Enhancing Pretraining Data Efficiency of Transformer Models using Probabilistic Context Free Grammars

Nandan Sarkar
Yale University
CS
nandan.sarkar@yale.edu

Lucas Torroba Hennigen
MIT
EECS/CSAIL
lucastor@mit.edu

Arman Cohan
Yale University
CS
arman.cohan@yale.edu

Yoon Kim
MIT
EECS/CSAIL
yoonkim@mit.edu

Abstract

Neural language models built on Transformer architectures have achieved remarkable success in Natural Language Processing (NLP) tasks but rely heavily on massive datasets, posing challenges for data-efficient training on smaller scales. This research addresses these limitations by pretraining language models on synthetic data generated using Probabilistic Context-Free Grammars (PCFGs), which capture the hierarchical and recursive structure of natural language. We train a PCFG on a 50-million-token subset of the Wikipedia corpus and use it to generate synthetic corpora in two formats: plain surface-form text and linearized tree representations with explicit bracketed structure. These are used to pretrain a 94M-parameter decoder-only Transformer model. After pretraining, models are fine-tuned on the original Wikipedia corpus and evaluated on both BLiMP - a benchmark for syntactic generalization - and validation perplexity. We experiment with varying pretraining data sizes, from 0.5M to 50M tokens, and also include baseline comparisons using data from randomly initialized PCFGs and synthetic text generated by both trained and untrained language models. Our findings show that pretraining on PCFG-generated data can improve syntactic generalization and perplexity in some cases - particularly at small scales and when mixing surface-form and tree-structured data - but these improvements are modest and inconsistent. Surprisingly, models pretrained on unstructured synthetic data from language models often performed as well or better. This suggests that the presence of hierarchical structure in pretraining data is not, on its own, sufficient to reliably induce stronger inductive biases. These results challenge prior assumptions and underscore the need to investigate what specific properties of pretraining data—beyond explicit structure—most effectively support inductive bias formation in low-resource settings.

1 Introduction

Transformer-based language models have driven dramatic progress in natural language processing (NLP), achieving state-of-the-art results on a wide range of tasks. This progress, however, has come through increasingly larger models trained on increasingly massive datasets, often reaching hundreds of billions of parameters and trillions of tokens. In contrast, the average child is exposed to fewer than 100 million words by age 13 (Warstadt et al. [2023]), yet is capable of mastering multiple languages with remarkable grammatical precision. In humans, language acquisition is guided by

inductive biases: predispositions that shape how learners generalize beyond observed data, guiding language acquisition by constraining interpretation in systematic, often universal, ways (McCoy et al. [2020]). This indicates that Transformers may lack the inductive biases necessary for efficient language acquisition.

Even though natural language is a linear sequence, it is structured hierarchically. Sentences can be split into high level phrases, which can be further split into smaller units. Being able to understand and infer these hierarchical structures is crucial in our ability to understand and produce new sentences [Murty et al., 2023b, Ahuja et al., 2025]. A core question is whether similar hierarchical biases can be encoded into neural models to improve data efficiency and generalization. A growing body of work has sought to build such biases through architectural changes (Yoshida et al. [2024], Li et al. [2024], Murty et al. [2023a], Sartran et al. [2022])and training regimes (Ahuja et al. [2025], Zhao et al. [2023], Murty et al. [2023b], McCoy et al. [2020]).

In recent years, significant efforts have been dedicated to optimizing large-scale language model pretraining, yet there has been little progress in pretraining these models at human-sized data scales (Warstadt et al. [2023]). Focusing on smaller-scale, human-like pretraining offers several important benefits: it provides a controlled sandbox for developing techniques that improve data efficiency and could scale to larger models, it enables the construction of more cognitively plausible models of human language acquisition, and it lowers the computational barrier to entry, helping to democratize academic research on pretraining (Warstadt et al. [2023]). Therefore, in this work we investigate how to improve pretraining data efficiency to more human-like scales.

