Evaluating Transformer's Ability to Learn Mildly Context-Sensitive Languages

Shunjie Wang*
Independent Researcher
shunjiewang@hotmail.com

Shane Steinert-Threlkeld
University of Washington
shanest@uw.edu

Abstract

Despite the fact that Transformers perform well in NLP tasks, recent studies suggest that self-attention is theoretically limited in learning even some regular and context-free languages. These findings motivated us to think about their implications in modeling natural language, which is hypothesized to be mildly context-sensitive. We test the Transformer's ability to learn mildly context-sensitive languages of varying complexities, and find that they generalize well to unseen in-distribution data, but their ability to extrapolate to longer strings is worse than that of LSTMs. Our analyses show that the learned self-attention patterns and representations modeled dependency relations and demonstrated counting behavior, which may have helped the models solve the languages.

1 Introduction

Transformers (Vaswani et al., 2017) have demonstrated well-versed language processing capabilities and enabled a wide range of exciting NLP applications ever since its inception. However, Hahn (2020) shows that hard self-attention Transformers, as well as soft attention under some assumptions, fail at modeling regular languages with periodicity as well as hieararchical context-free languages eventually when presented with long enough sequences.

These theoretical limitations have since sparked the interest of the formal language community. A variety of formal languages, as well as formal models of computation such as circuits, counter automata, and predicate logic, have been studied to characterize the expressiveness of the architecture.

When it comes to probing an architecture's linguistic adequacy, a particular class of formal languages and formalisms naturally comes into sight: the *mildly context-sensitive* class (Joshi, 1985; Kallmeyer, 2010), the formal complexity class hypothesized to have the necessary expressive power for natural language.

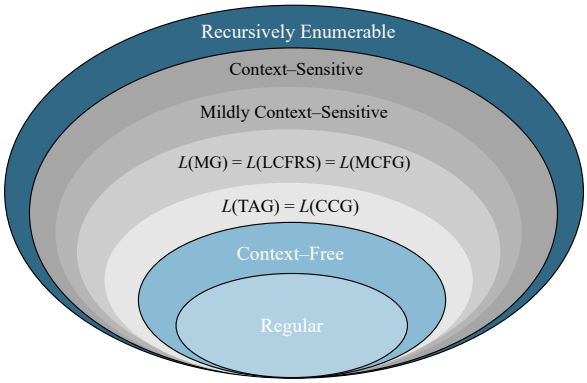


Figure 1: How certain MCSGs fit on the Chomsky hierarchy of languages in terms of their weak generative capacities (Stabler, 2011): CFL $\subset$ $L(\text{TAG}) = L(\text{CCG}) \subset L(\text{MG}) = L(\text{LCFRS}) = L(\text{MCFG}) \subset$ CSL. No formalism generates the largest set of MCSLs.

This motivates us to study the Transformer's ability to learn a variety of linguistically significant, mildly context-sensitive string languages of varying degrees of complexities. Specifically, we ask two research questions:

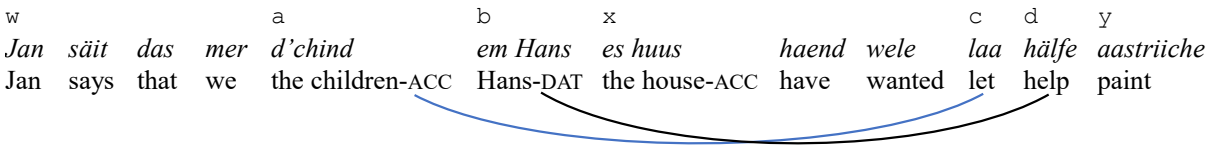
1. How well do Transformers learn MCSLs from finite examples, both in terms of generalizing to in-distribution data, as well as extrapolating to strings longer than the ones seen during training?

2. What kind of meaningful representations or patterns do the models learn?

Our contributions include that we extend current empirical studies on formal language learning with Transformers to the mildly context-sensitive class, and find that they generalize well to unseen strings within the same length range as training data, but their ability to extrapolate is worse than that of LSTMs. We also present analyses that suggest self-attention learned symbol dependency relations, and

*This work extends the author's master's thesis (Wang, 2021) done at University of Washington.

Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP, pages 271–283



'Jan says that we have wanted to let the children help Hans paint the house.'

Figure 2: Swiss German subordinate clauses allow n accusative NPs before m dative NPs, followed by n corresponding accusative object taking verbs before m corresponding dative object taking verbs. Shieber (1985) defined a homomorphism for Swiss German such that when intersecting with regular $\texttt{wa}^*\texttt{b}^*\texttt{xc}^*\texttt{d}^*\texttt{yz}$ yields non-context-free $\texttt{wa}^n\texttt{b}^m\texttt{xc}^n\texttt{d}^m\texttt{yz}$, which contradicts CFL's closure property under intersection with regular languages.

the representations encoded count information for some complex languages, which may have been useful for solving languages in this class.

2 Mildly Context-Sensitive Hypothesis

In search of computational models that have adequate power to generate natural language sentences while also assigning meaningful structural descriptions like trees to them, Chomsky (1956, 1959) defined context-sensitive grammars (CSG) and context-free grammars (CFG) as intermediate systems that lie between two extremities: the Turing machine which overgenerates and the finite-state automaton which undergenerates. A question that immediately follows the definitions is whether the CFG could serve as a computational model for natural language, which had been an open question for a few decades until it was settled by evidence such as Swiss German cross-serial dependency (Figure 2; Shieber, 1985) and Bambara vocabulary (Culy, 1985), which demonstrated the existence of natural languages that are supra-context-free.

However, although more restricted than the Turing machine, the CSG is also undesired as it still has much more generative capacity than natural languages should ever need, and as a result of that, is hard to parse efficiently. This motivated Tree-Adjoining Grammars (TAG; Joshi, 1985) and Combinatory Categorial Grammars (CCG; Steedman, 1987) among a few other weakly equivalent formalisms such that extend the CFG with just enough additional descriptive power so that phenomena like Swiss German cross-serial dependency can be treated, while not using the full CSG thus parsing can still be efficient. The properties of these formalisms with such additional power roughly characterize a class of languages and grammars that Joshi (1985) calls *mildly context-sensitive languages/grammars* (MCSL/MCSG).

Another related line of weakly equivalent formalisms, such as Linear Context-Free Rewriting Systems (LCFRS; Vijay-Shanker et al., 1987), Multiple Context-Free Grammars (MCFG; Seki et al., 1991), and Minimalist Grammars (MG; Stabler, 1997), further extend their expressive power beyond that of the TAG, as motivated by more complex phenomena like German scrambling (Becker et al., 1991). While no single grammar formalism generates the largest possible set of MCSLs that satisfy the formal characterization in Kallmeyer (2010), these are the closest approximations we have. Such differences in expressiveness formed a subhierarchy within this class (Figure 1), and the languages recognizable by TAGs and MGs and their respective equivalents, denoted as $L(\text{TAG})$ and $L(\text{MG})$, are therefore the language complexity classes that we examine in this work.

