Learning Syntax Without Planting Trees: Understanding Hierarchical Generalization in Transformers

Kabir Ahuja[1] **Vidhisha Balachandran**[2] **Madhur Panwar**[3] **Tianxing He**[4]
Noah A. Smith[1,5] **Navin Goyal**[6] **Yulia Tsvetkov**[1]
[1] University of Washington, USA [2] Microsoft Research, USA [3] EPFL, Switzerland
[4] Tsinghua University, China [5] Allen Institute for AI, USA [6] Microsoft Research, India
kahuja@cs.washington.edu

Abstract

Transformers trained on natural language data have been shown to exhibit hierarchical generalization without explicitly encoding any structural bias. In this work, we investigate sources of inductive bias in transformer models and their training that could cause such preference for hierarchical generalization. We extensively experiment with transformers trained on five synthetic, controlled datasets using several training objectives and show that, while objectives such as sequence-to-sequence modeling, classification, etc., often fail to lead to hierarchical generalization, the language modeling objective consistently leads to transformers generalizing hierarchically. We then study how different generalization behaviors emerge during the training by conducting pruning experiments that reveal the joint existence of subnetworks within the model implementing different generalizations. Finally, we take a Bayesian perspective to understand transformers' preference for hierarchical generalization: We establish a correlation between whether transformers generalize hierarchically on a dataset and if the simplest explanation of that dataset is provided by a hierarchical grammar compared to regular grammars exhibiting linear generalization. Overall, our work presents new insights on the origins of hierarchical generalization in transformers and provides a theoretical framework for studying generalization in language models.

1 Introduction

Natural language is structured hierarchically: Words are grouped into phrases or constituents, which can be further grouped to form higher-level phrases up to the full sentence. How well do neural networks trained on language data learn this hierarchical structure has been a subject of great interest (Tenney et al., 2019; Peters et al., 2018; Lin et al., 2019; Wu et al., 2020). A useful tool to understand the model- and dataset-specific properties that results in aforementioned phenomenon is the test for hierarchical generalization, i.e., evaluating the capability of a model to generalize to novel syntactic forms, which were unseen during training. A classic problem to test for hierarchical generalization is *question formation*, where given a declarative sentence, e.g., *My walrus does move the dogs that do wait.*, the task is to transform it into a question: *Does my walrus move the dogs that do wait?* The task is accomplished by moving one auxiliary verb to the front. The correct choice to move *does* in this example (rather than *do*), is predicted both by a *hierarchical rule* based on the phrase-structure syntax of the sentence, and by a *linear rule* that prescribes moving the *first* auxiliary.

Hence, as a test for hierarchical generalization, we can ask: For neural networks trained from scratch on data consistent with both hierarchical and linear rules (ambiguous data), do they learn to generalize hierarchically or do they learn a linear rule? This question has been well studied in past work for different neural network architectures and it has been shown that RNN and transformer architectures, which lack explicit tree-structure encoding, fail to generalize hierarchically (Frank and Mathis, 2007; McCoy et al., 2018, 2020; Petty and Frank, 2021; Mueller et al., 2022). However, Murty et al. (2023) showed that, surprisingly, when trained for a long time after attaining perfect training accuracy, transformers *do* start to generalize hierarchically, and named this phenomenon *Structural Grokking*.

In this work, we ask: *Why do transformers show hierarchical generalization, despite lacking architectural biases towards hierarchical*

Transactions of the Association for Computational Linguistics, vol. 13, pp. 121–141, 2025. https://doi.org/10.1162/tacl_a_00733
Action Editor: Omri Abend. Submission batch: 4/2024; Revision batch: 9/2024; Published 2/2025.

structure? We first explore if the choice of training objective can influence hierarchical generalization in transformers. Specifically, we consider five objectives—language modeling, sequence-to-sequence modeling, prefix language modeling, sequence classification and cloze completion—and study the generalization behavior of transformers trained from scratch[1] under five synthetic, controlled settings following prior work—English question formation (as described above), German question formation (Mueller et al., 2022), tense-reinflection (McCoy et al., 2020), passivization (Mueller et al., 2022), and simple agreement, a task that we construct. We find that only the language modeling objective consistently obtains strong hierarchical generalization, while the other objectives often fail, implicating *language modeling objective as a source of bias towards hierarchical generalization*.

Further, to understand how hierarchical structure is learned and represented during training, we propose two new attention head pruning strategies to discover subnetworks corresponding to different generalizations. We find that *both hierarchical and linear types of generalizations can be discovered as subnetworks in the trained model*, and these subnetworks continue to coexist over the course of training, despite the overall model performing closer to one kind of generalization over the other.

Finally, we attempt to explain why language modeling results in a preference towards hierarchical structure using the Bayesian framework from Perfors et al. (2011). Specifically, we consider generative probabilistic grammars (PCFGs) to model the simple agreement task data by constructing hierarchical grammars consistent with the hierarchical rule as well as regular grammars consistent with the linear rule. We then compare their posterior probabilities to understand which grammar has a better trade-off for the goodness of fit (given by the likelihood) and simplicity (given by the prior on grammars). Our analysis reveals a *correlation between transformer LMs generalizing hierarchically and hierarchical grammars having higher posterior, compared with regular grammars*, thereby providing a potential explanation to transformers' preference towards hierarchical generalization.

Our contributions can be summarized as:

1. We show that language modeling training objective acts as a source of inductive bias towards hierarchical generalization in transformers.

2. We show that when trained on ambiguous data, transformer LMs learn multiple rules to perform the task that are encoded as subnetworks, which once formed continue to co-exist throughout training.

3. Our Bayesian analysis suggests that transformers generalize hierarchically because the *hierarchical grammars that fit the data are often ''simpler'' compared to regular grammars*.

To our knowledge, the present work is the first to show that language modeling objective is a source of inductive bias for hierarchical generalization and to use the Bayesian perspective to explain hierarchical generalization in language models.

2 Background

Hierarchical Generalization. Hierarchical generalization is a form of systematic generalization, where, given instances generated from a hierarchical grammar, we evaluate the capability of a model to generalize to unseen syntactic forms. For example, consider the task of converting a declarative English sentence to a question:

1. (a) **Input**: My walrus does move the dogs that do wait .
   (b) **Output**: Does my walrus move the dogs that do wait ?

Notice that the task can be accomplished by moving one auxiliary verb to the front of the sentence. For sentences with two auxiliaries like 1a, as English speakers we know that the auxiliary to move is the one associated with the head verb in the sentence (i.e., *does*, which is associated with *move*, not *do*, which is associated with *wait*). Modeling this rule requires understanding the

[1] Since our aim is to understand hierarchical generalization in transformers in isolation, following prior work (e.g., Murty et al., 2023), we train transformer models from scratch, without any pretraining, eliminating the possibility of these models having any hierarchical bias due to pretraining on language data (Mueller et al., 2022).

Task	Examples
QF (German)	unsere Papageien **können** meinen Papagei, der gewartet **hat** , akzeptieren . → **können** unsere Papageien meinen Papagei, der gewartet **hat** , akzeptieren ? ihr Molch, der gegessen **hat** , **kann** lächeln . → **kann** ihr Molch, der gegessen **hat** , lächeln ?
Passivization	some tyrannosaurus entertained your **quail** behind your **newt** . → your **quail** behind your **newt** was entertained by some tyrannosaurus . the zebra upon the **yak** confused your **orangutans** . → your **orangutans** were confused by the zebra upon the **yak** .
Tense reinflection	my **zebra** by the **yak** swam . → my **zebra** by the **yak** swims . my **zebras** by the **yak** swam . → my **zebras** by the **yak** swim .
Simple Agreement	my **zebra** by the **yak** → swims my **zebras** by the **yak** → swim

Table 1: Examples from the different tasks we study, highlight indicates examples in the generalization set.

phrase structure of the language. We call this the *Hierarchical Rule*. One can alternatively consider a much simpler explanation, the *Linear Rule*: moving the first auxiliary in the sentence to the beginning. However, if we consider sentences with a different syntax (relative clause attached to subject instead of object) like the example below:

2. (a) **Input**: My walrus who doesn't sing does move .
   (b) **Linear rule output**: Doesn't my walrus who sing does move ?
   (c) **Hierarchical rule output**: Does my walrus who doesn't sing move ?

In this case, using the linear rule will result in an ungrammatical sentence, i.e., outputting 2b instead of 2c. We study the following question in our work: Consider neural networks trained from scratch on data consistent with both hierarchical and linear rules (like Example 1). When presented with sentences such as 2a do they generalize hierarchically (predicting 2c) or linearly (predicting 2b)?

Tasks and Datasets. In our study, we consider five tasks, including the question formation task above. Examples from all the tasks are provided in Table 1. All the tasks follow a common recipe: The training dataset has examples that are consistent with both hierarchical and linear rules. For evaluation, two variants of the test data are considered: an in-distribution test set, which follows the same distribution as the training data (i.e., also ambiguous with respect to the correct rule); and a generalization test set, which consists of examples which are only consistent with the hierarchical rule.

For *Question Formation*, we use the dataset from McCoy et al. (2020). We also experiment with *Question Formation in German* with the dataset from Mueller et al. (2022). The dataset here consists of sentences with the modals *können/kann* (can) or auxiliaries *haben/hat* (have/has), together with infinitival or past participle main verbs as appropriate, which can be moved to the front similar to English to form questions. We also consider *Passivization* from Mueller et al. (2022), where the task is to transform a sentence in active voice to passive, and *Tense Reinflection* (McCoy et al., 2020), which involves converting a sentence in past tense to present. Finally, we introduce a simplified version of the tense reinflection task, which we name *Simple Agreement*. Unlike others, simple agreement is a single-sentence task where only the *present*-tense sentences from the tense-inflection are considered. In this task, we evaluate the model's ability to generate the correct inflection of the main verb based on the context. Here, the hierarchical rule requires the verb to agree with the hierarchically determined subject and the linear rule requires it to agree with the most recent noun in the sequence. Our motivation for introducing this new task is that it is much more straightforward to construct probabilistic grammars for single-sentence tasks, while for sentence-transformation tasks like question formation, it can be non-trivial. We make use of the underlying probabilistic grammars heavily in §5, where we try to provide a Bayesian interpretation of hierarchical generalization in language models.

For all tasks involving transformation of inputs (i.e., all except simple agreement), the datasets also include input identity pairs. For example, for question formation, the dataset is augmented with declarative-declarative pairs. Importantly, the identity pairs in the training data also include input sentences whose corresponding outputs would disambiguate the data, i.e., are only satisfied by the hierarchical rule, although, such outputs are not present in the training data and hence

the transformation task remains ambiguous. This choice was made in McCoy et al. (2020) (and others) to familiarize the model with at least the input sentences for which the outputs remain unobserved. We provide full details of all the datasets in Appendix §A.2.

