Fall 2018

Seq2Seq Models with Dropout can Learn Generalizable Reduplication

Brandon Prickett
Aaron Traylor, *Brown University*
Joe Pater

Seq2Seq Models with Dropout can Learn Generalizable Reduplication

Brandon Prickett, Aaron Traylor, and **Joe Pater**
Linguistics Department
University of Massachusetts Amherst
bprickett@umass.edu, aaron_traylor@brown.edu,
pater@linguist.umass.edu

Abstract

Natural language reduplication can pose a challenge to neural models of language, and has been argued to require variables (Marcus et al., 1999). Sequence-to-sequence neural networks have been shown to perform well at a number of other morphological tasks (Cotterell et al., 2016), and produce results that highly correlate with human behavior (Kirov, 2017; Kirov & Cotterell, 2018) but do not include any explicit variables in their architecture. We find that they can learn a reduplicative pattern that generalizes to novel segments if they are trained with dropout (Srivastava et al., 2014). We argue that this matches the scope of generalization observed in human reduplication.

1 Introduction

Reduplication is a common morphological process in which all or part of a word is copied and added to one side of the word's stem. An example of reduplication occurring in the language Karao is given in (1):

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

man***ba***kal	→	man***baba***kal
'fight each other'		'fight each other'
(2 people)		(>2 people)

In the example above, the stem *ba* is reduplicated to create the affixed form *baba*. Berent (2013) discusses four different possibilities for how speakers could represent reduplication in their minds: (i) memorization of which reduplicated forms go with which stems, (ii) learning a function that copies all segments that undergo reduplication, (iii) learning a function that copies all feature values that undergo reduplication, or (iv) learning a function that uses algebraic symbols to copy the appropriate material, regardless of its segmental or featural content. She concludes that reduplication and similar processes in language involve the fourth possibility, which she labels an *identity function*. An identity function for reduplication is illustrated in (2), with *α* acting as a variable that represents the reduplicated sequence.

(2) Reduplication as an algebraic rule

$$\alpha \rightarrow \alpha\alpha$$

Marcus et al. (1999) came to a similar conclusion regarding reduplication and identity functions, after showing that infants could learn a reduplication-like pattern and generalize that pattern to novel segments. They used this as evidence against connectionist models of grammar, which do not typically include explicit variables[1] (see, for example, Elman, 1990; Rumelhart & McClelland, 1986). Both feed-forward and simple recurrent neural networks fail at learning generalizable identity functions (Berent, 2013; Marcus, 2001; Marcus et al., 1999; Tupper & Shahriari, 2016).

In this paper, we revisit these arguments against variable-free connectionist models in light of recent developments in neural network architecture and training techniques. Specifically, we test Sequence-to-Sequence models (Sutskever et al., 2014) with LTSM (Long Short-Term

[1] We use the term *explicit variable* to refer to the algebraic symbols that are often absent from connectionist theories of cognition. However, a number of connectionist models do incorporate explicit variables, such as the models in Marcus (2001), Smolensky and Legendre (2006), and Moreton (2012). See Pater (2018:§4) for further discussion of different hybrids of connectionist and symbolic approaches.

Memory; Hochreiter & Schmidhuber, 1997) and dropout (Srivastava et al., 2014). We find that the scope of generalization for the models is increased from copying segments to copying feature values when dropout is added. Additionally, we argue that variable-free feature copying is sufficient to model human generalization, contrary to Berent's (2013) claim that an algebraic identity function is necessary.

2 Background

The debate between connectionist and symbolic theories of grammar has largely been focused on the domain of morphology (for a review, see Pater, 2018). Reduplication was no exception, with standard connectionist models failing to learn the pattern (Gasser, 1993). Standard models also failed to generalize a reduplicative pattern in a way that mimicked human behavior (Marcus et al., 1999). Marcus (2001) argued that this was evidence of the need for variables in models of cognition. While supporters of connectionism pointed out issues with some of Marcus et al.'s (1999) conclusions (e.g. Seidenberg & Elman, 1999), they failed to show that a connectionist network with no variables could learn reduplication without being previously trained on a similar identity function (see Endress, Dehaene-Lambertz, & Mehler, 2007 for an overview of these studies).

