Can LSTMs and GRUs Generalize Subregular Patterns?

Cody St. Clair
Stony Brook University
cody.stclair@stonybrook.edu

Emily Peterson
Stony Brook University
emily.k.peterson@stonybrook.edu

Joanne Chau
Stony Brook University
choryan.chau@stonybrook.edu

Abstract

This paper acts as an extension to Avcu et al.'s *Subregular Complexity and Deep Learning* (2017), which focused on the usage of formal language theory and grammatical inference as tools in understanding the learning capabilities of deep neural networks. Their study included learning experiments on two types of Recurrent Neural Networks (RNNs), simple RNNs (s-RNN) and Long Short-Term Memory RNNs (LSTMs), on languages obtained from Strictly Local (SL) and Strictly Piecewise (SP) subregular language classes. This paper extends the experiments to include Gated Recurrent Units (GRU) and the Tier-Based Strictly Local (TSL) subregular language class. Our results show that GRU performed better than LSTM in all subregular languages. Unlike GRU, LSTM performed poorly in learning TSL languages. Notably, directionality did not have an effect on language learning in terms of subregular languages.

1 Introduction

This paper is an extension to Avcu et al.'s *Subregular Complexity and Deep Learning* (2017), who trained and tested simple Recurrent Neural Networks (s-RNNs) and Long Short-Term Memory Recurrent Neural Networks (LSTMs) on languages from the Strictly Local (SL) and Strictly Piecewise (SP) subregular language classes. Neural networks are widely used to solve many Natural Language Processing (NLP) problems. They are robust and able to handle a large variety of language modeling tasks, but understanding their patterns has continued to be difficult despite their wide usage. Formal language theory can help provide a more systematic approach to building an understanding of the learning capabilities of neural networks. By being able to associate different neural networks to their pattern of learning, we are able to provide better approaches to current and future NLP problems.

One of the main issues in their study is that they only focused on a small subset of neural networks and subregular language classes. Extending this research program to more neural networks with more hyperparameter options would provide more material for analysis. Using additional subregular language classes can provide additional information on these learning patterns. We are extending to the Tier-Based Strictly Local (TSL) class as there has been supporting evidence that there are natural language patterns that are not handled by either SL or SP classes. Raising to the TSL level will allow us to account for these language patterns.We also added a GRU learning model with both unidirectionality and bidirectionality and implemented dropout to the model. By creating an additional learning model, we are able to use the subregular language data available to understand its learning pattern.

The addition of a dropout hyperparameter was suggested by Avcu et al. (2017). Though it was not employed in our tests, we added the hyperparameter so that it is convenient to incorporate dropout in future work. Bidirectionality was added and tested in the hope that having information from both earlier and later in the string would allow the models to

learn the desired generalizations better than their unidirectional counterparts (though this turned out not to be the case, see section 4). We tested GRU-based models in addition to LSTMs to see how the two compare with each other on this generalization task. This seemed like a particularly interesting direction to explore, as GRUs seem to be growing in popularity over LSTMs in many applications.

We first explain the motivation for focusing on the subregular language classes and give natural language examples for each of the languages we chose in Section 2. A description of the datasets and models is given in Section 3 and results are given in Section 4. Section 5 discusses the results.

2 Subregular Phonology and Morphology

The Chomsky hierarchy is a hierarchical representation of the formal languages. It acts as a guideline for linguistic theories (Jäger and Roger, 2012). Formal languages are organized into a hierarchy ordered by increased complexity of the language. This hierarchy is shown in Figure 1 (Heinz, 2010). We decided to extend Avcu *et. al*'s data set of the Strictly Local (SL) and Strictly Piecewise (SP) languages to include Tier-Strictly Local languages (TSL).