Therefore, one hypothesis for the complexity of natural language is that it is mildly context-sensitive (henthforth MCS). There have been some challenges citing linguistic data requiring more power beyond MCS (Radzinski, 1991; Michaelis and Kracht, 1997; Kobele, 2006), but the validity of these claims remains controversial (Bhatt and Joshi, 2004; Clark and Yoshinaka, 2012), or no consensus has been reached on the need for more power (Graf, 2021). Thus, while acknowledging that whether the MCS hypothesis is true remains an open question, it is a reasonably good hypothesis that allows us to analyze natural languages meaningfully.

3 Related Work

Regarding the expressiveness of the Transformer, Pérez et al. (2019, 2021) established the Turing-completeness of the hard-attention Transformer. Bhattamishra et al. (2020b) proves the Turing-completeness of soft attention by showing that they can simulate RNNs. However, these results assumed arbitrary precision for weights and activations, had certain departures from the original archi-

tecture, and made the proofs through their unique task definitions.

In a practical case of formal language learning from finite examples, the Transformer's ability is known to be limited, even in the regular language class. Empirically, it was shown that Transformers of different self-attention variants have limited abilities to learn certain star-free languages, as well as non-star-free, periodic regular languages (Hahn, 2020; Bhattamishra et al., 2020a), although the latter may still be recognized theoretically with a simple modification to the architecture (Chiang and Cholak, 2022).

As for context-free languages with hiearchical structural analyses such as Dyck-n, Ebrahimi et al. (2020) empirically demonstrated one Transformer encoder setup in which such languages may be learned and observed stack-like behavior in self-attention patterns. Yao et al. (2021) proved and empirically showed that self-attention can learn Dyck-n with a bounded depth, although the boundedness reduces the CFL to regular. Additional empirical work on Dyck-n include Bernardy et al. (2021); Wen et al. (2023), among others.

Besides language recognition guided by the Chomsky hierarchy, another line of research investigates other alternative formal languages, such as counter-recognizable languages (Bhattamishra et al., 2020a) and first-order logic (Merrill and Sabharwal, 2022; Chiang et al., 2023), to characterize the expressiveness of Transformers.

This work introduces MCSGs into the empirical explorations through assessing the ability of Transformers in certain learning settings to learn a variety of string languages recognizable by MCSGs of different complexities, which have not yet been studied as a whole like languages in other classes. Occasionally, studies on the Transformer's learning ability worked with data that conveniently fall into this class, including a few counter languages that are also TAG-recognizable (Bhattamishra et al., 2020a), discontinuities in Dutch (Kogkalidis and Wijnholds, 2022), reduplication (Deletang et al., 2023), as well as a crossed parentheses language inspired by crossing dependency[1] (Papadimitriou and Jurafsky, 2023). This work complements these and other aforementioned related work by presenting a systematic evaluation guided by basic MCSL constructions and the subhierarchy within the class, as well as comparing each of the basic constructions against a less and a more complex counterparts.

4 Methodology

4.1 Task Setups

Our experiments use the original soft-attention Transformer with sinusoidal positional encoding (henthforth PE) as defined in Vaswani et al. (2017), and the architecture we use is based on a Transformer with just self-attention and feedforward sublayers, and depending on the task, we may use either unidirectional or bidirectional attention. We avoid using dropout as it could negatively impact performance since we are working with simple abstract formal languages. For each experiment, we also train an LSTM (Hochreiter and Schmidhuber, 1997) baseline for comparison. The implementations[2] for both the Transformer and the LSTM are from PyTorch (Paszke et al., 2019).

We use one of the following two established tasks for each of the languages depending on which better enables learning for the data. We further elaborate on the reasoning for the choice of task for each language in Appendix A.

Binary Classification (Bidirectional Attention) Following Weiss et al. (2018), a model g is said to recognize a formal language $L \in \Sigma^*$ if $f(g(w)) = 1$ for all and only strings $w \in L$. In our case, g is a Transformer encoder, and $g(w)$ is the representation of a positive or negative example w, which is averaged from each symbol's encoder output for all symbols in w. f is a fully connected linear layer that maps the pooled representation to a real number, which is then passed through the sigmoid function to obtain the class label 0 or 1 using a threshold of 0.5. The loss is the BCE loss between the prediction and the target label.

Next Character Prediction (Unidirectional Attention) For languages in which training with positive and negative examples is ineffective because too few examples are available or the set of possible negative examples is too large, we use this task, which only requires positive examples. Given a valid prefix of a string in a language at each timestep, the model is tasked to predict the next set of acceptable symbols, or predicts `[EOS]` if the prefix is already in the language. To do this, the Transformer outputs for each symbol in the string

[1] However, their construction is not based on the language on which Shieber (1985) based his argument, and a CFG analysis might exist for their string language.

[2] The code for the experiments is available at: https://github.com/shunjiewang/mcsl-transformer

		Mildly Context-Sensitive		
	CFL	$L(\text{TAG})$	$L(\text{MG}) = L(\text{MCFG})$	
	LESS COMPLEX	CANONICAL	MORE COMPLEX	SCRAMBLE
Copying	$ww^{\mathcal{R}}$	ww	www	
Crossing Dependency	$\mathtt{a}^n\mathtt{b}^m\mathtt{c}^m\mathtt{d}^n$	$\mathtt{a}^n\mathtt{b}^m\mathtt{c}^n\mathtt{d}^m$		O_2
Multiple Agreements	$\mathtt{a}^n\mathtt{b}^n$	$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$ $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n$	$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n\mathtt{e}^n$	MIX

Table 1: Languages this work studies, organized by complexities and basic MCSL constructions each represents or resembles.

are passed in parallel through a linear layer and then the sigmoid function using a threshold of 0.5 to obtain k-hot vectors of dimension $|\Sigma \cup \{\texttt{[EOS]}\}|$, where each dimension represents whether the symbol is in the set of next characters at the timestep. We consider the prediction for a string to be correct if all predicted k-hot vectors are correct. The loss is the individual symbol's BCE loss between the predicted and the target k-hot vectors summed and then averaged. A look-ahead mask is applied to prevent self-attention from attending to later positions, which indirectly offers positional information (Irie et al., 2019; Bhattamishra et al., 2020a; Haviv et al., 2022), thus making PE optional for the languages we study using this task, and makes the architecture in this task essentially a Transformer decoder.

4.2 Data

Following the categorization in Ilie (1997), we are interested in three basic constructions that should be contained in MCSLs:

1. copying: ww
2. crossing dependency: $\mathtt{a}^n\mathtt{b}^m\mathtt{c}^n\mathtt{d}^m$
3. multiple agreements: $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$

All these languages are TAG-recognizable. We try to compare each of the three languages with a similar but less complex context-free language, as well as a similar but more complex MG-recognizable language. We also investigate two relevant scramble languages that are also felicitously MCFG-recognizable.