Research in phonotactics has also supported the need for variables in models of language. Berent (2013) showed that Hebrew speakers generalized a phonotactic identity-based restriction in a way that she argued required variables. She presented various experimental results demonstrating that speakers would generalize the restriction to novel words, novel segments, and what she claimed to be novel feature values (for more on our interpretation of these findings, see §5.2). This ran contrary to the predictions of phonotactic learning models that did not include variables (Berent et al., 2012).

However, the models tested by Marcus et al. (1999) and Berent et al. (2012) were relatively simple compared to many modern neural network architectures. The modern model that we will examine is the Seq2Seq neural network (Sutskever et al., 2014), originally designed for machine translation. These models have been shown to perform well at learning a variety of morphological tasks (Cotterell et al., 2016), and produce results that highly correlate with human behavior (Kirov, 2017; Kirov & Cotterell, 2018).

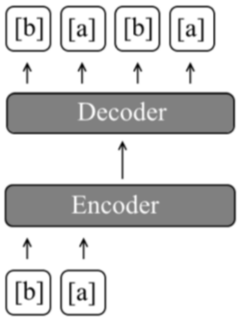


Figure 1: Illustration of Seq2Seq architecture modeling reduplication of the stem [ba].

Since these models include a number of recently-invented mechanisms, such as an encoder-decoder structure (Sutskever et al., 2014), Long Short-Term Memory layers instead of simple, recurrent ones (Hochreiter & Schmidhuber, 1997), and the possibility of dropout during training (Srivastava et al., 2014), it's unclear whether they will be limited in the same ways as their predecessors.

3 The Model

In this section, we will give a brief introduction to each of the mechanisms in our model that we consider to be relevant to the simulations presented in §4. For the documentation on the Python packages used to implement the model, see Chollet et al. (2015) and Rahman (2016). We chose to focus on Seq2Seq models because of their recent success in a number of linguistic tasks (summarized in §3.1). We leave exploring the differences between this architecture and its alternatives (such as simple recurrent networks) to future work.

3.1 The Seq2Seq Architecture

Seq2Seq neural networks have the ability to map from one string to another, without requiring a one-on-one mapping between the strings' elements (Sutskever et al., 2014). The model achieves this by using an architecture made up of an encoder and decoder pair. Each member in the pair is its own recurrent network, with the encoder processing the input string and the decoder transforming that processed data into an output string. The ability of these models' inputs and outputs to have independent lengths is useful for morphology, which usually involves adding or copying segments in a stem. An example of this for reduplication is shown in Figure 1.

In Figure 1, the encoder passes through the entire input string (i.e. the stem [ba]) before transferring information to the decoder. The decoder then unpacks this information, and gives a reduplicated form (i.e. [baba]) as output. In all of the simulations discussed in this paper, the encoder is *bidirectional*, meaning that it passes through the input string starting from both the left and right edges.

3.2 Long Short-Term Memory (LSTM)

LSTM (Hochreiter & Schmidhuber, 1997) is a kind of recurrent neural network layer which allows the model to store certain information in memory more easily than a simple recurrent layer could. While this architectural innovation was originally designed to address the problem of vanishing gradients (Bengio et al., 1994), it has been demonstrated that LSTM can also provide models with added representational power (Levy et al., 2018).

The way the model performs both of these tasks is by using *cell states*, bundles of interacting layers that can learn which features are important for the model to keep track of in a long-term way. During training, the network is not only keeping track of which information will allow it to predict the output from the input, but also which information at a given time step (i.e. at a given segment in the simulations presented here) will help it to predict the output at future time steps.

Vanishing gradients are not much of a concern in morphological learning, since input and output strings are relatively short in this domain of language. However, the effects of LSTM's added representational power on learning of reduplication have not yet been explored.