For all tasks excluding simple agreement, there are 100k training examples (50k transformation and 50k identity pairs) and 1k and 10k examples in in-distribution and generalization test sets, respectively. For simple agreement, we generate 50k training examples (and 1k/10k for test datasets).

Evaluation Metrics. Following prior work (McCoy et al., 2020; Mueller et al., 2024), for evaluating question formation (both English and German) we consider the *first-word accuracy*, i.e., given the declarative sentence as the input, evaluate whether the model predicts correct auxiliary for the first word in the generated question. For passivization, we evaluate using the *object noun accuracy*, which measures whether the correct noun was moved to the subject position. Finally, for tense-reinflection and simple-agreement, we consider *main-verb accuracy*, by evaluating if the main-verb (in the present tense) has the correct inflection, i.e., appropriately singular or plural based on the context. When evaluated on the in-distribution test set we denote it as *in-distribution accuracy* and *generalization* accuracy for the generalization test set.

3 How the Training Objective Influences Hierarchical Generalization

Prior work by McCoy et al. (2020), Petty and Frank (2021), and Mueller et al. (2022) used sequence-to-sequence training objective to train encoder-decoder models and found that RNNs and transformers do not exhibit hierarchical generalization. More recently, Murty et al. (2023) used a language modeling objective to train decoder-only transformers, which they found *did* generalize when trained for a sufficiently large number of epochs. To our best knowledge, this distinction about the choice of objective function has not been called out by prior work. Hence we conduct a systematic study to understand what effect the training objective has on hierarchical generalization.

3.1 Training Objectives

We consider the following five training objectives:

Language Modeling. Given a sequence of tokens, the language modeling objective trains the model to predict each token in a sequence given the preceding tokens. The model is optimized to minimize the negative log-likelihood of the sequences in the training data. For transformers, the language modeling objective is typically associated with decoder-only models like GPT (Brown et al., 2020). For the question formation task and the declarative-question pair from the Introduction, if $\boldsymbol{s} = \langle s_1, s_2, \ldots, s_{21} \rangle = \langle$*my, walrus, does, move, the, dogs, that, do, wait, .,* quest*, does, my, walrus, move, the, dogs, that, do, wait, ?*$\rangle$, the cross-entropy loss is computed over s_1 through s_{21}, each given the preceding tokens: $-\log p(\boldsymbol{s}) = -\sum_{i=2}^{21} \log p(s_i \mid s_1, \ldots, s_{i-1})$.

Sequence-to-sequence Modeling. The sequence-to-sequence (seq2seq) modeling objective (Sutskever et al., 2014), is used to train the model to generate a target sequence ($\langle s_{12}, \ldots, s_{21} \rangle$ in the above example) *given* an input sequence ($\langle s_1, \ldots, s_{11} \rangle$). This objective, which includes only the terms from $i = 12$ to 21 in equation above, is typically associated with an encoder-decoder model as used in the original transformer architecture (Vaswani et al., 2017). Note that the seq2seq objective is more suited for tasks with an explicit input and output (like question formation and tense inflection), but is not suitable for the simple agreement task. Hence, we skip the seq2seq objective for simple agreement.

Prefix Language Modeling. In this objective (Dong et al., 2019), we again generate the output text given the input (or ''prefix''), but we use a single transformer decoder instead of an encoder-decoder model. Unlike the original language modeling objective, here the loss is only computed over the output text and does not include the prefix. One modification that we make to how the prefix-LM objective is typically used, is that we use a causal mask for the prefix tokens as well instead of having bi-directional attention over the prefix tokens, since we found the latter to perform subpar (we compare the two in detail in §3.3).

Sequence Classification. Here the model is trained to map the entire sequence to a discrete

label. For example, for question formation the model is given the input declarative sentence and trained to predict the correct auxiliary from the set of auxiliary verbs (*do, does, don't, doesn't*) that should occur at the start of the question, i.e., a four-way classification task.

Cloze Completion. Here, the model is given a sequence of tokens with some tokens masked and trained to predict the masked tokens. For example, for the question formation task, we consider the declarative-interrogative pair and mask out tokens in the interrogative sentence at all positions where the auxiliaries could be present. Note that this objective is similar to masked language modeling as in Devlin et al. (2019); however, instead of masking tokens randomly, we mask the specific tokens as described above.[2] For the passivization task, we do not evaluate the cloze completion objective, because (unlike other tasks) the output sequence is significantly different from the input, which makes defining the masking strategy in this case non-trivial. Please refer to §A.3.1 for details of each objective for all five tasks.

3.2 Experimental Setup

We train transformer models from scratch with 8 heads and embedding dimension 512 for all experiments. Following Murty et al. (2023), for question formation and tense reinflection, we train transformer models with 6 layers and 4 layers, respectively, for all objectives excluding seq2seq. For the remaining tasks, we use 6-layer transformer encoder/decoder layers depending on the training objective. For the seq2seq objective, we use a 6-layer encoder/6-layer decoder model for all tasks. We use the Adam optimizer (Kingma and Ba, 2015) for training the model with a learning rate of 0.0001, following Murty et al. (2023) and use batch size of 8, training the model for 300k steps (24 epochs) for all tasks excluding simple agreement, which we train for 200k steps (32 epochs).

Baselines. By design of the test datasets, a model following the linear rule will obtain 100% in-distribution and 0% generalization accuracy. Only a model consistent with the hierarchical rule will obtain 100% accuracy on both test sets for all tasks.

[2]Our initial experiments with random-masking resulted in subpar performance, even on in-distribution test sets.

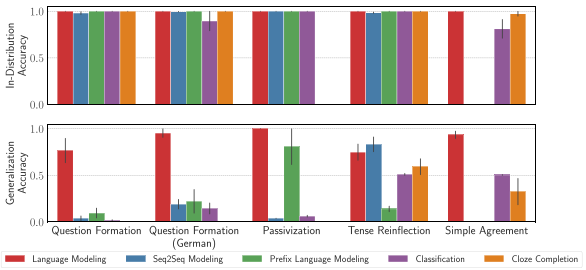


Figure 1: Effect of training objective on hierarchical generalization in transformers. The error bars correspond to the standard errors across 5 seeds.

3.3 Results

We compare the five objectives for the five tasks and show the results in Figure 1. Notice that while all the objectives almost always obtain close to 100% accuracy on the in-distribution test sets, there is much variation in the *generalization* accuracy. Particularly, we observe that only the language modeling objective consistently obtains high generalization accuracy on all five tasks, while models trained with other objectives often struggle. While seq2seq and prefix LM perform well on tense reinflection and passivization, respectively, they perform much worse on the other tasks. Thus, the choice of the objective is likely the reason behind the discrepancy in the results of Murty et al. (2023) and of Petty and Frank (2021) and Mueller et al. (2022).

We note that while the LM objective consistently achieves high generalization performance, it is not perfect as in the case of question formation and tense reinflection, where its average performance is roughly 76%. Recall that these reported numbers are averaged across 5 seeds. For all the tasks we find that there are seeds for which LMs achieve 100% generalization accuracy, which apart from the two exceptions discussed above, is not the case for other objectives. Interestingly, we observe (in Figure 1) that transformer LMs on average perform better on German question formation than the English version of the same task. We suspect this might be because the grammar used for generating the German dataset is structurally richer than the English grammar, as it also consists of both infinitival and past participle forms of the main verbs, while only infinitival forms are included in the English version. As noted in McCoy et al. (2018), presence of rich hierarchical cues in the data can aid in hierarchical generalization.

Robustness of Negative Results. We next check how robust are our negative results for non-language modeling objectives not exhibiting hierarchical generalization across different choices of hyperparameters. We consider model depth $\in \{2, 4, 6, 8, 10, 12, 16\}$, number of attention heads per layer $\in \{2, 4, 8, 16, 32\}$, and embedding dimension $\in \{64, 128, 256, 512, 1024\}$ for all the four non-LM objectives. We vary one hyperparameter at a time while keeping the other two fixed to the default values. Additionally, for prefix-LM we consider both the variants with causal attention and bidirectional attention (the original formulation from Dong et al. [2019]) on the input tokens, as we find this choice to influence the model's generalization capabilities. We provide results for the question formation task across different hyperparameter settings in Appendix Figure 4. As we can, see none of the hyperparameter settings results in matching the generalization performance of language modeling objective, which was 76% on average. The closest we get is 60% average generalization performance for prefix-LM with causal attention when using 12 layers, where 3 out of 5 seeds do end up with perfect generalization (not the case with any other objectives where none of the seeds succeed). We find this phenomenon to also hold for question formation German and passivization tasks (Figure 5 in Appendix), where seq2seq, prefix-LM (with bidirectional attention), classification, and cloze completion objectives fail to exhibit hierarchical generalization. However, for these tasks we do see prefix-LM with causal attention to perform on-par with language modeling. Overall, our results still show the most consistent trend for hierarchical generalization when using language modeling objective, and only using prefix-LM with causal attention, which is the most similar to language modeling out of all the objectives we study, to perform on par for some hyperparameter settings (typically with larger depth). The only difference between LM objective and prefix-LM objective with causal attention is the computation of loss over all tokens for the former, while only for the output tokens for the latter. Our results suggest that modeling loss over partial number of tokens in the sequence (e.g., only the output tokens and not the inputs) might be sufficient for hierarchical generalization in some cases as long as we perform autoregressive modeling. We leave further examination of this phenomenon for future work.

Takeaways. Overall, our experiments indicate language modeling as a source of inductive bias for the models to generalize hierarchically. As for why that might the case we do a more in-depth investigation in §5.

4 Discovering Subnetworks with Different Generalization Behaviors

The results from Murty et al. (2023) show that the transformer LMs obtain perfect in-domain accuracy much earlier during training, while generalization comes later. This suggests that the model might be implementing something like the linear rule early in training and eventually generalizes to the hierarchical rule. In this section, we explore whether these rules are implemented as subnetworks in the model and ask how these subnetworks evolve over the course of training.

Finding Subnetworks. Following Merrill et al. (2023), we use pruning to find the existence of subnetworks corresponding to different generalizations. In particular, we use the attention head pruning method from Voita et al. (2019), which introduces learnable gates for each attention head of a trained transformer model. Pruning is then performed by training these learnable gates while freezing the original model parameters, to minimize negative log-likelihood objective, but also adding an L_0-penalty as regularization to ensure sparsity. Since the L_0-norm is nondifferentiable, a stochastic relaxation is used, which considers the gates as random variables drawn from head-specific hard concrete distributions (Louizos et al., 2018).