4.2.1 Copying

Copy language $\{ww \mid w \in \{\mathtt{a}, \mathtt{b}\}^*\}$ is in $L(\text{TAG})$ (Joshi, 1985). Its context-free counterpart is the palindrome $\{ww^{\mathcal{R}} \mid w \in \{\mathtt{a}, \mathtt{b}\}^*\}$, where $w^{\mathcal{R}}$ is the reverse of the string w. Joshi (1985) indicates double copy language www is not in $L(\text{TAG})$. However, any multiple copying w^k is in $L(\text{MG})$ (Jäger and Rogers, 2012). Thus, we study www as the simplest strictly MG-recognizable language for copying.

We use the binary classification setup for this family of languages. To generate the strings, we enumerate each possible w in our chosen $|w|$ range and then duplicate w to produce $ww^{\mathcal{R}}$, ww, and www. Negative examples are random strings sampled from $\{\mathtt{a}, \mathtt{b}\}^*$ in the same length range as the positive examples, but are not in the set of positive examples.

4.2.2 Crossing Dependency

Cross-serial dependency language $\mathtt{a}^n\mathtt{b}^m\mathtt{c}^n\mathtt{d}^m$ is in $L(\text{TAG})$ (Joshi, 1985). Its context-free counterpart is the nesting dependency language $\mathtt{a}^n\mathtt{b}^m\mathtt{c}^m\mathtt{d}^n$. We use the next character prediction task for these two languages because the potential set of negative examples is too large.

Following Gers and Schmidhuber (2001), to recognize nested $\mathtt{a}^n\mathtt{b}^m\mathtt{c}^m\mathtt{d}^n$, when the input is $\mathtt{a}$, the next valid character is $\mathtt{a}$ or $\mathtt{b}$. As soon as the input becomes $\mathtt{b}$, the value of n is determined, then the next valid symbol is now $\mathtt{b}$ or $\mathtt{c}$. Once the input becomes $\mathtt{c}$, the value of m is also determined, and the next characters are deterministic from this point on. Lastly, we output [EOS] as soon as the final symbol in the input is consumed. Following the notation in Suzgun et al. (2019), we denote the above described input-target scheme as the following, where $\dashv$ denotes [EOS]:

$$\mathtt{a}^n\mathtt{b}^m\mathtt{c}^m\mathtt{d}^n \rightarrow (\mathtt{a}/\mathtt{b})^n(\mathtt{b}/\mathtt{c})^m\mathtt{c}^{m-1}\mathtt{d}^n \dashv$$

		IN-DISTR.	OOD	OOD
		$\lvert w\rvert \in [1, 11]$	$\lvert w\rvert = 12$	$\lvert w\rvert = 13$
$ww^{\mathcal{R}}$	Transf.	**99.5**$_{\pm 0.3}$	50.4$_{\pm 0.3}$	50.2$_{\pm 0.1}$
	LSTM	97.8$_{\pm 0.5}$	**96.0**$_{\pm 0.7}$	**96.0**$_{\pm 0.8}$
ww	Transf.	**99.5**$_{\pm 0.1}$	51.3$_{\pm 0.3}$	50.2$_{\pm 0.0}$
	LSTM	97.2$_{\pm 0.4}$	**95.7**$_{\pm 1.1}$	**90.4**$_{\pm 1.4}$
www	Transf.	**99.5**$_{\pm 0.2}$	51.0$_{\pm 0.5}$	50.5$_{\pm 0.2}$
	LSTM	99.4$_{\pm 0.1}$	**98.6**$_{\pm 0.5}$	**87.5**$_{\pm 6.2}$

Table 2: Palindrome/Copy/2-Copy: Transformers surpass LSTMs for in-distribution tests but fall to random guesses for OOD (null accuracy for OOD is 50%).

Trivially, this scheme can be generalized to the crossing dependency language:

$$\mathtt{a}^n\mathtt{b}^m\mathtt{c}^n\mathtt{d}^m \rightarrow (\mathtt{a}/\mathtt{b})^n(\mathtt{b}/\mathtt{c})^m\mathtt{c}^{n-1}\mathtt{d}^m \dashv$$

4.2.3 Multiple Agreements

Being simple counter languages, the multiple agreements family had previously been extensively studied. We complement the related work by adding an MG-recognizable language, as well as giving additional analyses on the learned pattern.

Both $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$, $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n$ are TAG-recognizable. Their context-free counterpart is $\mathtt{a}^n\mathtt{b}^n$. Joshi (1985) indicates $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n\mathtt{e}^n$ is not in $L(\text{TAG})$, but Stabler (1997) shows that it is in $L(\text{MG})$. Moreover, $\sigma_1^n, ..., \sigma_k^n$ for any arbitrary k is MG-recognizable (Jäger and Rogers, 2012). Thus, we study $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n\mathtt{e}^n$ as the simplest strictly MG-recognizable language in this family.

Since the dataset is very small as only one positive example is available for each n, we use the next character prediction task, which requires fewer examples and is also an established setup for learning these languages. Gers and Schmidhuber (2001); Suzgun et al. (2019) have proposed the input-target schemes for $\mathtt{a}^n\mathtt{b}^n$, $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$, and $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n$:

$$\mathtt{a}^n\mathtt{b}^n \rightarrow (\mathtt{a}/\mathtt{b})^n\mathtt{b}^{n-1} \dashv$$
$$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \rightarrow (\mathtt{a}/\mathtt{b})^n\mathtt{b}^{n-1}\mathtt{c}^n \dashv$$
$$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n \rightarrow (\mathtt{a}/\mathtt{b})^n\mathtt{b}^{n-1}\mathtt{c}^n\mathtt{d}^n \dashv$$

Trivially, this scheme can be generalized to the MG-recognizable $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n\mathtt{e}^n$:

$$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n\mathtt{e}^n \rightarrow (\mathtt{a}/\mathtt{b})^n\mathtt{b}^{n-1}\mathtt{c}^n\mathtt{d}^n\mathtt{e}^n \dashv$$

That is, the next valid character is a or b as long as the input is a, but once b occurs in the input, n will be determined and the next characters will be deterministic from this point on.

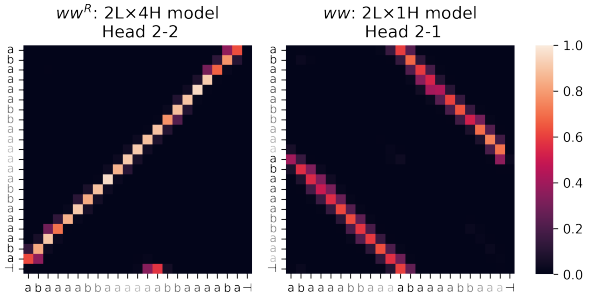


Figure 3: Anti-diagonal alignment for $ww^{\mathcal{R}}$, and forward and backward alignment for ww.

