuant], and [±syllabic]. The first three features allowed the model to differentiate between the consonants, while the last feature was used to differentiate between consonants and the vowel. The table in (3) shows these feature-segment relationships:³

(3) *Features for the toy language*

Segment	[voice]	[labial]	[continuant]	[syllabic]
[b]	+	+	-	-
[p]	-	+	-	-
[d]	+	-	-	-
[t]	-	-	-	-
[v]	+	+	+	-
[f]	-	+	+	-
[z]	+	-	+	-
[s]	-	-	+	-
[V]	-	-	-	+

The model was trained to map CV stems to reduplicated forms of the shape CVCV, where both C’s in the reduplicated form were identical to the original C in the stem. There were two types of simulation: in the first type, the model was exposed to all eight of the possible CV syllables in training. In the second type, the model was exposed to seven of the possible syllables, with [zV] being excluded from training. Each of the simulations reported here had 5000 epochs, with every exposure syllable occurring ten times per epoch, and initial weight values being randomized.

² For more on LSTM’s effect on neural networks’ linguistic representational power, see Linzen et al. (2016).

³ The model was given positive feature values for “+” and negative values for “-” in both its input and output vectors in training. For testing, any negative outputs were considered to be “-” and any positive outputs were “+.”

Multiple simulations were run with various parameter settings to ensure that the results presented here were an accurate representation of the model's ability.

3. Results

When the learner was exposed to a reduplication mapping for all eight of the possible CV stems, it correctly learned reduplication. The results for one of these simulations are shown in (4), with segmental representations to the left of the colons and the actual feature vectors the model received/returned to the right of the colons⁴.

<i>(4) Inputs at Testing</i>	→	<i>Outputs at Testing</i>
b: [10, 10, -10, -10] V: [-10, -10, -10, 10]	→	b: [1, 1, -1, -1,] V: [-1, -1, -1, 1,] b: [1, 1, -1, -1,] V: [-1, -1, -1, 1,]
p: [-10, 10, -10, -10] V: [-10, -10, -10, 10]	→	p: [-1, 1, -1, -1,] V: [-1, -1, -1, 1,] p: [-1, 1, -1, -1,] V: [-1, -1, -1, 1,]
d: [10, -10, -10, -10] V: [-10, -10, -10, 10]	→	d: [1, -1, -1, -1,] V: [-1, -1, -1, 1,] d: [1, -1, -1, -1,] V: [-1, -1, -1, 1,]
t: [-10, -10, -10, -10] V: [-10, -10, -10, 10]	→	t: [-1, -1, -1, -1,] V: [-1, -1, -1, 1,] t: [-1, -1, -1, -1,] V: [-1, -1, -1, 1,]
v: [10, 10, 10, -10] V: [-10, -10, -10, 10]	→	v: [1, 1, 1, -1,] V: [-1, -1, -1, 1,] v: [1, 1, 1, -1,] V: [-1, -1, -1, 1,]
f: [-10, 10, 10, -10] V: [-10, -10, -10, 10]	→	f: [-1, 1, 1, -1,] V: [-1, -1, -1, 1,] f: [-1, 1, 1, -1,] V: [-1, -1, -1, 1,]
z: [10, -10, 10, -10] V: [-10, -10, -10, 10]	→	z: [1, -1, 1, -1,] V: [-1, -1, -1, 1,] z: [1, -1, 1, -1,] V: [-1, -1, -1, 1,]

⁴ The use of 10/-10 was for convenience. It caused the model (whose output activation ranged from -1 to 1) to produce more extreme values in its output at testing, allowing for a less ambiguous interpretation of what the model had learned. I ran these same simulations with 1/-1 values in training and found similar results.

$$\begin{array}{lcl}
s: & [-10, -10, 10, -10] & \\
V: & [-10, -10, -10, 10] & \rightarrow \\
& & s: [-1, -1, 1, -1,] \\
& & V: [-1, -1, -1, 1,] \\
& & s: [-1, -1, 1, -1,] \\
& & V: [-1, -1, -1, 1,]
\end{array}$$