3.3 Dropout

Dropout is a method used in neural network training that helps models generalize correctly to items outside of their training data (Srivastava et al., 2014). It achieves this by having some units in the network "drop out" in each forward pass. This prevents the network from finding solutions that are too dependent on a small number of units. Practically speaking, this is implemented by setting a probability with which each unit will drop out (a hyper parameter set by the analyst) and then multiplying every unit's output by either a 0 or 1, depending on whether it has been randomly chosen to be dropped out or not. Which units are dropped

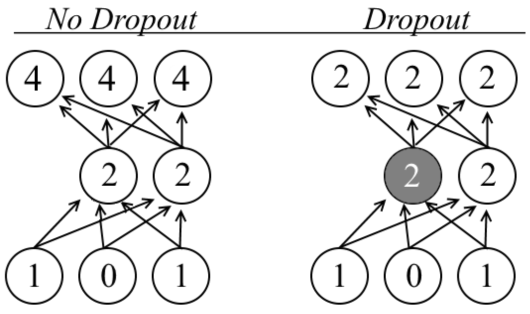


Figure 2: A simple, feed-forward network, with and without dropout. Each circle is a unit and each arrow is a connection. Dropped out units are in grey. Each unit's output (before dropout) is denoted by the number inside of it. All connections have a weight of 1 and all activation functions are f(x)=x.

out is resampled each forward pass, causing the network's solution to be more general than it might have been otherwise. This is illustrated for a single forward pass in a simple, feed-forward network on the right side of Figure 2. In this illustration, dropout causes the output units to have an activation of 2, instead of 4, because a unit in the middle layer is being dropped out and cannot contribute to the activations in the layer above it. For the simulations presented here that use dropout, it was applied with equal probability to all layers of the network.

4 Experiments

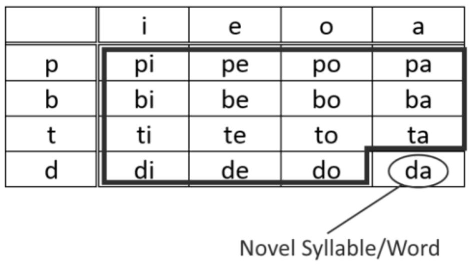
To test whether reduplication can be modeled by a neural network without explicit variables, we ran a number of simulations in which the model was trained on a reduplication pattern in a toy language and tested on how it generalized that pattern to novel data. To test what kind of generalization the model was performing, we set up different scenarios: one in which the model was tested on a novel syllable made up of segments it had seen reduplicating in its training data (§4.1), one in which the model was tested on a syllable made with a segment that it hadn't received in training (§4.2), and one in which the model was tested on a syllable with a novel segment containing a feature value that hadn't been presented in the training data (§4.3).

In the experiments presented here, a language's segments were each represented by a unique, randomly-produced vector of 6 features (excluding the simulations in §4.3), with feature values being either -1 or 1 (corresponding to the [-] and [+] used in standard phonological models). The inventory was divided into consonants and vowels by treating the first feature as [syllabic], i.e. any of the feature vectors that began with -1 were considered a consonant and any that began with 1 were considered a vowel. If an inventory had no vowels, one of its consonants was randomly chosen and its value for the feature [syllabic] was changed to 1.

The toy language for any given simulation consisted of all the possible CV syllables that could be made with that simulation's randomly created segment inventory. Crucially, before the data was given to the model, some portion of it was withheld for testing (see the subsections below for more information on what was withheld in each testing condition). The mapping that the model was trained on treated each stem (e.g. [ba]) as input and each reduplicated form (e.g. [baba]) as output. The model's input and output lengths were fixed to 2 and 4 segments, respectively (reflecting the fact that all the toy languages only had stems that were 2 segments long).

The models were trained for 1000 epochs, with training batches that included all of the learning data (i.e. learning was done in batch). The loss function that was being minimized was mean-squared error, and the minimization algorithm was RMSprop (Tieleman & Hinton, 2012). The models had 2 layers each in the encoder and decoder, with 18 units in each of these layers. All other parameters were the default values in the deep-learning Python package, Keras (Chollet et al. 2015).

To test whether the model generalized to withheld data, a relatively strict definition of success was used in testing. The model was given a withheld stem as input, and the output it predicted was compared to the correct output (i.e. the reduplicated form of the stem it was given). If every feature value in the predicted output had the same sign (positive/negative) as its counterpart in the correct output, the model was considered to be successfully generalizing the reduplication pattern. However, if any of the feature values did not have the same sign, that model was considered to be non-generalizing. While we only report the results from 25 runs in each condition, we ran many more while investigating various hyperparameter settings and possibilities about the construction of the training data. The results presented here are representative of the general pattern of results.



	i	e	o	a
p	pi	pe	po	pa
b	bi	be	bo	ba
t	ti	te	to	ta
d	di	de	do	da

Figure 3: Illustration of generalization to a novel syllable/word in a language with only eight segments. Specific IPA labels are hypothetical. Syllables surrounded by the black box were presented in training, while the circled syllable was withheld for testing.

