GCG-Based Artificial Languages for Evaluating Inductive Biases of Neural Language Models

Nadine El-Naggar **Tatsuki Kuribayashi** **Ted Briscoe**
Mohamed bin Zayed University of Artificial Intelligence
{nadine.naggar, tatsuki.kuribayashi, ted.briscoe}@mbzuai.ac.ae

Abstract

Recent work has investigated whether extant neural language models (LMs) have an inbuilt inductive bias towards the acquisition of attested typologically-frequent grammatical patterns as opposed to infrequent, unattested, or impossible patterns using artificial languages (White and Cotterell, 2021; Kuribayashi et al., 2024). The use of artificial languages facilitates isolation of specific grammatical properties from other factors such as lexical or real-world knowledge, but also risks oversimplification of the problem.

In this paper, we examine the use of Generalized Categorial Grammars (GCGs) (Wood, 2014) as a general framework to create artificial languages with a wider range of attested word order patterns, including those where the subject intervenes between verb and object (VSO, OSV) and unbounded dependencies in object relative clauses. In our experiments, we exemplify our approach by extending White and Cotterell (2021) and report some significant differences from existing results.

1 Introduction

Attested natural languages (NLs) often have different grammatical properties, such as different word orders, so it is reasonable to ask whether neural language models (LMs) have inductive biases towards specific properties, including different patterns of word order. There are thousands of NLs which differ along multiple semi-independent lexical and grammatical dimensions, so it is difficult to isolate specific properties to evaluate LMs' inductive biases using natural data (Mielke et al., 2019). To remedy this, artificial languages (ALs) have been used in order to create more controlled experiments. Researchers have designed ALs of varying complexities, ranging from lexically-simple but syntactically-complex formal languages, such as the irreducibly context-free Dyck languages or irreducibly indexed (mildly context-sensitive) languages such as cross-serial dependencies ($a^n b^n (c^n)$) (Hewitt et al., 2020), to putatively impossible languages based on permutations of English examples (Kallini et al., 2024).

White and Cotterell (2021) prioritise control of word order in their research. They generate ALs using a Probabilistic Context Free Grammar (PCFG), and use 6 parameters to reorder words and phrases to create 64 ALs with the same lexicon, with the aim of determining whether LMs exhibit an inductive bias towards specific orders. The same dataset of ALs is used by Kuribayashi et al. (2024) to explore a wider range of neural LMs. However, the use of a PCFG precludes the handling of (mildly) context-sensitive NL constructions and does not support a fully general account of unbounded filler-gap dependencies (Steedman, 1996). Furthermore, the use of a VP constituent in the base PCFG means Verb-Subject-Object (VSO) and OSV base orders cannot be represented in the languages created by White and Cotterell (2021).

We create a larger set of ALs that can be used to further test LMs for word order inductive biases covering a wider range of word orders. Specifically, we cover VSO and OSV orders, which represent approximately 8% of attested NLs according to typologists (Dryer and Haspelmath, 2013). Furthermore, we develop an extensible approach to defining ALs that supports the inclusion of mildly context-sensitive (indexed language) constructions, such as cross-serial dependencies, and a general approach to unbounded filler-gap dependencies. We introduce object relative clauses as one exemplar of an unbounded dependency into our extended dataset of ALs. We empirically test LMs on our artificial languages and find significant differences in results compared to existing studies (White and Cotterell, 2021; Kuribayashi et al., 2024), for example, a clearer preference of Transformers for subject-before-verb word orders. This suggests that using more complex, but arguably naturalistic ALs leads to rather different conclusions about the inductive bias of neural LMs

2 Background

2.1 Artificial languages

One line of research has used ALs to evaluate LMs capacity to learn ALs at different levels of the Chomsky hierarchy. Someya et al. (2024) use ALs to determine whether LMs can learn the properties of regular, context-free, and context-sensitive languages, such as nested and long-distance dependencies, and cross-serial dependencies. They find that LSTMs (Hochreiter and Schmidhuber, 1997), Stack-RNNs (Joulin and Mikolov, 2015), and Transformers (Vaswani et al., 2017) struggle to learn nested, long-distance, and cross-serial dependencies, but successfully learn regular languages. Other context-free languages, such as Dyck languages, and mildly context-sensitive languages, like $a^n b^n c^n$, have been used to test recurrent LM learning and generalization to longer sequences (Suzgun et al., 2019; Weiss et al., 2018; El-Naggar et al.) as well as establishing a correspondence between the different LM models and the levels of the Chomsky hierarchy (Delétang et al., 2023). One limitation of this research is that the ALs used diverge from NLs by using minimal vocabulary, many levels of nested dependencies, and so forth.

In another line of research, Chomsky et al. (2023) argued that neural LMs can learn both possible and impossible human languages, so cannot distinguish between them. Kallini et al. (2024) empirically address this claim, by developing putatively impossible AL variants by permutation and modification of an English dataset, following Ravfogel et al. (2019). They find that GPT-2 models struggle to learn the impossible languages, contradicting Chomsky's claim. However, it is difficult to determine precisely what makes the impossible ALs harder to learn because of the multi-dimensional nature of the altered English input.

White and Cotterell (2021) take inspiration from Ravfogel et al. (2019) but use ALs generated by a PCFG to examine the inductive biases of LMs towards different word orders. They use six parameters ('switches') which invert the order of daughter categories within distinct CF productions to determine the structure of their sentences, and evaluate LSTM and Transformer models on the ALs generated by the PCFGs defined by each distinct setting of these parameters. Extending this research, Kuribayashi et al. (2024) evaluate the performance of further cognitively-motivated LMs on the same ALs. However, as a consequence of the use of PCFGs containing a VP constituent, the ALs used by White and Cotterell (2021) and Kuribayashi et al. (2024) do not generate Verb-Subject-Object (VSO) or Object-Subject-Verb (OSV) word orders. In this paper, we generate a wider set of ALs using GCGs and replicate the experiments of Kuribayashi et al. (2024) on this new dataset. Our approach to controlled AL generation is, in principle, expressive enough to generate all attested NL constructions documented by linguists to date, so provides a general framework to support further AL-based investigation of neural LMs. In this paper, we exemplify this by also extending White and Cotterell (2021) dataset to include object relative clauses.

2.2 Categorial Grammar

Classic Categorial Grammar (CG) is a formalism which aims to represent NL syntax isomorphically with compositional semantics (Ajdukiewicz, 1935; Bar-Hillel, 1953). We focus on the syntactic generative properties of extensions to classical CG in this paper. The components of a CG are a lexicon pairing words with basic or functor categories, and a small set of rules defining how functor categories combine with basic categories syntactically and semantically. The "slash" notation is often used to indicate the direction of the arguments relative to the resulting category. For example, X/Y is a functor category looking for an argument basic category Y to the right to create result category X. In classical CG, there are just two rules **forward functional application** (a) or **backward functional application** (b), shown below.

(a) $X/Y \; Y \Rightarrow X$

(b) $Y \; X\backslash Y \Rightarrow X$

In English, a transitive verb like "met" is a functor category $(S\backslash NP)/NP$. The derivation shown below for "Kim met Sandy" shows both forward and backward application.

$$
\begin{array}{cccc}
\text{Kim} & \text{met} & \text{Sandy} & \\
\overline{\text{NP}} & \overline{(\text{S}\backslash\text{NP})/\text{NP}} & \overline{\text{NP}} & \\ \cline{2-3}
 & \multicolumn{2}{c}{\text{S}\backslash\text{NP}} & > \\ \cline{1-3}
\multicolumn{3}{c}{\text{S}} & <
\end{array}
$$

Most if not all of the variation between languages is captured by variation in the set of lexical categories assigned to words.

CG is equivalent to a binary-branching context-free grammar. There are extensions and generalizations of CG, such as Combinatory Categorial Grammar (CCG), (Steedman, 1996), which we refer to generically as Generalized Categorial Grammars (GCGs) (Wood, 2014). In CCG and GCGs, additional operations can be used to combine categories.

One such operation is **coordination**, where 2 constituents of the same category separated by conjunction can be combined into a single constituent of the same type,

$$X \text{ CONJ } X \Rightarrow X$$

Coordination (Φ) is shown in the derivation below.

$$\begin{array}{ccccc}
\text{Kim} & \text{and} & \text{Sandy} & \text{met} & \text{Felix} \\
\hline
\text{NP} & \text{CONJ} & \text{NP} & (\text{S}\backslash\text{NP})/\text{NP} & \text{NP} \\
\hline
\multicolumn{3}{c}{\text{NP}} \; {<}\Phi{>} & & \\
 & & & \multicolumn{2}{c}{\text{S}\backslash\text{NP}} \; {>} \\
\hline
\multicolumn{5}{c}{\text{S}} \; {<}
\end{array}$$

Forward composition and **backward composition** operations are utilized in CCG, where adjacent functions are composed. We show the rules of forward (a) and backward (b) composition below.

