Learning Syntax Without Planting Trees: Understanding When and Why Transformers Generalize Hierarchically

Kabir Ahuja [1] **Vidhisha Balachandran** [2] **Madhur Panwar** [3] **Tianxing He** [1] **Noah A. Smith** [1 4] **Navin Goyal** [3]
Yulia Tsvetkov [1]

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 failed to lead to hierarchical generalization, language modeling objective consistently led to transformers generalizing hierarchically. We then study how different generalization behaviors emerge during the training by conducting pruning experiments that reveal 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]University of Washington, Seattle, WA, USA [2]Carnegie Mellon University, Pittsburgh, PA, USA [3]Microsoft Research, India [4]Allen Institute for AI, USA. Correspondence to: Kabir Ahuja <kahuja@cs.washington.edu>, Yulia Tsvetkov <yuliat@cs.washington.edu>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

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). 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 & Mathis, 2007; McCoy et al., 2018; 2020; Petty & 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 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, pre-

fix language modeling, sequence classification and cloze completion – and study the generalization behavior of transformers trained from scratch 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*. Since our aim is to understand hierarchical generalization in transformers in isolation, following prior work, 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).

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. We also find that these disparate subnetworks exist due to the ambiguity in the training data, as we find different subnetworks to disappear upon adding disambiguating examples (i.e., only consistent with the hierarchical rule).

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 reveal a *correlation between transformer LMs generalizing hierarchically and hierarchical grammars having higher posterior, compared to regular grammars*, thereby providing a potential explanation to transformers' preference for 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*.

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 phrase structure of the language. We call this *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.

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.

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.

For all tasks involving transformation of inputs (i.e., all except simple agreement), the datasets also include input identity pairs. E.g., 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, however, 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 familiarise 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., 2023), for evaluating question formation (both English and German) we consider 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. 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. We denote the metrics as textitin-distribution accuracy and *generalization accuracy* depending on the test set used for evaluation.

3. How the Training Objective Influences Hierarchical Generalization

Prior work by McCoy et al. (2020), Petty & 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 isn't 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 seq2seq evaluation for this task.

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.

Sequence classification. Here the model is trained to map the entire sequence to a discrete label. E.g., 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 4-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. E.g., 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. [1] 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 inflection, 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 also considered other choices of number of layers for the seq2seq objectives and found results consistent with our findings (see Appendix Figure 5). We use Adam optimizer (Kingma & 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.

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 & Frank (2021) and Mueller et al. (2023). Interestingly, seq2seq and prefix-LM bear the greatest resemblance to the language modeling objective, as these two also involve generating the whole output sequence. The major difference between language modeling and these two objectives is that language modeling involves computing the loss over all the tokens, including the input tokens, which indicates that the corresponding loss terms from modeling the input tokens might be crucial for hierarchical generalization. Our hypothesis on why that might be important is that when considering loss over all the tokens, the model cannot just simply learn a trivial transformation (e.g., for question formation, move the first auxiliary to the beginning and copy rest of the tokens from input) from input sequence to output sequence to minimize the loss (as it needs to model the input token distribution as well).[2]

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

[2] This corresponds to the classical difference between a *generative* model, i.e., one trained to model the full distribution of the data (including inputs) and a *discriminative* one that models the conditional distribution of outputs given inputs.

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 75%. 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 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), the presence of rich hierarchical cues in the data can aid in hierarchical generalization.

Takeaways. Overall, our experiments indicate language modeling as a source of inductive bias for the models to generalize hierarchically. We hypothesize that the reason LMs learn the hierarchical rule rather than the linear rule is that modeling the hierarchical phrase structure is beneficial for modeling the distribution over full sequence of tokens. We will explore this hypothesis in more depth 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 they 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 generalization consistent with the linear rule, i.e., obtains 0% generalization accuracy and 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.5 (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 the three pruning methods to be unsuccessful obtaining 25% (i.e., random performance) on both the in-distribution and generalization test sets, providing further evidence that these subnetworks are not introduced by the pruning methods, and behaviors

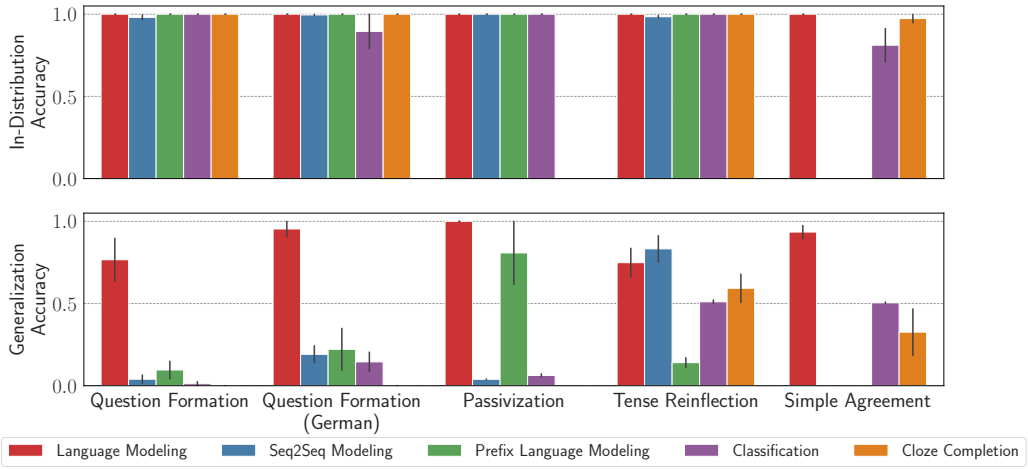


Figure 1: Effect of training objective on hierarchical generalization in transformers. The error bars correspond to the standard errors across 5 seeds.

akin to the two rules are implemented within the model.