4.1 Generalizing to Novel Syllables

Our first set of simulations tested whether the model could generalize to novel syllables. If the model failed at this task, then it would mean that it was memorizing whole words in the training data, rather than learning an actual pattern. Figure 3 illustrates this.

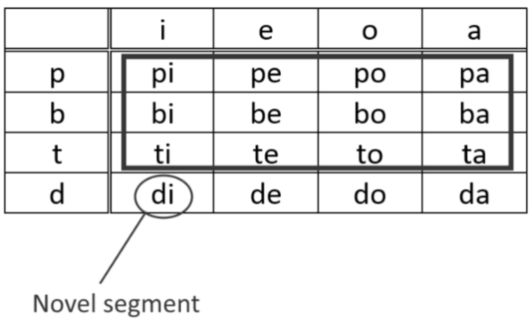
In Figure 3, [da] is the syllable that was withheld from training. This means that the model never saw the mapping from [da]$\rightarrow$[dada], but it did see the segments that make up [da]. For example, the training data [di] and [ba] would have demonstrated the behavior of [d] and [a] to the model, respectively.

For this condition, toy languages always contained 40 segments in their inventory, and the probability of a unit being dropped out was 0%.

The model successfully reduplicated syllables from training in all runs for this condition. Additionally, it generalized to novel syllables in 22 of the 25 simulations (88%). These results are summarized in Figure 6. This shows that a standard Seq2Seq model, with LSTM but no dropout, can perform generalization to novel syllables, and does so a majority of the time.

4.2 Generalizing to Novel Segments

The next scope of generalization described by Berent (2013) is generalization to novel segments. To test whether our model could achieve this, we created languages with inventories of 40 segments (as described above), but randomly chose a single consonant in each run to be withheld for testing. This is illustrated for a simplified example in Figure 4.



	i	e	o	a
p	pi	pe	po	pa
b	bi	be	bo	ba
t	ti	te	to	ta
d	di	de	do	da

Novel segment

Figure 4: Illustration of generalization to a novel segments in a language with only eight sounds. Specific IPA labels are hypothetical. Syllables surrounded by the black box were presented in training, while the circled syllable was withheld for testing.

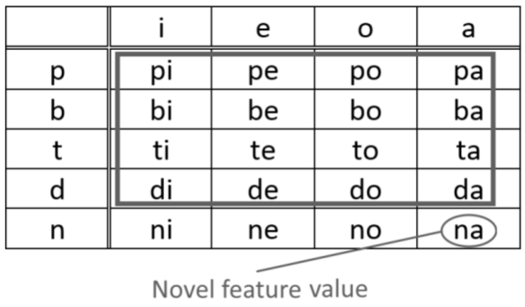
In Figure 4, the consonant [d] is never shown to the model in training. This means that the model has no experience with reduplicating this vector of feature values before testing. The model would have been exposed to each of the feature values making up this segment, though. For example, [t] and [b] would have exposed the model to the place and voicing features of [d], respectively.

For the results reported in this section, toy languages always had inventories of 40 segments. Two conditions were tested in regards to dropout: one in which dropout never happened (0%) and one which it happened to the majority of units in any given forward pass (75%).

When no dropout was applied, the model was unable to reliably generalize—with only 6 of the 25 runs achieving success on novel segments (24 out of 25 for trained segments). However, when dropout was applied with a probability of 75%, the model successfully generalized to novel segments in 15 of the 25 runs (25 out of 25 for trained segments). These results are summarized in Figure 6. This demonstrates that without dropout, our model does not reliably generalize to novel syllables, but that with dropout it does.