After completion of pruning, all gates are either fully open or closed, and a closed gate implies that the output of the corresponding head is zeroed-out in the computation of multi-head self-attention. Thus the pruning procedure does not modify any weights of the original model and merely performs subset selection on attention heads of the model. To find subnetworks consistent with different generalizations (linear-rule and hierarchical rule) we introduce three pruning strategies which differ in the data used for pruning:

1. **`Train-prune`** uses the original ambiguous training dataset to prune the attention heads. The subnetwork thus found is likely to be a compressed version of the full model.

2. **`Gen-prune`** uses a small fraction of the generalization set (1%) to prune the attention heads. If successful, this would yield a subnetwork consistent with hierarchical generalization—obtaining close to 100% generalization accuracy.

3. **`Train\Gen-prune`** involves minimizing the (negative log-likelihood) loss on the training data and *maximizing* it for the (1%) generalization data. In this case, successful pruning should yield a subnetwork that exhibits gen eralization consistent with the linear rule, i.e., obtains 0% generalization accuracy but obtains 100% in-distribution accuracy.

Experimental Setup. For pruning, we use a learning rate of 0.05, the L_0 penalty coefficient as 0.015, and train for 10k steps, which we found to work well across different pruning settings. We report the experiments for the question formation task and discuss the others in Appendix §A.4 (Figures 6, 7), for which we also obtain consistent results. Since we are interested in discovering subnetworks implementing hierarchical and linear rules, while pruning, we only use the negative log-likelihood of the first auxiliary in the question for computing the loss. To ensure that the discovered subnetworks are not just a by-product of the pruning procedure, we consider control groups, which are obtained by pruning randomly initialized networks.

Results. In Figure 2, we show the effect of different pruning methods on an intermediate model checkpoint, which does not yet generalize hierarchically. After `Train`-prune, roughly 80% heads of the full model are removed and in-distribution performance is conserved, though there is a drop in generalization performance (30% to 23%). After `Gen`-prune, we are able to find a subnetwork that achieves 100% generalization accuracy. This is striking, because the full network performed much worse. After `Train\Gen`-prune, we find a subnetwork that achieves 0% generalization accuracy while having 100% in-distribution performance; this subnetwork is behaviorally equivalent to the linear rule. Hence, these pruning experiments reveal the existence of subnetworks implementing different generalization behaviors. For the control groups, we find all three pruning methods to be unsuccessful obtaining 25% (i.e., random performance) on both the in-distribution and gen-

Figure 2: Pruning a transformer LM trained for 15000 steps using the three methods. Dark blocks mean the head is pruned and light means it is kept.

eralization test sets, providing further evidence that these subnetworks are not introduced by the pruning methods, and behaviors akin to the two rules are implemented within the language model.

We also analyze how these subnetworks evolve over the course of training and, interestingly, we find that once formed, these subnetworks continue to coexist over the course of training (Appendix Figure 8). This is true even when the behavior of the aggregate model becomes closer to the hierarchical rule with training; the competing linear-rule subnetwork does not really disappear. We hypothesize that the ambiguous training data (with two plausible generalizations, linear and hierarchical) is the reason for the existence of the subnetworks with very different generalization behaviors. To evaluate this hypothesis, we consider the case where the model is trained with *disambiguated* data—we augment the ambiguous training dataset with examples that are only consistent with the hierarchical rule and find the subnetwork corresponding to linear rule to disappear in this case (check Appendix Figures 8, 9 for details).

Effect of Head Capacity. We next investigate what effect the number of attention heads in the network has towards the presence of the two subnetworks. In addition to our original models trained with 8 heads per layer, we also train models with 16 heads, 4 heads, and 1 head and analyze the training dynamics using the same procedure as above. For models with 16 and 4 heads per layer (Figures 10a and 10b in the Appendix) we find results consistent with the 8 heads case—i.e., the two heads form early during the training and continue to co-exist throughout the training. Interestingly, for 1 head per layer case (Figure 10c in the Appendix), we find that both `Gen`-prune and `Train\Gen`-prune on average (across the 5 seeds) fail to discover their respective subnetworks. This is particularly noteworthy, because the full model with 1 head on average gets better generalization performance than the 4 heads per

layer model (0.7 for the former and 0.6 for the latter). Characterizing the behavior of the 1 head model remains an open question as we didn't find evidence of either of the two presupposed generalizations, which we leave for future work. Overall, these results show that the head capacity, i.e., the number of attention heads, might be crucial for the development of the subnetworks corresponding to distinct generalization behaviors.

5 A Bayesian Perspective on Hierarchical Generalization in Transformer LMs

A useful tool for understanding generalization in neural networks has been ''the simplicity bias'', the inductive bias towards simpler functions (De Palma et al., 2019) has been suggested as an explanation for why neural networks tend to generalize instead of overfitting the training data (Valle-Perez et al., 2019). Can we explain through ''simplicity'' the preference of the model towards hierarchical generalization over linear? Our main argument is that, when considering transformers trained with the language modeling objective, because the underlying data-generation process to be modeled produces each token in the full sequence, modeling the dependencies between the tokens hierarchically as opposed to learning a linear rule for each dependency might be simpler.[3] We leverage the Bayesian framework of Perfors et al. (2011), utilizing generative grammars to model data-generation processes corresponding to the hierarchical and linear rules, and operationalize the notion of simplicity and goodness of fit using the posterior probabilities of the grammars given the observed data. We then show a correlation between transformers' generalizing hierarchically and the training dataset being better explained using a hierarchical grammar than a grammar modeling the linear rule according to the posterior criterion.

5.1 Background

Operationalizing the Notion of Simplicity. Occam's Razor principle states that when two hypotheses explain the data equally well, the simpler one of the two is likely to be the correct one. This notion is mathematically formalized in Solomonoff's theory of inductive inference (Solomonoff, 1964) using a Bayesian approach by computing the posterior probabilities of the competing hypotheses and selecting the one with higher posterior, $p(h \mid D) \propto p(D \mid h) \cdot p(h)$. Here, $p(D \mid h)$ denotes the likelihood of the observed data D based on the hypothesis h, i.e., how well h fits the data D. $p(h)$ denotes the prior probability of h, which in Solomonoff's theory assigned higher values for simpler hypotheses h. Hence, by computing the posterior $p(h \mid D)$, Bayesian inference balances the tradeoff between the goodness of fit of a hypothesis (likelihood) and its simplicity (prior)—''Bayesian Occam's Razor''.

Probabilistic Grammars. Since our training objective is language modeling, we need to consider hypotheses that generate the entire sequence of tokens as represented in the training data. Following Perfors et al. (2011), we use generative grammars to model the data-generation process for language generation. For the purposes of this work we consider probabilistic context-free grammars (PCFGs) that can be represented using a 5-tuple, i.e., $G = \{V, \Sigma, R, S, \Theta\}$. Here, V denotes the set of nonterminal symbols that form phrases or constituents in a sentence, Σ denotes the set of terminal symbols or words in the sentences, $R \in V \times \{V \cup \Sigma\}^*$ denotes the set of production rules mapping phrases to sub-phrases or words, $S \in V$ is the start symbol that represents the whole sentence, and Θ denotes the probabilities on the production rules given each non-terminal. PCFGs are typically used to model the hierarchical phrase structure of a language. We can also apply some constraints to the form of production rules in R to obtain special cases (subsets) of CFGs. For example, regular grammars form a subset of CFGs whose production rules can be put into a right-linear form: $A \rightarrow bC$, where A and C are nonterminal symbols and b is a terminal.

We can view the data-generation process for dataset D using the probabilistic grammar G, and compute the posterior $p(G \mid D)$ to measure the simplicity and goodness of fit of a grammar G.

5.2 Method

We now give an overview of how we apply the Bayesian approach discussed above to understand hierarchical generalization in transformer LMs. We start by constructing a PCFG to model the hierarchical rule (denoted `CFG`) and a regular grammar (`Reg`) that generates data based on the linear

[3] Such an argument is implicit in the field of theoretical syntax where hierarchical representations are rife.

rule. We then generate data using both grammars – $\mathcal{D}_{\texttt{CFG}}$ from CFG and $\mathcal{D}_{\texttt{Reg}}$ from Reg. The intersection of the two datasets, $\mathcal{D}_{\texttt{CFG}} \cap \mathcal{D}_{\texttt{Reg}}$, contains ambiguous examples consistent with both the linear rule and hierarchical rule. We will use this as our training corpus $\mathcal{D}_{\text{train}}$. We then compute the posterior probabilities for both CFG and Reg given $\mathcal{D}_{\text{train}}$ and select the one with the higher posterior: $G^* = \arg\max_{G \in \{\texttt{CFG},\texttt{Reg}\}} p(G \mid \mathcal{D}_{\text{train}})$. We then train a transformer language model on $\mathcal{D}_{\text{train}}$, and check if it generalizes according to G^*. Specifically, if $G^* = \texttt{CFG}$, does the transformer follow the hierarchical rule, and if $G^* = \texttt{Reg}$, does the transformer follow the linear rule? The selection of G^* is intended to simulate ''idealized'' Bayesian learning, and check to what extent the transformer's learning behavior matches G^* across different scenarios. Below, we provide details of each of the steps.

Task. For this study we consider the simple agreement task, as constructing hierarchical and linear grammars for its data is straightforward.[4]

Constructing Grammars. Following Perfors et al. (2011), we hand-construct the CFG and regular grammars. The CFG is constructed so that each verb agrees with the hierarchically connected subject, while the regular grammar is constructed to follow the linear rule (verb agrees with the most recent noun). The constructed grammars are assigned uniform probabilities for the production rules, i.e., given a nonterminal, all productions are equally likely. For CFGs, we use Chomsky Normal Form for productions: Each production rule is of the form $A \to BC$ or $A \to a$, where A, B, C are nonterminals and a is a terminal symbol. Similarly, for the regular grammar Reg, we use the right-linear form for every rule: $A \to bC$ or $A \to a$. Like Perfors et al. (2011), we also adopt a type-based approach for constructing the grammars: Terminal symbols Σ instead of being the word tokens (e.g., *walrus*, *sing*) are syntactic categories (e.g., singular-noun, intransitive-verb, etc.), so that we can use these grammars to strictly model abstract syntactic structures and not vocabulary-type frequencies, and it also gives us a manageable number of possible generations by the grammars.