When the model was not exposed to all eight of the possible syllables in training, it only learned how to apply reduplication correctly to those syllables that it was trained on. The results for the input [zV] at testing when it was withheld from training are given in (5). The results for every other CV syllable in this simulation were identical to the results given in (4).

$$\begin{array}{lcl}
(5) \text{ Inputs at Testing} & & \text{Outputs at Testing} \\
z: & [10, 10, -10, -10] & b: [1, 1, -1, -1] \\
V: & [-10, -10, -10, 10] & V: [-1, -1, -1, 1] \\
& & b: [1, 1, -.99999994, -1] \\
& & V: [-1, -1, -1, 1]
\end{array}$$

Instead of mapping [zV] to [zVzV] as the reduplication rule would require, the model that had [zV] withheld from its training data mapped [zV] to [bVbV]. Which segment [z] becomes in the incorrectly reduplicated form varies from simulation to simulation, but the fact that the output is not [zVzV] stays the same.

4. Discussion

It is possible for neural nets to model reduplication-like processes, but they typically need some added piece of architecture that was designed for that purpose (see, for example, Gu et al. 2016 and Alhama 2017). This paper describes the results of a series of simulations that were designed to test whether a general-purpose Seq2Seq neural network could learn a reduplication pattern in a generalizable way. The results suggest that without a special mechanism for the task, even state-of-the-art neural networks struggle with learning this type of mapping.

References

- Alhama, Raquel G. (2017). *Computational modelling of artificial language learning*. Dissertation, Institute for Logic, Language and Computation (ILLC) at the University of Amsterdam.
- Fariz, Rahman (2016). *Seq2seq*. URL: <https://github.com/farizrahman4u/seq2seq>
- Gu, J., Lu, Z., Li, H., & Li, V. O. (2016). *Incorporating copying mechanism in sequence-to-sequence learning*. arXiv preprint arXiv:1603.06393.
- Hochreiter, Sepp and Jürgen Schmidhuber (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.
- Kirov, Christo (2017). Recurrent Neural Networks as a Strong Domain-General Baseline for Morpho-Phonological Learning. Poster presented at the *2017 Meeting of the Linguistic Society of America*.
- Linzen, Tal, Emmanuel Dupoux, and Yoav Goldberg (2016). Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics* 4: 521–535.
- Marcus, Gary F. (2003). *The algebraic mind: Integrating connectionism and cognitive science*. MIT press.

- Marcus, G. F., Sugumaran Vijayan, S. Bandi Rao, and Peter M. Vishton (1999). Rule learning by seven-month-old infants. *Science* 283.5398: 77-80.
- McCarthy, J. J. (1981). A prosodic theory of nonconcatenative morphology. *Linguistic inquiry*, 12(3), 373-418.
- McCarthy, John, and Alan Prince (1995). Faithfulness and reduplicative identity. In *University of Massachusetts occasional papers 18: Papers in Optimality Theory*: 249-384. GLSA, University of Massachusetts, Amherst.
- Nag, Dev (2016). *seq2seq: the clown car of deep learning*.
URL: <https://medium.com/@devnag/seq2seq-the-clown-car-of-deep-learning-f88e1204dac3>
- Pinker, Steven, and Alan Prince (1988). "On language and connectionism: Analysis of a parallel distributed processing model of language acquisition." *Cognition* 28.1: 73-193.
- Štekauer, Pavol, Salvador Valera, and Livia Kórtvélyessy (2012). *Word-formation in the world's languages: a typological survey*. Cambridge University Press.
- Tupper, Paul, and Bobak Shahriari (2016). *Which Learning Algorithms Can Generalize Identity-Based Rules to Novel Inputs?* arXiv preprint arXiv:1605.04002.