# Can Transformers Learn SCFGs? An Analysis of Pretraining on Synthetic Texts

**Anonymous ACL submission**

## Abstract

We explore the ability of Transformers to generalize Synchronous Context-Free Grammars (SCFG), i.e. to learn a particular grammar just from example strings. Two experiments were conducted. The first experiment explored Transformers' capacity to translate between synthetic languages corresponding to the source and target side of an SCFG grammar. The second experiment sought for a Transformer configuration which would be capable of SCFG parsing, i.e. identifying the ability to recognize licensed SCFG pairs of strings based on only positive and negative training examples. With a sufficiently large model, Transformers proved capable to learn this task to a high accuracy (97.8%) even for very long inputs, longer than any training items.

## 1 Introduction

Transformers (Vaswani et al., 2023) are, in principle, incapable of handling unbounded strings of context-free grammars due to their fixed-size architecture (Hahn, 2020). Yet, they excel in many natural language processing tasks, including machine translation, where they achieve near-human quality (Popel et al., 2020), as well as image (Dosovitskiy et al., 2020) and speech (Dong et al., 2018) processing. Translation, however, often requires modeling complex structural alignments between source and target languages, which, for some language pairs, can be described by synchronous context-free grammars (SCFGs) (Chiang, 2006). SCFGs impose a "doubly bracketing" structure—parallel hierarchical dependencies in both languages—making them particularly challenging for Transformers, as their theoretical limitations suggest a need for model size to scale with input length (Hahn, 2020).

While these theoretical constraints are well-established, their practical implications remain underexplored. How large must a Transformer be to process SCFGs of varying complexity? How many examples, and of what diversity, are needed to learn such grammars? This paper studies the empirical behavior of Transformers on synthetic SCFGs to estimate these practical limits and investigate their learnability. Our work complements theoretical analyses (Hahn, 2020; Merrill et al., 2021, 2022; Weiss et al., 2021) by providing experimental evidence on the interplay between model size, training data, and grammar complexity.

We focus on two tasks: **translation**, where Transformers must induce and apply SCFG rules to map between synchronized languages, and **parsing**, where they determine whether a string pair conforms to the grammar. Our experiments aim to assess whether Transformers can generalize from examples—inferring hierarchical rules rather than memorizing surface patterns. To this end, we:

- **Design SCFG templates** that emphasize key properties, such as deep recursion and nested productions, while maintaining simplicity (e.g., limited vocabulary size).
- **Train two Transformer architectures**:
  1. **Encoder-Decoder** for translation between synchronized languages. Success here suggests the model infers hierarchical dependencies, indicating rule induction rather than pattern memorization.
  2. **Encoder-Only** for parsing, predicting whether sentence pairs conform to the SCFG's rules, even for longer inputs. This tests generalization beyond the training set.

Our results provide empirical insights into the constants governing Transformer performance: the model size (e.g., heads, layers) required for inputs of given lengths and the number and diversity of examples needed to learn SCFGs of varying complexity.

## 2 Synchronous Context-Free Grammars

SCFGs extend standard Context-Free Grammars (CFGs) by generating pairs of related strings simultaneously (Chiang, 2006). SCFGs consist of production rules with dual right-hand sides (rhs), referred to as the "source rhs" and "target rhs". Each production synchronously generates the same non-terminals on both sides, though their positions may differ, allowing flexibility in the order between source and target outputs. Non-terminals in SCFGs are linked via indices: numbers that connect each source non-terminal to its corresponding target non-terminal, enabling synchronous generation.

### 2.1 Experimental Grammar Design

The constructed grammar for this study incorporates five critical features:

- **Deep recursion**: Enables complex hierarchical structures.
- **Compact vocabulary**: Limited terminal symbols (vocabulary of the two languages).
- **Binary form**: Pre-binarized to ensure compatibility with the CYK-like SCFG parser, which requires the grammar to be in Chomsky Normal Form. This is guaranteed by productions with either two non-terminals or a terminal.
- **No $\epsilon$-productions**: $\epsilon$-productions are not used at all in this whole experiment because they would introduce an ulterior level of complexity, especially for the parsing stage.
- **Ambiguity**: A single token on the source right-hand side may correspond to multiple possible translations on the target right-hand side, introducing ambiguity in the process.