4.3 Generalizing to Novel Feature Values

The most powerful form of generalization Berent (2013) discusses is generalization to novel feature values, which would signify the acquisition of a proper identity function. In the context of reduplication, this would involve correctly applying the process to a stem that includes feature values never seen in training. For example, if all of the consonants in training were oral, but the process generalized to nasal consonants, this would demonstrate generalization to the novel feature value [+nasal]. This is shown in Figure 5.



	i	e	o	a
p	pi	pe	po	pa
b	bi	be	bo	ba
t	ti	te	to	ta
d	di	de	do	da
n	ni	ne	no	na

Novel feature value

Figure 5: Illustration of generalization to a novel feature values in a language with only nine sounds. Specific IPA labels are hypothetical. Syllables surrounded by the black box were presented in training, while the circled syllable was withheld for testing.

In the example above, the syllable [na] represents a novel feature value. If a model only learned a function that learned to copy individual feature values from the stem into the reduplicant, it wouldn't generalize to this kind of novel feature value correctly. This generalization can only occur if the model learns to copy the reduplicant irrespective of individual features.

For the results reported here, toy languages always contained 43 segments in their inventory, and these were not produced randomly (this made it easier to ensure that a particular feature value could be withheld). A variety of other segment inventories were tested, with no changes in the model's performance. Results are presented here for simulations using 0% and 75% dropout probability, although numerous other values for this were also tested.

Regardless of whether dropout occurred, the model never generalized to novel feature values. These results are summarized in Figure 6. This shows that a standard Seq2Seq model, regardless of whether it has dropout, cannot generalize to novel feature values. We discuss in §5.2 why we do not see this limitation as a flaw in terms of modeling human language learning.

5 Discussion

5.1 Summary of Results

The results for each simulation can be viewed side-by-side in Figure 6. The findings from this series of experiments showed that even without dropout, a Seq2Seq model is not simply learning to

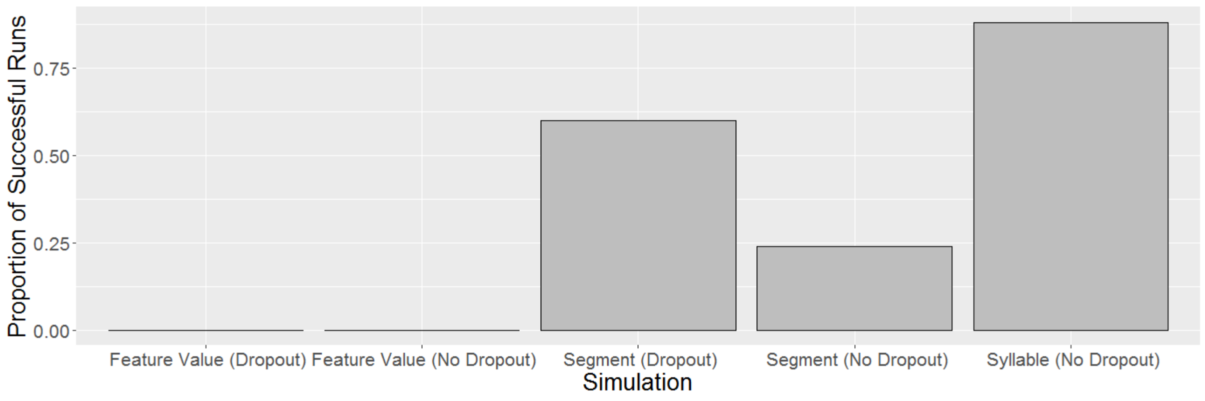


Figure 6: Full summary of results. Bars show the proportion of successful runs out of 25.

memorize <stem, reduplicant> pairs, since it was able to generalize reduplication to novel stems (i.e. novel syllables). We also showed that the model, when using dropout in training, can reliably generalize reduplication to novel segments.

5.2 Can humans generalize to novel feature values?

When discussing generalization, Berent (2013) used evidence from Hebrew speakers' phonotactic judgments to support the idea that they learn a true identity function when acquiring their native phonology. The judgments centered around a phonotactic restriction in Hebrew that prohibits the first two consonants in a stem from being identical. For example, the word *simem* 'he intoxicated' is grammatical, while the nonce word **sisem* is not. Berent (2013) showed that speakers generalized this pattern by having them rate the acceptability of various kinds of nonce forms.

The first results she discussed demonstrated Hebrew speakers generalizing to novel words. These words were made up of segments that were attested in Hebrew. Speakers in this experiment rated words with s-s-m stems (like **sisem* above) as significantly less acceptable than words with s-m-m and p-s-m stems. This demonstrated that Hebrew speakers were doing more than just memorizing the lexicon of their language (i.e. that they could extract phonotactic patterns).