Training dynamics. We now consider how these different subnetworks evolve over the course of the model training. To this end, we save checkpoints after every 1000 steps of training and perform the three kinds of pruning on those checkpoints. As shown in Figure 3(b), the "linear-rule" subnetwork becomes discoverable through pruning at roughly 6000 training steps – indicated by the 0% generalization performance for `Train\Gen`-prune curve and 100% in-distribution performance in Figure 3(a). Soon after, we see in Figure 3(b), the formation of a (hierarchical) generalization subnetwork, at around 9000 training steps, where the subnetwork obtained using `Gen`-prune obtains 100% generalization accuracy (the full network at this point only obtains 5% generalization performance). While there are occasional spikes in the generalization performance of subnetworks found using `Train\Gen`-prune during the first-thirds of the training, during majority of the training (especially the latter two-thirds) both the linear and hierarchical generalization subnetworks continue to be discoverable over the course of training – as indicated by stable 100% generalization accuracy of subnetworks found by `Gen`-prune and 0% generalization accuracy for `Train\Gen`-prune subnetworks in Figure 3(b) .

Our experiments reveal that, throughout training, there is competition between the two sub-networks, and while the behavior of the aggregate model becomes closer to the hierarchical rule with training, the competing linear-rule subnetwork does not really disappear. All three pruning methods are unsuccessful on the control group (randomly initialized networks), providing further evidence that these subnetworks are not introduced by the pruning methods, and behaviors akin to the hierarchical and linear rules are implemented within the language model.

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 (We add 10k such examples to the existing dataset containing 100k examples). We plot the training dynamics of the model trained with this disambiguated data in Figure 3(c). The full model without any pruning in this case, as expected, generalizes perfectly after a few thousand training steps without any noise, in contrast with generalization accuracy of the full-model trained on ambiguous data in Figure 3(b). More interestingly, Figure 3(c) shows that `Train\Gen`-prune fails to yield subnetworks that obtain 0% generalization accuracy, in contrast to the ambiguous data case in Figure 3(b). To make sure this is not due to the choice of our pruning hyperparameters, we conduct an extensive hyperparameter search, consisting of 128 combinations of the pruning learning rate, regularization penalty, and pruning steps (using Bayesian optimization) and still fail to find the setting where the `Train\Gen`-prune succeeds for the disambiguated data case (see Figure 8 in Appendix). This strongly suggests that the "linear-rule" subnetwork is never formed in the language model when it doesn't have a reason to be learned – when the alternate generalization behavior is no longer applicable to the entire training dataset.

We also experiment with the opposite disambiguation setup, where we augment the training data with examples only consistent with the linear rule, and in line with our findings, we find that the `Gen`-prune fails to find a subnetwork with 100% generalization accuracy – no subnetwork consistent with hierarchical rule is formed (Figure 9 in Appendix).

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.

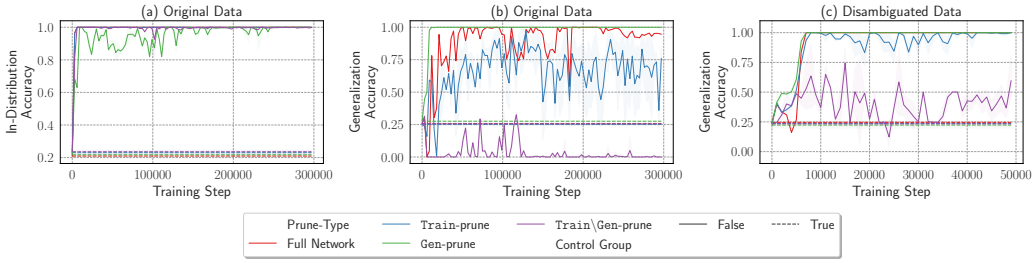


Figure 3: Tracking training dynamics with respect to the three pruning methods's subnetworks and the full network. (a) and (b): in-distribution and generalization accuracies of the LMs trained on the original ambiguous question formation data after pruning using the three methods, (c): generalization 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, while 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).

Hence, *ambiguity in the training data appears to drive the joint existence of these contrasting subnetworks.*

5. Why Do Transformer-Based LMs Generalize Hierarchically?

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, since 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. 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 (check Appendix §A.4.1 for a detailed background.) 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. Method

We now give an overview of how we apply the Bayesian approach discussed above to explain why transformer language models generalize hierarchically. We start by hand-constructing a probabilistic context-free grammar (PCFG) to model the hierarchical rule (denoted CFG) and a regular grammar (Reg) that generates data based on the linear 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 $\mathcal{D}_{\text{train}}$. Note that given $\mathcal{D}_{\text{train}}$, there can be grammars other than the CFG and Reg 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 $S \rightarrow a_1 a_2 \cdots a_n$. Here $a_i's$ are terminal symbols (words). One-state grammars are equivalent to finite state automata with a single state and hence permit any terminal symbol (i.e. any word) to follow any other. We also include these two grammars in our analysis.

We then compute the posterior probabilities for the four grammars given $\mathcal{D}_{\text{train}}$ and select the one with the higher posterior: $G^* = \arg\max_{G \in \mathcal{G}} p(G \mid \mathcal{D}_{\text{train}})$, where $\mathcal{G} = \{\texttt{CFG}, \texttt{Reg}, \texttt{Flat}, \texttt{One-State}\}$. Note that since we are primarily interested in comparing posteriors of different grammars, we can estimate the posterior by computing the likelihood and prior and taking product of the two, i.e., $p(G|D) \propto p(D \mid G)\, p(G)$. The prior $p(G)$ is chosen to encode the simplicity of a grammar, e.g. a grammar with many rules will have a lower prior probability. The likelihood $p(D|G)$ measures how well the grammar G fits D. Hence by computing the posterior we measure the tradeoff between the simplicity and goodness of fit for different grammars. We provide the details of how the priors and likelihood are computed in Appendix A.4.3