For both context-free and regular grammars we generate two variants, depending on the diversity of the sentence types generated by them:

Small Grammars `CFG-S` and `Reg-S`: Here we construct CFG and regular grammars that only generate 18 sentence types. Recall that a sentence type is a sequence of syntactic categories, e.g., sentences like *The walrus sings* can be represented by sentence type *determiner singular-noun intransitive-verb*. The hand-constructed CFG-S has 15 nonterminals and 21 production rules and Reg-S has 14 nonterminals and 22 production rules. Out of the 18 sentence types generated by both the grammars, 12 are common between the two (ambiguous) and 6 remaining in CFG-S that are only consistent with the hierarchical rule and 6 only consistent with linear rule in Reg-S.

Large Grammars `CFG-L` and `Reg-L`. Here, we consider larger grammars, which can generate much more diverse sentence types—180 sentence types. The major difference here is that these grammars are allowed to generate relative clauses with both the subject or object in the sentence. CFG-L has 25 nonterminals and 38 productions, while Reg-L has 41 nonterminals and 63 productions. Note that based on these numbers alone it is evident that we need much more complex regular grammars to generate diverse sentence types. Out of the 180 sentence types generated by each grammar, 120 are common between the two.

Generating Datasets. We generate the sentence types from each of the 4 grammars—$\mathcal{D}_{\texttt{CFG-S}}$, $\mathcal{D}_{\texttt{Reg-S}}$, $\mathcal{D}_{\texttt{CFG-L}}$, and $\mathcal{D}_{\texttt{Reg-L}}$. As mentioned before, the training dataset is constructed by considering the sentence types common between the CFG and corresponding regular grammar. We have $\mathcal{D}_{\text{train-S}} = \mathcal{D}_{\texttt{CFG-S}} \cap \mathcal{D}_{\texttt{Reg-S}}$ for the small grammars, and $\mathcal{D}_{\text{train-L}} = \mathcal{D}_{\texttt{CFG-L}} \cap \mathcal{D}_{\texttt{Reg-L}}$ for the larger ones. Note that these are the datasets of sentence-types, and transformers are trained on sentences. To generate sentences from these corpora of sentence types, we repeatedly sample sentence types, and replace the syntactic categories with the allowed tokens for that category (e.g., *determiner* can be replaced with *the*, *our*, *my*, etc.). Using this procedure we generate a corpus of 50k sentences from $\mathcal{D}_{\text{train-S}}$ and 50k sentences

[4] Question formation and tense reinflection involve pairs of sentences, where the second sentence is a transformed version of the first. Such sentence pairs would likely require more complex frameworks like synchronous grammars (Aho and Ullman, 1969), which we leave to future work.

from $\mathcal{D}_{\text{train}-\text{L}}$. To simplify the notation, we denote training data for transformers by $\mathcal{D}_{\text{train}-\text{S}}$ and $\mathcal{D}_{\text{train}-\text{L}}$ as well.

The generalization test sets are generated by considering the sentence types that are unique to a specific grammar. For example, we can have the test set $\mathcal{D}^{\texttt{Hier}}_{\text{test}-\text{S}} = \mathcal{D}_{\texttt{CFG-S}} \setminus \mathcal{D}_{\texttt{Reg-S}}$, which contains sentence types that are unique to $\mathcal{D}_{\texttt{CFG-S}}$ and hence only consistent with the hierarchical rule and not the linear rule. Similarly, $\mathcal{D}^{\texttt{Lin}}_{\text{test}-\text{S}} = \mathcal{D}_{\texttt{Reg-S}} \setminus \mathcal{D}_{\texttt{CFG-S}}$, consists of sentence types consistent only with the linear rule. We can equivalently define $\mathcal{D}^{\texttt{Hier}}_{\text{test}-\text{L}}$ and $\mathcal{D}^{\texttt{Lin}}_{\text{test}-\text{L}}$. While talking about the two datasets in general and not specifically about the small (S) or large (L) variants, we just use the notation $\mathcal{D}^{\texttt{Hier}}_{\text{test}}$ and $\mathcal{D}^{\texttt{Lin}}_{\text{test}}$.

Computing the Posterior for Each Grammar. Now that we have the four grammars constructed, we can compute the posteriors for the grammars given the corresponding training datasets. Note that, since we are only interested in comparing the posteriors of CFG and regular grammars, we can estimate the posterior by computing the likelihood and prior and taking product of the two, i.e., $p(G \mid D) \propto p(D \mid G)\, p(G)$. Recall that the prior probability of a grammar can be computed by calculating the probability of each of the choices that goes into defining that grammar: $p(G) = p(|V|) \prod_{k=1}^{|V|} p(P_k)\, p(\theta_k) \prod_{i=1}^{P_k} p(R_{k,i})$. Here, $|V|$ is the number of nonterminals, P_k is the number of productions from the k^{th} nonterminal with the probabilities of each production given by $\theta_k \in [0,1]^{P_k}$, and $R_{k,i}$ denotes the right hand side of the ith production rule from the kth nonterminal. Following Perfors et al. (2011), we use a geometric prior on $p(|V|)$ and $p(P_k)$, flat Dirichlet prior on θ_k, and compute $R_{k,i}$ depending on the nature of the production rule. We provide detailed computations of the prior in Appendix §A.5.1.

The likelihood $p(D \mid G)$, measures the probability that the dataset D is generated from the grammar G. For m sentence types in the dataset D, the likelihood is given by $p(D \mid G) = \prod_{i=1}^{m} p(S_i \mid G)$, where S_i's denote the sentence types in D. $p(S_i \mid G)$ is computed by taking product of the probabilities of production rules used to derive S_i using G (including adding the probabilities when multiple parses are possible for S_i). Note that computing $p(S_i \mid G)$ requires estimating the production probabilities θ_k from each nonterminal. We use the Inside-Outside algorithm (Baker, 1979) to obtain an approximate maximum likelihood estimate of the production probabilities on the dataset D.

Other Choices of Grammars. Given our generated training datasets ($\mathcal{D}_{\text{train}-\text{S}}, \mathcal{D}_{\text{train}-\text{L}}$), there can be grammars other than the four we constructed that can generate these datasets. In their analysis, Perfors et al. (2011) also consider two subsets of the regular grammars: Flat and One-state. Flat grammars have production rules which are the list of memorized sentences, i.e., of the form $S \to a_1 a_2 \cdots a_n$. Here $a_i's$ are terminal symbols and there are no nonterminals other than S. One-state grammars are equivalent to finite state automata with a single state and hence permit any terminal symbol to follow any other. We also include these two grammars in our analysis.

Further, even among the class of context-free and regular grammars, there might exist grammars with better posteriors on the training datasets $\mathcal{D}_{\text{train}-\text{S}}$ and $\mathcal{D}_{\text{train}-\text{L}}$ than the ones that we hand-construct. To remedy this, we also experiment with applying local search on our constructed grammars, using Bayesian model merging (BMM) (Stolcke and Omohundro, 1994) to minimize the grammars while improving the posterior on the respective training datasets.

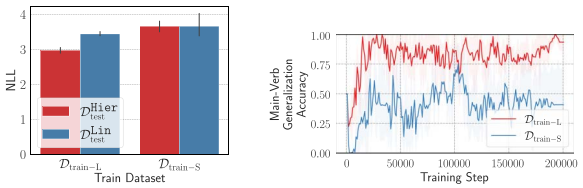
Explaining Generalization in Transformers. Recall that our goal has been to quantify the notion of simplicity of the two competing hypotheses (hierarchical and linear rule), which are consistent with the training data used to train transformer-based LMs. Our aim is to check whether the trained transformer LM exhibits generalization consistent with choosing the simpler (i.e., larger posterior) grammar. We evaluate this by comparing the negative log-likelihood (NLL) assigned by the transformer LM to the test sets corresponding to the two generalizations. For example, for the transformer model trained using data derived from $\mathcal{D}_{\text{train}-\text{L}}$, we evaluate its NLL (averaged over all examples) on the generalization test sets derived from the two grammars $\mathcal{D}^{\texttt{Hier}}_{\text{test}-\text{L}}$ and $\mathcal{D}^{\texttt{Lin}}_{\text{test}-\text{L}}$, and check if it assigns a lower NLL to the test data coming from the grammar with higher posterior. For a more intuitive metric, we also compute the main-verb accuracy from §3.1.

5.3 Results

Comparing Posteriors. The log-probabilities for all the hand-constructed grammars on the two

Grammar	$\mathcal{D}_{\text{train}-\text{L}}$ (120 types)			$\mathcal{D}_{\text{train}-\text{S}}$ (12 types)		
	log-Prior	log-Likelihood	log-Posterior	log-Prior	log-Likelihood	log-Posterior
`CFG`	−367	−639	**−1006**	−169	−34	−203
`Reg`	−619	−616	−1235	−190	**−30**	−220
`Flat`	−4567	**−574**	−5141	−281	**−30**	−311
`One-State`	**−58**	−2297	−2355	**−51**	−121	**−172**

Table 2: Comparing the log-probabilities for the grammars given $\mathcal{D}_{\text{train}-\text{L}}$ and $\mathcal{D}_{\text{train}-\text{S}}$.



(a) Negative log-likelihood (NLL)

(b) Main-verb generalization accuracy

Figure 3: Performance of transformer models trained on the $\mathcal{D}_{\text{train}-\text{L}}$ and $\mathcal{D}_{\text{train}-\text{S}}$ datasets.

datasets is provided in Table 2. On both datasets, the one-state grammar gets the highest *prior*, which is expected as it is the simplest grammar that we study, but also has the lowest log-likelihood. The flat grammars fit both the datasets the best and have the highest log-likelihood, which is also expected since a flat grammar memorizes the training data, but it comes at a cost of increased complexity as indicated by the lowest prior.

For the high diversity dataset $\mathcal{D}_{\text{train}-\text{L}}$, we observe that the CFG best balances the tradeoff between the simplicity and goodness of fit, obtaining the highest posterior. This shows why it would be more beneficial to model this dataset using a hierarchical phrase structured grammar than a linear grammar. However, when we consider the low-diversity dataset $\mathcal{D}_{\text{train}-\text{S}}$, while the CFG still obtains a better posterior than the regular grammar, it is the one-state grammar obtains the highest posterior out of all the grammars. This is consistent with the findings of Perfors et al. (2011), who found that for small corpora, one-state grammars often obtain higher posteriors than the context-free and regular grammars. In such cases, learning the distribution of syntactic category sequences, without abstract nonterminals, wins out on the Bayesian criterion.