(a) $X/Y \; Y/Z \Rightarrow X/Z$

(b) $Y\backslash Z \; X\backslash Y \Rightarrow X\backslash Z$

Composition (B) is shown in the derivation below.

$$\begin{array}{cccc}
\text{the elf} & \text{on} & \text{the shelf} & \text{laughed} \\
\hline
\text{NP} & (\text{NP}\backslash\text{NP})/\text{NP} & \text{NP} & \text{S}\backslash\text{NP} \\
\hline
 & \multicolumn{2}{c}{\text{NP}\backslash\text{NP}} \; {>} & \\
 & \multicolumn{3}{c}{\text{S}\backslash\text{NP}} \; {<}\mathbf{B} \\
\hline
\multicolumn{4}{c}{\text{S}} \; {<}
\end{array}$$

Permutation is included in our GCG as a more computationally tractable alternative to type raising in CCG. We use the version from Briscoe (1997, 2000), which allows for a cyclic permutation of the functor arguments without changing their directionality. The definition of permutation is as follows:

$$(X|Y_1)...|Y_n \Rightarrow (X|Y_n)|Y_1$$

Permutation (P) is shown in the derivation below.

$$\begin{array}{ccc}
\text{Kim} & \text{met} & \text{Sandy} \\
\hline
\text{NP} & (\text{S}\backslash\text{NP})/\text{NP} & \text{NP} \\
 & (\text{S}/\text{NP})\backslash\text{NP} \; {<}\mathbf{P}{>} & \\
\hline
\multicolumn{2}{c}{\text{S}/\text{NP}} \; {<} & \\
\hline
\multicolumn{3}{c}{\text{S}} \; {>}
\end{array}$$

We develop our ALs from a GCG utilizing these rules of application, coordination, composition, and permutation.

3 Dataset

As a first case study employing our GCG to create ALs, we mostly reproduce the dataset of White and Cotterell (2021) using GCG but also add some novel word order constructions. Specifically, we adapt the parameters defined by White and Cotterell (2021) to create a GCG for each of the 64 AL configurations they define. We then created lexicons for SOV and VOS languages to create an additional 32 ALs for VSO and OSV languages. We also extend each AL with object relative clauses as an exemplar of a potentially unbounded dependency ('filler-gap') construction.

3.1 The Lexicon

We define lexical syntactic categories, e.g., `NP`, first, as listed in Table 1, and then define a set of lexicons. We use a set of mostly English words that is of the same size and has the same categories as White and Cotterell (2021), including singular and plural nouns, and past and present tense verbs, but we ignore subject-verb number agreement, in our initial, simple setting. In addition, following White and Cotterell (2021), we avoid lexical ambiguity, and thus each word in the lexicon is assigned to exactly one category. Following White and Cotterell (2021), we use subject and object markers in all the artificial languages.

3.2 Dataset Generation

Dataset generation involves several steps:

1. **Determining the GCG categories:** We set a GCG lexical syntactic category (e.g., `SCOMP\S`) for each of word types (e.g., `COMP`), as shown in Table 1. These GCG categories are parameterized by seven word order parameters shown in Table 2. For example, if the `S` parameter in Table 2 is set to 0 (head-final), the GCG syntactic type of `VI` (*walked*) should be $\text{S}\backslash\text{NP}_{\text{SUBJ}}$ as follows:

$$\begin{array}{ccc}
\text{Kim} & \text{ga} & \text{walked} \\
\hline
\text{NP} & \text{NP}_{\text{SUBJ}}\backslash\text{NP} & \text{S}\backslash\text{NP}_{\text{SUBJ}} \\
\hline
\multicolumn{2}{c}{\text{NP}_{\text{SUBJ}}} \; {<} & \\
\hline
\multicolumn{3}{c}{\text{S}} \; {<}
\end{array}$$

Category	GCG syntactic type	Example
NP (Noun Phrase)	NP	**Kim** ga kissed **Sandy** o
SUBJ (Subject Marker)	$NP_{SUBJ}\backslash NP$	Kim **ga** kissed Sandy o
OBJ (Object Marker)	$NP_{SUBJ}\backslash NP$	Kim ga kissed Sandy **o**
ADJ (Adjective)	$NP\|NP$	**red** car ga ran
VT (Transitive Verb)	$(S\|NP_{SUBJ})\|NP_{OBJ}$	Kim ga **kissed** Sandy o
VI (Intransitive Verb)	$S\|NP_{SUBJ}$	red car ga **ran**
VCOMP (Complementary Verb)	$(S\|NP_{SUBJ})\|SCOMP$	Kim ga **believed** that Sandy ga lied
COMP (Verb Complement)	$SCOMP\|S$	Kim ga believed **that** Sandy ga lied
CONJ (Conjunction)	var\var/var	Kim **and** Sandy ga ate
PREP (Preposition)	$(NP\|NP)\|NP$	elf **on** shelf ga laughed
REL (Relativizer)	$(NP_{SUBJ}\|NP_{SUBJ})\|(S\|NP_{OBJ})$	man ga **whom** I ga met laughed

Table 1: Lexical syntactic categories used in our artificial grammar. The bars "|" in the GCG lexical categories indicate either forward- or back-slash, which is controlled by word order parameters in Table 2. The examples in the English grammar are also shown, where the word(s) belonging to the category being described are shown in bold.

Param.	Description	0 (head-final)	1 (head-initial)
S	Order of subject and verb	$VI \rightarrow S\backslash NP_{SUBJ}$ $VT \rightarrow (S\backslash NP_{SUBJ})\|NP_{OBJ}$ $VCOMP \rightarrow (S\backslash NP_{SUBJ})\|SCOMP$	$VI \rightarrow S/NP_{SUBJ}$ $VT \rightarrow (S/NP_{SUBJ})\|NP_{OBJ}$ $VCOMP \rightarrow (S/NP_{SUBJ})\|SCOMP$
VP	Order of object and verb	$VT \rightarrow (S\|NP_{SUBJ})\backslash NP_{OBJ}$ $VCOMP \rightarrow (S\|NP_{SUBJ})\backslash SCOMP$ $REL \rightarrow (NP_{SUBJ}\|NP_{SUBJ})\|(S\backslash NP_{OBJ})$	$VT \rightarrow (S\|NP_{SUBJ})/NP_{OBJ}$ $VCOMP \rightarrow (S\|NP_{SUBJ})/SCOMP$ $REL \rightarrow (NP_{SUBJ}\|NP_{SUBJ})\|(S/NP_{OBJ})$
O	Order of subject and object	Restriction to make an S precede O as canonical word order	Restriction to make an O precede S as canonical word order
COMP	Position of complementizer	$COMP \rightarrow SCOMP\backslash S$	$COMP \rightarrow SCOMP/S$
PP	Postposition or preposition	$PREP \rightarrow (NP\backslash NP)/NP$	$PREP \rightarrow (NP/NP)\backslash NP$
ADJ	Order of adjective and noun	$ADJ \rightarrow NP/NP$	$ADJ \rightarrow NP\backslash NP$
REL	Position of relativizer	$REL \rightarrow (NP_{SUBJ}/NP_{SUBJ})\backslash(S\|NP_{OBJ})$	$REL \rightarrow (NP_{SUBJ}\backslash NP_{SUBJ})/(S\|NP_{OBJ})$

Table 2: Word order parameters and their associated GCG categories. "A→B" indicates A|B (A is expanded to B) in the GCG derivation.

In contrast, if S is set to 1 (head-initial), the possible word order will be like:

$$
\begin{array}{ccc}
\text{walked} & \text{Kim} & \text{ga} \\
\hline
S/NP_{SUBJ} & NP & NP_{SUBJ}\backslash NP \\
\cline{2-3}
 & \multicolumn{2}{c}{NP_{SUBJ}} \; < \\
\hline
\multicolumn{3}{c}{S} \; <
\end{array}
$$

Different ALs are generated by different combinations of the seven word-order parameters, which control the directionalities in the lexical categories, resulting in different word orders (Table 2).