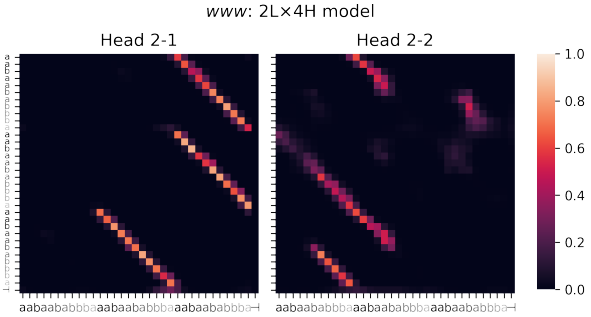


Figure 4: www expects six alignments in total, usually distributed across heads, where half are better aligned, and the other half partially aligned.

4.2.4 Scramble Languages

Joshi (1985) argued that MCSGs should only handle limited cross-serial dependency like the type in Dutch (Bresnan et al., 1982) and Swiss German, but not as in MIX $= \{w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^* \mid \lvert w\rvert_{\mathtt{a}} = \lvert w\rvert_{\mathtt{b}} = \lvert w\rvert_{\mathtt{c}}\}$, where $\lvert w\rvert_\sigma$ denotes the number of occurrences of symbol σ in string w, that is, the language of strings with an equal number of a's, b's, and c's, but the symbols can occur in any order, and can thus be seen as scrambled $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$.

MIX resembles an extreme case of free word order and is not recognizable by the TAG (Kanazawa and Salvati, 2012). However, it turned out that the language is in $L(\text{MCFG})$ (Salvati, 2015). A related language $O_2 = \{w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}^* \mid \lvert w\rvert_{\mathtt{a}} = \lvert w\rvert_{\mathtt{c}} \wedge \lvert w\rvert_{\mathtt{b}} = \lvert w\rvert_{\mathtt{d}}\}$, which can be seen as scrambled $\mathtt{a}^n\mathtt{b}^m\mathtt{c}^n\mathtt{d}^m$, is also in $L(\text{MCFG})$ (Salvati, 2015).

We investigate the two scramble languages using the binary classification task as the model may benefit from seeing the whole string at once to directly model the occurrences of each symbol. For MIX, the positive examples exhaustively enumerate all permutations of $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$ in the chosen n range, and the negative examples enumerate the remaining strings in $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^*$ within the same range

to help the model better eliminate most wrong hypotheses. As for O_2, we enumerate permutations of $\mathtt{a}^n\mathtt{b}^m\mathtt{c}^n\mathtt{d}^m$ in the chosen n, m range, and the negative examples are from the remaining strings in $\{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}^*$ that are in the same range. Negative examples for both languages also include strings where $|w|_\sigma = 0$. The train-test split is performed over each sequence length separately rather than over the entire dataset, so strings of different lengths appear in all splits.

Table 1 summarizes all languages studied in this work, and we also include detailed dataset statistics in Appendix B.

5 Experiments

Each model is evaluated on three sets: an in-distribution held-out test set, an out-of-distribution (henthforth OOD) set with strings longer than the ones seen during training, and a second OOD set with even longer strings. We report the mean and standard deviation of test accuracies in three runs with different random seeds.

We then visualize the heads from our best-performing runs that most clearly demonstrate highly interpretable patterns upon visual inspection, but note that all visualized patterns do recur across different configurations.

5.1 Copying

Our Transformer models learned ww and the related $ww^\mathcal{R}, www$ with high accuracy and outperformed LSTMs in in-distribution tests. However, in the two OOD tests, only LSTMs were able to extrapolate, while the accuracies of the Transformers are close to random guesses (Table 2).

We identified certain heads that align the substrings in different diagonalities to measure the similarity of a string to itself (Figure 3). For ww, the gold alignment is to align the first w against the second and vice versa. In the visualized run, we find that among all positive examples in the test set, 93.4% of the time the highest query-key attention is on the gold alignment. For $ww^\mathcal{R}$, the gold alignment expects the head of a substring to attend to the tail of the other substring, thus resulting in an anti-diagonal pattern, and 94.8% of the time the highest attention is on the gold alignment diagonals among all positive examples in the test set.

For www, a gold alignment requires each substring to attend to the other two, resulting in a total of six alignments, which we did get during training as shown in Figure 4. The six alignments are distributed across heads in a multi-head model, where only three of the six are better aligned, while the other partial alignments appear to be auxiliary if were at all useful for inference. In the visualized model, the three clearer alignments in the first head on average match the gold alignment 86.1% of the time over positive examples in the test set, while the accuracy is 39.0% for the other three partial alignments.

		IN-DISTR.	OOD	OOD
		$n, m \in [1, 50]$	n or $m \in [51, 100]$	n or $m \in [101, 150]$
$\mathtt{a}^n\mathtt{b}^m\mathtt{c}^m\mathtt{d}^n$	Tr.+PE	$99.8_{\pm 0.2}$	$6.5_{\pm 1.3}$	$0.0_{\pm 0.0}$
	Tr.−PE	$\mathbf{100.0}_{\pm 0.0}$	$98.0_{\pm 0.3}$	$23.0_{\pm 3.1}$
	LSTM	$\mathbf{100.0}_{\pm 0.0}$	$\mathbf{100.0}_{\pm 0.0}$	$\mathbf{100.0}_{\pm 0.0}$
$\mathtt{a}^n\mathtt{b}^m\mathtt{c}^n\mathtt{d}^m$	Tr.+PE	$\mathbf{100.0}_{\pm 0.0}$	$7.2_{\pm 1.6}$	$0.0_{\pm 0.0}$
	Tr.−PE	$\mathbf{100.0}_{\pm 0.0}$	$92.3_{\pm 1.2}$	$27.0_{\pm 14.0}$
	LSTM	$\mathbf{100.0}_{\pm 0.0}$	$\mathbf{99.2}_{\pm 1.3}$	$\mathbf{81.3}_{\pm 12.3}$

Table 3: Nesting/Crossing: not using sinusoidal PE helped with extrapolation. Note that the two OOD sets include both strings where only one of n, m is OOD, e.g., $\mathtt{a}^1\mathtt{b}^{100}\mathtt{c}^1\mathtt{d}^{100}$, and strings where both n, m are OOD, e.g., $\mathtt{a}^{51}\mathtt{b}^{100}\mathtt{c}^{51}\mathtt{d}^{100}$.

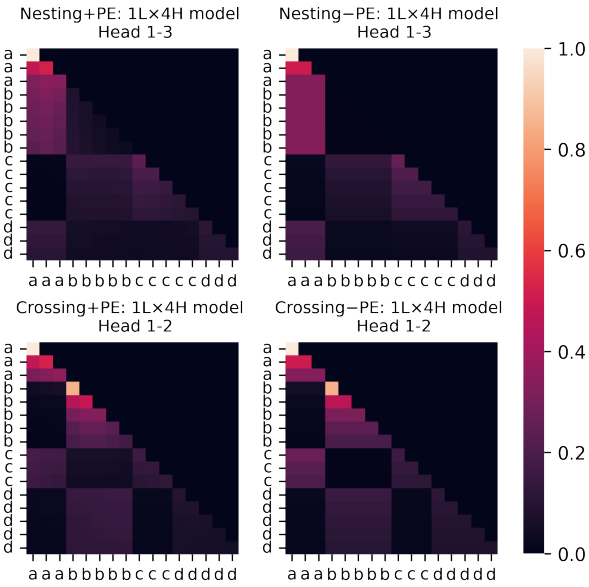


Figure 5: Crossing shows a checkerboard pattern as the result of correctly identifying pairwise dependents, while nesting has a similar pattern except for different symbol dependencies.