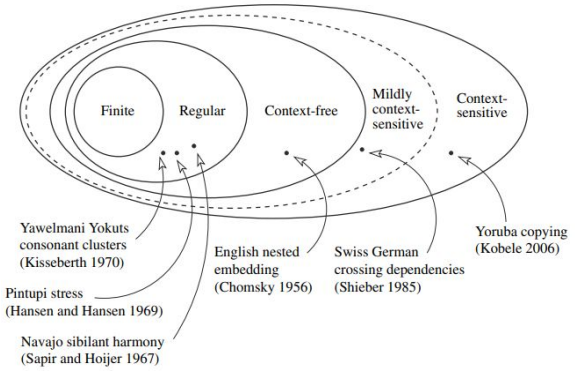

Figure 1: Chomsky Hierarchy (Heinz, 2010)

We chose to explore these three language classes because it has been argued that most phonological and morphological constraints can be explained under the subregular hierarchy. Specifically, such constraints can be explained entirely in SL, SP and TSL (Aksënova et al. 2016). "A subregular language is a set of strings that can be described without employing the full power of Finite State Automatas (FSAs)." (Jäger and Rogers, 2012). As mentioned by Avcu et. al, formal languages as targets of learning have been around for many years. Because the grammars generated from such formal languages are known and the experiment is controlled, utilizing formal languages as targets may provide insights due to their relative complexity.

2.1 Strictly Local

Below are examples of pluralization in English. In English, when a noun is pluralized, in orthography, it is marked with a 's' or 'es'. If the word ends in a voiceless consonant, the plural marking will be [s], if it is a [s] or [z] sound, the plural marking is [ɪz], otherwise, the plural marking will be [z]. Disregarding the sound constraint, in English, pluralization is done by suffixation and suffixation only. Pluralization in English can never be achieved with prefixation. It belongs to the SL class, as it can be explained in terms of forbidden bigram substrings.

(1)	bʌs	'bus'	bʌs-**ɪz**	'buses'
(2)	kæt	'cat'	kæt-**s**	'cats'
(3)	dɔg	'dog'	dɔg-**z**	'dogs'

2.2 Strictly Piecewise

Oftentimes, languages can belong to multiple different subregular classes. For example, in (6) and (7)[1] (Heinz, 2019), we have what is known as Samala Sibilant Harmony. According to native speakers of Samala, a word can not contain both [s] and [ʃ], regardless of distance between the two sounds.

[1] * indicates ungrammaticality

(4) ha∫xintilawa∫
'his former Indian name'
*ha**s**xintilawa∫
*ha∫xintilawa**s**

(5) **s**i**s**huleqpeyu**s**
'they two want to follow it'
***s**i∫huleqpeyu**s**
***s**i**s**huleqpeyu∫

This constraint can be explained by both SP and TSL languages. In SP terms, it can be described with negative constraints forbidding the co-occurrence of [s] and [∫], which could be written as *[s]...[∫] and *[∫]...[s]. In TSL terms, we can have a negative grammar that rules out *[s][∫] and *[∫][s] on a tier. Constructing languages that weren't members of more than one class was difficult because of this.

2.3 Tier-Based Strictly Local

Not all long-distance dependencies in phonology and morphology can be classified as Strictly Piecewise. For example, the phonological phenomena of long distance dissimilation and the requirement of exactly one primary stress (culminativity), along with some common and uncommon morphological phenomena, all fall outside the SP class but within the Tier-Based Strictly Local class. An example of such a pattern is shown in examples (6) and (7)[2], which illustrate Indonesian circumfixation (Aksënova et al. 2016).

(6) tinggi 'high'
ke-tingg-**ian** 'altitude'
(7) mahasiswa 'big pupil (student)'
ke-mahasiswa-**an** 'student affairs'

In these examples, if a string begins with *ke-*, it must end with *-an*, regardless of word length. Furthermore, *ke-* and *-an* cannot appear in a string alone.

[2] The - is used to mark morphological boundaries in order to make the data easier to read and interpret.

3 Experiments

We trained a total of 72 models on training sets from nine subregular languages. The datasets are further explained in Section 3.1. The target task was simple binary classification of strings as accepted ("True") or not accepted ("False") according to the language in question.