We obtain consistent findings with some subtle differences for the grammars minimized using the BMM algorithm, which we detail in Appendix §A.5.2 and Table 3.

Sensitivity to Prior. Since choosing the prior is subjective and can influence results, we conduct a sensitivity analysis by varying the parameters of the geometric distributions used to define $p(|V|)$ and $p(P_k)$. We consider 49 different combinations of these parameters and find the results from Table 2 to hold for all choices (Appendix Figure 11).

Performance of Transformer-based LMs. We train the transformer-based LMs on the two datasets ($\mathcal{D}_{\text{train}-\text{L}}, \mathcal{D}_{\text{train}-\text{S}}$) and evaluate their generalization based on the $\mathcal{D}_{\text{test}}^{\texttt{Hier}}$ and $\mathcal{D}_{\text{test}}^{\texttt{Lin}}$ test sets. We use the same experimental setup as discussed in §3.2. In Figure 3a, we see for the models trained on the low-diversity dataset $\mathcal{D}_{\text{train}-\text{S}}$ that the model obtains similar negative log-likelihood values on both test sets, implying that the model has no preference for generalizing according to the linear rule or the hierarchical rule. For this dataset, neither the CFG nor the regular grammar were optimal in terms of the posterior probabilities, so we observe that the transformer's learning behavior is consistent with the ''idealized'' setup above. For the models trained on the $\mathcal{D}_{\text{train}-\text{L}}$ dataset, however, we see that the model learns to generalize hierarchically, with the NLL on the $\mathcal{D}_{\text{test}}^{\texttt{Hier}}$ test set being significantly lower than that on the $\mathcal{D}_{\text{test}}^{\texttt{Lin}}$ test set.

We see these findings reflected on the main-verb accuracy metric as well (Figure 3b), where the model trained on the $\mathcal{D}_{\text{train}-\text{L}}$ dataset obtains close to 100%, while the one trained on the $\mathcal{D}_{\text{train}-\text{S}}$ dataset obtains close to 50% generalization accuracy, again showing no preference for a hierarchical or linear rule. We also verify that these results are not just a by-product of the choice of hyperparameters, and train transformer models with different number of layers on the $\mathcal{D}_{\text{train}-\text{S}}$ dataset, and in none of the cases did we observe the models exhibiting preference for hierarchical generalization (see results in Appendix Figure 12).

Takeaways. Our results indicate that when transformers-based LMs are trained on data that can be generated from multiple grammars with different complexities, they exhibit generalization behavior consistent with the simplest grammar. Under tested conditions, we find that when the data is syntactically diverse, a hierarchical grammar not only fits the data well but also is simpler compared to the regular grammars with linear

agreements (in agreement with Perfors et al., 2011). In contrast, for low diversity data, we find that a non-hierarchical grammar gets higher posterior and transformers trained on such data do not exhibit any preference for hierarchical generalization.

Limitations. David Marr's famous three levels of analysis of an information processing system (e.g., human brain or AI) includes the *computational level*, where we consider what does an ideal solution to the problem that the system is solving looks like; *algorithmic level*, what algorithm might solve the problem in question; and *implementation level*, where we study how the representation and algorithm are implemented physically (Marr, 1982). In our work we only focused on the computational level analysis, i.e., understanding the behavior of transformer LMs in terms of an ideal (Bayesian) learner. Our results only provide a correlation between transformers generalizing hierarchically and training data being explained more effectively (based on the posterior criterion) by a hierarchical grammar than a regular grammar, showing an agreement with the idealised learner. However, we do not make any claim about the algorithmic and implementation level details in transformer LMs—i.e., whether these models internally learn the underlying grammars. There is some evidence in prior work showing transformer LMs to learn the probabilistic grammars when trained on data generated from them, e.g., Allen-Zhu and Li (2023), show transformer LMs internally learning the underlying PCFG when trained on data generated from the same. While this provides some encouraging support for our Bayesian interpretation, further investigation of the internal mechanisms of transformer LMs under our experimental setup is needed to establish a causal connection, which we leave to explore in future work.

6 Conclusion

In our work, we studied when and why transformers exhibit hierarchical generalization when trained on an otherwise ambiguous data consistent with both linear and hierarchical rules. There are multiple directions that can be explored in the future. While our results suggest language modeling as a source of hierarchical bias, it still remains unclear why hierarchical generalization is delayed (i.e., grokking). While the experiments concerning our Bayesian interpretation only involved the simple agreement tasks for which it was possible to construct CFGs, in future it would be interesting to explore methods to model the simplicity and goodness of fit for competing hypotheses for tasks involving transformation of an input sentence to output sentence. While we show a correlation between the Bayesian interpretation and the behavior of transformer LMs, future work can also look at establishing a causal connection between the two.

Acknowledgments

The authors would like to thank the anonymous reviewers and the action editor for their feedback and suggestions for new experiments and revisions that helped greatly improve the paper. We gratefully acknowledge support from the National Science Foundation under CAREER grant no. IIS2142739, and NSF grants no. IIS2125201 and IIS2203097. This work was also supported in part by gift funding from Google, Microsoft Research, and OpenAI.

References

Alfred V. Aho and Jeffrey D. Ullman. 1969. Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences*, 3:37–56. https://doi.org/10.1016/S0022-0000(69)80006-1

Zeyuan Allen-Zhu and Yuanzhi Li. 2023. Physics of Language Models: Part 1, Learning Hierarchical Language Structures. *ArXiv e-prints*, abs/2305.13673. Full version available at http://arxiv.org/abs/2305.13673

James K. Baker. 1979. Trainable grammars for speech recognition. *Journal of the Acoustical Society of America*, 65. https://doi.org/10.1121/1.2017061

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher

Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Giacomo De Palma, Bobak Kiani, and Seth Lloyd. 2019. Random deep neural networks are biased towards simple functions. *Advances in Neural Information Processing Systems*, 32.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation.

Robert Frank and Donald Mathis. 2007. Transformational networks. *Models of Human Language Acquisition*, 22.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings*.

Yongjie Lin, Yi Chern Tan, and Robert Frank. 2019. Open sesame: Getting inside BERT's linguistic knowledge. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 241–253, Florence, Italy. Association for Computational Linguistics.

Christos Louizos, Max Welling, and Diederik P. Kingma. 2018. Learning sparse neural networks through L_0 regularization. In *International Conference on Learning Representations*.

David Marr. 1982. The Philosophy and the Approach. In *Vision*, pages 8–37. W. H. Freeman.

R. Thomas McCoy, Robert Frank, and Tal Linzen. 2018. Revisiting the poverty of the stimulus: Hierarchical generalization without a hierarchical bias in recurrent neural networks. In *Proceedings of the 40th Annual Meeting of the Cognitive Science Society, CogSci 2018*, pages 2096–2101. The Cognitive Science Society.

R. Thomas McCoy, Robert Frank, and Tal Linzen. 2020. Does syntax need to grow on trees? Sources of hierarchical inductive bias in sequence-to-sequence networks. *Transactions of the Association for Computational Linguistics*, 8:125–140. https://doi.org/10.1162/tacl_a_00304

William Merrill, Nikolaos Tsilivis, and Aman Shukla. 2023. A tale of two circuits: Grokking as competition of sparse and dense subnetworks. In *ICLR 2023 Workshop on Mathematical and Empirical Understanding of Foundation Models*.

Aaron Mueller, Robert Frank, Tal Linzen, Luheng Wang, and Sebastian Schuster. 2022. Coloring the blank slate: Pre-training imparts a hierarchical inductive bias to sequence-to-sequence models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1352–1368, Dublin, Ireland. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.findings-acl.106

Aaron Mueller, Albert Webson, Jackson Petty, and Tal Linzen. 2024. In-context learning generalizes, but not always robustly: The case of syntax. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4761–4779, Mexico City, Mexico. Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.naacl-long.267

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. 2022. Transformers can do Bayesian inference. In *International Conference on Learning Representations*.

Shikhar Murty, Pratyusha Sharma, Jacob Andreas, and Christopher Manning. 2023. Grokking of hierarchical structure in vanilla transformers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 439–448,

Toronto, Canada. Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.acl-short.38

Amy Perfors, Joshua B. Tenenbaum, and Terry Regier. 2011. The learnability of abstract syntactic principles. *Cognition*, 118(3):306–338. https://doi.org/10.1016/j.cognition.2010.11.001, PubMed: 21186021

Matthew E. Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Brussels, Belgium. Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1179

Jackson Petty and Robert Frank. 2021. Transformers generalize linearly. ArXiv:2109.12036 [cs].

Ray J. Solomonoff. 1964. A formal theory of inductive inference. Part I. *Information and Control*, 7(1):1–22. https://doi.org/10.1016/S0019-9958(64)90223-2

Andreas Stolcke and Stephen Omohundro. 1994. Inducing probabilistic grammars by Bayesian model merging. In *International Colloquium on Grammatical Inference*, pages 106–118. Springer. https://doi.org/10.1007/3-540-58473-0_141

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 27.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? Probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Guillermo Valle-Perez, Chico Q. Camargo, and Ard A. Louis. 2019. Deep learning generalizes because the parameter-function map is biased towards simple functions. In *International Conference on Learning Representations*.

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*, volume 30. Curran Associates, Inc.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1580

Zhiyong Wu, Yun Chen, Ben Kao, and Qun Liu. 2020. Perturbed masking: Parameter-free probing for analyzing and interpreting BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4166–4176, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.383

A Appendix

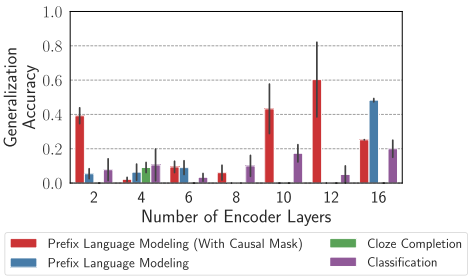
A.1 Code and Datasets

We release our code and datasets for public use: https://github.com/kabirahuja2431/transformers-hg.

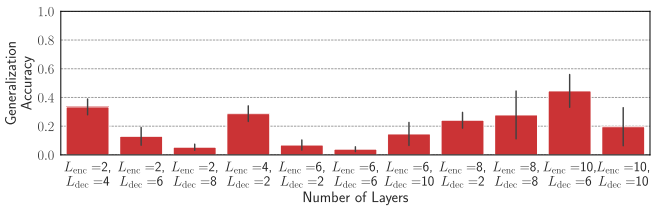
A.2 Tasks and Datasets

Below we provide the details of the five tasks.