2. **Generating the grammars:** We use the seven binary parameters (Table 2) to generate our 96 grammars based on GCG. The parameters, except for O, are the same as White and Cotterell (2021), and the O parameter biases the S-O order (as a part of postprocessing). This is needed because the permutation operation for the VT will eliminate the bias regarding the order of S and O, so to align the experimental settings with White and Cotterell (2021), we add this parameter. The O parameter is set to either 0 or 1 only when the subject and object are positioned on the same side of a (transitive) verb (SOV, OSV, VSO, VOS); otherwise, the O parameter is automatically determined by the first two parameters of S and VP (SVO and OVS). This process results in 96 grammars – less than the mathematically possible combinations of seven binary parameters (2^7=128). Each language is associated with a specific combination of parameter assignments and denoted, for example, as 0001111 (S=0, VP=0, O=0, COMP=1, PP=1, ADJ=1,

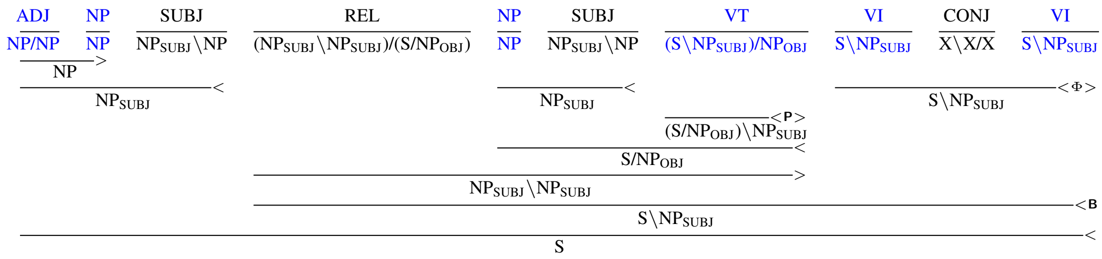


Figure 1: Example of a template and its derivation. The sentence structure is like "Tall man whom she met walked and talked." The word categories shown in black (e.g., SUBJ) correspond to a single lexical item (e.g., ga). The remaining categories in blue have several candidates of lexical items, and these are uniformly sampled from the predefined dictionary.

Algorithm 1 Template Generation Algorithm

Require: Set of word categories $\mathcal{C}$, 96 parsers $[p_1, \cdots, p_{96}]$
Initialize empty dictionary $ValidTemplates$
for $length = 3$ to 10 **do**
  for each sequence of $c \in \mathcal{C}^{length}$ **do** ▷ Generate all word category sequences
    if c matches heuristics **then**
      skip ▷ Exclude immediately invalid templates
    end if
    for each parser p_i in 96 parsers **do**
      if p_i successfully parses c **then**
        Add c to $ValidTemplates[i]$ ▷ Select grammatically valid templates
      end if
    end for
  end for
end for
return $ValidTemplates$

Algorithm 2 Generating Sentences from Templates

Input: Valid templates T, dictionary D mapping word category $c \in \mathcal{C}$ to lexical items $V_c = D[c]$
Output: Set of grammatical sentences S
$S \leftarrow \emptyset$
for each template $t \in T$ **do**
  for 0 to 500 **do**
    $s \leftarrow$ dummy string of length $|t|$
    for each category c_i in $t = [c_1, \cdots, c_n]$ **do**
      Randomly sample $w_i \sim D[c_i]$ (uniform distribution)
      $s[i] = w_i$
    end for
    if $s \notin S$ **then**
      Add s to S
    end if
  end for
end for
return S

REL=1).

3. **Template Generation:** To cover all possible valid syntactic structures in each of our 96 ALs, we first enumerate all possible sequences of word categories (e.g., "`NP ADJ VT CONJ REL...`"), up to length 10, in a brute-force manner. We then parse these sequences with a GCG parser with the corresponding grammar configuration.[1] Word category sequences, and by extension, sentences created from them, are considered grammatically valid if we obtain at least one derivation resulting in `S` based on the GCG parser. An example of a valid template is shown in Figure 1. This template generation is summarized in Algorithm 1. Note that in order to make this process more efficient, we apply some heuristics (detailed in Appendix A.1) to eliminate templates that cannot result in a valid sentence.

4. **Sentence Generation:** Once we have our templates for each of the 96 grammars, we generate 500 sentences for each template in each grammar by random sampling of the lexicon. We ensure that all of the generated sentences are unique by removing duplicate sentences when they occur. This is shown in Algorithm 2.

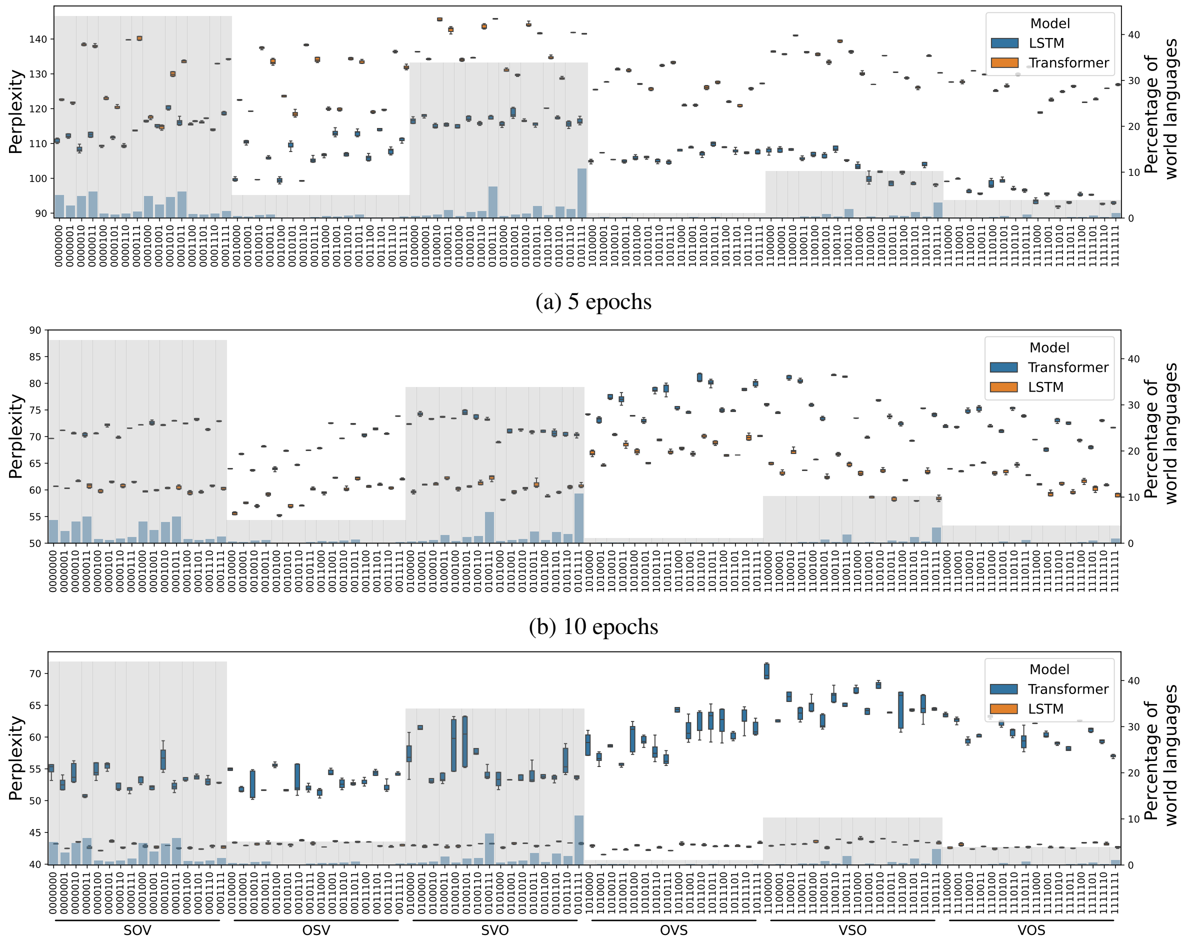
5. **Sampling from the Datasets:** Similarly to the dataset size per grammar as White and Cotterell (2021), we randomly sample 50K sentences from the datasets generated for each grammar. We also ensure that all sampled sentences are distinct. These datasets are the ones that we use in our experiments.

4 Experiments

4.1 Settings

We evaluate the same models as White and Cotterell (2021), which are the LSTM (Hochreiter and Schmidhuber, 1997) and Transformer (Vaswani

[1]We adapt the NLTK CCGChartParser (Bird et al., 2009), removing type raising and adding the permutation operation as defined by Briscoe (1997, 2000), and use this to parse our templates.



