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]

[1] University of Washington [2] Carnegie Mellon University [3] Microsoft Research
[4] Allen Institute for AI
kahuja@cs.washington.edu

ABSTRACT

Transformers trained on natural language data have been shown to learn its hierarchical structure and generalize to sentences with unseen syntactic structures 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 generalization behavior to emerge. We extensively experiment with transformer models trained on multiple synthetic datasets and with different training objectives and show that while other objectives e.g. sequence-to-sequence modeling, prefix language modeling, often failed to lead to hierarchical generalization, models trained with the language modeling objective consistently learned to generalize hierarchically. We then conduct pruning experiments to study how transformers trained with the language modeling objective encode hierarchical structure. When pruned, we find joint existence of subnetworks within the model with different generalization behaviors (subnetworks corresponding to hierarchical structure and linear order). Finally, we take a Bayesian perspective to further uncover transformers' preference for hierarchical generalization: We establish a correlation between whether transformers generalize hierarchically on a dataset and whether the simplest explanation of that dataset is provided by a hierarchical grammar compared to regular grammars exhibiting linear generalization.

https://github.com/kabirahuja2431/transformers-hg

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 the neural network models trained on language data learn this phrase structure of human language has been a subject of great interest. A flurry of past work have shown that syntax trees can be recovered from recurrent neural network (RNN) and transformer-based models trained on large-scale language corpora (Tenney et al., 2019, Peters et al., 2018, Lin et al., 2019, Wu et al., 2020). While these studies provide useful evidence of the aforementioned phenomenon, they do not shed light on the architectural choices, training paradigms or dataset characteristics that lead models to learn the phrase structure of language.

A useful tool to understand these model and dataset specific properties is through 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 says to move the *first* auxiliary. Hence, as a test for hierarchical generalization, we can ask, for neural networks trained from scratch on data that is consistent with both hierarchical and linear rules (i.e.,

ambiguous data), do they learn to generalize hierarchically or do they learn a linear rule (e.g., moving the first auxiliary to the beginning of the sentence)?

This question has been well-studied in past work for different neural network architectures. In particular, McCoy et al. (2020) showed that recurrent neural networks fail to generalize hierarchically when trained on ambiguous data, and only using tree-structured networks (Chen et al., 2017, 2018), which use explicit parses as inputs, leads to hierarchical generalization. Petty and Frank (2021) and Mueller et al. (2022) observed the same for transformers. However, Murty et al. (2023a) showed that, surprisingly, when trained for a long time after attaining perfect training accuracy, transformers *do* start to generalize hierarchically. They named this phenomenon *Structural Grokking*, because it resembles "grokking" as observed by Power et al. (2022) (where neural networks start to generalize long after they have overfit the training data).

While the results of Murty et al. (2023a) suggest that transformers are capable of exhibiting hierarchical generalization despite being trained on ambiguous data, it remains unclear why they exhibit such a preference. In our 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 in our study – language modeling, sequence-to-sequence modeling, prefix language modeling, sequence classification, and cloze completion, and compare the hierarchical generalization exhibited by transformers under these objectives. As a test for hierarchical generalization, in addition to the English question formation task described above, we also include German question formation (Mueller et al., 2022); tense-reinflection (McCoy et al., 2020), which converts a sentence in the past tense to the present; passivization (Mueller et al., 2022), i.e., converting a sentence in active voice to passive; and simple agreement, a synthetic task that we construct to check if the model can predict correct agreement between the verb and subject in a declarative sentence. To better understand how different generalization behaviors are implemented within the trained networks, we propose two new attention head pruning strategies to discover subnetworks corresponding to different generalizations (hierarchical and linear rules).

Finally, to understand why language modeling results in bias towards hierarchical structure, we utilize the Bayesian framework from Perfors et al. (2011) and consider generative probabilistic grammars (PCFGs) modeling the simple agreement task. Specifically, we construct hierarchical grammars (consistent with the hierarchical rule) as well as regular grammars that generate the data linearly and hence are consistent with the linear rule. We then compare the posterior probabilities of the two grammars, to understand which grammar has a better trade-off for the goodness of fit (measured using the likelihood) and simplicity (by calculating the prior on grammars), thereby explaining the preference of transformers for hierarchical or linear generalization.

Since our aim is to understand hierarchical generalization in transformers in isolation, following McCoy et al., 2020, Murty et al., 2023a, we train transformer models from scratch, without any pretraining, eliminating the possibility of these models having bias towards hierarchical generalization due to having been trained on language data before (Mueller et al., 2022). For the same reason, we also use synthetic datasets for training and evaluation that exclusively measure the inductive biases of these models towards hierarchical or linear generalization. Due to the controlled nature of our setup, the transformer models that we train are small (6 layers and 512 hidden size).