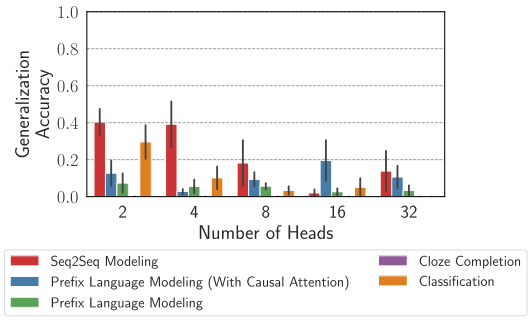
1. Question Formation. As described above, the task is to transform a declarative sentence into a question. We use the dataset from McCoy et al. (2020) for this task, which was constructed from a context-free grammar (CFG) with three sentence types varying in the existence and position of the relative clause (RC) in the sentence: (i) no RC, e.g., sentence *The walrus does sing*; (ii) RC attached to the object, e.g., sentence 1a; and (iii) RC attached to the subject, e.g., sentence 2a. The training data includes (i) declarative-question pairs where the task is to take a declarative sentence and generate a question as output and (ii) identity pairs where the task requires copying an input declarative sentence. The declarative-question pairs in the training set only contain sentences without any RC or with RC attached to the object. Importantly, the identity pairs in the training data also include sentences with RC *on the subject*, to expose the



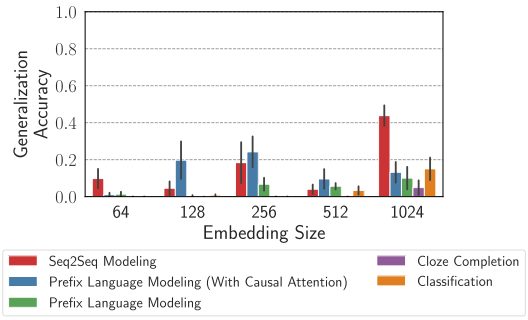
(a) Effect of model depth for PrefixLM and Cloze Completion training objectives



(b) Effect of model depth for Seq2Seq training objective.



(c) Effect of number of heads for different training objectives



(d) Effect of embedding size for different training objectives

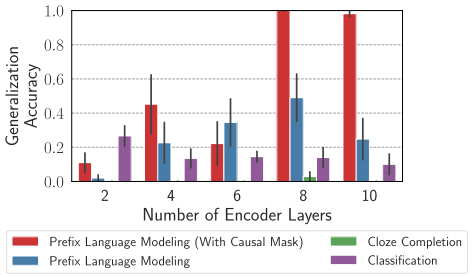
Figure 4: Robustness of negative results for non-language modeling objectives across different hyperparameter setting for the question formation task. For each experiment, we vary one hyperparameter while keeping the other two fixed to default values, i.e., 6 layers, 8 heads, and 512 embedding dimension. Note that the language modeling objective achieves an average generalization accuracy of 0.76 with the default hyperparameters.

Grammar	$\mathcal{D}_{\text{train-L}}$ (120 types)			$\mathcal{D}_{\text{train-S}}$ (12 types)		
	log-Prior	log-Likelihood	log-Posterior	log-Prior	log-Likelihood	log-Posterior
CFG*	−345	−639	**−984**	−112	−42	**−155***
Reg*	−393	−658	−1051	−125	−34	−159
Flat	−4567	**−574**	−5141	−281	**−30**	−311
One-State	**−58**	−2297	−2355	**−51**	−121	−172

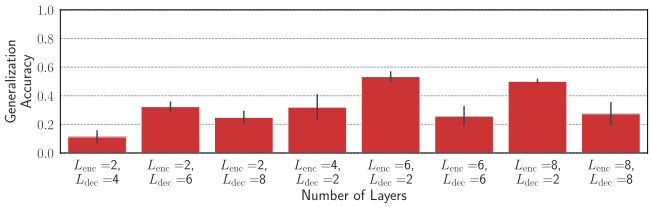
Table 3: Comparing the log-probabilities for each of the 4 grammars after performing BMM on the CFG and Reg grammars given the training datasets $\mathcal{D}_{\text{train-L}}$ and $\mathcal{D}_{\text{train-S}}$. The superscript $*$ symbol on log-posterior for CFG* on $\mathcal{D}_{\text{train-S}}$ indicates that while the results show highest posterior for this grammar, after minimization the grammar no longer models the hierarchical rule and starts to also generate sentence types consistent with the linear rule.

model to sentences of this type (McCoy et al., 2020). During training a token quest or decl is added to specify whether to perform question formation or the copy task. Following past work (e.g., Murty et al., 2023), we evaluate the model on the *first-word accuracy*—given the declarative sentence as the input, evaluate whether the model predicts correct auxiliary for the first word in the generated question.

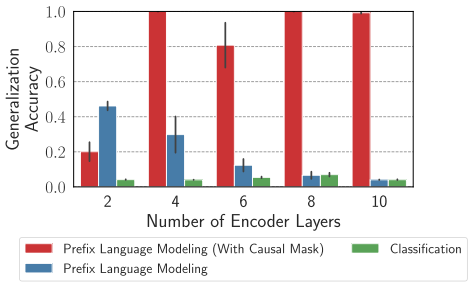
2. Question Formation (German). This is the same task as above, but the sentences are in German instead of English. We use the dataset from Mueller et al. (2022), consisting of sentences with the modals *können/kann* (can) or auxiliaries *haben/hat* (have/has), together with infinitival or past participle main verbs as appropriate, which can be moved to the front similar to English to form questions. The dataset construction and



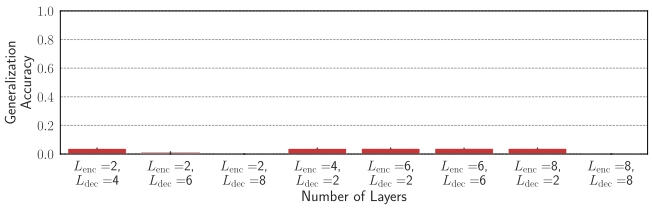
(a) Effect of model depth for PrefixLM, Classification, and Cloze Completion training objectives on the question formation German task.



(b) Effect of model depth for Seq2Seq training objective on the question formation German task.



(c) Effect of model depth for PrefixLM, Classification, and Cloze Completion training objectives on the passivization task.



(d) Effect of model depth for Seq2Seq training objective on the passivization task.

Figure 5: Robustness of negative results for non-language modeling objectives across different model depths for question formation German and passivization tasks. Note that the language modeling objective achieves roughly an average generalization accuracy of 1.0 with the default hyperparameters. We omit plots for heads and embedding dimension in interest of space, but we obtain consistent results with what we observe here.

evaluation metrics remain identical to the English version.

3. Passivization. The task here is to transform an active sentence to passive. We use the dataset from Müller et al. (2022) and similar to question formation, active-passive pairs in the training data are ambiguous—compatible with both rules: the hierarchical rule which involves identifying the object in the sentence and moving it to the front, and the linear rule that moves the second noun in the sentence to front. Such sentences correspond to ones without a prepositional phrase (PP) or with a PP on the object. For generalization, the model is evaluated on sentences with PP on the subject, for which only hierarchical rule is applicable. The training data here also is augmented with identity active-active pairs which consist of sentences of all the three types. For evaluation, following Mueller et al. (2022), we consider *object noun accuracy*, which measures whether the correct noun was moved to the subject position.

4. Tense Reinflection. In this task, we are given a sentence in the past tense, and the task is to transform it into present tense. While performing the transformation to present tense, the model has to figure out from the context whether each verb should be singular or plural (*-s* suffix) in the present tense. In this case, the hierarchical rule requires each verb to agree with the hierarchically determined subject and the linear rule requires a verb to agree with the most recent noun in the sequence. We use the same dataset as McCoy et al. (2020), where, similar to question formation, the training dataset contains tense reinflection pairs (past-present) that are consistent with both rules, and identity pairs (past-past) for copying. The models are evaluated using *main-verb accuracy*, which is calculated as the fraction of examples in the test set for which the generated present tense sentence has the correct main verb.

5. Simple Agreement. We also introduce a simplified version of the tense reinflection task. Unlike others, simple agreement is a single-sentence task where only the *present*-tense sentences from the tense-inflection are considered. In this task, we evaluate the model's ability

to generate the correct inflection of the verb at the end of the sentence. The hierarchical and linear rules are defined in the same way as tense reinflection. For evaluation, since from the context it is no longer clear what should be the correct verb, we use *main-verb contrastive accuracy*, which is calculated by considering each sentence in the test dataset (e.g., *my **zebra** by the yaks swims*), forming the prefix (*my **zebra** by the yaks*) and checking if the model assigns the higher probability to the correct inflection of the main verb in the original sentence (*swims* vs. *swim*).

A.3 Training Objectives and Hierarchical Generalization

A.3.1 Details About Training Objectives

Here we detail the input-output structure for all objectives concerning the five tasks that we study.

Language Modeling. As discussed in the main text, for the question formation task we simply consider the sequence s as declarative-question pair (or declarative-declarative pair for copy task), e.g., $s = \{\text{my}, \text{walrus}, \cdots, \texttt{quest}, \text{does}, \cdots, \text{move}, ?\}$. Similarly, for passivization it is the active-passive sentence pair (or active-active); for tense reinflection it is the pair of past and present tense sentence (or past-past), and for simple agreement it is simply the single input sentence.

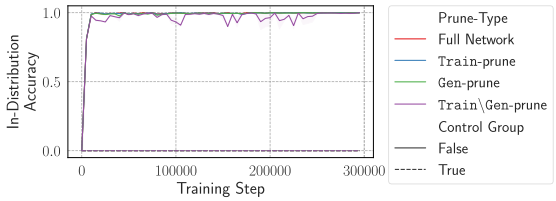
Sequence-to-sequence Modeling and Prefix Language Modeling. The inputs for the two objectives are the declarative sentence (or active sentence for passivization and past tense sentence for tense reinflection) and the outputs sequences are the corresponding questions (or passive sentence/present tense sentence depending on the task). Note that all four tasks allow identity pairs, hence the outputs can be the same as the inputs when `decl` token is provided at the end of the input. One modification that we make to how the prefix-LM objective is typically used is that we use a causal mask for the prefix tokens as well instead of having bi-directional attention over the prefix tokens, since we found the latter to perform subpar in our initial experiments (unstable in-distribution performance).