We explore whether inductive syntactic biases can be imparted through the data itself, specifically investigating whether explicitly hierarchical pretraining data can improve the syntactic generalization and data efficiency of transformer language models. We test this by pretraining models on data generated from Probabilistic Context-Free Grammars (PCFGs), which capture core aspects of natural language's recursive, hierarchical structure. In this way, we generate large quantities of synthetic text for pretraining that is structurally rich and unambiguous. We evaluate two variants of this data: one as surface-form text (just words without explicit brackets) and one as linearized tree representations (with brackets indicating depths of words in the corresponding tree generated from the PCFG). Our goal is to determine whether exposing models to syntactically structured data during pretraining can effectively induce inductive biases that improve downstream generalization in data-constrained settings.

However, our findings diverged from our original expectations. Contrary to work such as Hu et al. [2025], which finds that pretraining (what they refer to as "pre-pretraining") on formal languages with hierarchical structure enables language models to achieve lower loss and better syntactic generalization with less data, we find that pretraining on synthetic, PCFG-generated hierarchical data—at least in our setup—did not consistently impart the kinds of syntactic inductive biases we aimed to induce. While some gains were observed, particularly with pretraining on mixed surface-form and tree-structured corpora, improvements were modest and often failed to outperform our baselines. These results raise important questions about what kinds of structured data and training paradigms are actually effective in inducing inductive biases.

The remainder of the paper is organized as follows: Section 2 reviews related work and provides necessary background information; Section 3 outlines our methodology; Section 4 presents experimental results and a discussion of key findings; and Section 5 concludes with directions for future work.

2 Background

2.1 Inducing Hierarchical Inductive Biases in Language Models

The hypothesis that human learners possess strong hierarchical inductive biases has motivated a line of work investigating whether similar biases can be built into neural models. Several recent studies examine how inductive biases related to syntactic structure can be acquired or enhanced during training, which we use to motivate our approach.

One such study, *Universal Linguistic Inductive Biases via Meta-Learning* [McCoy et al., 2020], explores how meta-learning can simulate the emergence of language-specific inductive biases from a universal learning prior. The authors define a distribution over possible languages, $p(L)$, and train a neural network that learns to learn: for each sampled language L_i, the model is trained on

L_i to produce a task-specific model M_i, which is then evaluated. Based on this evaluation, the meta-initialization M_0 is updated, encouraging it to acquire inductive biases that generalize across the language distribution. Over time, M_0 becomes a better starting point for rapid generalization to new languages. This setup provides a compelling framework for understanding how inductive biases—potentially including hierarchical structure—can be acquired in data-efficient ways. In our work, we similarly aim to induce syntactic biases, but through the data itself rather than meta-learning.

The work *Grokking of Hierarchical Structure in Vanilla Transformers* [Murty et al., 2023b] shows that standard transformer architectures can eventually learn to represent and generalize over hierarchical syntactic structures, even without explicit inductive bias, if trained long enough. In a carefully controlled setup, the authors observe a phase transition: performance on in-domain data saturates early, while generalization to out-of-distribution hierarchical structures emerges much later. This delayed generalization, termed "grokking", suggests that hierarchical competence can emerge naturally in transformers, but only with extensive training. Our work explores whether pretraining on structured data can encourage this generalization to occur earlier or more robustly.

In *Do Transformers Parse While Predicting the Masked Word?* [Zhao et al., 2023], the authors explore whether masked language models like BERT implicitly learn syntactic structure. Training on PCFG-generated data, they show that the inside-outside algorithm—the classical dynamic programming method for parsing—optimizes the masked language modeling objective. This suggests that, when trained on such data, transformers may implicitly recover the algorithmic structure of syntactic parsing, even without supervision. These results underscore the compatibility of PCFG-generated data with Transformer objectives and motivate our decision to use PCFGs as a tool for injecting syntactic structure.

Learning Syntax Without Planting Trees [Ahuja et al., 2025] investigates the conditions under which hierarchical generalization arises in transformers. The authors test various training objectives and datasets, finding that only the standard language modeling objective reliably induces hierarchical generalization across syntactic tasks. Moreover, they find that the presence of unambiguous training examples—those with clear hierarchical structure—leads to the formation of specialized subnetworks that support this generalization. These findings highlight two important insights for our work: first, that language modeling is an effective vehicle for learning syntax; and second, that unambiguous syntactic data, such as those provided by PCFG-generated data, are particularly valuable for inducing inductive biases.