We then train a transformer language model on $\mathcal{D}_{\text{train}}$, and check if it generalizes according to G^*. This is done by constructing generalization test sets by considering the sentence types that are unique to a specific grammar. E.g., we can have the test set $\mathcal{D}^{\texttt{Hier}}_{\text{test}} = \mathcal{D}_{\texttt{CFG}} \setminus \mathcal{D}_{\texttt{Reg}}$, which contains sentence types that are unique to $\mathcal{D}_{\texttt{CFG}}$ and hence only consistent with the hierarchical rule and not the linear rule. Similarly, $\mathcal{D}^{\texttt{Lin}}_{\text{test}} = \mathcal{D}_{\texttt{Reg}} \setminus \mathcal{D}_{\texttt{CFG}}$, consists of sentence types consistent only with the linear rule. We can then evaluate if the transformer model trained on the ambiguous data $\mathcal{D}_{\text{train}}$, assigns a higher-likelihood to $\mathcal{D}^{\texttt{Hier}}_{\text{test}}$ or $\mathcal{D}^{\texttt{Lin}}_{\text{test}}$. Specifically, if $G^* = \texttt{CFG}$, does the transformer follow the hierarchical rule, and if $G^* = \texttt{Reg}$, does the transformer follow the linear rule? Note that if G^* is either a Flat or One-State grammar, the model should show no preference towards either linear or hierarchical rule, and assign similar likelihoods to both $\mathcal{D}^{\texttt{Hier}}_{\text{test}}$ and $\mathcal{D}^{\texttt{Lin}}_{\text{test}}$. 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. We provide details of each of these steps in Appendix §A.4.

Experimental Setup. For this study we consider the simple agreement task, as constructing hierarchical and linear grammars for its data is straightforward.[3] We also consider two variants of both context-free and regular grammars, depending on the diversity of the sentence types generated by them. 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*. In particular we construct small grammars `CFG-S` and `Reg-S`, each generating 18 sentence types, out of which 12 are common between the two and hence ambiguous. Similarly, we generate large grammars `CFG-L` and `Reg-L` which generate more diverse sentence types – 180 by each grammar, out of which 120 are common between the two. We denote the training and test sets generated from these two classes of grammars using the $-S$ and $-L$ suffixes, for small and large grammars – e.g. $\mathcal{D}_{\text{train}-\text{L}}$ denotes the data generated from `CFG-L` and `Reg-L`, and $\mathcal{D}_{\text{train}-\text{S}}$ denotes the data generated from `CFG-S` and `Reg-S`.

We evaluate trained transformers by comparing the negative log-likelihood (NLL) assigned by the models to the test sets corresponding to the two grammars. For a more intuitive metric, we also compute the main-verb accuracy from §3.1.

5.2. Results

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

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 $\mathcal{D}^{\texttt{Hier}}_{\text{test}}$ and $\mathcal{D}^{\texttt{Lin}}_{\text{test}}$ test sets. We use the same experimental setup as discussed in §3.2. In Figure 4a, 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 $\mathcal{D}^{\texttt{Hier}}_{\text{test}}$ test

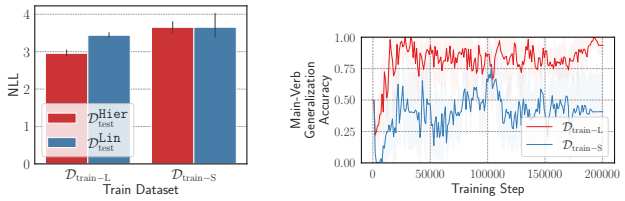
[3] Tasks like Question formation involve pairs of sentences, where the second sentence is a transformed version of the first. Such sentence pairs would likely require more complex frameworks like synchronous grammars (Aho & Ullman, 1969), which we leave to future work.

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 $\mathcal{D}_{\text{train}-\text{S}}$.

set being significantly lower than that on the $\mathcal{D}_{\text{test}}^{\text{Lin}}$ test set.

We see these findings reflected on the main-verb accuracy metric as well (Figure 4b), 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.



(a) Negative log-likelihood (NLL)

(b) Main-verb generalization accuracy

Figure 4: Performance of transformer models trained on the $\mathcal{D}_{\text{train}-\text{L}}$ and $\mathcal{D}_{\text{train}-\text{S}}$ datasets.

Takeaways. Our results indicate that when transformers-based LMs exhibit hierarchical generalization, despite being trained on ambiguous data, under the tested conditions, the hierarchical grammar not only fits the data well but is also simpler compared to the regular grammar with linear agreements. For the cases where this condition does not hold, we observe that the trained transformers exhibit no preference for hierarchical generalization.

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.

References

Aho, A. V. and Ullman, J. D. Syntax directed translations and the pushdown assembler. *J. Comput. Syst. Sci.*, 3:37–56, 1969. URL https://api.semanticscholar.org/CorpusID:205894705.

Baker, J. K. Trainable grammars for speech recognition. *Journal of the Acoustical Society of America*, 65, 1979. URL https://api.semanticscholar.org/CorpusID:121084921.

Bhaskar, A., Friedman, D., and Chen, D. The heuristic core: Understanding subnetwork generalization in pretrained language models, 2024.

Bhattamishra, S., Patel, A., Kanade, V., and Blunsom, P. Simplicity bias in transformers and their ability to learn sparse Boolean functions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5767–5791, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.317. URL https://aclanthology.org/2023.acl-long.317.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners, 2020.

Chen, A., Shwartz-Ziv, R., Cho, K., Leavitt, M. L., and Saphra, N. Sudden drops in the loss: Syntax acquisition, phase transitions, and simplicity bias in MLMs. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=MO5PiKHELW.

Chomsky, N. *Language and Mind*. Cambridge University Press, 1 edition, 1968.

Chomsky, N. *Language and Learning: The Debate Between Jean Piaget and Noam Chomsky*. Harvard University Press, 1980.

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

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T. (eds.), *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)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Dong, L., Yang, N., Wang, W., Wei, F., Liu, X., Wang, Y., Gao, J., Zhou, M., and Hon, H.-W. Unified language model pre-training for natural language understanding and generation. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/c20bb2d9a50d5ac1f713f8b34d9aac5a-Paper.pdf.

Frank, R. and Mathis, D. Transformational networks. *Models of Human Language Acquisition*, 22, 2007.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Lewis, J. D. and Elman, J. L. Learnability and the statistical structure of language: Poverty of stimulus arguments revisited. In *Proceedings of the 26th annual Boston University conference on language development*, volume 1, pp. 359–370. Citeseer, 2001.