(a) 5 epochs

(b) 10 epochs

(c) Early-stop

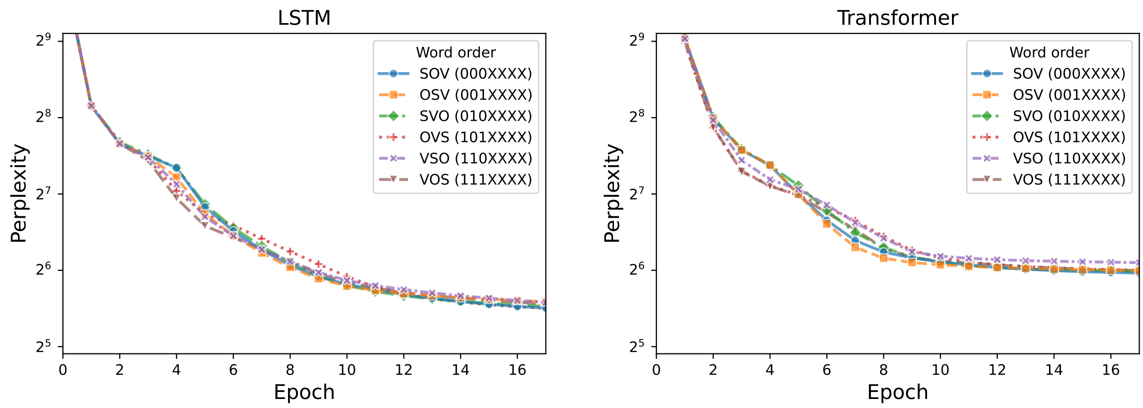
Figure 2: PPLs over 96 grammars. The blue and orange box plots correspond to Transformer and LSTM, respectively. The bars in the graph show the percentage of world languages for each grammar (blue) and word order group, e.g., SOV (gray).

et al., 2017) models. We evaluate perplexity (PPL) over the sentences of the different word orders and investigate the inductive biases that models may have towards specific word order configurations. For each of our 96 languages, similarly to Kuribayashi et al. (2024), the 50K sentences are divided across 5 runs. In each run, the 10K sequences are divided into train/dev/test split with a ratio of 8:1:1. Different random seeds are used in each run. We will basically follow the experimental settings in White and Cotterell (2021) and Kuribayashi et al. (2024) but also extend some analyses focusing on learning dynamics across different training epochs, rather than focusing only on a specific epoch (10 epochs in Kuribayashi et al. (2024)) or the end of learning based on specific criteria (early stopping with patience of 5 in White and Cotterell (2021)).

4.2 Results

What kind of language is harder to learn? Following White and Cotterell (2021); Kuribayashi et al. (2024), we show the PPL distribution across 96 grammars in Figure 2. The distributions at 5 epochs (Figure 2a), 10 epochs (Figure 2b), and the end of training based on early-stopping (consistently longer than 10 epochs; Figure 2c) are reported. Comparing our early-stopping results with those reported in White and Cotterell (2021) with the same stopping criteria, we replicate a high-level trend that Transformers exhibit more PPL variations than LSTMs. At the same time, we observe a somewhat clearer preference of Transformers toward head-final word orders (grammars with many `0`s) than reported in White and Cotterell (2021).

We also observe a dynamic change in word order preference during training. Specifically, at the earlier training phase (5 epochs; Figure 2a), the



(a) LSTM

(b) Transformer

Figure 3: The PPL trajectories for different S-O-V word orders and models (measured on validation data in the early-stopping setting). The y-axis is logarithmic. For better visibility of the preference transition, we cut off large PPLs (y-axis) in the first few epochs and results after the 18th epoch (x-axis), but there is almost no PPL difference across different word order conditions in these epochs.

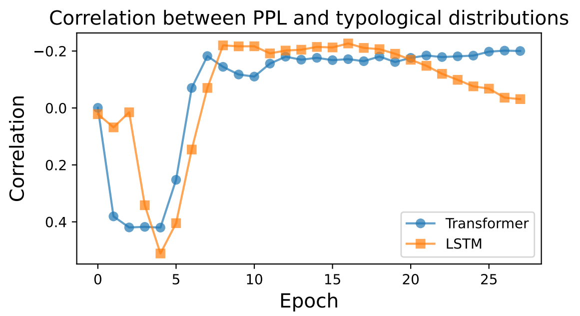
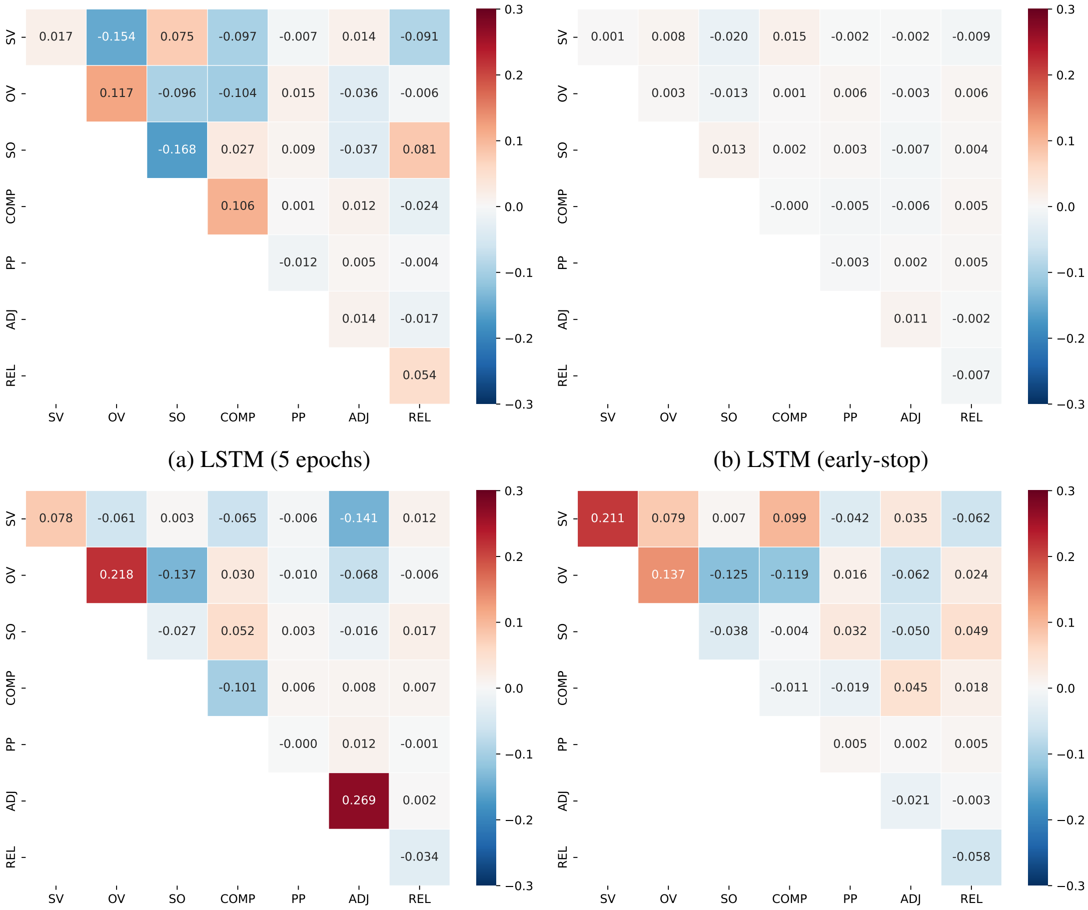


Figure 4: Correlations between PPL and typological distributions, which are measured in each epoch during training (on validation data in the early-stopping setting). The correlations from five runs are averaged. To highlight that a negative correlation is expected, the y-axis is inverted.

PPL tends to be lower in head-initial languages (grammars with many 1s) or more neutral than in the latter phase (early-stopping), which indicates that head-initial languages can be more efficiently learned at first, and then head-final languages outperforms ultimately. Comparing these dynamic preference changes (head-initial→head-final) with the diachronic word order changes in the world's languages, our results, interestingly, contrast with the common view that natural languages have evolved from head-final (SOV) to more neutral (SVO) or head-initial (VSO/VOS) ones (Gell-Mann and Ruhlen, 2011). Figure 3 further summarizes this dynamic change in word order preference.

Typological (mis)alignment The percentage of world languages for each grammar and word order group is superimposed on Figure 2 (blue and gray bars). To calculate these typological distributions, we basically adopted the statistics used in Kuribayashi et al. (2024) and enriched them by integrating the S-O order statistics from Dryer and Haspelmath (2013) and complementizer position statistics from Skirgård et al. (2023). The two distributions of PPLs and word order frequencies are compared using Pearson correlation coefficients, following Kuribayashi et al. (2024). After 5 epochs, the correlation between PPLs and typological distributions was 0.40 ($p<0.05$) and 0.25 ($p<0.05$) for LSTM and Transformer, respectively. The positive correlation indicates that the **worse** the PPL is, the **more frequent** the word order is in the world, contrasting with the common claim that natural language is optimized toward better predictability (Gibson et al., 2019; Hahn et al., 2020). After further training in the early-stopping setting, the correlation scores decreased to 0.05 (not significant) and -0.33 ($p<0.05$) for LSTM and Transformer, respectively. These dynamics are shown in Figure 4, where the correlation between typological distributions and PPL distributions for each training epoch is reported. There is a general trend that stable results (i.e., not changing suddenly in adjacent epochs) and better typological correlations are obtained at the later phase of training, but the typological alignment of the LSTM ultimately decreased and lost word order preferences as shown in Figure 2c.