5.2 Crossing Dependency

The in-distribution tests were solved almost perfectly by all three model setups, including a Transformer decoder setup where we remove PE and

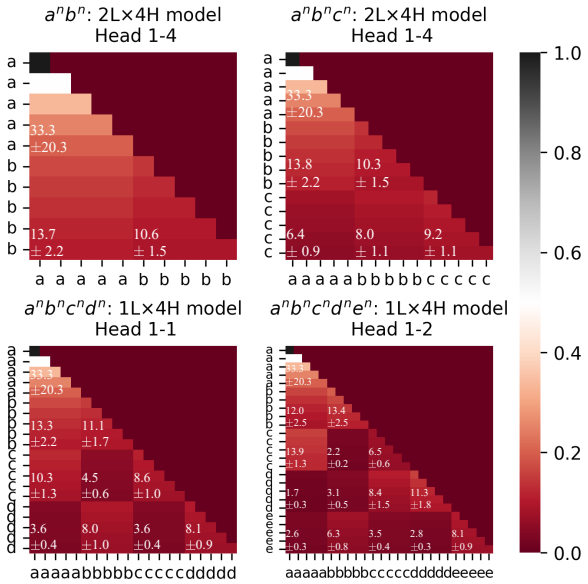


Figure 6: Every occurrence of a query symbol attends to every occurrence of a key symbol using similar attention values, thus low variance among the values. The query symbols also attend to different key symbols with different weights, thus resulting in a grid pattern.

		IN-DISTR.	OOD	OOD
		$n \in [1, 50]^*$	$n \in [51, 100]$	$n \in [101, 150]$
$\mathtt{a}^n\mathtt{b}^n$	Tr.−PE	**100.0**$_{\pm 0.0}$	**100.0**$_{\pm 0.0}$	91.3$_{\pm 8.4}$
	LSTM	**100.0**$_{\pm 0.0}$	**100.0**$_{\pm 0.0}$	**100.0**$_{\pm 0.0}$
$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$	Tr.−PE	**100.0**$_{\pm 0.0}$	**100.0**$_{\pm 0.0}$	36.0$_{\pm 14.2}$
	LSTM	**100.0**$_{\pm 0.0}$	**100.0**$_{\pm 0.0}$	**100.0**$_{\pm 0.0}$
$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n$	Tr.−PE	**100.0**$_{\pm 0.0}$	**100.0**$_{\pm 0.0}$	24.0$_{\pm 10.2}$
	LSTM	**100.0**$_{\pm 0.0}$	**100.0**$_{\pm 0.0}$	48.7$_{\pm 13.6}$
$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n\mathtt{e}^n$	Tr.−PE	**100.0**$_{\pm 0.0}$	85.3$_{\pm 15.4}$	3.3$_{\pm 4.7}$
	LSTM	**100.0**$_{\pm 0.0}$	**100.0**$_{\pm 0.0}$	**100.0**$_{\pm 0.0}$

Table 4: Multiple Agreements: Transformers without PE demonstrated the ability to extrapolate, though in general still not as good or consistent as LSTMs. *The in-distribution set is held-out and only has strings with $n \in \{5, 15, 25, 35, 45\}$.

rely only on indirect positional information from the causal mask. For OOD tests, we evaluate models on strings where both n, m are OOD, as well as strings in which only one of n, m is OOD. We find that removing PE and relying only on causal mask helped with the Transformer's extrapolation, which is consistent with findings in Bhattamishra et al. (2020a) on other languages. However, such ability is still worse than that of LSTMs (Table 3).

Figure 5 shows that the Transformer's attention formed a checkerboard pattern for recognizing crossing, as the result of each symbol in the query attends to every occurrence of itself and its dependent in the key but not to the non-dependents. As for nesting, the pattern is very similar except that the dependency relations are different. Models trained with or without sinusoidal PE end up learning very similar patterns, except that without PE, the attention from one query symbol to every occurrence of a key symbol is uniformly distributed, resulting in a stack of color bands on the attention map of the visualized head.

The attention maps suggest that in the optimal case, each symbol in query identifies to which other symbol in key it is pairwise dependent, and then in the visible portion without look-ahead mask, distributes its attention to every occurrence of itself and the dependent, and gives zero attention to the other pair. We measure as an example how accurately the visualized head in the crossing model without PE has implemented this optimum, and we find that across all in-distribution test set datapoints, keys that expect 100% of the attention weights from each symbol in query have received on average 93.0% of the attention weights.

5.3 Multiple Agreements

We follow the established finding in Bhattamishra et al. (2020a) and only consider the Transformer decoder without sinusoidal PE for these languages, as training with PE was ineffective in pilot experiments. Transformers without PE demonstrated the ability to extrapolate, although on average they are still not as good or consistent as LSTMs (Table 4).

We annotate the mean and standard deviation in percentages among all attention values from one query symbol to one key symbol in Figure 6. It can be observed that every input alphabet symbol attends to different symbols in the output alphabet using a different weight, but the attention values to each occurrence of the same symbol are similar, and thus have low variance, culminating in the grid pattern we see on the maps. The differences in weights from a query symbol suggested a particular dependency analysis learned by that run.

Unlike what we have discussed for copying and crossing, the multiple agreements strings do not have a definite dependency relation to learn, and many possible analyses exist for the same string, e.g., any two symbols in the alphabet could be pairwise dependent, or all symbols in the alphabet could be dependent on each other (cf. Joshi (1985)). Thus, from run to run, to which key sym-

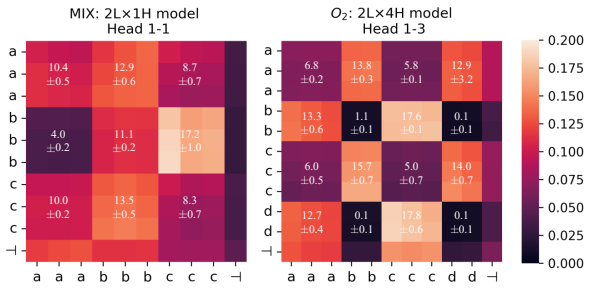


Figure 7: MIX resembles multiple agreements in that the attention weights from one query symbol to one key symbol are similar. O_2 resembles the checkerboard in crossing although it does not ignore non-dependents.

bol the query symbol gives the most attention, and how much attention is given to each symbol, could indeed vary. Also, the lack of a gold analysis has determined that the model cannot simply focus on some of the symbols, and it is crucial for every symbol to attend to every symbol as we see here.