Lin, Y., Tan, Y. C., and Frank, R. Open sesame: Getting inside BERT's linguistic knowledge. In Linzen, T., Chrupała, G., Belinkov, Y., and Hupkes, D. (eds.), *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 241–253, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4825. URL https://aclanthology.org/W19-4825.

Liu, Z., Kitouni, O., Nolte, N. S., Michaud, E., Tegmark, M., and Williams, M. Towards understanding grokking: An effective theory of representation learning. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 34651–34663. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/dfc310e81992d2e4cedc09ac47eff13e-Paper-Conference.pdf.

Louizos, C., Welling, M., and Kingma, D. P. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1Y8hhg0b.

Macwhinney, B. The childes project: tools for analyzing talk. *Child Language Teaching and Therapy*, 8, 01 2000. doi: 10.1177/026565909200800211.

McCoy, R. T., Frank, R., and Linzen, T. Revisiting the poverty of the stimulus: hierarchical generalization without a hierarchical bias in recurrent neural networks. *ArXiv*, abs/1802.09091, 2018. URL https://api.semanticscholar.org/CorpusID:3580012.

McCoy, R. T., Frank, R., and Linzen, T. 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, 2020. doi: 10.1162/tacl_a_00304. URL https://aclanthology.org/2020.tacl-1.9.

Merrill, W., Tsilivis, N., and Shukla, A. 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*, 2023. URL https://openreview.net/forum?id=8GZxtu46Kx.

Millidge, B. Grokking 'grokking', 2023. URL http://www.beren.io/2022-01-11-Grokking-Grokking/.

Mueller, A. and Linzen, T. How to plant trees in language models: Data and architectural effects on the emergence of syntactic inductive biases. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11237–11252, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.629. URL https://aclanthology.org/2023.acl-long.629.

Mueller, A., Frank, R., Linzen, T., Wang, L., and Schuster, S. Coloring the blank slate: Pre-training imparts a hierarchical inductive bias to sequence-to-sequence models. In Muresan, S., Nakov, P., and Villavicencio, A. (eds.), *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 1352–1368, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.106. URL https://aclanthology.org/2022.findings-acl.106.

Mueller, A., Webson, A., Petty, J., and Linzen, T. In-context Learning Generalizes, But Not Always Robustly: The Case of Syntax, November 2023. URL http://arxiv.org/abs/2311.07811. arXiv:2311.07811 [cs].

Müller, S., Hollmann, N., Arango, S. P., Grabocka, J., and Hutter, F. Transformers can do bayesian inference. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KSugKcbNf9.

Murty, S., Sharma, P., Andreas, J., and Manning, C. Grokking of hierarchical structure in vanilla transformers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 439–448, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-short.38. URL https://aclanthology.org/2023.acl-short.38.

Nanda, N., Chan, L., Lieberum, T., Smith, J., and Steinhardt, J. Progress measures for grokking via mechanistic interpretability, January 2023. URL https://arxiv.org/abs/2301.05217v2.

Perfors, A., Regier, T., and Tenenbaum, J. B. Poverty of the stimulus? a rational approach. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 28, 2006.

Perfors, A., Tenenbaum, J. B., and Regier, T. The learnability of abstract syntactic principles. *Cognition*, 118(3):306–338, 2011. ISSN 0010-0277. doi: https://doi.org/10.1016/j.cognition.2010.11.001. URL https://www.sciencedirect.com/science/article/pii/S0010027710002593.

Peters, M. E., Neumann, M., Zettlemoyer, L., and Yih, W.-t. Dissecting contextual word embeddings: Architecture and representation. In Riloff, E., Chiang, D., Hockenmaier, J., and Tsujii, J. (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1499–1509, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1179. URL https://aclanthology.org/D18-1179.

Petty, J. and Frank, R. Transformers Generalize Linearly, September 2021. URL http://arxiv.org/abs/2109.12036. arXiv:2109.12036 [cs].

Power, A., Burda, Y., Edwards, H., Babuschkin, I., and Misra, V. Grokking: Generalization beyond overfitting on small algorithmic datasets. *ArXiv*, abs/2201.02177, 2022. URL https://api.semanticscholar.org/CorpusID:245769834.

Reali, F. and Christiansen, M. H. Structure dependence in language acquisition: Uncovering the statistical richness of the stimulus. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 26, 2004.

Redington, M., Chater, N., and Finch, S. Distributional information: A powerful cue for acquiring syntactic categories. *Cognitive Science*, 22(4):425–469, 1998. doi: https://doi.org/10.1207/s15516709cog2204_2. URL https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog2204_2.

Shah, R. [AN #159]: Building agents that know how to experiment, by training on procedurally generated games. 2023. URL https://www.lesswrong.com/posts/zvWqPmQasssaAWkrj/an-159-building-agents-that-know-how-to-experiment-by.

Solomonoff, R. A formal theory of inductive inference. part i. *Information and Control*, 7(1):1–22, 1964. ISSN 0019-9958. doi: https://doi.org/10.1016/S0019-9958(64)90223-2. URL https://www.sciencedirect.com/science/article/pii/S0019995864902232.

Stolcke, A. and Omohundro, S. Inducing probabilistic grammars by bayesian model merging. In *International Colloquium on Grammatical Inference*, pp. 106–118. Springer, 1994.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

Tenney, I., Xia, P., Chen, B., Wang, A., Poliak, A., McCoy, R. T., Kim, N., Durme, B. V., Bowman, S., Das, D., and Pavlick, E. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SJzSgnRcKX.

Thilak, V., Littwin, E., Zhai, S., Saremi, O., Paiss, R., and Susskind, J. The slingshot mechanism: An empirical study of adaptive optimizers and the grokking phenomenon, 2022.

Valle-Perez, G., Camargo, C. Q., and Louis, A. A. Deep learning generalizes because the parameter-function map is biased towards simple functions. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rye4g3AqFm.

Varma, V., Shah, R., Kenton, Z., Kramár, J., and Kumar, R. Explaining grokking through circuit efficiency, September 2023. URL https://arxiv.org/abs/2309.02390v1.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, , and Polosukhin, I. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., and Titov, I. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In Korhonen, A., Traum, D., and Màrquez, L. (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5797–5808, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1580. URL https://aclanthology.org/P19-1580.