Finally, *Between Circuits and Chomsky: Pre-pretraining on Formal Languages Imparts Linguistic Biases* [Hu et al., 2025] presents a study with motivations and methodology closely aligned with ours. The authors demonstrate that "pre-pretraining"—that is, pretraining on formal languages with explicit hierarchical structure prior to exposure to natural language—leads to more efficient training: models converge faster and achieve better syntactic generalization compared to models trained on an equivalent amount of natural language alone. Notably, they suggest that a single token of formal language may substitute for multiple tokens of natural language during pretraining. While our work shares their central motivation, our findings counter their conclusions: although we observe similar performance gains relative to our standard baseline, these gains do not consistently surpass alternative baselines in our setting. We expand on these findings and their implications in Section 4.

Together, these works motivate our approach of using synthetically generated, unambiguous hierarchical data, derived from PCFGs, to pretrain language models. By doing so, we aim to inject inductive biases that promote hierarchical generalization, potentially accelerating or amplifying the kinds of structural learning observed in the studies above.

2.2 PCFG Definition

PCFGs are a stochastic extension of context-free grammars that assign probabilities to each grammar rule. Formally, a PCFG is defined as a 5-tuple $G = (N, \Sigma, R, S, q)$, where:

- N is a finite set of non-terminal symbols,
- Σ is a finite set of terminal symbols,
- R is a set of production rules $\alpha \to \beta\gamma$, with $\alpha \in N$ and $\beta, \gamma \in N \cup \Sigma$,
- $S \in N$ is the start symbol,

- $q(\alpha \to \beta\gamma)$ is a probability distribution over rules, such that:

$$\sum_{\alpha \to \beta\gamma \in R : \alpha = X} q(\alpha \to \beta\gamma) = 1 \quad \text{for each } X \in N$$

and $q(\alpha \to \beta\gamma) \geq 0$ for all $\alpha \to \beta\gamma \in R$.

The probability of a complete parse tree t is defined as the product of the probabilities of the rules used to generate it

$$p(t) = \prod_{i=1}^{n} q(\alpha_i \to \beta_i \gamma_i)$$

where each $\alpha_i \to \beta_i \gamma_i$ is a rule in the tree. PCFGs are useful for modeling ambiguity in natural language, as they allow ranking of parse trees by likelihood (Collins [2011]). We leverage PCFGs to generate synthetic training data with explicit hierarchical structure. This allows us to investigate whether pretraining on such unambiguous data can imbue transformer models with inductive biases that support more efficient and generalizable syntactic learning. Specifically, we utilize the Simple PCFG (Liu et al. [2023]) formulation that simplifies the grammar by assuming that left and right branches in the rules are generated independently. That is, rule probabilities are factorized as

$$q(\alpha \to \beta\gamma) = q(\beta \mid \alpha) \cdot q(\gamma \mid \alpha),$$

allowing efficient training and generation at scale.

3 Methodology

In this section, we outline our full experimental pipeline. At a high level, our approach involves four steps: (1) training a PCFG on a real-world corpus, (2) generating synthetic data from the learned PCFG, (3) pretraining a transformer language model on this synthetic data, and (4) fine-tuning the model on the same natural language corpus used to train the PCFG.

3.1 Training the PCFG

We train a Simple-PCFG with 4096 non-terminal and 8192 terminal symbols and a 30k vocabulary on a 50 million token subset of the Wikipedia corpus ([Foundation, 2023]). We chose a 50 million token subset to approximate the mid-range of linguistic exposure typically experienced by a child.

3.2 Data Generation

We generate multiple pretraining corpora using the trained PCFG. For baselines, we also generate pretraining corpora from a randomly initialized PCFG, as well as trained and randomly initialized language models.