The grammar generation system extends the `nltk.grammar` library that handles common CFGs (Bird et al., 2009). The implementation preserves the original module's interface while incorporating functionality for synchronous production.

The grammars used for the generations of pairs are visible in Appendix A.

#### 2.1.1 Language Generation Parameters

The generation process of grammar $G$ is controlled by parameters that affect:

- **Recursion probability**: Decreases logarithmically with derivation depth. Tuning this parameter determines whether to prioritize shallower trees or more complex ones.
- **Maximum length**: It limits the generation of sentences to a certain upper bound.

The final distribution of the sentences, sorted by their length, is visible in Figure 1 and Figure 2 of Appendix B, while the distribution of the terminals that belong to the two synchronous grammars (source rhs and target rhs) are visible in Figure 3 and Figure 4 of Appendix C.

A Synchronous CYK Parser, also called Bi-Text Parser(Chiang, 2006), was implemented from scratch in order to clean out all the sentences that are common both in the good grammar $G$ and in the wrong grammars $G'$ and $rand$.

## 3 First Experiment: Machine Translation

This experiment evaluates the performance of a sequence-to-sequence Transformer model for machine translation, implemented using the `FairSeq` library. (Ott et al., 2019)
`FairSeq`, a PyTorch-based framework developed by Meta, abstracts low-level complexities, enabling efficient experimentation with Transformer architectures and hyperparameters. The objective was to train a Transformer on paired sentences and assess its ability to generate accurate target sentences from source sentences during testing. Specifically, we evaluated whether the generated translations align with the gold-standard test set or remain consistent within the specified $G$ when paired with the source sentence - this is checked using the parser.

While promising results would demonstrate the transformer's translation capabilities, they may not fully reveal its generalization ability, as it could rely on simplistic token mappings between synthetic languages. To address this, a more complex experiment is described in Section 4.

### 3.1 Dataset Preparation

The dataset comprises 100,000 sentence pairs, split into three subsets: training(66%), validation(22%), and test(12%).

### 3.2 Model Configuration and Results

The Transformer model was trained on the binarized dataset using an encoder-decoder architecture. The best-performing configuration, which achieved a remarkable 97% accuracy on 10,000 test samples, is detailed in Appendix D under **Model 1**. This configuration demonstrated that the Transformer effectively learned to translate between the synthetic languages. However, reducing the model's capacity or training duration degraded performance significantly. The worst configuration tested, referred

as **Model 2** in Appendix D, yielding only 1.17% accuracy on generated sentence pairs.

These results highlight the importance of sufficient model capacity and training duration for effective translation in this context.

## 4 Second Experiment: Acceptor

The first experiment demonstrated the capability of a Transformer to generate a target-language sentence that correctly translates a given source sentence. This experiment investigates whether a Transformer can classify a pair of input sentences as generatable by a SCFG (Chiang, 2006).

The Acceptor Transformer is an encoder-only Transformer designed to perform a binary classification task. It takes as input a pair of sentences from source and target languages and outputs a token: "Y" (accepted) if the pair conforms to the target grammar, or "n" (rejected) otherwise.

Successful classification of sentence pairs as within or outside the SCFG's language would indicate the transformer's ability to generalize over data, rather than relying solely on memorization of large datasets. Such capability suggests an understanding of the grammar's complex structure, characterized by recursion and potential ambiguity.

### 4.1 Dataset Preparation

The experiment utilizes two different datasets for the experiment. Both datasets are filtered through a parser to exclude any pairs that could interfere with Transformer training. The datasets are designed for the following purposes:

- **Small Training Set**: A relatively small dataset (200,000 pairs) tests the transformer's efficiency in learning from limited data.
- **Validation on Similar Data**: Short sentences (length 2–14, matching training) with a overlap of approximately 15% evaluate whether the model performs well on familiar sentences.
- **Testing on Complex Data**: Longer, entirely unseen sentences (length 15–100) assess generalization, as these sentences are more complex. (The yellow histograms in Appendix H show the exact quantity of examples per each length using both the random grammar and the "anti-grammar" $G'$.)