5.4 Scramble Languages

We use the macro F-1 score as the metric for this set of languages, since our data generation is skewed towards negative examples. Despite the seeming complexity of the data, Transformers are able to solve MIX and O_2 in-distribution test sets perfectly, while LSTMs also have very high scores. However, the MIX OOD sets are challenging for both models, while LSTMs outperformed Transformers in solving the OOD sets for O_2 (Table 5).

Since $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \subset$ MIX and $\mathtt{a}^n\mathtt{b}^m\mathtt{c}^n\mathtt{d}^m \subset O_2$, we use unscrambled strings in the visualizations for readability in Figure 7. MIX has a pattern that resembles the one in multiple agreements in that the amount of attention from one symbol in query to one symbol in key among all occurrences is similar and has low variance, as we annotated in the visualized example. Similarly, O_2 has a pattern that resembles the checkerboard in crossing, although it is not the case that the queries ignore non-dependents. However, it is still evident that each query in O_2 identified which two symbols should form pairwise dependents and used similar attention weights to the pair. Do note that although we visualized the unscrambled strings for readability, the similar attention and low variance properties hold for other scrambled strings.

As an additional analysis, we probe the MIX representations to see what information is encoded. One possibility is the count for the symbol occurrences at each timestep, which directly follows from MIX's definition. We decode the MIX embeddings for a full counting target that maintains the ongoing tallies for all 3 symbols, as illustrated in Table 6. We also include a control task (Hewitt and Liang, 2019) target that is the random shuffle of the counting target, and if the probing model trained on this target has a higher error, we would be more confident that the count is actually present in the representations, rather than the counting results following from the power of the probing model.

Similar to the methodology in Wallace et al. (2019), we used an MLP regressor prober with 1 hidden layer, ReLU activations, and MSE loss. We trained the prober for up to 300 epochs with early stopping. On the in-distribution test set, the prober using the counting target has an MSE of 0.21 and a Pearson correlation of 0.929 between the target and the predicted count values. This contrasts with the control task target which has an MSE of 1.33. We find this to be suggestive that the learned representations contain count information, which may have been useful for solving scramble languages.

		IN-DISTR.	OOD	OOD
		$\lvert w\rvert_\sigma \in [1,4]$	$\lvert w\rvert_\sigma = 5$	$\lvert w\rvert_\sigma = 6$
MIX	Transf.	**100.0**$_{\pm 0.0}$	65.6$_{\pm 2.9}$	45.7$_{\pm 6.3}$
	LSTM	**100.0**$_{\pm 0.0}$	70.3$_{\pm 10.5}$	49.0$_{\pm 15.5}$
		$\lvert w\rvert_\sigma \in [1,3]$	$\lvert w\rvert_\sigma \in [1,4]$	$\lvert w\rvert_\sigma \in [1,5]$
O_2	Transf.	**100.0**$_{\pm 0.0}$	60.5$_{\pm 8.5}$	45.1$_{\pm 10.1}$
	LSTM	**100.0**$_{\pm 0.0}$	**100.0**$_{\pm 0.0}$	**98.6**$_{\pm 0.4}$

Table 5: Scramble macro F-1 (%): the models performed perfectly for in-distribution tests, but MIX OOD sets are challenging to both models, whereas LSTMs outperformed Transformers for O_2 OOD sets. Note that we made sure that the three tests for O_2 do not share datapoints when generating them.

	Counting Target			Control Task		
	#a	#b	#c	#a	#b	#c
a	[1	0	0]	[2	2	2]
b	[1	1	0]	[1	0	0]
c	[1	1	1]	[2	1	1]
a	[2	1	1]	[1	1	0]
b	[2	2	1]	[1	1	1]
c	[2	2	2]	[2	2	1]

Table 6: Target examples for $w =$ abcabc. The counting target is the count of each symbol at each timestep; the control task target is the random shuffle of the counting target.

6 Discussion and Conclusion

We empirically studied the Transformer's ability to learn a variety of linguistically significant MCSLs. The significance of the languages is two-fold: they represent a hypothesized upper bound for the complexity of natural language, and they are the abstractions of the motivating linguistic phenomena. Overall, the Transformers performed well in in-distribution tests and are comparable to LSTMs, but their ability to extrapolate is limited. In our next character prediction experiments with Transformer decoders, removing the sinusoidal PE alleviated the problem, which is an established empirical finding for some formal languages and natural language, but this technique is not always generalizable to other data, nor does it work in the encoder since the decoder can rely on the indirect positional information from the causal mask in absence of PE.

Transformers leveraged the attention mechanism to score the similarity between the substrings. In our analyses, the learned self-attention's alignments often reflect the symbol dependency relations within the string, which had been useful for MCSLs because of the rich and complex dependencies in the languages. In a more complex language like MIX, Transformers had implicitly learned some form of counting behavior that may have helped solve the language.

Within the same family of languages spanning across complexity classes, the learned patterns are similar and no significant differences in behaviors are observed in the reduced or added complexity languages. This may suggest that we cannot draw parallels between the MCSG formalisms and the Transformer's expressiveness directly, like some other formal models such as circuits (Hao et al., 2022; Merrill et al., 2022) do. However, this work serves as an example of how we may draw inspiration from the rich MCSL scholarships to motivate work in NLP, as they help us examine the linguistic capacity of current and future NLP models.

Limitations

An empirical study on formal language learning is always inconveniently insufficient, as there is always some string length upper bound that any experiment can get to or reasonably work with, so any conclusions drawn are based on an unintentionally bounded dataset, which could weaken the argument about learnability in general as the dataset might form a language with reduced complexity.

In addition, the roles of the other heads, the feed-forward sublayer, etc. are not investigated. Therefore, we cannot definitively say how self-attention directly contributed to inference, despite learning meaningful and interpretable patterns (cf. Wen et al. (2023)).

Ideally, we would complement the empirical findings with theoretical constructions on whether and how the MCSLs can be learned, which is lacking in the current work. However, the empirical results serve as the foundation towards that goal. Especially, the highly interpretable self-attention patterns could inspire us and hint at what the theoretical constructions would look like.

References

Tilman Becker, Aravind K. Joshi, and Owen Rambow. 1991. Long-distance scrambling and Tree Adjoining Grammars. In *Fifth Conference of the European Chapter of the Association for Computational Linguistics*, Berlin, Germany. Association for Computational Linguistics.

Jean-Philippe Bernardy, Adam Ek, and Vladislav Maraev. 2021. Can the transformer learn nested recursion with symbol masking? In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 753–760, Online. Association for Computational Linguistics.

Rajesh Bhatt and Aravind Joshi. 2004. Semilinearity is a syntactic invariant: A reply to Michaelis and Kracht 1997. *Linguistic Inquiry*, 35(4):683–692.

Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. 2020a. On the Ability and Limitations of Transformers to Recognize Formal Languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7096–7116, Online. Association for Computational Linguistics.