3.1 Data

We chose three languages per each of our three subregular classes. All languages drew strings from the alphabet $\Sigma = \{a, b, c, d\}$. The SL languages can be described in terms of banned bigram substrings. SL.0 banned one substring, SL.1 banned three, and SL.2 banned four. The SP languages correspond directly to the SL languages, and can be described in terms of banned bigram subsequences. SP.0 banned one subsequence, SP.1 banned three, and SP.2 banned four. The TSL languages did not correspond to the SL and SP languages in this way; instead, they represent certain other phenomena in natural languages chosen based on examples from Aksënova et al. (2016). TSL.0 modeled circumfixation by requiring that strings beginning with a end in b, and that strings ending in b begin with a. TSL.1 modeled the requirement of exactly one primary stress per word by requiring that every string contain exactly one b. TSL.2 modeled the Swahili pattern described earlier; b could only appear either at the end of a string or in second position after an a.

Datasets for each language were generated using the Pynini python library for compiling string grammars using finite state transducers (Gorman 2016). We owe much of the data generation code to Kostyszyn et al (2019). Each dataset contained a list of randomly generated strings. Half the strings were generated from a finite state acceptor (FSA) representing the language in question. These strings were labeled "True." The other half were generated by the

complement of the FSA and labeled "False." There were two groups of datasets per language. The first group contained sets of 10,000 strings each, while the second group contained sets of 100,000 strings each. In each group, there were five datasets: Training, Development/Validation, and three Tests. The Training, Development, and Test 1 sets contained "short" strings of lengths of 10 to 19 characters. There were no duplicate strings among these three sets. Tests 2 and 3 contained "long" strings of lengths of 31 to 50 characters. These tests were meant to test the models' ability to generalize the patterns they learned to longer sequences. Test 2 allowed duplicates. Test 3 consisted of strings in "adversarial pairs;" the first string in each pair was a "long" string from the language, and the second string was an edit distance of 1 away from the first, but was not in the language. For four model types, three subregular classes, three languages per class, two set sizes, and three tests each, there were 216 test sets in total.

3.2 Hypotheses

We formed a number of hypotheses about how the models would perform based on common intuitions about the abilities of RNNs as well as on the findings of Avcu et al. (2017). First, since bidirectionality has been shown to improve on the performance of simple RNNs and LSTMs on certain tasks which involve learning long-distance dependencies (Graves & Schmidhuber 2005), we expected the bidirectional models to outperform the unidirectional ones. We also expected that the models trained on 100k-string sets would do better than those trained on 10k-sets, since it is generally expected that more training data leads to better performance. We expected performance to be high on Test 1, since the Training and Test 1 sets contain strings of equal length. We expected Test 2 performance to suffer in comparison, since its strings were at least ten characters longer than the training strings. Finally, we expected performance on Test 3 to suffer due to the extreme similarity between "True" and "False" strings.

3.3 Model

Our model is essentially a reimplementation of the model of Kostyszyn et al. (2019) and Avcu et al. (2017), which was originally built in PyTorch. Our reimplementation instead uses Tensorflow/Keras. Unless otherwise noted, Keras's default parameter settings were used. A summary of the model is presented in Figure 2. We extended the original model by adding the GRU as a possible RNN type and adding bidirectional and dropout hyperparameters (though we did not employ dropout in any of our tests). It is our hope that this model, as well as our preliminary results, will promote future investigation in the area of subregular pattern recognition in neural models.