Regression analysis Figure 5 shows quantitative statistics on which word order parameters are asso-



(a) LSTM (5 epochs) (b) LSTM (early-stop)

(c) Transformer (5 epochs) (d) Transformer (early-stop)

Figure 5: Coefficients of word order parameters (and their interactions) estimated by the regression models to predict PPL from word order parameters

ciated with the PPL differences. Similarly to White and Cotterell (2021), we train a regression model to predict PPLs from word order parameters and their interaction terms.[2] Positive coefficients for a single word-order parameter (diagonal elements of matrices in Figure 5) indicate that head-initial assignment leads to **worse** PPLs. Positive coefficients for interaction terms indicate that the consistent head-directionality between the two parameters leads to **worse** PPLs, and these are expected to be negative if the common patterns of consistent head-directionalities in natural language are from learners' biases. The coefficients for interaction terms are frequently positive; thus Transformers and LSTMs do not exhibit inductive biases toward typologically plausible, consistent head-directionality, which is consistent with the results in White and Cotterell (2021).

The coefficient matrices also suggest that both training setting differences (e.g., Figures 5a vs. 5b) and model architecture differences (e.g., Figures 5a vs. 5c) had an impact on the results. As for the `REL` parameter, where our inclusion of object relative clauses may impact results, we did not observe previously reported trends, for example, a relatively large positive interaction between `OV` and `REL` reported in White and Cotterell (2021) disappeared.

4.3 Discussion

There are several possible reasons that could explain the differences between our findings and those of White and Cotterell (2021) and Kurib-

[2]We used the statsmodels package (Seabold and Perktold, 2010). The formulation is `PPL` ∼ `SV*OV + SV*SO + SV*COMP + SV*PP + SV*ADJ + SV*REL + OV*SO + OV*COMP + OV*PP + OV*ADJ + OV*REL + SO*COMP + SO*PP + SO*ADJ + SO*REL + COMP*PP + COMP*ADJ + COMP*REL + PP*ADJ + PP*REL + ADJ*REL`, where each parameter is a binary factor with dummy coding (head-final as 0 and head-initial as 1), and `X*Y` represents to both main effects of `X` and `Y` and their interaction effect of `X:Y`. We normalized PPL scores with min-max scaling. In contrast to White and Cotterell (2021), we did not include the sentence-level random effect because our dataset does not have strict alignment between sentences across different grammars.

ayashi et al. (2024). One reason may be that the GCG-generated datasets are potentially more complex than the PCFG-generated datasets used by White and Cotterell (2021) and Kuribayashi et al. (2024). Our datasets include some long-distance dependencies, and in some cases, as a result of permutation, more flexible word orders. Another source of the difference is the addition of 32 grammars (VSO and OSV), which were not included in previous studies. At the same time, we simplified the grammar to omit subject-verb number agreement in this study; thus, the impact of adding such strict agreement rules should be considered in future work.

The dynamic change of word order preference over training epochs emphasizes the effect of inductive biases from training hyperparameters (e.g., training length) beyond model architectures. We have reported experiments only using specific LSTM and Transformer LMs (see Appendix B), but as an orthogonal endeavor to refining ALs, testing a more diverse set of models, including syntactic LMs (Kuribayashi et al., 2024) and more comprehensive exploration of model configurations (e.g., layer numbers, parameter sizes), should yield further insights.

5 Conclusions

In this paper, we extend the work of White and Cotterell (2021) and create a broader set of ALs to evaluate the inductive biases of LMs towards different word orders. This includes the OSV and VSO word orders that were not represented in previous works (White and Cotterell, 2021; Kuribayashi et al., 2024) and permits the inclusion of constructions, which can represent more complex or flexible structures and orders, including longer distance dependencies. We evaluate LSTM and Transformer learning of our ALs and calculate perplexity. We find that the models prefer head initial languages, which contrasts with the findings obtained in previous work. This is intriguing and raises questions that we intend to address and explore further in future work.

We intend to investigate the effects of different training settings and paradigms, on the learning of different language configurations. We also intend to investigate and explore how the models generalize beyond the training data, e.g., to longer sequences. We also intend to investigate and understand model learning and behavior when exposed to different types of long-distance dependencies, such as nested dependencies and cross-serial dependencies, as they occur in NLs. The lexicon we use here disregards verb tenses and number agreement. In future work, we plan to extend our lexicon to contain more detail about the specific features of words and, in general, inject more realistic properties into our ALs.

Limitations

In this work, we use artificial languages to evaluate our LMs' inductive biases. Artificial languages, though controlled, often do not reflect many of the properties and complexities of natural languages, such as subject-verb agreement, lexical ambiguity, and long-distance dependencies. We do not currently distinguish between nouns of different pluralities or verbs of different tenses in our lexicon. More critically, the meaning of sentences in our artificial language is nonsensical in the sense that terminal lexical symbols are randomly sampled, while natural language will have selectional preferences (Hopkins, 2022), or more generally, grounding to events/propositions in the real world. Although our study is a step in the direction of resolving such limitations with GCG, in the future, we plan to extend our lexicon and grammar, including crosslingual perspectives (Xu et al., 2025; Yang et al., 2025), to include more detail and more realistic properties of natural language step-by-step. There is also room to explore the design of typologically impossible/implausible features (Hunter, 2025). Our artificial languages go beyond context-free, and allow us to evaluate the different types of longer-distance dependencies, which we have not explored in detail in this work, but plan to address in the future.

Such future work should also include more in-depth ablations on what kind of additional complexity, compared to the existing PCFG data, affected the results. The evaluation framework also has room to be extended; for example, we can evaluate the compositional generalization of LMs using out-of-domain, longer sequences in evaluation. It will also be fruitful to integrate the perspective of interpretability research to answer how and why LMs struggle with specific word order languages internally.

From an engineering perspective, our dataset generation pipeline can be improved. We first generated possible word sequences in a brute-force manner, and then these were filtered with some heuristic rules and a CCG parser. This brute-force process will limit generation of a corpus with longer sentence lengths, and should be replaced with a more efficient method.

Lastly, while the training paradigms we use in this work are very commonly used, our tested LMs are limited with respect to, e.g., their parameter size, types, and training procedures. In the future, we would like to develop a better understanding of the learning dynamics and explore LM learning of our ALs using different learning paradigms.

Ethical Statement

The data used in this paper is artificial data based mostly on English words. It does not contain any sensitive information or any information that poses any risks. We have no ethical concerns with the contents of this paper.

Acknowledgment

We appreciate constructive feedback from anonymous reviewers.

References

Kazimierz Ajdukiewicz. 1935. Die syntaktische konnexitat. *Studia philosophica*, pages 1–27.

Yehoshua Bar-Hillel. 1953. A quasi-arithmetical notation for syntactic description. *Language*, 29(1):47–58.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly.

Ted Briscoe. 1997. Co-evolution of language and of the language acquisition device. In *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics, Madrid, Spain*, pages 418–427.

Ted Briscoe. 2000. Grammatical acquisition: Inductive bias and coevolution of language and the language acquisition device. *Language*, 76(2):245–296.

Noam Chomsky, Ian Roberts, and Jeffrey Watumull. 2023. Noam chomsky: The false promise of chatgpt. *The New York Times*, 8.

Grégoire Delétang, 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, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online (v2020.4)*. Zenodo.

Nadine El-Naggar, Pranava Madhyastha, and Tillman Weyde. Exploring the long-term generalization of counting behavior in rnns. In *I Can't Believe It's Not Better Workshop: Understanding Deep Learning Through Empirical Falsification*.

Murray Gell-Mann and Merritt Ruhlen. 2011. The origin and evolution of word order. *Proceedings of the National Academy of Sciences*, 108(42):17290–17295.

Edward Gibson, Richard Futrell, Steven P Piantadosi, Isabelle Dautriche, Kyle Mahowald, Leon Bergen, and Roger Levy. 2019. How efficiency shapes human language. *Trends Cogn. Sci.*, 23(5):389–407.

Michael Hahn, Dan Jurafsky, and Richard Futrell. 2020. Universals of word order reflect optimization of grammars for efficient communication. *Proc. Natl. Acad. Sci. U. S. A.*, 117(5):2347–2353.

John Hewitt, Michael Hahn, Surya Ganguli, Percy Liang, and Christopher D. Manning. 2020. Rnns can generate bounded hierarchical languages with optimal memory. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 1978–2010.