Sequence Classification. For question formation, the input is the declarative sentence, and the output is the four possible auxiliary tokens, $\{\text{do}, \text{does}, \text{don}'\text{t}, \text{doesn}'\text{t}\}$ for English and $\{\text{können}, \text{kann}, \text{haben}, \text{hat}\}$ for German. For passivization task, the input is the sentence in active voice and the output is the subject of the passive sentence, which can be any of the 26 nouns in the datasets vocabulary. For tense reinflection, the input is the sentence in past tense and the output is the present tense form of the main-verb in the input sentence (18 classes corresponding to the verbs in dataset). For simple agreement, the input is the sequence of tokens until the main verb and the model needs to predict the main-verb as a multi-label (across vocabulary of 18 verbs) classification task. The classification head for all tasks excluding tense reinflection, is attached to the last token in the sequence. For tense reinflection it is attached to the main-verb in the input sentence as otherwise the linear-rule which uses the noun most recent to the main-verb might not be appropriate. We also use causal mask for all tasks, as we found the models to perform better on in-distribution test set in our initial experiments when using it. Also, note that due to the nature of the objective, identity pairs are not supported.

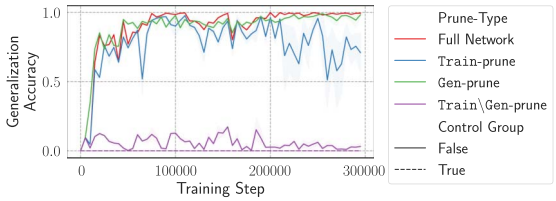
Cloze Completion. For the question formation task, we consider the declarative-interrogative pair and mask out tokens in the interrogative sentence at all positions where the auxiliaries could be present. Specifically, we have mask tokens where i) the auxiliary is present in the interrogative sentence or ii) the auxiliary was present in the original declarative sentence. The model is trained to predict the correct auxiliary at the right positions and `<EMPTY>` if an auxiliary is not present at a particular position. Similarly, for tense reinflection, we consider the past-present sentence pair, mask out all the verbs in the present tense sentence and train the model to predict the right form of the verbs. In the simple agreement task, we consider only the present tense sentence, mask out all the verbs and train the model to predict them. Here also we found using causal mask helps in better in-distribution performance and hence use it in all our experiments.

A.4 Pruning for Tasks Other than Question Formation

In the main paper under §4 our results on the discovery of subnetworks with different generalization performances were performed on question formation task. Here, we provide the results for tense reinflection and simple agreement. For

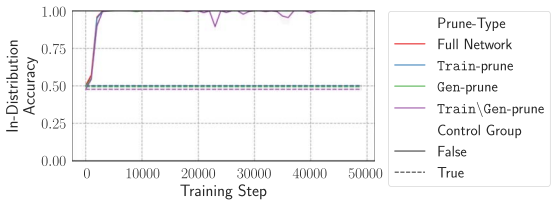


(a) In-distribution accuracy for different pruning methods across training (original ambiguous training data)

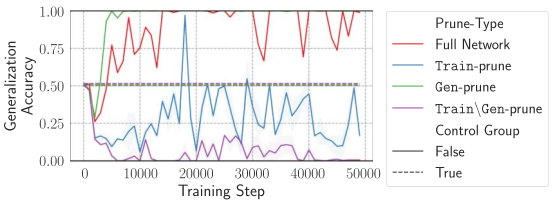


(b) Generalization accuracy for different pruning methods across training (original ambiguous training data)

Figure 6: Tracking training dynamics w.r.t. to the three pruning methods for tense reinflection task.



(a) In-distribution accuracy for different pruning methods across training (original question formation training data)



(b) Generalization accuracy for different pruning methods across training (original question formation training data)

Figure 7: Tracking training dynamics w.r.t. to the three pruning methods for simple agreement task.

tense-reinflection, we slightly modify the pruning procedure. Since, for tense reinflection, we need to generate the entire present tense sequence to check if the predicted main verb is in the correct form, we compute the loss over all output tokens during pruning unlike question formation, where only the loss on the first auxiliary in the question was computed. Due to this, for `Train\Gen`-prune training becomes highly unstable as the procedure involves minimizing training and maximizing the test loss. Hence, we propose an alternate `Train\Gen`-prune procedure for this task, where we generate a ‘‘linear-rule’’ version of the generalization set, where the sentence pairs are generated in such a way that they are only consistent with the linear-rule. Note that this can be done by simply taking a past tense sentence in the generalization set and flipping the inflection of the main-verb based on the agreement with the most recent noun preceding the verb. Note that similar to `Gen`-prune, here also we only use 1% of the total data from the ‘‘linear-rule’’ generalization set for pruning to avoid the possibility of overfiting. For simple agreement the procedure remains same as question formation, with the only difference that the loss is computed on the main-verb in this case during pruning instead of the auxiliary. Pruning results for the two tasks are provided in Figures 6 and 7. We find results consistent with our findings for question formation task here as well, where the ‘‘linear-rule’’ and ‘‘hierarchical-rule’’ subnetworks can be found using pruning and continue to co-exist over the course of training.

A.5 Grammar Details

A.5.1 Prior Computation

The prior probability of a grammar can be computed by calculating the probability of each of the choices that goes into defining that grammar:

$$p(G) = p(|V|) \prod_{k=1}^{|V|} p(P_k)\, p(\theta_k) \prod_{i=1}^{P_k} p(R_{k,i}). \quad (1)$$

Here, $|V|$ is the number of nonterminals, P_k is the number of productions from the k^{th} nonterminal with the probabilities of each production given by $\theta_k \in [0,1]^{P_k}$, and $R_{k,i}$ denotes the right hand side of the ith production rule from the kth nonterminal. Following Perfors et al. (2011), we use a geometric prior on $p(|V|)$ and $p(P_k)$, flat Dirichlet prior on θ_k, and compute $R_{k,i}$ depending on the nature of the production rule.

Recall that the geometric distribution is given by $p(n;p) = (1-p)^{n-1}p$, where p is a parameter of the geometric distribution, often interpreted as

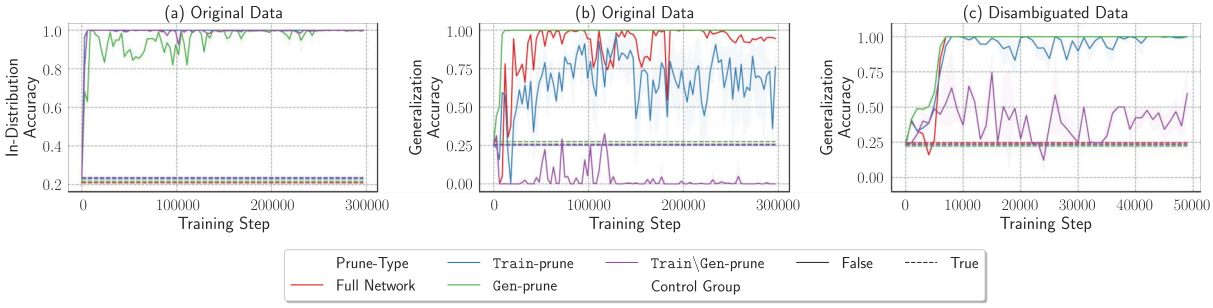


Figure 8: Tracking training dynamics with respect to the three pruning methods's subnetworks and the full network. (a) and (b): in-distribution and generalization accuracies of the LMs trained on the original ambiguous question formation data after pruning using the three methods, (c): generalization accuracy after pruning the model trained on disambiguated data. For models trained with original data, we can discover sub-networks consistent with hierarchical rule as well as the linear rule, indicated by the 0% generalization performance and 100% in-distribution performance for `Train\Gen`-prune curve, and 100% accuracy on both test sets for `Gen`-prune curve, roughly after 9k training steps. On the other hand, for the models trained with disambiguated data, linear rule subnetwork is not found (indicated by the curve corresponding to `Train\Gen`-prune never approaching 0% generalization accuracy). Note that disambiguated data was constructed by augmented original training dataset with 10k additional examples which are only consistent with the hierarchical rule.

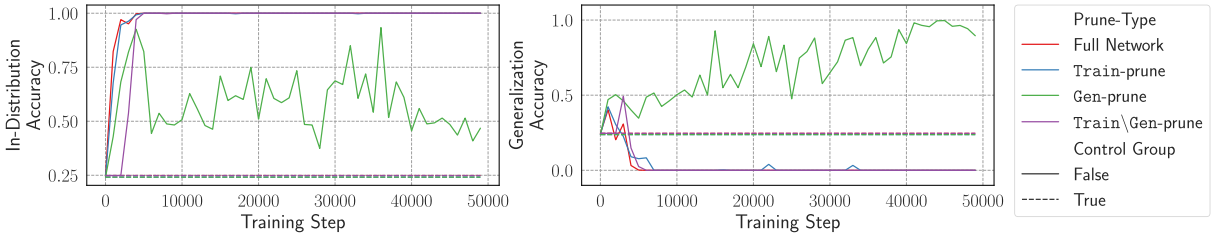
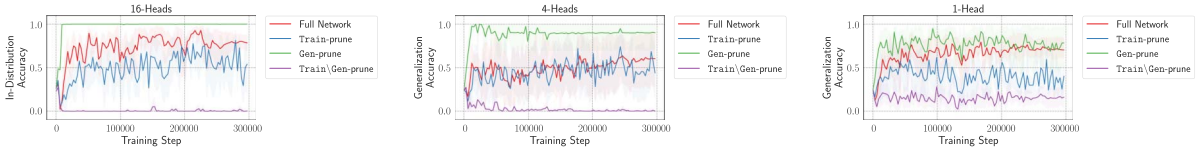


Figure 9: Training dynamics of transformer LM trained with question-formation data which is disambiguated with examples consistent only with the linear rule (by augmenting 10k such examples to the original 100k ambiguous examples). As can be seen, the full network in this case after a few thousand steps plateaus at 0% generalization performance, which is expected since only the linear rule is applicable to the entire dataset. Further, even `Gen`-prune in this case fails to find subnetworks with 100% in-distribution as well as 100% generalization performance. While further during training, `Gen`-prune does find subnetworks with higher generalization performance, the in-distribution performance at these points is very low, meaning the subnetwork isn't actually consistent with the hierarchical rule.