Zhang, E., Lepori, M. A., and Pavlick, E. Instilling inductive biases with subnetworks, 2024. URL https://arxiv.org/abs/2310.10899.

A. Appendix

A.1. Related Work

Language acquisition in humans. The problem of question formation has been well studied in the linguistics and cognitive science literature, where it has been observed that children from a very young age can produce grammatical output consistent with the hierarchical rule. In particular, the poverty of stimulus argument (Chomsky, 1968; 1980) asserts that children are unlikely to observe sufficient data to rule out order rule during language acquisition, and hence knowledge of the language being phrase structured must be innately specified (*linguistic nativism*). On the other hand, as a critique to the nativist argument, the *empiricist* argument (Redington et al., 1998; Lewis & Elman, 2001; Reali & Christiansen, 2004) states that distributional and statistical regularities in the data can be used to explain why children choose a hierarchical rule over an order-based rule. A critique of the empiricist argument is that it ignores the hierarchical phrase structured nature of natural language (Perfors et al., 2006) . Works like (Perfors et al., 2006; 2011) address this critique to empiricist argument using a Bayesian perspective on grammar induction and show that given child-directed corpus, an ideal learner can infer that a hierarchical grammar is simpler and fits the data as well as a linear grammar, without having this knowledge specified innately.

Hierarchical generalization in neural networks. Studying hierarchical generalization in neural networks has had its roots in empiricist arguments for language acquisition. Lewis & Elman (2001) showed that a simple recurrent network language model trained on the CHILDES dataset (Macwhinney, 2000) (designed for studying language of and directed to young children), would assign higher probabilities to questions constructed using the hierarchical rule than the order rule. A critique of the above work has been that it doesn't model the relation between the declarative and the question, hence failing to fully address the original poverty of stimulus argument. (Frank & Mathis, 2007) trained simple recurrent networks on the transformational task (form a question from the declarative) and found some evidence of the networks generalizing hierarchically, though the performance was found to depend heavily on the auxiliaries.

McCoy et al. (2018) used the setup from (Frank & Mathis, 2007) and performed a more thorough study on hierarchical generalization in RNNs (trained as seq2seq models), finding that while these models exhibit limited generalization performance, using attention and training on data with additional syntactic cues can help improve the performance. McCoy et al. (2020) studied the architectural inductive biases in RNNs influencing hierarchical generalization, and found

that only using a tree-structured model would consistently lead to hierarchical bias. Petty & Frank (2021); Mueller et al. (2022) corroborated these findings for transformers, finding networks to generalize linearly instead of hierarchically. In contrast to these findings, recently (Murty et al., 2023), showed that transformers, when trained for longer duration – way beyond saturating in-distribution performance – started exhibiting hierarchical generalization.

While all of these works train neural network models from scratch, recently there has been work on understanding hierarchical generalization in transformer models pretrained on large amounts of naturalistic language data. (Mueller & Linzen, 2023) found that pretraining encoder-decoder transformers on corpora like Wikipedia or CHILDES results in hierarchical bias in these models, though training on CHILDES was found to be orders of magnitude more sample-efficient towards imparting this bias. (Mueller et al., 2023) studied hierarchical generalization during in-context learning in language models, finding large variance in performance across different models. They found this variance to be explained by the composition of training data and particularly found the models trained on code to generalize better.

Grokking. One puzzle in deep learning generalization is the phenomenon of "grokking," where neural network are observed to start generalizing long after having overfit the training data (Power et al., 2022). Numerous efforts have been made to understand grokking and why it occurs. Millidge (2023) conjecture that for overparameterized networks the optimal set (i.e., the set of all parameter values resulting in 0 training loss) corresponds to a manifold in parameter space and stochastic gradient descent essentially acts as a random walk in this manifold, eventually hitting the parameters that generalize. The other explanations rely on *simplicity bias*, hypothesizing that the solutions that generalize are simpler but slower to learn (Shah, 2023; Nanda et al., 2023; Bhattamishra et al., 2023; Varma et al., 2023). Thilak et al. (2022) explain grokking from an optimization standpoint and show it to happen at the onset of a phenomenon they call as "slingshot mechanism," identified by spikes in the training loss which result in increased norm of the final-layer weights. (Liu et al., 2022) attempt to explain grokking through the theory of representation learning, identifying four phases during training and grokking occurring in a "Goldilocks zone" between two of these phases.

Training dynamics and subnetwork generalization. Merrill et al. (2023), identify dense and sparse subnetworks in the transformer models trained on a sparse-parity task and found the model starting to generalize as the norm of the sparse subnetwork undergoes rapid norm growth. Chen et al. (2024) identify emergence of syntactic attention structure in transformer masked language models, resulting from sudden drops in the loss, leading to the model subsequently acquiring different linguistic capabilities. Zhang et al. (2024) identifies task-specific subnetworks in a trained neural network model and use them to instill inductive biases to models being trained from scratch. In concurrent work, Bhaskar et al. (2024) find, using pruning, and for BERT-based models finetuned on NLP tasks like natural language inference and paraphrase identification, the existence of subnetworks that exhibit same in-domain performance but very different out-of-distribution generalization performance. This finding is in line with our observations about the presence of subnetworks consistent with different generalization behaviors. However, due to the nature of our problem, we are further able to show what specific behaviors these subnetworks associate with, how each of these evolves over the course of training, and suggest why these subnetworks co-exist during training.

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 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 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. Grammar details

A.4.1. 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 formalised 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 explains 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 subphrases 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: $A \to bC$, 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.

A.4.2. Constructing Grammars and Datasets

Constructing grammars for simple agreement. 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 (each verb in the sentence agrees with the most recent noun). The constructed grammars are assigned uniform probabilities for the production rules i.e., given a nonterminal, all the productions are equally likely. For an example of productions from both the grammars see Figures 10 and 11 in the Appendix. For constructing CFG, we use Chomsky Normal Form for the 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 of productions: Every rule is of the form $A \to bC$ or $A \to a$.