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

Mark Hopkins. 2022. Towards more natural artificial languages. In *Proceedings of the 26th Conference on Computational Natural Language Learning (CoNLL)*, pages 85–94, Abu Dhabi, United Arab Emirates (Hybrid).

Tim Hunter. 2025. Kallini et al. (2024) do not compare impossible languages with constituency-based ones. *Computational Linguistics*, pages 1–10.

Armand Joulin and Tomás Mikolov. 2015. Inferring algorithmic patterns with stack-augmented recurrent nets. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 190–198.

Julie Kallini, Isabel Papadimitriou, Richard Futrell, Kyle Mahowald, and Christopher Potts. 2024. Mission: Impossible language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 14691–14714.

Tatsuki Kuribayashi, Ryo Ueda, Ryo Yoshida, Yohei Oseki, Ted Briscoe, and Timothy Baldwin. 2024. Emergent word order universals from cognitively-motivated language models. In *Proceedings of the*

62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024, pages 14522–14543.

S. J. Mielke, Ryan Cotterell, Kyle Gorman, Brian Roark, and Jason Eisner. 2019. What kind of language is hard to language-model? In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4975–4989.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Shauli Ravfogel, Yoav Goldberg, and Tal Linzen. 2019. Studying the inductive biases of rnns with synthetic variations of natural languages. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 3532–3542.

Skipper Seabold and Josef Perktold. 2010. Statsmodels: Econometric and statistical modeling with python. In *Proceedings of the 9th Python in Science Conference 2010 (SciPy 2010), Austin, Texas, June 28 - July 3, 2010*, pages 92–96.

Hedvig Skirgård, Hannah J. Haynie, Damián E. Blasi, Harald Hammarström, Jeremy Collins, Jay J. Latarche, Jakob Lesage, Tobias Weber, Alena Witzlack-Makarevich, Sam Passmore, Angela Chira, Luke Maurits, Russell Dinnage, Michael Dunn, Ger Reesink, Ruth Singer, Claire Bowern, Patience Epps, Jane Hill, Outi Vesakoski, Martine Robbeets, Noor Karolin Abbas, Daniel Auer, Nancy A. Bakker, Giulia Barbos, Robert D. Borges, Swintha Danielsen, Luise Dorenbusch, Ella Dorn, John Elliott, Giada Falcone, Jana Fischer, Yustinus Ghanggo Ate, Hannah Gibson, Hans-Philipp Göbel, Jemima A. Goodall, Victoria Gruner, Andrew Harvey, Rebekah Hayes, Leonard Heer, Roberto E. Herrera Miranda, Nataliia Hübler, Biu Huntington-Rainey, Jessica K. Ivani, Marilen Johns, Erika Just, Eri Kashima, Carolina Kipf, Janina V. Klingenberg, Nikita König, Aikaterina Koti, Richard G. A. Kowalik, Olga Krasnoukhova, Nora L.M. Lindvall, Mandy Lorenzen, Hannah Lutzenberger, Tônia R.A. Martins, Celia Mata German, Suzanne van der Meer, Jaime Montoya Samamé, Michael Müller, Saliha Muradoğlu, Kelsey Neely, Johanna Nickel, Miina Norvik, Cheryl Akinyi Oluoch, Jesse Peacock, India O.C. Pearey, Naomi Peck, Stephanie Petit, Sören Pieper, Mariana Poblete, Daniel Prestipino, Linda Raabe, Amna Raja, Janis Reimringer, Sydney C. Rey, Julia Rizaew, Eloisa Ruppert, Kim K. Salmon, Jill Sammet, Rhiannon Schembri, Lars Schlabbach, Frederick W.P. Schmidt, Amalia Skilton, Wikaliler Daniel Smith, Hilário de Sousa, Kristin Sverredal, Daniel Valle, Javier Vera, Judith Voß, Tim Witte, Henry Wu, Stephanie Yam, Jingting Ye, Maisie Yong, Tessa Yuditha, Roberto Zariquiey, Robert Forkel, Nicholas Evans, Stephen C. Levinson, Martin Haspelmath, Simon J. Greenhill, Quentin D. Atkinson, and Russell D. Gray. 2023. Grambank reveals the importance of genealogical constraints on linguistic diversity and highlights the impact of language loss. *Science Advances*, 9.

Taiga Someya, Ryo Yoshida, and Yohei Oseki. 2024. Targeted syntactic evaluation on the chomsky hierarchy. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation, LREC/COLING 2024, 20-25 May, 2024, Torino, Italy*, pages 15595–15605.

Mark Steedman. 1996. Surface structure and interpretation.

Mirac Suzgun, Yonatan Belinkov, Stuart M Shieber, and Sebastian Gehrmann. 2019. Lstm networks can perform dynamic counting. In *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pages 44–54.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

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, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 740–745.

Jennifer C. White and Ryan Cotterell. 2021. Examining the inductive bias of neural language models with artificial languages. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 454–463.

Mary McGee Wood. 2014. *Categorial grammars (RLE linguistics b: Grammar)*. Routledge.

Tianyang Xu, Tatsuki Kuribayashi, Yohei Oseki, Ryan Cotterell, and Alex Warstadt. 2025. Can language models learn typologically implausible languages? *Preprint*, arXiv:2502.12317.

Xiulin Yang, Tatsuya Aoyama, Yuekun Yao, and Ethan Wilcox. 2025. Anything goes? a crosslinguistic study of (im)possible language learning in lms. *Preprint*, arXiv:2502.18795.

A Dataset Details

A.1 Heuristics Used in Template Generation

In order to make the template generation process more efficient, we apply some heuristics to eliminate templates that would not result in valid sentences in any of our artificial languages. We eliminate templates with the following properties:

1. Shorter than 3 words (the shortest valid sentence in all grammars is 3 words),
2. Starting with a conjunction,
3. Ending with a conjunction,
4. Containing 2 consecutive conjunctions,
5. Containing 2 consecutive prepositions,
6. Starting with subject or object markers,
7. The total number of subject and object markers is greater than the number of NPs,
8. A complementizer appears in the template without a complement verb.

A.2 Restrictions Applied to Parser

In order to parse our templates and assign them to the suitable languages, we adapt the NLTK CCGChartParser (Bird et al., 2009) by disabling type raising, which is included in Combinatory Categorial Grammar (CCG) (Steedman, 1996) and implement and integrate the permutation operation as defined by Briscoe (1997, 2000), which is included in Generalized Categorial Grammar (GCG) (Wood, 2014). We disallow crossed composition and restrict the composition operations in the parser to forward and backward composition.

In the NLTK CCGChartParser, restrictions can be applied to prevent composition, crossing, and substitution by adding ",","." or "_", respectively, before the argument when defining the grammar. When we implement permutation, we introduce an additional character "@" that prevents permutation from being applied.

When defining our grammars, we restrict permutation to categories with S functors only, i.e., verbs. Additionally, in order to restrict the subject and object markers to only combine with NP, we restrict composition when defining the NP_{SUBJ} and NP_{OBJ} categories in the grammar.

Using GCGs to create our artificial languages can allow for flexible word orders as a result of permutation. This would result in OSV sentences being present in SOV datasets, VSO sentences being present in VOS datasets and vice versa. We inhibit permutation when parsing templates into OSV, SOV, VOS and OVS languages, except in the sentences where a REL category is present. This way, there is a clearer distinction between these languages.

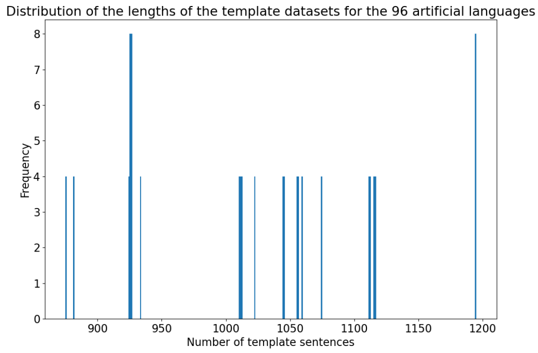


Figure 6: Histogram showing the distribution of the number of templates in the 96 artificial languages

A.3 Dataset Statistics

We calculate statistics for our 96 artificial languages and the templates from which we generate the sentences to provide more insight into the properties of the datasets.

We calculate the average sequence length for the templates and sentences used in evaluation, and they are both approximately 9.42 words long. We count the number of sequences in each template and plot the distribution of them in Figure 6. The smallest and largest template files consist of 875 and 1195 template sequences, respectively. We calculate the average template size as 1022.75 sequences.

We show the number of overlapped sentences and overlapped templates, and the percentage of overlapped sentences and templates in Figures 7,8,9, and 10. As shown in the heatmaps, there is some overlap in the templates for the different languages (Figures 9 and 10). However, there is negligible overlap between the datasets used for experiments (Figures 7 and 8).