The dataset is composed of pairs of sentences built as shown in Appendix E.

**Dataset $G$ + $G'$**

The first dataset is derived from the chosen $G$ and an "anti-grammar" $G'$ designed to be structurally similar but to generate a small percentage of incorrect sentence pairs. The two grammars are thought to have all reachable productions, all non-terminals on the left-hand side able to produce both unary and binary productions (which means respectively to yield terminals or call other non-terminals) and to have about 30 total productions. The "anti-grammar" $G'$ introduces a new non-terminal and some indices inversion with the goal of deviating the standard grammar $G$ from its normal behaviour.

**Dataset $G$ + random**

The second dataset is built with pair sentences generated by $G$ and randomly generated pair sentences. The random grammar uses, as for $G'$, the same terminal vocabulary of $G$ and even mimics the same probabilities of picking them. The strength of this methodology lies in the fact that all terminals are completely randomly placed in the sentence, therefore following no structure tree produced by the previously mentioned algorithm. This will result in more robustness for our experiment.

Another dataset of this type was built for a third grammar named $G_{14}$ and shown in Appendix A, more complex than $G$, with more productions, terminals and more recursion in its productions which has also given positive results.

### 4.2 Model Setup

Multiple encoder-only Transformer configurations were trained and evaluated on the two datasets to determine which dataset yields superior performance and whether randomly generated incorrect pairs enhance generalization over SCFGs. The primary objective was to identify the optimal number of attention heads, model dimension, and number of layers for improved classification accuracy and to derive scaling laws that facilitate generalization over SCFGs.

Results indicate that both datasets produce comparable performance, with configurations yielding the best and worst performances consistent across datasets. The randomly generated incorrect pairs (Dataset 2) achieve slightly higher accuracies (approximately 1% improvement). Poorly performing configurations typically feature a small model dimension and limited dataset exposure (e.g., few epochs). Many experiments have been conducted

with different configurations of the model. The two extremes are reported in the following sections.

### 4.3 Worst Configuration

The Transformer architecture for the reported results features 3 layers, 16 attention heads, a 64-dimensional model and embedding space, trained for 5 epochs with a batch size of 256. Results for the two datasets are presented in Appendix G.

This configuration predominantly predicts "n" for longer sentences, despite reasonable validation performance, suggesting limited generalization to unseen test data. See Appendix H, Figure 7.

### 4.4 Best Configuration

The architecture for the reported results features 5 layers, 32 attention heads, a 256-dimensional model and 128-dimensional embedding space, trained for 13 epochs with a batch size of 256. Results for the two datasets are presented in Appendix G while the accuracy performances on test set are visible in Figure 8 and Figure 9 of Appendix H. The learning curve and the loss of the model are shown in Appendix F.

### 4.5 Future Work

To build on the findings of this study, the following directions for future research are proposed:

**Exploration of Complex Grammars**: Investigate other and more sophisticated grammars, including those that closely resemble natural languages. This includes designing and testing SCFGs with increased numbers of terminals and production rules.

**Grammatical Property Injection**: Embedding specific grammatical properties into natural-language-like SCFGs. We aim to identify which grammatical properties facilitate or hinder translation tasks, providing deeper insights into the models' ability to learn structural recursion. We already have results (Appendix F) that suggest that a more complex grammar, having both a greater number of eligible productions and a larger terminal vocabulary, requires larger model dimension and embedding space as well as more training time to converge.

**Development of Scaling Laws**: Conduct extensive experiments to derive scaling laws (Kaplan et al., 2020) that optimize Transformer performance across a wide range of grammars. These laws would quantify the relationships between model parameters (e.g., number of attention heads, layers, and model dimensionality) and dataset size to maximize translation accuracy. Preliminary observations suggest that configurations with 16 or 32 attention heads, 3 layers, and a model dimensionality of 256 to 512 are promising.