Trained PCFG Data Generation. Once trained, we generate synthetic data from the grammar. To do this, we begin by sampling from the start symbol and recursively apply production rules—each selected according to the learned rule probabilities during training—until a complete derivation consisting solely of terminal symbols is produced. Each derivation yields two types of data: 1) a sentence in its surface form, and 2) a linearized tree string corresponding to the sentence's hierarchical structure where each word is annotated with bracketing that captures its depth in the syntactic hierarchy. We pretrain models on each data type independently, as well as on a mixture of both surface-form and tree-structured inputs.

Data generation for Baselines. As baselines, we generate parallel datasets from a randomly initialized PCFG with the same vocabulary and non-terminal count. We further generate synthetic text using a trained baseline language model (trained on the original 50 million token subset of the Wikipedia Corpus with no pretraining), and a randomly initialized language model.

Mixed Corpora. We also create mixed datasets consisting of both text and tree representations, marked with special tokens to indicate the input type.

128-Shuffle-Dyck Data. Finally, to directly compare with the strongest results reported by Hu et al. [2025], we generated pretraining corpora using 128-Shuffle-Dyck data—their best-performing formal language—to evaluate whether similar improvements would hold in our experimental setup.

3.3 Model Architecture, Pretraining, and Finetuning

All experiments use a custom ∼94M parameter decoder-only Transformer model based on the `lit-gpt` ([AI, 2023]) codebase. The model uses 16 transformer layers, 8 attention heads, and an embedding dimension of 512. For tokenization, we use a 30K Byte Pair Encoding vocabulary trained on the full 50M token Wikipedia corpus. In experiments involving synthetic tree representations, we modify the tokenizer to assign special token IDs to custom bracketing symbols.

We vary the amount of pretraining data across all conditions: 0.5M, 1M, 10M, 30M, and 50M tokens. All models are pretrained for five epochs. Fine-tuning is performed on the original 50M-token Wikipedia corpus using the same architecture and tokenizer for an additional five epochs.

Furthermore, we found that training only on the structure of the data—rather than the semantics—led to better results across experiments. In practice, this corresponds to resetting the input and output embeddings of the language model after pretraining and before fine-tuning. This step encourages the model to rely completely on structural patterns learned from the data. Because this strategy consistently outperformed not resetting the input/output embeddings, we apply embedding resets in all experiments and report only those results in Table 1.

3.4 Evaluation

To assess whether structured pretraining improves syntactic generalization, we evaluate each model on the BLiMP benchmark Warstadt et al. [2020], focusing specifically on the syntax and syntax-adjacent tracks. These tracks consist of minimal pairs of sentences designed to test grammatical acceptability across a range of syntactic phenomena. A model is scored as correct if it assigns a higher probability to the grammatical sentence. We report average accuracy across categories. We also report validation perplexity on the Wikipedia corpus to measure the model's overall language modeling capability.

4 Results and Discussion

Table 1 presents the performance of all pretraining configurations, evaluated on both BLiMP syntactic accuracy and Wikipedia validation perplexity. We highlight several key findings below:

First, we find that pretraining on **trained PCFG-generated trees** consistently improves perplexity relative to the baseline and yields strong syntactic generalization. The 50M-token model achieves a BLiMP accuracy of 0.6178—the highest among all PCFG-based configurations. This supports the idea that exposing models to explicit hierarchical structure during pretraining can guide them toward more syntactically informed representations. However, these gains do not scale monotonically with data volume: the lowest perplexity is observed in the 0.5M-token model, which nonetheless underperforms the baseline on BLiMP. Meanwhile, the 50M-token model exhibits higher perplexity than smaller configurations (though still lower than baseline), but attains the best syntactic accuracy overall.

Models pretrained on **trained PCFG-generated surface text** exhibit more variable behavior. While the 10M-token model performs competitively with a BLiMP accuracy of 0.6106, the 50M-token version fails to match this performance and underperforms both the baseline and several lesser-trained models. This pattern suggests that while surface-form PCFG samples retain syntactic richness, they may lose inductive effectiveness at scale, potentially due to a lack of explicit structural signals that are otherwise available in tree-structured inputs.