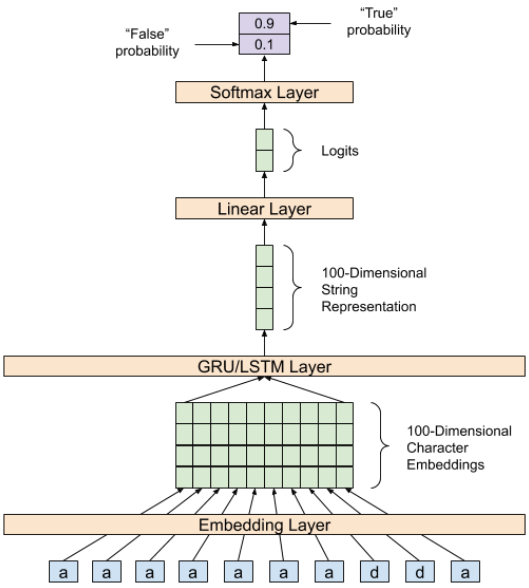

Figure 2: Summary of our model

The model consists of a typical embedding layer, which converts the input characters into dense, learned embeddings. All of our models used 100-dimensional embeddings. This sequence of embeddings for each word is then passed through an RNN (either a GRU or an

LSTM in our tests), which constructs a single vector representation of the input word, with the same number of dimensions as the embeddings. The RNN layer may be either unidirectional or bidirectional. This is passed through a simple dense layer with a linear activation function to produce two logit values, one representing true and the other representing false. Finally, the logits are passed through a softmax function to obtain a probability distribution. If "True" is the more likely option, then the model will predict that the input string is in the target language, while if "False" is more likely, it will predict the string is not in the language.

During training, the output probabilities are compared with ground truth values, which are 2-dimensional one-hots, so that if a training string is in the language, its "True" probability is 1, and if it is not then its "False" probability is 1. Loss values are then computed as a binary cross entropy between the model's output probabilities and the ground truth. Our code for data generation, models, training, a ReadMe and evaluation can be found at https://github.com/emkp/CSE538_FinalProject.

4 Results

Accuracy and F_1 scores for the LSTM and GRU models are given in Figures 3 and 4, respectively. Adding bidirectionality had no effect on the scores for any of the models. Both types of model reached perfect or near-perfect accuracy on many tests, but they struggled in the two more complex SP languages, especially on Test 3. In general, performance was best on the SL languages. Both kinds of models performed perfectly on all the 100k SL tests and on all TSL.0 tests. The GRUs also performed perfectly on all SP.0 tests.

Whereas the GRUs performed similarly on TSL and SL, the LSTMs fell behind on TSL. They suffered significantly on TSL.1 and TSL.2, especially in the smaller datasets. The 10k TSL.1 LSTM performed at 75.16% accuracy on Test 2, but performed worse than chance on Test 3 at 45.35%. The 10k TSL.2 LSTM had similar issues; it performed barely above chance at 54.43% accuracy on Test 2, and only slightly better (65.8%) on Test 3. The corresponding F_1 scores reveal more information about the problem; the models were apparently drastically over-assigning True values. This problem was mitigated by increasing the dataset size to 100k for TSL.1 (100%), but not for TSL.2 (69.39%).

The LSTMs did even worse on the SP languages. For these experiments, Test 1 was no challenge, and the Test 2 results were nearly on par with Test 1, except for the puzzling result of 51.93% accuracy on the 10k SP.0 Test 2. This result was particularly strange, since SP.0 was the simplest (fewest banned subsequences) of the SP languages, and the corresponding SP.1 and SP.2 models had nearly perfect accuracy scores. The SP LSTMs also struggled on Test 3, with scores between 50% and 76.6% for all but 100k SP.0 Test 3.

