The Fine Line between Linguistic Generalization and Failure in Seq2Seq-Attention Models

Noah Weber[*], **Leena Shekhar**[*], **Niranjan Balasubramanian**
Stony Brook University, NY
{nwweber, lshekhar, niranjan}@cs.stonybrook.edu

Abstract

Seq2Seq based neural architectures have become the go-to architecture to apply to sequence to sequence language tasks. Despite their excellent performance on these tasks, recent work has noted that these models usually do not fully capture the linguistic structure required to generalize beyond the dense sections of the data distribution (Ettinger et al., 2017), and as such, are likely to fail on samples from the tail end of the distribution (such as inputs that are noisy (Belinkov and Bisk, 2018) or of different lengths (Bentivogli et al., 2016)). In this paper, we look at a model's ability to generalize on a simple symbol rewriting task with a clearly defined structure. We find that the model's ability to generalize this structure beyond the training distribution depends greatly on the chosen random seed, even when performance on the standard test set remains the same. This suggests that a model's ability to capture generalizable structure is highly sensitive. Moreover, this sensitivity may not be apparent when evaluating it on standard test sets.

1 Introduction

It is well known that language has certain structural properties which allows natural language speakers to make "infinite use of finite means" (Chomsky, 1965). This structure allows us to generalize beyond the typical machine learning definition of generalization (Valiant, 1984) (which considers performance on the distribution that generated the training set), permitting the understanding of any utterance sharing the same structure, regardless of probability. We refer to this notion as *linguistic* generalization [1].

Many problems in NLP are treated as sequence to sequence tasks with solutions built on seq2seq-attention based models. While these models perform very well on standard datasets and also appear to capture some linguistic structure (Williams et al., 2018; Belinkov et al., 2017; Linzen et al., 2016), they also can be quite brittle, typically breaking on uncharacteristic inputs (Lake and Baroni, 2018; Belinkov and Bisk, 2018).

Due to the high capacity of these models, it is not unreasonable to expect them to learn *some* structure from the data. However, learning structure is not a sufficient condition to achieving linguistic generalization. If this structure is to be usable on data outside the training distribution, the model must learn the structure *without* additionally learning patterns specific to the training data.

In this work, we look at the feasibility of training seq2seq-attention models so they generalize in this linguistic sense. We train models on a symbol replacement task with a well defined generalizable structure. The task is simple enough that all models achieve near perfect accuracy on the standard test set, i.e., where the inputs are drawn from the same distribution as that of the training set. We then test these models for linguistic generalization by creating test sets of uncharacteristic inputs, i.e., inputs that are not typical in the training distribution but still solvable given that the generalizable structure was learned. Our results show that generalization is highly sensitive[2]; even changes in the random seed can drastically affect the ability to generalize. This suggests that the line between generalization and failure is quite fine, and may not be feasible to reach by tuning alone.

2 Symbol Rewriting Task

Real world NLP tasks are complex, and as such, it can be difficult to precisely define what a model

[*]These authors contributed equally to this work.

[1]From here on, mentions of generalization refer to the linguistic kind.

[2]The sensitivity of generalization is also hinted at in McCoy et al. (2018) who additionally note performance variations across initializations

Proceedings of the Workshop on Generalization in the Age of Deep Learning, pages 24–27
New Orleans, Louisiana, June 5, 2018.

should and should not learn during training. As done in previous work (Lake and Baroni, 2018; Rodriguez and Wiles, 1998), we ease analysis by looking at a simple formal task. The task is set up to mimic (albeit, in an oversimplified manner) the input-output symbol alignments and local syntactic properties that models must learn in many natural language tasks, such as translation, tagging and summarization. The task is defined over sequences of symbols, $\{x_1, ...x_n | x_i \in X\}$, where X is the input alphabet. Each symbol $x \in X$ is uniquely associated with its own output alphabet Y_x. Output is created by taking each individual symbol x_i in the sequence and rewriting it as any sequence of k symbols from Y_{x_i}. To do the task, the model must learn alignments between the input and output symbols, and preserve the simple local syntactic conditions (every group of k symbols must come from the same input alphabet Y_x). As an example, let $X = \{A, B\}$, $Y_A = \{\text{Consonants}\}$, $Y_B = \{\text{Vowels}\}$, and $k = 2$. Then a valid output for the input BA would be $aupt$. For our task, $|X| = 40$ and each x_i has a corresponding output alphabet Y_{x_i} of size 16.