Satwik Bhattamishra, Arkil Patel, and Navin Goyal. 2020b. On the computational power of transformers and its implications in sequence modeling. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 455–475, Online. Association for Computational Linguistics.

Joan Bresnan, Ronald M. Kaplan, Stanley Peters, and Annie Zaenen. 1982. Cross-serial dependencies in Dutch. *Linguistic Inquiry*, 13(4):613–635.

David Chiang and Peter Cholak. 2022. Overcoming a theoretical limitation of self-attention. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7654–7664, Dublin, Ireland. Association for Computational Linguistics.

David Chiang, Peter Cholak, and Anand Pillay. 2023. Tighter bounds on the expressivity of transformer encoders. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 5544–5562. PMLR.

Noam Chomsky. 1956. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124.

Noam Chomsky. 1959. On certain formal properties of grammars. *Information and Control*, 2(2):137–167.

Alexander Clark and Ryo Yoshinaka. 2012. Beyond semilinearity: Distributional learning of parallel multiple context-free grammars. In *International Conference on Grammatical Inference*, pages 84–96. PMLR.

Christopher Culy. 1985. *The Complexity of the Vocabulary of Bambara*, pages 349–357. Springer Netherlands, Dordrecht.

Gregoire Deletang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, and Pedro A Ortega. 2023. Neural networks and the chomsky hierarchy. In *The Eleventh International Conference on Learning Representations*.

Javid Ebrahimi, Dhruv Gelda, and Wei Zhang. 2020. How can self-attention networks recognize Dyck-n languages? In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4301–4306, Online. Association for Computational Linguistics.

Felix A Gers and Jürgen Schmidhuber. 2001. LSTM recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks*, 12(6):1333–1340.

Thomas Graf. 2021. Minimalism and computational linguistics. Lingbuzz/005855.

Michael Hahn. 2020. Theoretical Limitations of Self-Attention in Neural Sequence Models. *Transactions of the Association for Computational Linguistics*, 8:156–171.

Yiding Hao, Dana Angluin, and Robert Frank. 2022. Formal language recognition by hard attention transformers: Perspectives from circuit complexity. *Transactions of the Association for Computational Linguistics*, 10:800–810.

Adi Haviv, Ori Ram, Ofir Press, Peter Izsak, and Omer Levy. 2022. Transformer language models without positional encodings still learn positional information. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1382–1390, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.

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

Lucian Ilie. 1997. On computational complexity of contextual languages. *Theoretical Computer Science*, 183(1):33–44. Formal Language Theory.

Kazuki Irie, Albert Zeyer, Ralf Schlüter, and Hermann Ney. 2019. Language Modeling with Deep Transformers. In *Proc. Interspeech 2019*, pages 3905–3909.

Gerhard Jäger and James Rogers. 2012. Formal language theory: refining the chomsky hierarchy. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 367(1598):1956–1970.

Aravind K. Joshi. 1985. *Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions?*, Studies in Natural Language Processing, page 206–250. Cambridge University Press.

Laura Kallmeyer. 2010. *Parsing Beyond Context-Free Grammars*, 1st edition. Springer Publishing Company, Incorporated.

Makoto Kanazawa and Sylvain Salvati. 2012. MIX is not a tree-adjoining language. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 666–674, Jeju Island, Korea. Association for Computational Linguistics.

Gregory Michael Kobele. 2006. *Generating copies: An investigation into structural identity in language and grammar*. Ph.D. thesis, University of California, Los Angeles.

Konstantinos Kogkalidis and Gijs Wijnholds. 2022. Discontinuous constituency and BERT: A case study of Dutch. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3776–3785, Dublin, Ireland. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

William Merrill and Ashish Sabharwal. 2022. A Logic for Expressing Log-Precision Transformers. *arXiv e-prints*, page arXiv:2210.02671.

William Merrill, Ashish Sabharwal, and Noah A. Smith. 2022. Saturated transformers are constant-depth threshold circuits. *Transactions of the Association for Computational Linguistics*, 10:843–856.

Jens Michaelis and Marcus Kracht. 1997. Semilinearity as a syntactic invariant. In *Logical Aspects of Computational Linguistics*, pages 329–345, Berlin, Heidelberg. Springer Berlin Heidelberg.

Benjamin Newman, John Hewitt, Percy Liang, and Christopher D. Manning. 2020. The EOS decision and length extrapolation. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 276–291, Online. Association for Computational Linguistics.

Isabel Papadimitriou and Dan Jurafsky. 2023. Pretrain on just structure: Understanding linguistic inductive biases using transfer learning. *arXiv e-prints*, page arXiv:2304.13060.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Jorge Pérez, Pablo Barceló, and Javier Marinković. 2021. Attention is turing-complete. *Journal of Machine Learning Research*, 22(75):1–35.

Jorge Pérez, Javier Marinković, and Pablo Barceló. 2019. On the turing completeness of modern neural network architectures. In *International Conference on Learning Representations*.

Daniel Radzinski. 1991. Chinese number-names, tree adjoining languages, and mild context-sensitivity. *Computational Linguistics*, 17(3):277–300.

Sylvain Salvati. 2015. MIX is a 2-MCFL and the word problem in $\mathbb{Z}^2$ is captured by the IO and the OI hierarchies. *Journal of Computer and System Sciences*, 81(7):1252–1277.

Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191–229.

Stuart M. Shieber. 1985. *Evidence Against the Context-Freeness of Natural Language*, pages 320–334. Springer Netherlands, Dordrecht.

Edward Stabler. 1997. Derivational minimalism. In *Logical Aspects of Computational Linguistics*, pages 68–95, Berlin, Heidelberg. Springer Berlin Heidelberg.

Edward P. Stabler. 2011. Computational perspectives on Minimalism. In Cedric Boeckx, editor, *The Oxford Handbook of Linguistic Minimalism*, pages 617–643. Oxford University Press, Oxford.

Mark Steedman. 1987. Combinatory grammars and parasitic gaps. *Natural Language and Linguistic Theory*, 5(3):403–439.

Mirac Suzgun, Yonatan Belinkov, and Stuart M. Shieber. 2019. On evaluating the generalization of LSTM models in formal languages. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2019*, pages 277–286.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

K. Vijay-Shanker, David J. Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *25th Annual Meeting of the Association for Computational Linguistics*, pages 104–111, Stanford, California, USA. Association for Computational Linguistics.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do NLP models know numbers? probing numeracy in embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5307–5315, Hong Kong, China. Association for Computational Linguistics.

Shunjie Wang. 2021. Evaluating transformer's ability to learn mildly context-sensitive languages. Master's thesis, University of Washington.

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2018. On the practical computational power of finite precision RNNs for language recognition. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 740–745, Melbourne, Australia. Association for Computational Linguistics.

Kaiyue Wen, Yuchen Li, Bingbin Liu, and Andrej Risteski. 2023. (Un) interpretability of transformers: a case study with dyck grammars.