B Model Details

Hyperparameters of the Transformer and LSTM LMs are shown in Table 3, which is the same as Kuribayashi et al. (2024). Models are trained with the Fairseq (Ott et al., 2019) toolkit.

Fairseq model	share-decoder-input-output-embed	True
	embed_dim	128
	ffn_embed_dim	512
	layers	2
	heads	2
	dropout	0.3
	attention_dropout	0.1
	#params.	462K
Optimizer	algorithm	AdamW
	learning rates	5e-4
	betas	(0.9, 0.98)
	weight decay	0.01
	clip norm	0.0
Learning rate scheduler	type	inverse_sqrt
	warmup updates	400
	warmup init learning rate	1e-7
Training	batch size	512 tokens
	sample-break-mode	none
	epochs	10

(a) Transformer.

Fairseq model	share-decoder-input-output-embed	True
	embed_dim	128
	hiden_size	512
	layers	2
	dropout	0.1
	#params.	3,547K
Optimizer	algorithm	AdamW
	learning rates	5e-4
	betas	(0.9, 0.98)
	weight decay	0.01
	clip norm	0.0
Learning rate scheduler	type	inverse_sqrt
	warmup updates	400
	warmup init learning rate	1e-7
Training	batch size	512 tokens
	sample-break-mode	none
	epochs	10

(b) LSTM.

Table 3: Model hyperparameters

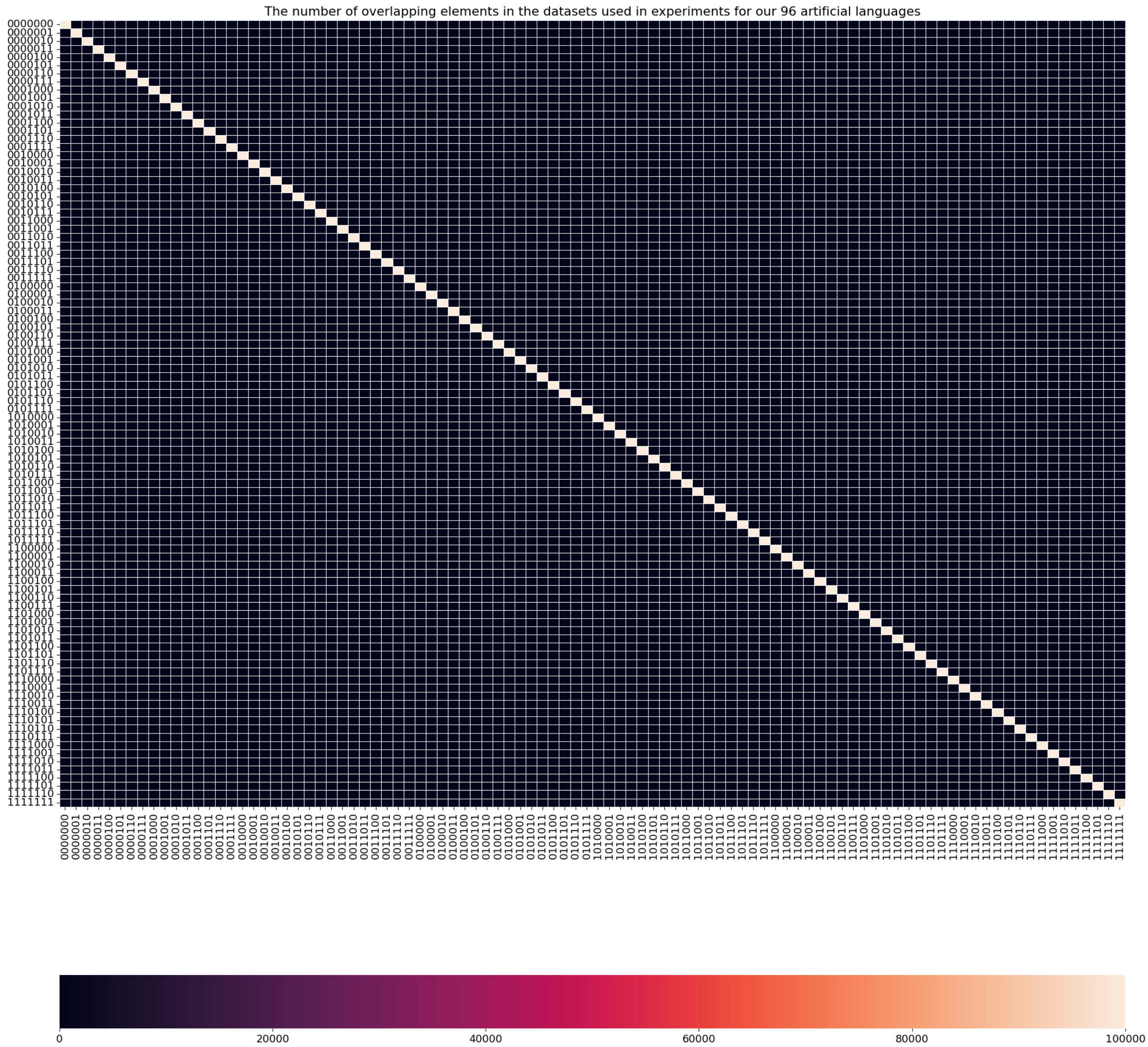


Figure 7: Heatmap showing the number of overlapping elements in the datasets for the 96 artificial languages we use in experiments.

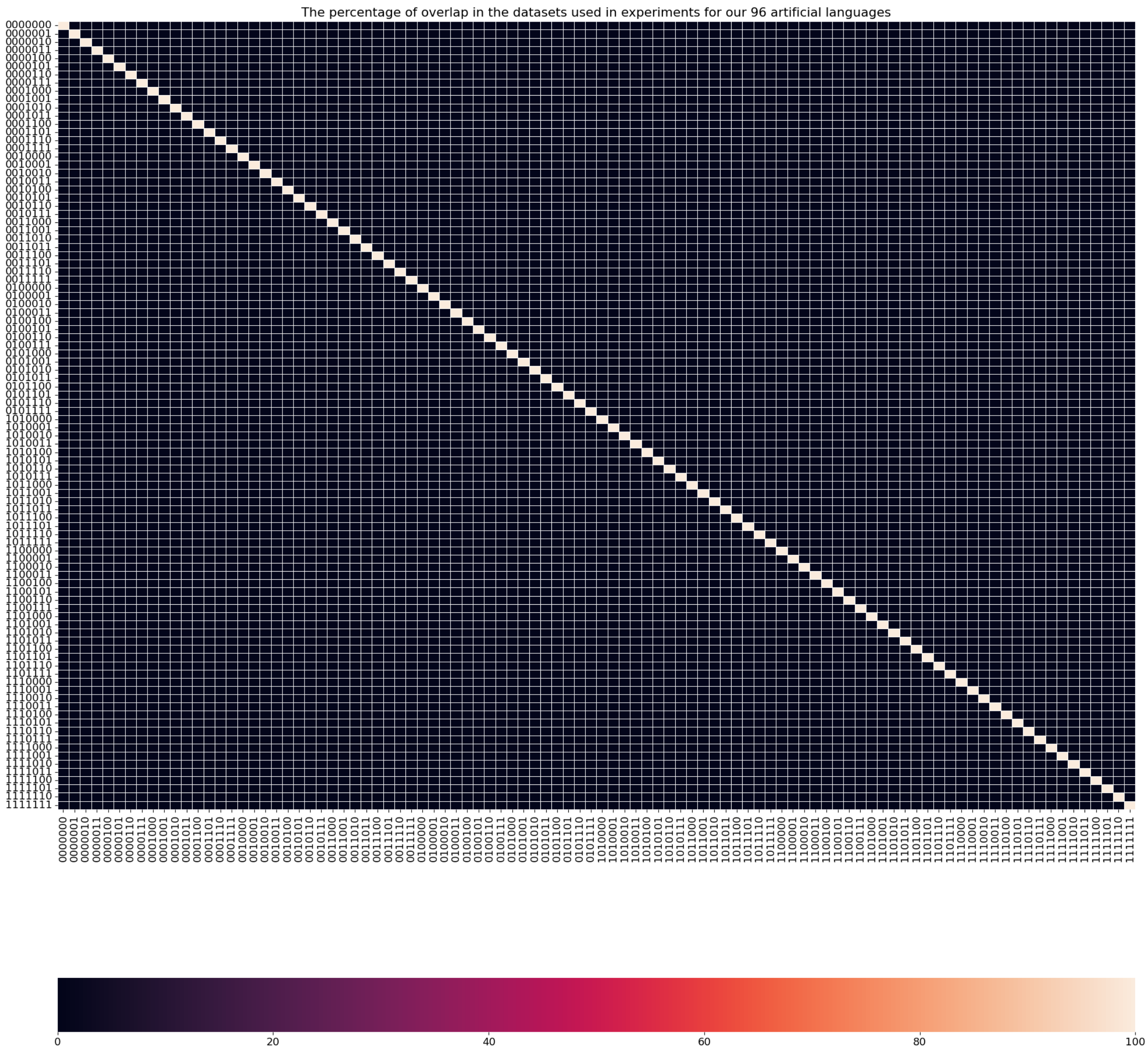


Figure 8: Heatmap showing the percentage of overlapping elements in the datasets for the 96 artificial languages we use in experiments.