To generalize to any input sequence, a model must: **(1)** learn the generalizable structure - the alignments between input and output alphabets, and **(2)** *not* learn any dependencies among input symbols or sequence length. To test the extent to which **(2)** is met, we train[3] seq2seq-attention models with 100,000 randomly generated samples with inputs uniformly generated with lengths 5-10 and no input symbol appearing more than once in a single sample. If the model learned alignments without picking up other dependencies among input symbols or input lengths then the resulting model should have little problem in handling inputs with repeated symbols or different lengths, despite never seeing such strings.

For evaluation we trained 50 different models with the *same* configuration, chosen with a validation set, but with *different* random seeds. We created 4 different test sets, each with 2000 randomly generated samples. The first test set consists of samples that are characteristic of the training set, having lengths 5-10 and no repeats (**Standard**). The second set tests the model's ability to generalize to repeated symbols in the input (**Repeat**). The third and fourth sets test its ability to generalize to different input lengths, strings of length 1-4 (**Short**) and 11-15 (**Long**) respectively.

[3] A detailed account of model training, regularization, and tuning is provided in the supplementary material.

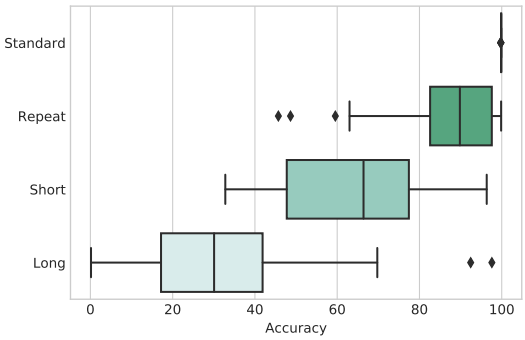

Figure 1: Accuracy % distribution across 50 runs with different random seeds on the four test sets.

3 Results and Conclusions

The distribution of model accuracy[4] measured at instance level on the four test sets across all the 50 seeds is given in Figure 1. All models perform above 99% on the standard set, with a deviation well below 0.1. However, the deviation on the other two sets is much larger, ranging from 13.39 for the repeat set to 20.63 for the long set. In general, the model performs better on the repeat set than on the short and long sets. Performance on the short and long sets is not always bad, some seeds giving performances of above 95% for either the short or long set. Ideally, we would like a seed which performs good on all the test sets; however, this seems hard to obtain. The highest average performance across the non standard test sets for any seed was 79.52%. Learning to generalize for both the repeated and longer inputs seems even harder, with the Pearson correlation between performance on the repeat and long sets being -0.71.

The variability in generalization on uncharacteristic inputs (and thus, the extent of linguistic generalization) given different random seeds is alarming, particularly given the fact that the standard test set performance remains mostly the same regardless. The task presented here was easy and simple to analyze, however, future work may be done on natural language tasks. If these properties hold it might indicate that a new evaluation paradigm for NLP should be pushed; one that emphasizes performance on uncharacteristic inputs in addition to the data typically seen in training.

[4] We compute accuracy as $\frac{\text{\# times the model produced a valid output}}{\text{\# samples}}$.

References

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. *6th International Conference on Learning Representations* .

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 861–872.

Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. 2016. Neural versus phrase-based machine translation quality: a case study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 257–267.

Noam Chomsky. 1965. *Aspects of the Theory of Syntax*. The MIT Press, Cambridge.

Allyson Ettinger, Sudha Rao, Hal Daum III, and Emily M Bender. 2017. Towards linguistically generalizable nlp systems: A workshop and shared task. In *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9(8):1735–1780.

Brenden Lake and Marco Baroni. 2018. Still not systematic after all these years: On the compositional skills of sequence-to-sequence recurrent networks. *ICLR 2018* .

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *TACL* 4:521–535.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*. The Association for Computational Linguistics, pages 1412–1421.

R. Thomas McCoy, Robert Frank, and Tal Linzen. 2018. Revisiting the poverty of the stimulus: hierarchical generalization without a hierarchical bias in recurrent neural networks. *CoRR* abs/1802.09091.