The next results that Berent (2013) presented involved Hebrew speakers generalizing the pattern to novel segments. The segments of interest were /tʃ/, /dʒ/ and /w/, all of which are not present in native Hebrew stems. Even when these non-native phonemes were used, Hebrew speakers rated words whose first two consonants were identical (e.g. dʒ-dʒ-r) as worse than those that did not violate the phonotactic restriction (e.g. r-dʒ-dʒ). This demonstrated that speakers had not just memorized a list of consonants that cannot cooccur (e.g. *pp, *ss, *mm, etc.) while acquiring their phonological grammar.

Finally, Berent (2013) showed that speakers can generalize the pattern to the segment [θ], which she claimed represented generalization to a novel feature value (which she referred to as *across the board generalization*). While we agree with her conclusions regarding the first two sets of results, we find her interpretation of this final result to be less convincing. The novel feature value that she claimed was represented by [θ] was the feature value [Wide]. However, this is a fairly non-standard phonological feature. Using a more standard featural representation for [θ], such as [+anterior, +continuant, -strident] (Chomsky & Halle, 1968), would mean that [θ] does not represent a novel feature value for Hebrew, since the language contains other, native, [+anterior], [+continuant], and [-strident] sounds.

Returning to reduplication, a relevant generalization experiment is Marcus et al. (1999). In that experiment, infants generalized a reduplication-like pattern to novel words with novel segments, but not with novel feature values. All of the words in Marcus et al.'s testing phase used feature values that would have been familiar to the infants from their native language of English.

To our knowledge, no experiment has shown humans generalizing to truly novel feature values. Because of this, we cannot conclude whether our model's results from §4.3 are human-like or not.

5.3 Why did dropout allow the model to generalize to novel segments?

While we've shown that dropout increases the Seq2Seq model's scope of generalization, it still remains unclear why this form of regularization succeeds for this task. One hypothesis is that dropout causes certain training data to be indistiguishable from crucial testing data. For example, if training includes the stems [pa] and [da], but [ta] is withheld, a model without dropout would not generalize to the novel item because it was never trained on reduplicating [t]. However, when dropout is applied, in a subset of epochs, the unit activations distinguishing [t] from [d] will be dropped out. This will allow the model to learn how to reduplicate a syllable that is ambiguous between [ta] and [da]. These ambiguous training epochs could provide enough information to the model for it to learn a function that generalizes correctly to withheld segments.

This explanation could suggest that other forms of regularization, such as an L2 prior, that don't create a similar kind of ambiguity, may not have the same effect on reduplication learning.

5.4 Future Work

A number of avenues exist for furthering this line of research. First of all, as mentioned in §5.2, the full picture of how humans behave in regards to generalizing reduplication-like patterns is still an open question. Additionally, more direct modeling of some of the experimental data that does exist on this subject (e.g. Marcus et al., 1999) could help to shed more light on how well a Seq2Seq network with no explicit variables can model such results.

The simulations outlined here could also be made to more realistically mimic natural language. Stems of varying lengths, and with various syllabic structures could pose an interesting challenge to any model of reduplication. A reduplication process that copies only part of the stem could also test whether the model is capable of learning a more complex identity function. Additionally, presenting the data in a way that mimics the exposure children receive could be useful, since infants are not directly presented with stem-reduplicant pairs in isolation.

Future research could also address the effects of dropout presented here. First of all, since dropout is one of many different regularization methods (Wager et al., 2013), testing its alternatives could be useful. And if it's the case that dropout allows a model to learn in a more human-like way, then adding dropout to models of other domains of language (such as phonotactics) should also be explored.

5.5 Conclusion

In summary, we found that a Seq2Seq model could learn a simple reduplication pattern and generalize that pattern to novel items. Dropout increased the model's scope of generalization from novel syllables to novel segments, demonstrating a human-like behavior that has been previously used as an argument against connectionist models with no explicit variables (for other alternatives to explicit variables in neural networks, see Garrido Alhama, 2017; Gu et al., 2016). These results weaken this line of argument against connectionism.