These directions aim to enhance the generalizability of our findings and provide a deeper understanding of Transformer capabilities in learning and translating recursive grammatical structures.

## 5 Conclusions

We empirically examined the learnability of Synchronous CFGs by transformers. Our results differed from the initial expectations. Contrary to our hypothesis, the sequence-to-sequence Transformer demonstrated the ability to learn the general structure of the grammar, despite the vast number of possible recursive syntactic structures of only a fraction was demonstrated in the training data. Encoder-only also learned to identify sequences licensed by the SCFG at the accuracy of 96.7%, offering a practically appealing alternative to exact parsing.

### Limitations

This study inevitably has limitations. The experiment currently demonstrates that Transformers can learn some specific grammars with a limited set of terminals (far fewer than what would be needed for natural languages) and a small number of production rules (in the range 30–50, in contrast to thousands required to simulate a natural language). Despite the high recursion in these rules, the simplicity of the grammars restricts the generalizability of the findings. Moreover, natural languages have more complex alignments, not captured in our grammars, such as non-1:1 token alignments and the absence of $\epsilon$ productions. The results for grammar $G_{14}$ (a more complex grammar than $G$), shown in Figure 6, suggest that the best configuration from the Section 4 experiment can still learn a more complex grammar. However, achieving the same prediction accuracy as for $G$ requires different model tuning.

## References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

David Chiang. 2006. An introduction to synchronous grammars. Technical report, University of Notre Dame.

Linhao Dong, Shuang Xu, and Bo Xu. 2018. Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5884–5888.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszok, and Neil Houlsby. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*. Presented at ICLR 2021.

Michael Hahn. 2020. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *Preprint*, arXiv:2001.08361.

William Merrill, Michael Hahn, and Ashish Sabharwal. 2022. The computational complexity of transformers. *arXiv preprint arXiv:2205.08337*.

William Merrill, Ashish Sabharwal, and Noah Schwartz. 2021. Formal language theory meets modern nlp. In *Proceedings of the 2021 ACL Workshop on Benchmarking: Past, Present and Future*, pages 46–52. Association for Computational Linguistics.

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 the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Martin Popel, Marketa Tomkova, Jakub Tomek, Łukasz Kaiser, Jakob Uszok, Ondřej Bojar, and Zdeněk Žabokrtské. 2020. Human translation quality is achievable with deep learning. *Nature Communications*, 11(1):4296.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need. *Preprint*, arXiv:1706.03762.

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2021. Thinking like transformers. In *Proceedings of the 9th International Conference on Learning Representations (ICLR)*. Presented at ICLR 2021.

# A Grammar definition

1. $G$: (manually created)

```
S -> A{1} B{2} // B{2} A{1}
A -> A{1} B{2} // A{1} B{2}
A -> C{1} F{2} // C{1} F{2}
B -> B{1} F{2} // F{2} B{1}
B -> D{1} A{2} // D{1} A{2}
C -> C{1} D{2} // D{2} C{1}
C -> F{1} B{2} // B{2} F{1}
D -> F{1} A{2} // F{1} A{2}
D -> D{1} C{2} // D{1} C{2}
F -> D{1} B{2} // B{2} D{1}
F -> F{1} C{2} // C{2} F{1}
A -> a // e
A -> b // f
A -> a // g
B -> b // f
B -> a // e
C -> c // g
C -> d // f
D -> d // h
D -> c // g
F -> c // g
F -> a // h
```