Following (Perfors et al., 2011), we 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., determiner, 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*. Different sentence types in this case differ by the plurality of the nouns (singular or plural), type of verbs (transitive or intransitive), and presence or absence of prepositional phrases accompanying the nouns. The resulting hand-constructed `CFG-S` in this case has 15 nontermi-

nals and 21 production rules and Reg-S has 14 nonterminals and 22 production rules. Both grammars have the same 8 terminals. 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: In this case we consider larger grammars, which can generate much more diverse sentence types – 180 sentence types. The major difference with the smaller grammars here is that they are allowed to generate relative clauses, which can be present at both the subject and 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, and the remaining sentence types are only generated by the specific grammars (following either hierarchical or linear rule).

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 $\mathcal{D}_{\text{train}-\text{S}} = \mathcal{D}_{\texttt{CFG-S}} \cap \mathcal{D}_{\texttt{Reg-S}}$ for the small grammars, and $\mathcal{D}_{\text{train}-\text{L}} = \mathcal{D}_{\texttt{CFG-L}} \cap \mathcal{D}_{\texttt{Reg-L}}$ for the larger ones. Note that these are the datasets of sentence-types, and transformers are trained on sentences. To generate sentences from these type corpora, 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 $\mathcal{D}_{\text{train}-\text{S}}$ and 50k sentences from $\mathcal{D}_{\text{train}-\text{L}}$. Note that the simple agreement experiments in §3.1, were performed using the latter dataset derived from $\mathcal{D}_{\text{train}-\text{L}}$.

The generalization test sets are generated by considering the sentence types that are unique to a specific grammar. E.g., 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 $\mathcal{D}^{\texttt{Lin}}_{\text{test}}$.

A.4.3. Posterior Computation

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., $p(G|D) \propto p(D \mid G)\, p(G)$. 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:

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

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 $p(P_k)$. 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 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 $A \to BC$ or $A \to a$, hence the probability of the right hand side can be given by, $p(R_{k,i}) = \frac{1}{2}\frac{1}{|V|^2}1(|R_{k,i}| = 2) + \frac{1}{2}\frac{1}{|\Sigma|}1(|R_{k,i}| = 1)$. Since the regular grammars are in the right linear form i.e. productions of the form $A \to bC$ or $A \to a$, we can compute $p(R_{k,i}) = \frac{1}{2}\frac{1}{|\Sigma|}\frac{1}{|V|}1(|R_{k,i}| = 2) + \frac{1}{2}\frac{1}{|\Sigma|}1(|R_{k,i}| = 1)$. 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 inclusion or exclusion of $p(\Sigma)$ doesn't make a difference.[4]

The likelihood $p(D \mid G)$, measures the probability that the dataset D is generated from the grammar G. For m sentence

[4] 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.

types in the dataset D, the likelihood is given by

$$p(D \mid G) = \prod_{i=1}^{m} p(S_i \mid G), \tag{2}$$

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. Hence, having computed both the prior $p(G)$ and $p(D \mid G)$, we can compute the posterior $p(G \mid D)$.

A.4.4. Local search using Bayesian model merging.

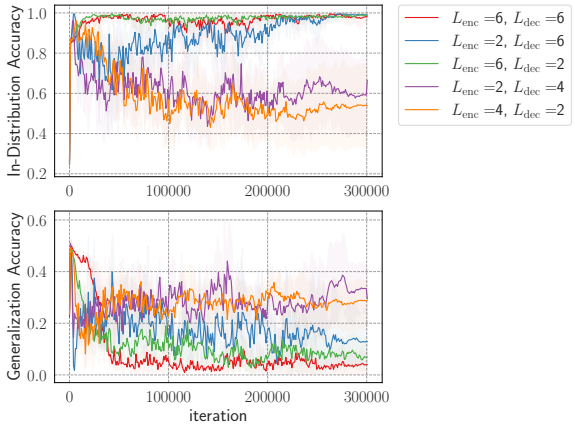
The hand-constructed grammars that we consider in our study might not be optimal in terms of the posterior given the training data. E.g., there might be some redundant production rules or non-terminals, which can be removed by merging two or more non-terminals. We use the Bayesian Model Merging (BMM) algorithm from Stolcke & Omohundro (1994); 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. E.g. for production rules $A \to BC$, $A \to BD$, we can merge C and D to F, resulting the new production rules: $A \to BF$, $F \to C$, and $F \to D$. 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 obtained, while should assign high likelihood to the ambiguous training data, it might no longer be consistent with the held out sentence types, e.g., $\mathcal{D}_{\text{test}}^{\texttt{Hier}}$ or $\mathcal{D}_{\text{test}}^{\texttt{Lin}}$, 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 `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 `CFG-L`, the grammar remains consistent with the hierarchical rule even after merging.

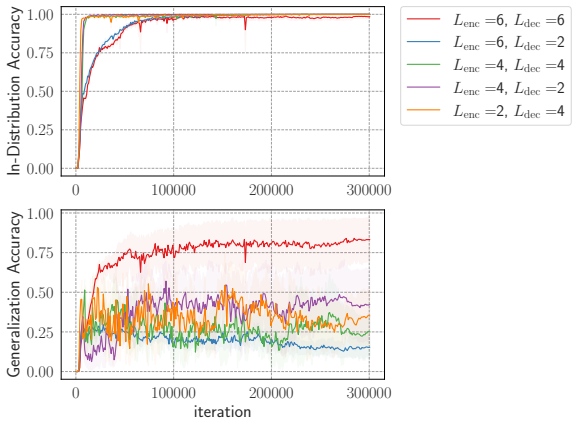
The log-probabilities after applying BMM algorithm are provided in Table 3. For the $\mathcal{D}_{\text{train-L}}$ dataset, we find that our results remain consistent with those for hand-constructed grammars in Table 2: $\texttt{CFG-L}^*$ obtains a lower posterior than $\texttt{Reg-L}^*$. On the other hand for the $\mathcal{D}_{\text{train-S}}$ dataset, $\texttt{CFG-S}^*$ ends up with a higher posterior than the `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.