Shunyu Yao, Binghui Peng, Christos Papadimitriou, and Karthik Narasimhan. 2021. Self-attention networks can process bounded hierarchical languages. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3770–3785, Online. Association for Computational Linguistics.

A Additional Experiment Details

Choice of Task In pilot experiments, training crossing and multiple agreements with the binary classification task was unsuccessful, and our analyses suggest that they learned spurious statistical

	Train		Dev		Test		OOD-1		OOD-2	
	POS	NEG	POS	NEG	POS	NEG	POS	NEG	POS	NEG
	$\|w\| \in [1, 11]$						$\|w\| = 12$		$\|w\| = 13$	
$ww^{\mathcal{R}}$	2858	2863	609	617	624	603	4096	4096	8192	8192
ww	2873	2847	622	603	592	635	4095	4095	8192	8192
www	2894	2836	607	621	592	637	4096	4096	8192	8192
	$n, m \in [1, 50]$						n or $m \in [51, 100]$		n or $m \in [101, 150]$	
$\mathtt{a}^n\mathtt{b}^m\mathtt{c}^m\mathtt{d}^n$	1750	—	375	—	375	—	7500	—	12500	—
$\mathtt{a}^n\mathtt{b}^m\mathtt{c}^n\mathtt{d}^m$	1750	—	375	—	375	—	7500	—	12500	—
	$n \in [1, 50]$						$n \in [51, 100]$		$n \in [101, 150]$	
$\mathtt{a}^n\mathtt{b}^n$	40	—	5	—	5	—	50	—	50	—
$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$	40	—	5	—	5	—	50	—	50	—
$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n$	40	—	5	—	5	—	50	—	50	—
$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n\mathtt{e}^n$	40	—	5	—	5	—	50	—	50	—
	$\|w\|_\sigma \in [1, 4]$						$\|w\|_\sigma = 5$		$\|w\|_\sigma = 6$	
MIX	25497	51645	5462	11034	5467	11145	756756	408111	756756	960501
	$\|w\|_\sigma \in [1, 3]$						$\|w\|_\sigma \in [1, 4]$		$\|w\|_\sigma \in [1, 5]$	
O_2	56518	289590	12111	62006	12115	62156	44100	1135444	56700	3030880

Table 7: Dataset statistics and label distribution. The next character prediction task uses only positive examples. OOD sets for scramble languages are downsampled. Negative examples for scramble languages also include strings where $|w|_\sigma = 0$.

cues. The task was especially difficult for multiple agreements, where only one example is available for each n. For crossing, we tried to use an equal number of positive and negative examples, but that is not enough for the model to rule out alternative wrong hypotheses. On the other hand, if we follow what we did for O_2 and enumerate all possible negative examples in a length range, since crossing has a much wider length range than O_2, this will lead to an explosion of negative examples and is impractical to work with. Teaching these two sets of languages with the binary classification setup may still be possible, but the negative examples likely need to be carefully curated, so that we avoid an explosion of negative examples over positive examples, but still have enough negative datapoints to help the model eliminate most wrong hypotheses such as the spurious cue ones.

The `[EOS]` Decision In the binary classification task, since we are not scoring or generating a string, the decision on whether to add `[EOS]` to strings is arbitrary. Newman et al. (2020) suggest that without `[EOS]`, models may extrapolate better to longer strings. We tried the setup without `[EOS]` in pilot experiments but found no significantly better performance for our Transformer models in the studied languages. We chose to include `[EOS]` and use the `[EOS]` embeddings as the sentence representations for LSTMs in this task.

	Learning Rate	Min. Delta	Max. Epochs	Patience
Copying	{1e-3, 5e-4, 1e-4}	1e-4	200	20
Crossing-LSTM	{1e-2, 5e-2, 1e-3}	1e-4	150	20
Crossing-Transf.	{1e-3, 5e-4, 1e-4}	1e-4	150	20
Multiple-LSTM	{1e-2, 5e-2, 1e-3}	1e-4	3000	400
Multiple-Transf.	{1e-3, 5e-4}	1e-4	3000	400
MIX	{1e-3, 5e-4, 1e-4}	1e-5	50	5
O_2	{5e-4, 3e-4, 1e-4}	1e-5	50	5

Table 8: Search space and choices for certain hyperparameters per language.

B Training Details

The datasets for all languages are generated exactly once, and are used across hyperparameter tuning and the final experiments. We record the random seeds used for generation for reproducibility. We try to enumerate all examples in our chosen length range except for the OOD sets of the scramble languages because of an explosion of datapoints as strings get longer, and we downsample in this case. For MIX, positive examples are capped at 756756; negative examples for a given string length are capped at 25200. For O_2, examples for a given string length are capped at 6300.

We then do a 70%-15%-15% Train-Dev-Test random split of the data. This is performed over

	LSTM		Transformer			
	d_{model}	LR	d_{model}	LR	#layers	#heads
$ww^{\mathcal{R}}$	64	1e-4	64	5e-4	2	4
ww	64	5e-4	32	5e-4	2	1
www	64	1e-4	64	5e-4	2	4
$\mathtt{a}^n\mathtt{b}^m\mathtt{c}^m\mathtt{d}^n$	64	5e-2	64	1e-3	1	4
$\mathtt{a}^n\mathtt{b}^m\mathtt{c}^n\mathtt{d}^m$	64	1e-2	64	1e-3	1	4
$\mathtt{a}^n\mathtt{b}^n$	64	1e-2	64	5e-4	2	4
$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$	64	1e-2	64	5e-4	2	4
$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n$	64	1e-2	64	5e-4	1	4
$\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n\mathtt{d}^n\mathtt{e}^n$	64	5e-2	64	5e-4	1	4
MIX	64	5e-4	64	1e-4	2	1
O_2	64	5e-4	64	5e-4	2	4

Table 9: Final hyperparameters used in experiments per language.

the entire dataset, except for scramble languages, in which we split by each string length separately since the dataset is skewed towards longer strings as a result of enumerating all distinct permutations for a given string length. For multiple agreements languages, since the dataset is too small to be randomly split, we manually picked strings with $n \in \{5, 15, 25, 35, 45\}$ as the test set and strings with $n \in \{6, 16, 26, 36, 46\}$ as the development set. We give detailed statistics of the dataset size and the label distribution in Table 7.

We first tune the hyperparameters for each of the languages on both the Transformer and the LSTM. All models tune d_{model} in $\{16, 32, 64\}$, and for Transformers, we also tune the number of layers in $\{1, 2\}$ and the number of heads in $\{1, 2, 4\}$. The search space for the learning rate per language is in Table 8. We tuned the hyperparameters using grid search and picked the configuration with the lowest development loss for each language. The final set of hyperparameters is in Table 9. For both tuning and training, we also use early stopping with manually selected minimum delta, patience, and the maximum number of epochs as shown in Table 8. All experiment runs used a batch size of 32, and the AdamW optimizer (Loshchilov and Hutter, 2019). The embedding layer of the Transformer is initialized using a uniform distribution in the range $[-0.1, 0.1]$.