2. $G'$ ("anti-grammar"):

```
S -> A{1} B{2} // B{2} A{1}
A -> F{1} E{2} // F{1} E{2}
A -> C{1} F{2} // C{1} F{2}
B -> A{1} F{2} // F{2} A{1}
B -> D{1} A{2} // D{1} A{2}
C -> D{1} D{2} // D{2} D{1}
C -> B{1} B{2} // B{2} B{1}
D -> F{1} A{2} // F{1} A{2}
D -> S{1} C{2} // S{1} C{2}
F -> D{1} B{2} // B{2} D{1}
F -> F{1} C{2} // C{2} F{1}
E -> D{1} C{2} // D{1} C{2}
E -> F{1} A{2} // A{2} F{1}
A -> d // e
A -> c // f
A -> b // g
B -> a // f
B -> c // e
C -> d // g
C -> d // h
D -> c // h
D -> c // g
F -> a // g
F -> b // e
E -> b // h
```

3. $G_{14}$: (generated by a Grammar Generator)

```
S -> N{1} Y{2} // N{1} Y{2}
Y -> A{1} R{2} // R{2} A{1}
N -> D{1} N{2} // N{2} D{1}
R -> S{1} A{2} // S{1} A{2}
D -> N{1} R{2} // N{1} R{2}
N -> D{1} R{2} // R{2} D{1}
D -> R{1} N{2} // N{2} R{1}
R -> Y{1} D{2} // Y{1} D{2}
N -> A{1} N{2} // N{2} A{1}
A -> Y{1} N{2} // Y{1} N{2}
N -> R{1} A{2} // R{1} A{2}
A -> R{1} R{2} // R{1} R{2}
Y -> R{1} Y{2} // Y{2} R{1}
D -> A{1} R{2} // R{2} A{1}
Y -> k // m
R -> k // s
A -> d // m
Y -> d // t
D -> l // p
D -> i // s
A -> i // v
Y -> k // s
A -> h // u
Y -> l // v
```

## B Distribution of the sentences

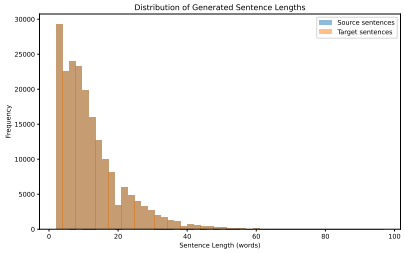


Figure 1: Distribution of 100,000 positive examples.

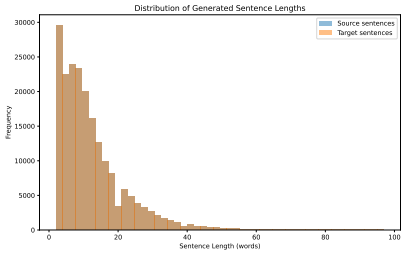


Figure 2: Distribution of 100,000 negative examples.

## C Distribution of the terminals

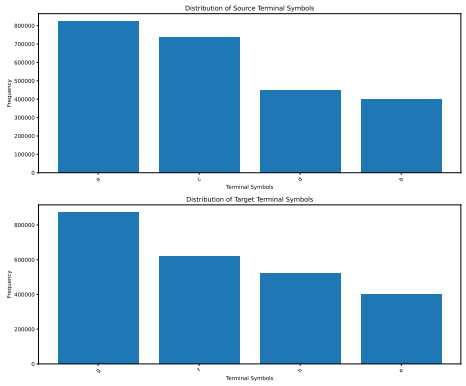


Figure 3: Distribution of source and target terminals in $G$.

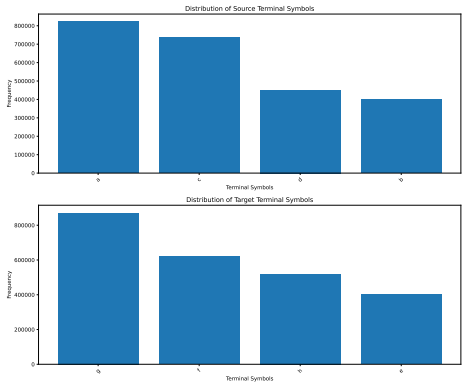


Figure 4: Distribution of source and target terminals in $random$, programmed to follow the same distribution as $G$.