LSTM, no dropout

			Unidirectional						Bidirectional					
			SL.0		SL.1		SL.2		SL.0		SL.1		SL.2	
			F-score	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy
SL	100k	Test1	1	1	1	1	1	1	1	1	1	1	1	1
		Test2	1	1	1	1	1	1	1	1	1	1	1	1
		Test3	1	1	1	1	1	1	1	1	1	1	1	1
	10k	Test1	1	1	1	1	0.9996	0.9996	1	1	1	1	0.9996	0.9996
		Test2	1	1	1	1	1	1	1	1	1	1	1	1
		Test3	1	1	1	1	0.86402	0.8436	1	1	1	1	0.86402	0.8436
			SP.0		SP.1		SP.2		SP.0		SP.1		SP.2	
			F-score	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy
SP	100k	Test1	1	1	1	1	0.99998	0.99998	1	1	1	1	0.99998	0.99998
		Test2	0.99999	0.99999	1	1	1	1	0.99999	0.99999	1	1	1	1
		Test3	0.99989	0.99989	0.67547	0.51956	0.76561	0.69385	0.99989	0.99989	0.67547	0.51956	0.76561	0.69385
	10k	Test1	0.9998	0.9998	1	1	1	1	0.9998	0.9998	1	1	1	1
		Test2	0.07433	0.5193	1	1	0.9999	0.9999	0.07433	0.5193	1	1	0.9999	0.9999
		Test3	0.03088	0.5042	0.6668	0.5003	0.76587	0.6943	0.03088	0.5042	0.6668	0.5003	0.76587	0.6943
			TSL.0		TSL.1		TSL.2		TSL.0		TSL.1		TSL.2	
			F-score	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy
TSL	100k	Test1	1	1	1	1	1	1	1	1	1	1	1	1
		Test2	1	1	0.99996	0.99996	0.99992	0.99992	1	1	0.99996	0.99996	0.99992	0.99992
		Test3	1	1	1	1	0.75339	0.67267	1	1	1	1	0.75339	0.67267
	10k	Test1	1	1	1	1	1	1	1	1	1	1	1	1
		Test2	1	1	0.66959	0.7516	0.16339	0.5443	1	1	0.66959	0.7516	0.16339	0.5443
		Test3	1	1	0.38863	0.4535	0.49809	0.658	1	1	0.38863	0.4535	0.49809	0.658

Figure 3: Results for unidirectional and bidirectional LSTM models with no dropout

GRU, no dropout

			Unidirectional						Bidirectional					
			SL.0		SL.1		SL.2		SL.0		SL.1		SL.2	
			F-score	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy
SL	100k	Test1	1	1	1	1	1	1	1	1	1	1	1	1
		Test2	1	1	1	1	1	1	1	1	1	1	1	1
		Test3	1	1	1	1	1	1	1	1	1	1	1	1
	10k	Test1	1	1	1	1	1	1	1	1	1	1	1	1
		Test2	1	1	1	1	0.9999	0.9999	1	1	1	1	0.9999	0.9999
		Test3	0.9995	0.9995	1	1	0.951113	0.9486	0.9995	0.9995	1	1	0.951113	0.9486
			SP.0		SP.1		SP.2		SP.0		SP.1		SP.2	
			F-score	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy
SP	100k	Test1	1	1	1	1	0.99999	0.99999	1	1	1	1	0.99999	0.99999
		Test2	1	1	1	1	1	1	1	1	1	1	1	1
		Test3	1	1	1	1	0.806335	0.75982	1	1	1	1	0.806335	0.75982
	10k	Test1	1	1	0.9999	0.9999	0.998899	0.9989	1	1	0.9999	0.9999	0.998899	0.9989
		Test2	1	1	1	1	1	1	1	1	1	1	1	1
		Test3	1	1	0.670736	0.5091	0.766284	0.695	1	1	0.670736	0.5091	0.766284	0.695
			TSL.0		TSL.1		TSL.2		TSL.0		TSL.1		TSL.2	
			F-score	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy	F-score	Accuracy
TSL	100k	Test1	1	1	1	1	1	1	1	1	1	1	1	1
		Test2	1	1	1	1	0.99997	0.99997	1	1	1	1	0.99997	0.99997
		Test3	1	1	1	1	0.986738	0.98656	1	1	1	1	0.986738	0.98656
	10k	Test1	1	1	1	1	1	1	1	1	1	1	1	1
		Test2	1	1	1	1	0.9999	0.9999	1	1	1	1	0.9999	0.9999
		Test3	1	1	1	1	0.986777	0.9866	1	1	1	1	0.986777	0.9866