Mixed tree/text corpora perform well at small scales, with the mixed data 0.5M-token model achieving 33.0128 perplexity, the best among all PCFG-based configurations. However, we find diminishing returns in perplexity and BLiMP score as training data size increases, suggesting that mixing formats may introduce ambiguity. This is interesting, as we thought that simultaneously exposing models to bracketed and unbracketed structure might reinforce hierarchical inductive biases.

When comparing to **randomly initialized PCFGs**, we find that structure alone (even without trained probabilities) can be effective. The 50M-token random PCFG tree model achieves 0.6128 BLiMP accuracy, nearly matching the best trained-PCFG model, and the 0.5M-token random PCFG tree model outperforms its trained counterparts in perplexity. This suggests that the presence of consistent, recursive structure, even when randomly generated, can facilitate generalization.

We also evaluate models pretrained on **LM-generated synthetic data**. Surprisingly, pretraining on text generated by a randomly initialized LM yields the best perplexity across all configurations (32.8733), while the strongest BLiMP performance overall comes from pretraining on 0.5M tokens of text generated by a baseline LM trained on our Wikipedia corpus (0.6180). These results challenge the notion that syntactic structure alone drives improvements in generalization.

Finally, while **128-Shuffle-Dyck data**—the best-performing formal language from Hu et al. [2025]—does show modest improvements over the baseline in both metrics, its performance is outpaced by several of our PCFG and LM-based variants, suggesting that benefits from such formal pretraining may not easily transfer across training regimes and model configurations.

Together, these findings complicate the picture painted by prior work. While structured data can yield clear benefits, its effectiveness appears to depend on the balance of format, quantity, and training strategy.

5 Conclusion and Future Work

This study set out to explore whether syntactic inductive biases can be imparted through pretraining on PCFG-generated data. While we observed modest improvements in both syntactic accuracy and perplexity in certain configurations, our results suggest that structured data alone may not reliably enhance generalization. These findings open up several promising avenues for future work. In particular, we aim to explore other hybrid pretraining strategies that combine structured and unstructured data. Prior research has shown that mixing formal languages with random language data during pretraining can lead to improved generalization. Extending this, we plan to test whether blending PCFG-generated data with random LM-generated sequences or natural language samples can help balance inductive bias with lexical diversity, potentially yielding stronger downstream performance. Additionally, we will investigate alternative syntactic formalisms and data curation techniques to further enhance pretraining data efficiency at human-like scales.

Configuration	BLiMP	Perplexity
Baseline	0.5922	35.1104
Pretrained on Trained PCFG Generated Trees:		
0.5M trees	0.5892	33.4093
1M trees	0.5847	33.0997
10M trees	0.5976	33.7538
30M trees	0.5938	34.9081
50M trees	0.6178	34.8878
Pretrained on Trained PCFG Generated Text:		
0.5M text	0.6042	33.2507
1M text	0.5891	33.3791
10M text	0.6106	35.8758
30M text	0.5943	35.9077
50M text	0.5787	35.3094
Mixed (dummy tokens for tree/text):		
0.5M each text/trees	0.6017	33.0128
1M each text/trees	0.5973	34.0068
10M each text/trees	0.5918	35.1650
30M each text/trees	0.5904	35.8313
50M each text/trees	0.5833	37.1077
Pretrained on Random Init. PCFG Generated Trees:		
0.5M trees	0.5899	33.1817
1M trees	0.5795	33.2966
10M trees	0.6010	34.3512
30M trees	0.5849	34.9798
50M trees	0.6128	35.6820
Pretrained on Random Init. PCFG Generated Text:		
0.5M text	0.6062	33.3579
1M text	0.5996	33.0559
10M text	0.5985	34.7514
30M text	0.5850	35.6974
50M text	0.6022	36.0656
Pretrained on Baseline LM Generated Text:		
0.5M text	**0.6180**	32.9118
1M text	0.5964	33.1910
10M text	0.5928	35.5734
30M text	0.5899	35.8268
50M text	0.5842	37.1504
Pretrained on Random Init. LM Generated Text:		
0.5M text	0.6028	**32.8733**
1M text	0.6016	34.0712
10M text	0.5965	34.4882
30M text	0.5816	36.3063
50M text	0.5929	37.1646
Pretrained on 128-Shuffle-Dyck Data:		
0.5M tokens	0.5888	33.0605
1M tokens	0.5953	33.2389
10M tokens	0.5938	33.5984
30M tokens	0.6072	35.0559
50M tokens	0.5920	35.8672

Table 1: BLiMP accuracy and validation perplexity for all pretraining configurations. Bolded values indicate the best overall performance on each metric.