the probability of success, and a geometric distribution models the probability of success after n trials. Hence, choosing a geometric prior penalizes the grammars with a large number of nonterminals ($|V|$) and productions per nonterminal (P_k). In our experiments we use $p = 0.5$, following Perfors et al. (2011), but we conduct a sensitivity analysis on the choice of this parameter. For θ_k, we use a flat (i.e., $\alpha = 1$) Dirichlet prior, a popular choice for modeling probabilities for categorical distributions ($K - 1$ simplex). Note that since the Dirichlet is a continuous distribution, the probability of any specific θ_k is zero and we use the discrete relaxation from (Perfors et al., 2011) to model $p(\theta_k)$. The probability of the production rule $p(R_{k,i})$, depends on the type of grammar. For CFGs, since we consider them in CNF, the production rules are of the form $A \rightarrow BC$ or $A \rightarrow a$, hence the probability of the right hand side can be given by, $p(R_{k,i}) = \frac{1}{2}\frac{1}{|V|^2}\mathbb{1}(|R_{k,i}| = 2) + \frac{1}{2}\frac{1}{|\Sigma|}\mathbb{1}(|R_{k,i}| = 1)$. Since the regular grammars are in the right linear form i.e., productions of the form $A \rightarrow bC$ or $A \rightarrow a$, we can compute $p(R_{k,i}) = \frac{1}{2}\frac{1}{|\Sigma|}\frac{1}{|V|}\mathbb{1}(|R_{k,i}| = 2) + \frac{1}{2}\frac{1}{|\Sigma|}\mathbb{1}(|R_{k,i}| = 1)$. One might notice that we are missing the probability of number of terminal symbols $p(\Sigma)$ in the prior equation. We ignore this because both the CFG and regular grammars have the same number of terminals in our experiments, and since we are interested in just comparing the probabilities, the



(a) Training dynamics for Transformer LM with 16 heads per attention layer

(b) Training dynamics for Transformer LM with 4 heads per attention layer

(c) Training dynamics for Transformer LM with 1 head per attention layer

Figure 10: Effect of head capacity on the presence of hierarchical and linear generalization subnetworks. Runs are averaged over 5 seeds.

inclusion or exclusion of $p(\Sigma)$ doesn't make a difference.[5]

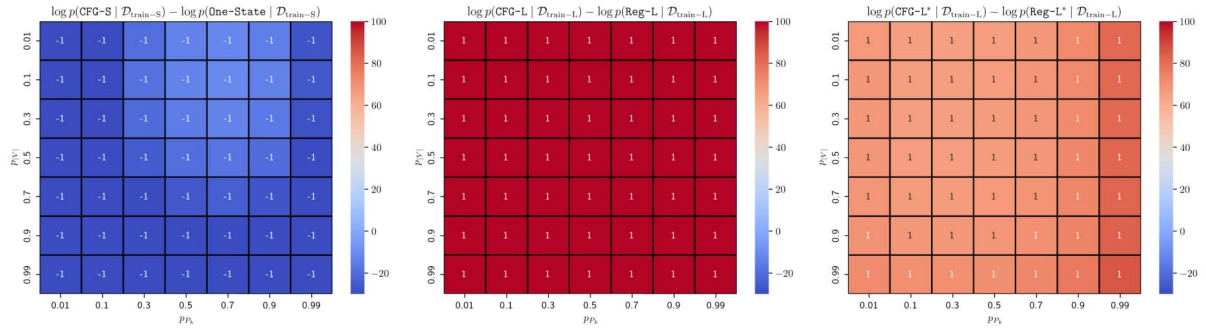
A.5.2 Local Search Using Bayesian Model Merging

The hand-constructed grammars that we consider in our study might not be optimal in terms of the posterior given the training data. For example, there might be some redundant production rules or non-terminals, which can be removed by merging two or more non-terminals. We use the Bayesian model merging (BMM) algorithm from Stolcke and Omohundro (1994) and Perfors et al. (2011) to perform a local search for grammars with higher posteriors starting from the hand-constructed ones. The algorithm works as follows: We start from the initial grammar and iterate over all possible merges. A merge involves replacing two non-terminal symbol by a single new non-terminal and adding two new productions mapping the new non-terminal to the older ones. For example, for production rules $A \rightarrow BC$, $A \rightarrow BD$, we can merge C and D to F, resulting the new production rules: $A \rightarrow BF$, $F \rightarrow C$, and $F \rightarrow D$. For each merge, we thus obtain a new grammar, and compute its posterior. We then select the grammar with the highest posterior (greedy search) and repeat the procedure with this new grammar. If no merge results in a grammar with higher posterior than the initial grammar, we terminate the search. We denote the context free grammars after merge as $\texttt{CFG}^*$ ($\texttt{CFG-S}^*$ and $\texttt{CFG-L}^*$) and regular grammars as $\texttt{Reg}^*$ ($\texttt{Reg-S}^*$ and $\texttt{Reg-L}^*$).

An important detail to note here is that while performing the merging algorithm, we use the ambiguous corpus $\mathcal{D}_{\text{train-L}}$ or $\mathcal{D}_{\text{train-S}}$ for computing the posteriors and hence searching the right set of merges. The final grammar obtain while it should assign high likelihood to the ambiguous training data, might no longer be consistent with the held out sentence types, e.g., $\mathcal{D}^{\texttt{Hier}}_{\text{test}}$ or $\mathcal{D}^{\texttt{Lin}}_{\text{test}}$, and hence the final grammars obtained might not strictly model the linear or hierarchical rules. To check if such a situation arises in our case, we compare the set of all generations from a grammar before and after merging. If the two are same, it implies that the grammar continues to be consistent with both the ambiguous and unambiguous sentence types, and hence obey the linear or order rule of the original hand-constructed grammar. We find that for $\texttt{CFG-S}$, after applying the merging algorithm, the grammar obtained is no longer consistent with just the hierarchical rule and starts to also generate sentence types consistent with the linear rule. This implies that **for the low-diversity data case, even using a CFG it is better to avoid modeling the hierarchical rule given the ambiguous data**. For $\texttt{CFG-L}$, the grammar remains consistent with the hierarchical rule even after merging.

The log-probabilities after applying BMM algorithm are provided in Table 3. For the $\mathcal{D}_{\text{train-L}}$ dataset, we find that our results remain consistent with those for hand-constructed grammars in Table 2: $\texttt{CFG-L}^*$ obtains a lower posterior than $\texttt{Reg-L}^*$. On the other hand for the $\mathcal{D}_{\text{train-S}}$ dataset, $\texttt{CFG-S}^*$ ends up with a higher posterior than the $\texttt{One-State}$ grammar. However, as noted above after minimization $\texttt{CFG-S}^*$ is no longer consistent with the hierarchical rule, i.e., doesn't generate sentences where verbs only agree with the hierarchically connected nouns. Hence, our observations that for the lower-diversity case, modeling the hierarchical rule is not optimal according the posterior criterion, remains consistent here as well.

[5] One might also notice that $p(G)$ allows some probability for generating the same rule more than once; it ''leaks'' probability mass. No prior literature, to our knowledge, suggests that this should pose a problem to our analysis.

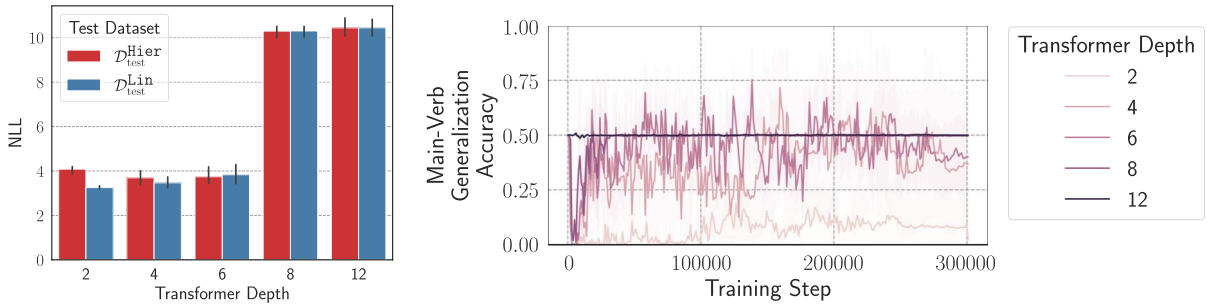


(a) Low diversity data case – $\mathcal{D}_{\text{train}-\text{S}}$ for hand-constructed grammars.

(b) High diversity data case – $\mathcal{D}_{\text{train}-\text{L}}$ for hand-constructed grammars.

(c) High diversity data case – $\mathcal{D}_{\text{train}-\text{L}}$ for BMM minimized grammars.

Figure 11: Sensitivity analysis on varying the geometric distribution parameter $p_{|V|}$ for $p(|V|)$ and p_{P_k} for $p(P_k)$. We plot the difference between the log-posterior of the CFG and the other grammar with the highest posterior, which is `One-State` for $\mathcal{D}_{\text{train}-\text{S}}$ and `Reg-L` (or `Reg-L`* for BMM minimized case) for $\mathcal{D}_{\text{train}-\text{L}}$. The values in the heatmaps correspond to the sign of the difference between the posteriors (1 for positive and -1 for negative). A positive sign implies that the CFG has the higher posterior than the alternate grammar and negative sign implies otherwise. For each of these combinations, we find that for $\mathcal{D}_{\text{train}-\text{S}}$ case, consistent with Table 2 the `CFG-S` always obtain a lower posterior compared to the `One-State` grammar. Similarly for the `CFG-L` and `Reg-L`, the findings are also consistent across all 49 combinations i.e., `CFG-L` always obtain a higher posterior than `Reg-L`. This holds for the BMM minimized grammars as well, where for $\mathcal{D}_{\text{train}-\text{L}}$ case `CFG-L` always obtain a higher posterior than `Reg-L`. Note that since after minimization on the smaller grammars (`CFG-S` and `Reg-S`), we are left with no grammar obeying the hierarchical rule, we skip sensitivity analysis for that case.



(a) Negative log-likelihood (NLL) on the $\mathcal{D}^{\texttt{Hier}}_{\text{test}}$ and $\mathcal{D}^{\texttt{Lin}}_{\text{test}}$ test datasets.

(b) Main-verb generalization accuracy on $\mathcal{D}^{\texttt{Hier}}_{\text{test}}$ test set.

Figure 12: Do transformer models trained on $\mathcal{D}_{\text{train}-\text{S}}$ ever show hierarchical generalization? We vary the depth of the transformer-LM (number of decoder layers) and find in no case, transformer exhibiting hierarchical generalization. Interestingly, for smaller depths, we see the models generalizing according order rule, indicated by lower NLL on $\mathcal{D}^{\texttt{Lin}}_{\text{test}}$ than $\mathcal{D}^{\texttt{Hier}}_{\text{test}}$ and a main-verb accuracy of roughly around 0%/ when transformer depth is 2. For depths greater than 4, we observe starts to show no preference for either the linear or hierarchical rule.