Figure 4: Results for unidirectional and bidirectional GRU models with no dropout

5 Discussion

We will first compare the results of our LSTM experiments with the results from Avcu et al. (2017). Whereas Avcu et al. reported mean accuracy scores over 10 runs of each experiment, we were limited to one run per experiment due to time constraints (it took 17 hours to complete one run per experiment on a laptop CPU). To establish the reliability of our results, we will have to repeat the experiments and calculate mean scores. Please consider the results below to be preliminary and in need of

replication. We will then discuss what we have learned given our hypothesis.

5.1 Comparison with Previous Work

In general, our LSTMs' performance scores were more extreme than those of Avcu et al.; our highest scores (100%) were higher than theirs, and our lowest scores were lower than theirs. This is not too surprising, since our experiments are not exactly the same -- they used slightly different minimum and maximum lengths for the "short" and "long" strings, for example. The experiments of ours that are most directly comparable with the previous paper are those for the unidirectional LSTMs trained on SL.2 and SP.2.

We can compare these models with the previous paper's SL2 and SP2 experiments with vectors of size 100. In their paper, the SL2 results stood out as peculiar because "the SL2 experiments, which are arguably the simplest patterns to learn, were among the worst results in all the experiments" (Avcu et al. 2017). This specific result was not replicated in our results; our LSTMs performed within one percentage point of 100% on SL.2 for Tests 1 and 2. However, there was a drop in accuracy on Test 3 (84.36%), which was not seen in Avcu et al.

Furthermore, our results did seem to reproduce a different (but related), broader trend of theirs, which is that there was a drop in accuracy for LSTMs on SP languages in general. This trend appeared in our results even though we included only SP languages with forbidden bigram subsequences, whereas Avcu et al. only saw this trend for SP languages with larger forbidden subsequences (i.e. 4-grams and 8-grams), and not with forbidden bigrams. What's more, for our results, the trend is only pronounced in the Test 3 SP experiments. In other words, whereas Avcu et al found a drop in accuracy for LSTMs on SP languages as the length of the forbidden ngram sequences increased, we found a similar drop in accuracy, but related not to ngram length, but rather to the "difficulty" of the Test set and the *number* of forbidden (bigram) subsequences. This result reinforces the need for more investigation into deep networks' performance on the SP class.

5.2 Further Discussion

We will now address our hypotheses from section 3.2. First, the bidirectional models did not outperform the unidirectional ones as expected; in fact, the results were exactly the same for unidirectional and bidirectional versions of the models. We suspect that this might have something to do with the alphabet being so small. For example, consider the SP.2 language. It disallows the following subsequences: *ba, bb, bd,* and *ab*. In consequence, if a string contains a *b* early in the string, the only choice for the rest of the string is a long stretch of repeated *c*s. We therefore generated many strings ending in stretches of 20, 30, or more *c*s in a row. A model trained on such strings might come to learn the wrong generalization; that is, to simply accept strings that have a lot of *c*s. It's not clear that adding bidirectionality to a model would help it overcome this obstacle. However, if we used a bigger alphabet, there would be more options of characters to use besides *c*, and the model might be able to generalize better. Future research should investigate the effects of increasing the alphabet size.

In terms of data sets, the models trained on the larger (100k) datasets did tend to outperform those trained on the smaller sets, as expected. One exception is the Test 3 result for the SP.2 LSTM, which saw no improvement from 10k to 100k. This indicates that a lack of sufficient data is not the source of the drop in accuracy on this language. As expected, Test 1 proved to be the "easiest" test -- or at least not significantly more

difficult than Tests 2 and 3 -- for all experiments. Performance on the other two tests was highly related to the model type and language class. We will compare performance across these parameters next.