## D Seq2Seq Model Tables

| Parameter | Model 1 | Model 2 |
|---|---|---|
| Enc. Layers | 6 | 1 |
| Dec. Layers | 6 | 1 |
| Enc. Emb. Dim. | 512 | 256 |
| Dec. Emb. Dim. | 512 | 256 |
| Epochs | 15 | 1 |

Table 1: Comparison of Model Configurations

## E Dataset pairs

Dataset pairs follow this structure:

`<CLS> (src) <SEP> (tgt) <LABEL> (Y/n)`

Where (src) is a string $\in$ the source language described by $G$ and (tgt) is a string $\in$ the target language described by $G$. <CLS>, <SEP>, <LABEL> are special tokens and "Y" or "n" represent the final labels: accepted or rejected.

## F Learning

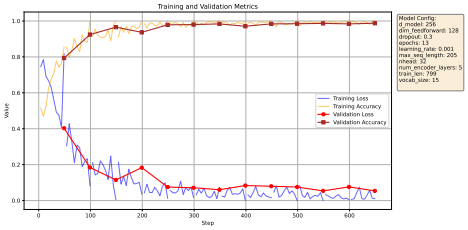


Figure 5: Learning curve of the model for $G$+random

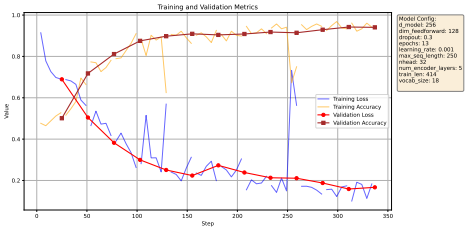


Figure 6: Learning curve of the same model for a more complex grammar $G_{14}$+random

## G Acceptor Best/Worst Configurations

| **Metric** | $G$+$G'$ | $G$+**random** |
|---|---|---|
| Training accuracy | 97.38% | 97.32% |
| Validation accuracy | 86.23% | 91.48% |
| Test accuracy | 49.86% | 50.16% |
| 'Y' labels | 1.16% | 0.37% |
| 'n' labels | 98.57% | 99.95% |

Table 2: Performance results for the worst Transformer configuration.

| **Metric** | $G$+$G'$ | $G$+**random** |
|---|---|---|
| Training accuracy | 98.12% | 99.12% |
| Validation accuracy | 97.74% | **98.75%** |
| Test accuracy | 95.62% | **96.70%** |
| 'Y' labels | **99.88%** | 99.74% |
| 'n' labels | 91.35% | **93.67%** |

Table 3: Performance results for the best Transformer configuration.

## H Acceptor performances

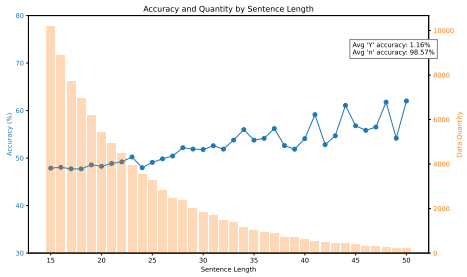


Figure 7: Worst configuration performance on $G + G'$.

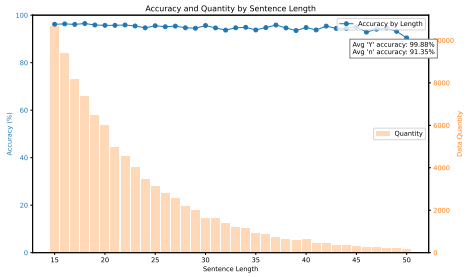


Figure 8: Best performance using $G + G'$.

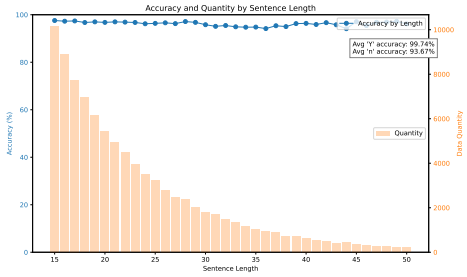


Figure 9: Best performance using $G$ + random.