A.5. 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 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.

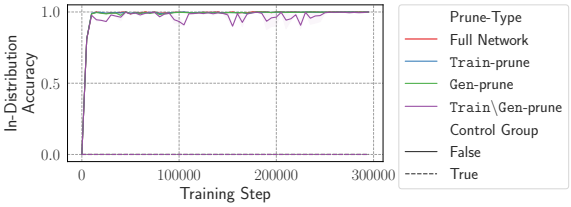


(a) Question formation

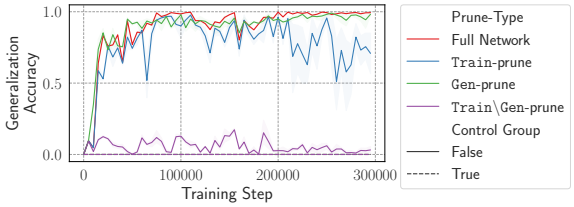


(b) Tense reinflection

Figure 5: Effect of the depth towards hierarchical generalization in seq2seq models on question formation and tense inflection tasks. For question formation task, we see that none of the combinations result in high generalization accuracy and the best combinations that get around 40% generalization accuracy have poor in-distribution performance. For tense reinflection, only on using 6 encoder and 6 decoder layers (what is reported in the paper), results in high generalization performance (including seeds that achieve 100% accuracy), but no other combination results in good performance.

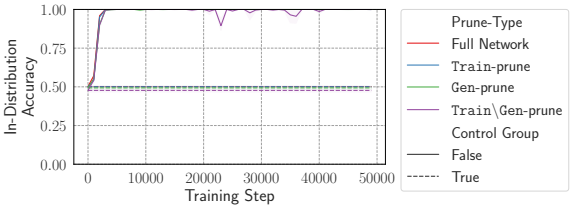


(a) In-distribution accuracy for different pruning methods across training (original question formation training data)

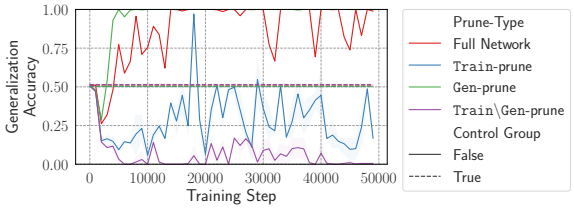


(b) Generalization accuracy for different pruning methods across training (original question formation training data)

Figure 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

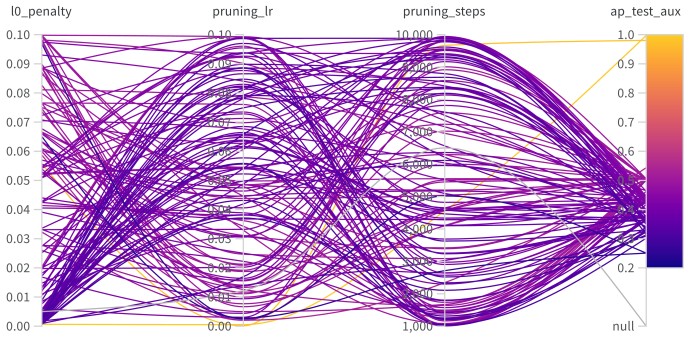


Figure 8: Performing `Train\Gen`-prune using different pruning hyperparameters, L_0 regularization penalty, pruning learning rate, and pruning steps. `ap_test_aux` denotes the generalization accuracy after pruning. We try 128 combinations of these parameters using Bayesian optimization and in no case we find a subnetwork obtaining 0% generalization accuracy. At best case we find subnetworks with 25% accuracy which correspond to a random baseline for this task (since there are 4 choices of the auxiliary verbs).

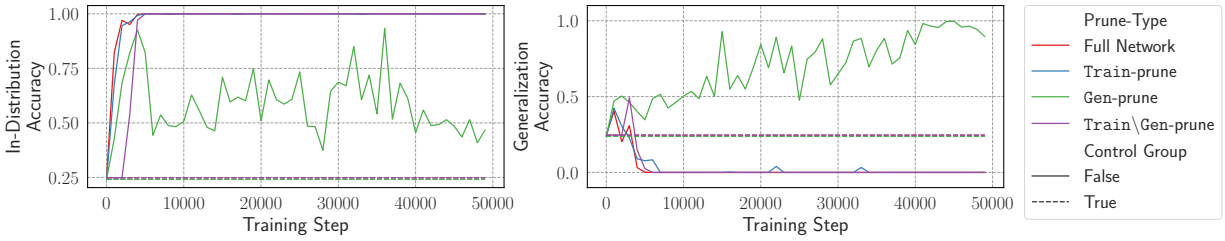


Figure 9: Training dynamics of transformer LM trained with question-formation data which is disambiguated with examples consistent only with the linear rule (by augmenting 10k such examples to the original 100k ambiguous examples). As can be seen, the full network in this case after a few thousand steps plateaus at 0% generalization performance, which is expected since only the linear rule is applicable to the entire dataset and hence the model is more likely to learn linear generalization in this case. 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.

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 super-script $*$ 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.

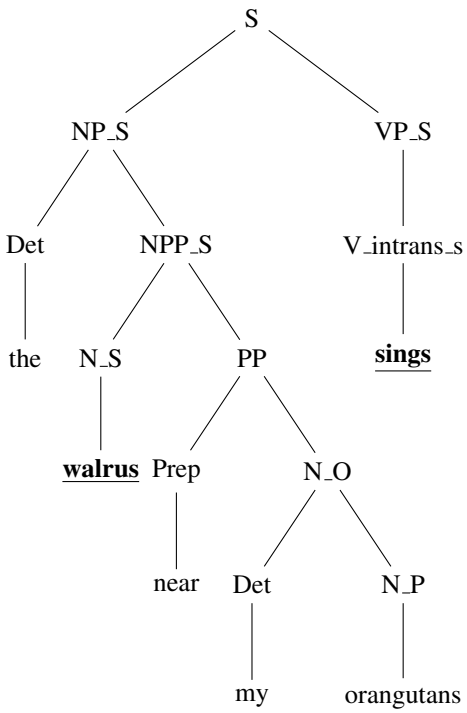


Figure 10: Example of production from the CFG with agreement following hierarchical rule