References

Kabir Ahuja, Vidhisha Balachandran, Madhur Panwar, Tianxing He, Noah A. Smith, Navin Goyal, and Yulia Tsvetkov. Learning syntax without planting trees: Understanding hierarchical generalization in transformers. *arXiv preprint arXiv:2404.16367*, 2025. URL https://arxiv.org/abs/2404.16367.

Lightning AI. Litgpt. https://github.com/Lightning-AI/litgpt, 2023.

Michael Collins. Probabilistic context-free grammars (pcfgs). https://www.cs.columbia.edu/~mcollins/courses/nlp2011/notes/pcfgs.pdf, 2011. Lecture notes, Columbia University.

Wikimedia Foundation. Wikimedia wikipedia dataset, 2023. URL https://huggingface.co/datasets/wikimedia/wikipedia.

Michael Y. Hu, Jackson Petty, Chuan Shi, William Merrill, and Tal Linzen. Between circuits and chomsky: Pre-pretraining on formal languages imparts linguistic biases. *arXiv preprint arXiv:2502.19249*, 2025. URL https://arxiv.org/abs/2502.19249.

Jiaoda Li, Jennifer C. White, Mrinmaya Sachan, and Ryan Cotterell. A transformer with stack attention. In *Findings of the Association for Computational Linguistics: NAACL 2024*, 2024. URL https://arxiv.org/abs/2405.04515.

Wei Liu, Songlin Yang, Yoon Kim, and Kewei Tu. Simple hardware-efficient pcfgs with independent left and right productions. *arXiv preprint arXiv:2310.14997*, 2023. URL https://arxiv.org/abs/2310.14997.

R. Thomas McCoy, Erin Grant, Paul Smolensky, Thomas L. Griffiths, and Tal Linzen. Universal linguistic inductive biases via meta-learning. In *Proceedings of the 42nd Annual Meeting of the Cognitive Science Society*, 2020. URL https://arxiv.org/abs/2006.16324.

Shikhar Murty, Pratyusha Sharma, Jacob Andreas, and Christopher D. Manning. Pushdown layers: Encoding recursive structure in transformer language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023a. URL https://arxiv.org/abs/2310.19089.

Shikhar Murty, Pratyusha Sharma, Jacob Andreas, and Christopher D. Manning. Grokking of hierarchical structure in vanilla transformers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023b. URL https://arxiv.org/abs/2305.18741.

Laurent Sartran, Samuel Barrett, Adhiguna Kuncoro, Miloš Stanojević, Phil Blunsom, and Chris Dyer. Transformer grammars: Augmenting transformer language models with syntactic inductive biases at scale. *arXiv preprint arXiv:2203.00633*, 2022. URL https://arxiv.org/abs/2203.00633.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. Blimp: The benchmark of linguistic minimal pairs for english. *Transactions of the Association for Computational Linguistics*, 8:377–392, 2020. doi: 10.1162/tacl_a_00321. URL https://doi.org/10.1162/tacl_a_00321.

Alex Warstadt, Leshem Choshen, Aaron Mueller, Adina Williams, Ethan Wilcox, and Chengxu Zhuang. Call for papers – the babylm challenge: Sample-efficient pretraining on a developmentally plausible corpus. *arXiv preprint arXiv:2301.11796*, 2023. URL https://arxiv.org/abs/2301.11796.

Ryo Yoshida, Taiga Someya, and Yohei Oseki. Tree-planted transformers: Unidirectional transformer language models with implicit syntactic supervision. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024. URL https://arxiv.org/abs/2402.12691.

Haoyu Zhao, Abhishek Panigrahi, Rong Ge, and Sanjeev Arora. Do transformers parse while predicting the masked word? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023. URL https://arxiv.org/abs/2303.08117.