We also predicted that Test 3 would be a challenge to the models. For GRUs, this played out only for the 10k SP.1 and SP.2 experiments, with accuracy scores of 50.91% and 69.5%, respectively. In other words, while the GRUs were not generally challenged by the TSL languages, they did have trouble generalizing the more complex SP patterns to longer strings. The LSTMs, on the other hand, struggled significantly on both SP and TSL, with the exception of TSL.0. Again, these results suggest that GRUs are better suited than LSTMs to generalizing TSL patterns, but that both struggle with SP patterns. The GRUs did perfectly (or nearly) on all Test 1 and Test 2 experiments. Our prediction that performance on Test 2 would suffer played out reliably only only for SP.0, TSL.1, and TSL.2, a puzzling result. In general, however, GRUs outperformed the LSTMs wherever the LSTMs had less-than-perfect scores. It's unclear why this should be.

Again, all of these experiments are in need of more training runs to confirm the reliability of the results. It is possible that, with a different random initialization of the character embeddings, the performance drops seen here could be lessened. However, if this were the case, it would point to another problem with RNNs' learning capabilities -- that the results are highly dependent on the random seed. As Weber et al. (2018) point out, such dependence on the random seed is a sign that the models' ability to generalize is highly sensitive, which would be a considerable weakness of the deep learning approach.

6 Conclusion

This paper described a set of experiments for probing the learning capabilities of LSTMs and GRUs in terms of subregular complexity. We focused on the models' ability to learn and generalize long-distance dependencies. The results show that GRUs generalize more successfully than LSTMs. Adding bidirectionality to the models had no effect on performance. We also found that though GRUs outperformed LSTMs in general, both model types struggled with the Strictly Piecewise language. Furthermore, LSTMs did significantly worse in learning the TSL patterns.

Using our training and model scripts, this approach to probing neural networks' learning capabilities can be easily scaled and extended to include other subregular classes and other model architectures. Future research should also examine the effects of varying the alphabet size, language complexity within classes, and other hyperparameters. Augmenting the networks with other common techniques such as dropout and attention may also have significant effects on the results of experiments on neural networks and subregular languages.

References

Aksënova, A., Graf, T., & Moradi, S. (2016, August). Morphotactics as tier-based strictly local dependencies. In Proceedings of the 14th sigmorphon workshop on computational research in phonetics, phonology, and morphology (pp. 121-130).

Avcu, E., Shibata, C., & Heinz, J. (2017). Subregular complexity and deep learning. arXiv preprint arXiv:1705.05940.

Gorman, K. (2016, August). Pynini: A Python library for weighted finite-state grammar compilation. In Proceedings of the SIGFSM Workshop on Statistical

NLP and Weighted Automata (pp. 75-80).

Graf, T. (2015, December). Computational Unity Across Language Modules [Powerpoint Slides]. Retrieved from https://thomasgraf.net/doc/talks/Graf15MSUSubregulartalk.pdf.

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural networks, 18(5-6), 602-610.

Heinz, J. (2019). Class 6: Rawski LSA [Powerpoint Slides]. Retrieved from http://jeffreyheinz.net/classes/davis2019/materials/class6-Rawski-LSA.pdf.

Heinz, J. (2010). Learning long-distance phonotactics. Linguistic Inquiry, 41(4), 623-661.

Heinz, J., Rawal, C., & Tanner, H. G. (2011, June). Tier-based strictly local constraints for phonology. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human language technologies (pp. 58-64).

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

Jäger, G. & Rogers, J. (2012). Formal language theory: refining the Chomsky hierarchy.

Kostyszyn, K., Tuck, E., Belliveau, S., Gao, T., & Heinz, J. (2019). SBFST - Classify the Language of Your DFA. GitHub repository. https://github.com/kkostyszyn/SBFST_2019

Weber, N., Shekhar, L., & Balasubramanian, N. (2018). The fine line between linguistic generalization and failure in Seq2Seq-attention models. arXiv preprint arXiv:1805.01445.