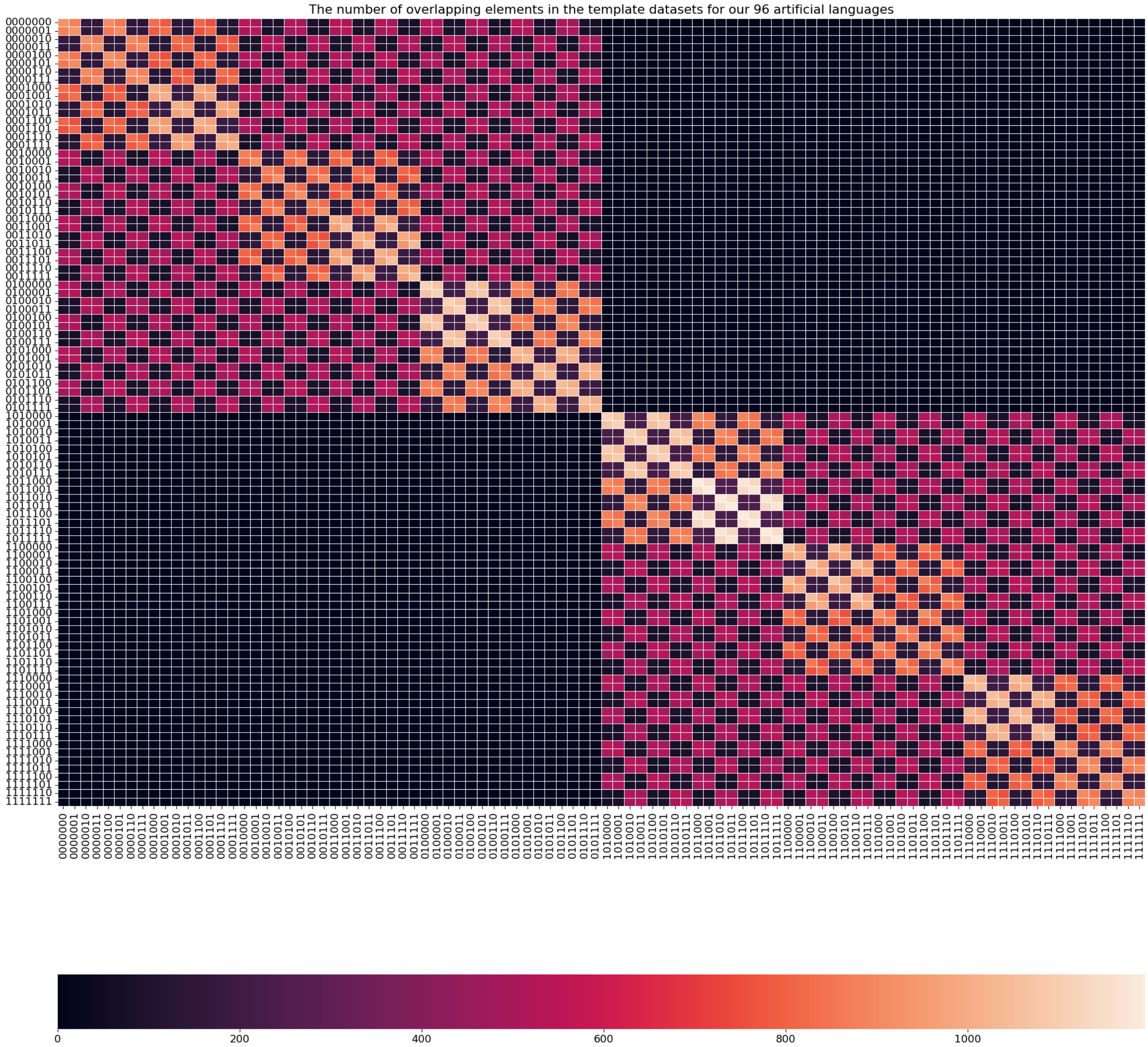


Figure 9: Heatmap showing the number of overlapping elements in the template datasets for the 96 artificial languages.

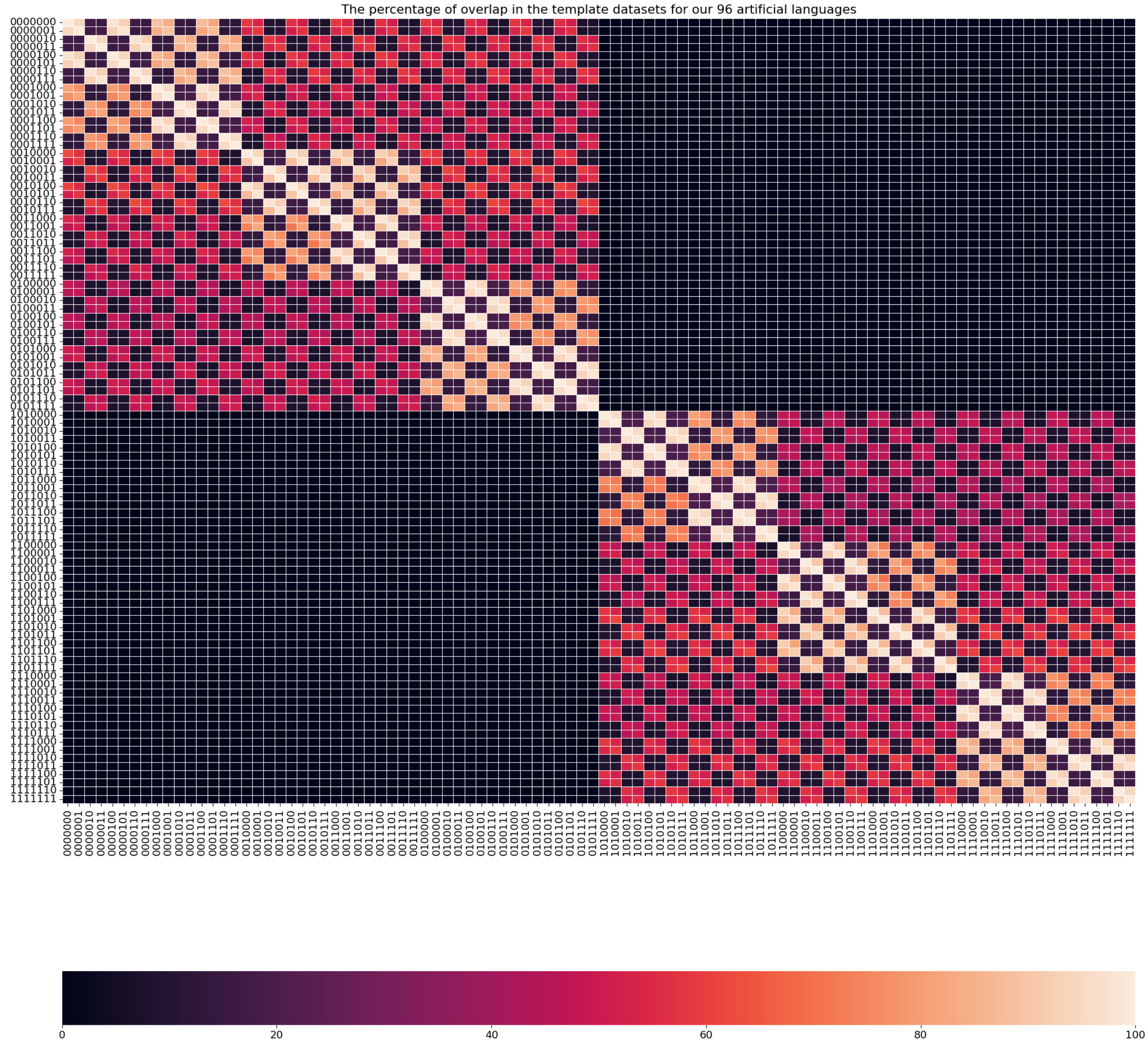


Figure 10: Heatmap showing the percentage of overlapping elements in the template datasets for the 96 artificial languages.