Paul Rodriguez and Janet Wiles. 1998. Recurrent neural networks can learn to implement symbol-sensitive counting. In *Advances in Neural Information Processing Systems*. pages 87–93.

L. G. Valiant. 1984. A theory of the learnable. *Commun. ACM* 27(11):1134–1142.

Adina Williams, Andrew Drozdov, and Samuel R. Bowman. 2018. Do latent tree learning models identify meaningful structure in sentences? *Transactions of the ACL (TACL)* abs/1709.01121.

A Supplemental Material

A.1 Model and Training Details

The models we use are single layer, unidirectional, seq2seq LSTMs (Hochreiter and Schmidhuber, 1997) with bilinear attention (Luong et al., 2015) and trained with vanilla SGD. To determine the epoch to stop training at, we create a validation set of 2000 samples with the same characteristics as the training set, i.e., of length 5-10 with no repeated symbols. Training is stopped once accuracy[5] on the validation set either decreases or remains unchanged. The size of the hidden state and embeddings were chosen such that they were as small as possible without reducing validation accuracy, giving a size of 32.

Tuning hyperparameters is often done on a validation set drawn from the same distribution as the training set (as we often don't know the exact form of uncharacteristic inputs, with the exception of noisy inputs) which motivated our decision to use a validation set of characteristic inputs to decide the epoch to stop at. However, we noticed only small variation in the validation performance upon using different learning rates and dropout probabilities (where dropout was applied to the input and output layers). In order to fine tune these parameters to avoid extreme overfitting, we created another validation set consisting of 5000 samples of "uncharacteristic" inputs, i.e., inputs with repeated symbols and varying from length 3-12. These two hyperparameter values were set to 0.125 and 0.1, respectively, according to the performance on this validation set, averaged across a set of randomly chosen random seeds. Further training details are listed in Table 1.

A.2 Symbol Rewriting Task Examples

Here we provide a simple example of the task. If the input symbol A maps to any permutations of a_1, a_2, or a_3, and B maps to permutations of b_1, b_2, or b_3. Each a_i and b_i has 2 possible values, a_{i1} or a_{i2} and b_{i1} or b_{i2} respectively. Thus, mapping an input symbol to 48 $(8 * 3!)$ possible permutations. A possible valid output for the input AB is $a_{21}a_{32}a_{11}b_{32}b_{11}b_{22}$. Note that any such permutation is valid and permutations are selected at random when generating the data. We allow this stochasticity in the outputs in order to prevent the model from resorting to pure memorization. Ta-

[5] Defined in section 3

LSTM Layers	1
WE/LSTM size	32
Attention	Bilinear
Batch size	64
Optimizer	SGD
LR	0.125
Max gradient norm	5
Dropout	0.1

Table 1: Model details.

	Standard	Repeat	Short	Long
Size	2k	2k	2k	2k
Src Length	5-10	5-10	1-4	11-15
Tgt Length	15-30	15-30	3-12	33-45

Table 2: Details about the four test sets used in our experiments.

ble 2 provides further information on the 4 different test sets.

A.3 Model Performance

We provide the summary statistics across all runs (50 different random seeds) in Table 3, which gives the mean, standard deviation, minimum, and maximum accuracies across all random seeds. We additionally provide a sample of performances for some individual random seeds in Table 4, with the highest and lowest accuracies in each column highlighted.

	Standard	Repeat	Short	Long
Mean	99.85	86.67	64.36	32.09
Std.	0.03	13.39	18.61	20.63
Min.	99.73	45.70	32.80	0.15
Max.	99.88	99.85	96.35	97.60

Table 3: Accuracy % summarized across all 50 runs with different random seeds.

Seed	Standard	Repeat	Short	Long
2787	**99.88**	94.65	42.05	23.05
5740	99.86	**45.70**	56.55	**97.60**
10000	99.86	98.55	**32.80**	**0.15**
14932	**99.73**	87.05	42.20	29.75
28897	99.87	**99.85**	47.40	1.40
30468	99.87	86.35	**96.35**	12.90

Table 4: Accuracy % on the test sets for selected runs out of 50 with different random seeds.