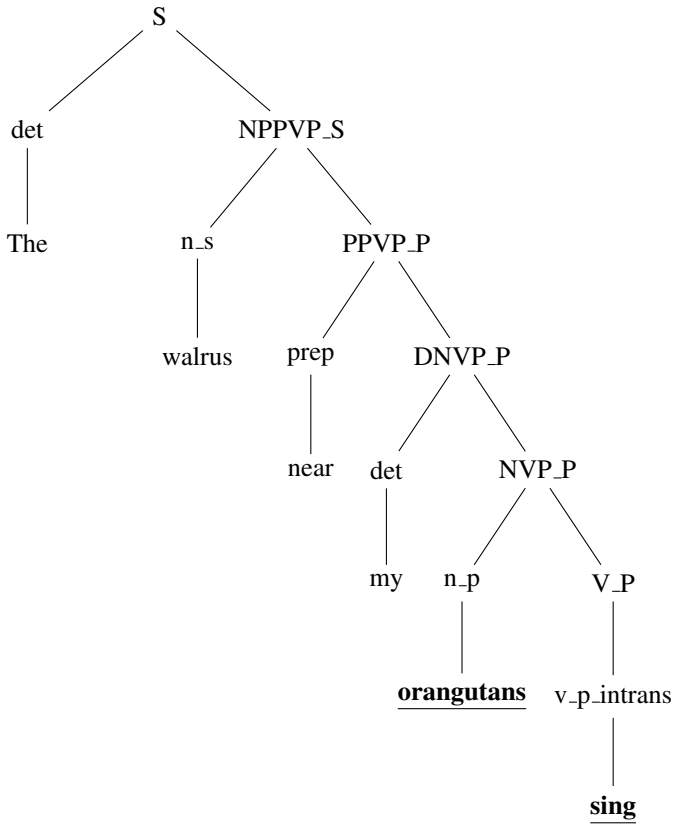
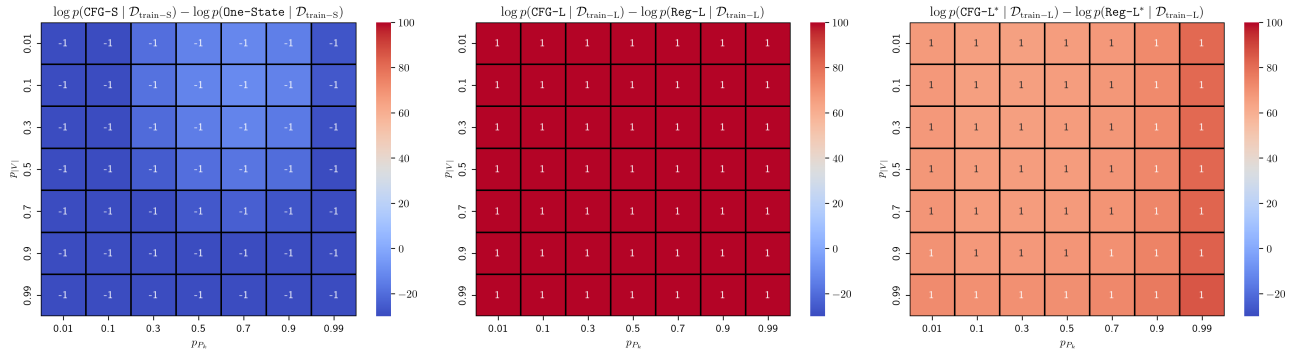


Figure 11: Example of production from the linear grammar with agreement following order rule

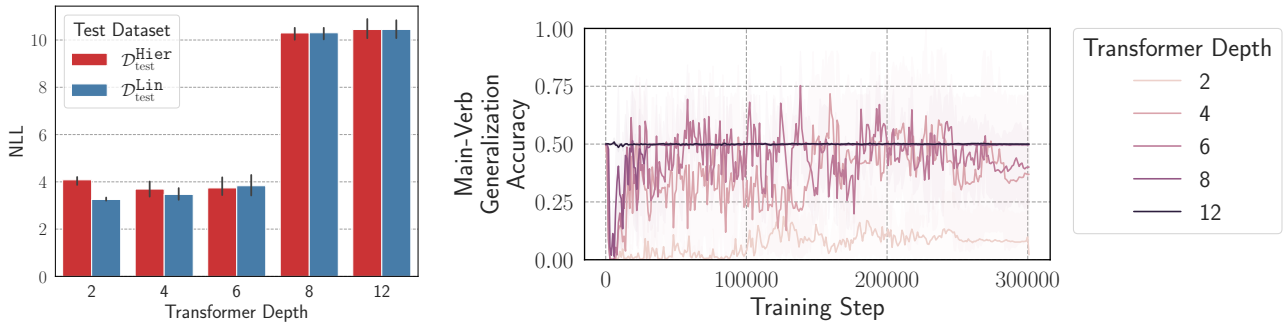


(a) Low diversity data case – $\mathcal{D}_{\texttt{train-S}}$ for hand-constructed grammars.

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

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

Figure 12: 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 $\mathcal{D}_{\texttt{train-S}}$ and `Reg-L` (or `Reg-L`* for BMM minimized case) for $\mathcal{D}_{\texttt{train-L}}$. 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 $\mathcal{D}_{\texttt{train-S}}$ 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 $\mathcal{D}_{\texttt{train-L}}$ 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 $\mathcal{D}_{\texttt{test}}^{\texttt{Hier}}$ and $\mathcal{D}_{\texttt{test}}^{\texttt{Lin}}$ test datasets.

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

Figure 13: Do transformer models trained on $\mathcal{D}_{\texttt{train-S}}$ 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 $\mathcal{D}_{\texttt{test}}^{\texttt{Lin}}$ than $\mathcal{D}_{\texttt{test}}^{\texttt{Hier}}$ 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.