Acknowledgments

The authors would like to thank the members of the UMass Sound Workshop, the members of the UMass NLP Reading Group, Tal Linzen, and Ryan Cotterell for helpful feedback and discussion. Additionally, we would like to thank the SIGMORPHON reviewers for their comments. This work was supported by NSF Grant #BCS-1650957.

References

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166.

Berent, I. (2013). The phonological mind. *Trends in Cognitive Sciences*, *17*(7), 319–327.

Berent, I., Wilson, C., Marcus, G. F., & Bemis, D. K. (2012). On the role of variables in phonology: Remarks on Hayes and Wilson 2008. *Linguistic Inquiry*, *43*(1), 97–119.

Chollet, F., & others. (2015). Keras. Retrieved January 18, 2018, from https://github.com/keras-team/keras

Chomsky, N., & Halle, M. (1968). *The sound pattern of English.* New York, NY: Harper & Row.

Cotterell, R., Kirov, C., Sylak-Glassman, J., Yarowsky, D., Eisner, J., & Hulden, M. (2016). The SIGMORPHON 2016 shared task—morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology* (pp. 10–22).

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*(2), 179–211.

Endress, A. D., Dehaene-Lambertz, G., & Mehler, J. (2007). Perceptual constraints and the learnability of simple grammars. *Cognition*, *105*(3), 577–614.

Garrido Alhama, R. (2017). Computational modelling of Artificial Language Learning. *Dissertation*, Institute for Logic, Language and Computation (ILLC) at the University of Amsterdam.

Gasser, M. (1993). *Learning words in time: Towards a modular connectionist account of the acquisition of receptive morphology*. Indiana University, Department of Computer Science.

Gu, J., Lu, Z., Li, H., & Li, V. O. (2016). Incorporating copying mechanism in sequence-to-sequence learning. *ArXiv Preprint ArXiv:1603.06393*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Kirov, C. (2017). Recurrent Neural Networks as a Strong Domain-General Baseline for Morpho-Phonological Learning. In *Poster presented at the 2017 Meeting of the Linguistic Society of America*.

Kirov, C., & Cotterell, R. (2018). *Recurrent Neural Networks in Linguistic Theory: Revisiting Pinker & Prince (1988) and the Past Tense Debate*.

Levy, O., Lee, K., FitzGerald, N., & Zettlemoyer, L. (2018). Long Short-Term Memory as a Dynamically Computed Element-wise Weighted Sum. *ArXiv Preprint ArXiv:1805.03716*.

Marcus, G. (2001). *The algebraic mind*. Cambridge, MA: MIT Press.

Marcus, G., Vijayan, S., Rao, S. B., & Vishton, P. M. (1999). Rule learning by seven-month-old infants. *Science*, *283*(5398), 77–80.

Moreton, E. (2012). Inter-and intra-dimensional dependencies in implicit phonotactic learning. *Journal of Memory and Language*, *67*(1), 165–183.

Pater, J. (2018). Generative linguistics and neural networks at 60: foundation, friction, and fusion. *Language*.

Rahman, F. (2016). *seq2seq: Sequence to Sequence Learning with Keras*. Python. Retrieved from https://github.com/farizrahman4u/seq2seq

Rumelhart, D., & McClelland, J. (1986). On learning the past tenses of English verbs. In J. McClelland & D. Rumelhart (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* (Vol. 2: Psychological and Biological Models, pp. 216–271). The MIT Press.

Seidenberg, M. S., & Elman, J. L. (1999). Do infants learn grammar with algebra or statistics? *Science*, *284*(5413), 433f–433f.

Smolensky, P., & Legendre, G. (2006). *The harmonic mind: From neural computation to optimality-theoretic grammar (Cognitive architecture), Vol. 1*. MIT press.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*(1), 1929–1958.

Štekauer, P., Valera, S., & Körtvélyessy, L. (2012). *Word-formation in the world’s languages: a typological survey*. Cambridge University Press.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104–3112).

Tieleman, T., & Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, *4*(2), 26–31.

Tupper, P., & Shahriari, B. (2016). Which Learning Algorithms Can Generalize Identity-Based Rules to Novel Inputs? *ArXiv Preprint ArXiv:1605.04002*.

Wager, S., Wang, S., & Liang, P. S. (2013). Dropout training as adaptive regularization. *Advances in Neural Information Processing Systems*, 351–359.