Our contributions:

- We discover that *the choice of the training objective affects hierarchical generalization in transformers*. Among five training objectives and five datasets, we find that only the language modeling objective consistently obtains strong hierarchical generalization across different tasks. This highlights that modeling the entire sequence of tokens (input and output) is critical for learning hierarchical structure.
- We find that *different types of generalizations consistent with the training data* (e.g., hierarchical and linear rules) *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. Further, we find 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, utilizing the Bayesian framework from Perfors et al. (2011), we show a correlation between transformer LMs generalizing hierarchically and hierarchical grammars having higher posterior, compared to regular grammars that follow the linear rule. This suggests that transformers generalize hierarchically because the *hierarchical grammars that fit the data are often "simpler" compared to regular grammars*. We also identify a case where this does not hold i.e. regular grammars have a higher posterior than hierarchical grammars, and show that transformers fail to generalize hierarchically in this case.

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

in language models. Our work takes steps towards understanding hierarchical generalization in language models, and we hope that our analysis method will be useful to study other forms of generalization in these models.

2 Background

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

1. (a) **Input**: My walrus does move .
   (b) **Output**: Does my walrus move ?
2. (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. While for sentences of type 1a, with only a single auxiliary this is trivial, for sentences of type 2a, 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. This linear rule is independent of the hierarchical structure of the sentence. However, consider sentences of type 3 below:

3. (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 ?

First, notice that sentence 3a has a different syntactic structure compared to the sentence 2a, as the relative clause *who doesn't sing* accompanies the subject, unlike in example 2 where *that do wait* modified the object (*the dogs*). In this case, using the linear rule to form question will result in an ungrammatical sentence, i.e., outputting sentence 3b instead of sentence 3c. In this work, we study the following question: Consider neural networks trained from scratch on data consistent with both hierarchical and linear rules (e.g., examples 1, 2). When presented with sentences such as 3a do they generalize hierarchically (predicting 3c) or do they learn a linear rule (predicting 3b)?

Tasks and datasets. In our study, we consider five tasks, including the question formation task above. Examples from all the tasks (excluding English question formation) 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., has the same syntactic forms and is 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. 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 1a; (ii) RC attached to the object, e.g., sentence 2a; and (iii) RC attached to the subject, e.g., sentence 3a. The training data includes (a 50-50 split) (i) declarative-question pairs where the task is to take a declarative sentence and generate a question as output and (ii) auxiliary 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. As such, the training dataset is consistent with both the hierarchical and linear rules and ambiguous in terms of which rule is applicable. Importantly, the auxiliary 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 task or the copy task.[1] Following McCoy et al. (2020) and Murty et al. (2023a), we evaluate the model on the *first-word accuracy*, i.e., given the declarative sentence as the input, we evaluate whether the model predicts the correct auxiliary for the first word in the question in its generation.

[1]Notice that the dataset consists of sentences with explicit auxiliary verbs – *my walrus does move*, instead of the less marked *my walrus moves*. This choice was made by McCoy et al. (2020), to study the problem as auxiliary fronting, i.e., moving the auxiliary verb to the beginning, and thereby studying the two rules (hierarchical and linear).

Table 1: Examples from the different tasks we study in our work. highlighted text indicates examples in the generalization set.

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

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.[2] Here again, the linear rule is to move the first modal or auxiliary to the front, and the hierarchical rule requires moving the token associated with the main verb. 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. The dataset from Müller et al. (2022) is constructed such that it contains active sentences of three types: (i) without any prepositional phrase (PP), (ii) with a PP on the object, and (iii) with a PP on the subject. Similar to question formation dataset, the active-passive pairs in the training dataset are constructed only using the sentences of type (i) and (ii). These two sentence types are again 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. Like question formation, the training data 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 tense reinflection, 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 that include past-form of sentences whose present form can only be generated using the hierarchical rule. 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 other tasks, simple agreement is a single-sentence task where only the *present*-tense sentences from the tense-inflection are used for training. In this task ,we evaluate the model's ability to generate the correct inflection of the verb at the end of the sentence. E.g., when given the prefix *my **zebra** by the yaks* as input, does the model assign higher likelihood to the singular verb form *swims* (correct) or the plural form *swim* (incorrect), as the continuation. 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*).

[2]Note that in German, negation is represented using another word *nicht* which is not fronted with the auxiliary (*can't* becomes *kann nicht*), hence Mueller et al. (2022) do not use the auxiliaries with negation for German, like we have for the English version.

For all tasks excluding simple agreement, there are 100k training examples (50k transformation pairs 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).

3 How the Training Objective Influences Hierarchical Generalization

We now discuss how the choice of training objective can influence hierarchical generalization in transformers. Prior work by McCoy et al. (2020), Petty and Frank (2021), and Mueller et al. (2022) used a 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. (2023a) used a language modeling objective to train a decoder-only transformer, which they found *did* generalize hierarchically when trained for a sufficiently large number of epochs – well beyond the point of achieving perfect training task accuracy. To the best of our 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 in our study:

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), and the loss is computed over *all* tokens in the sequence. 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:

$$-\log p(\boldsymbol{s}) = -\sum_{i=1}^{21} \log p(s_i \mid s_1, \ldots, s_{i-1}). \tag{1}$$

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 (e.g., from the example above, $\langle s_{12}, \ldots, s_{21} \rangle$) *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 1, 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 do not evaluate the seq2seq objective for simple agreement.

Prefix language modeling. In the prefix language modeling objective (Dong et al., 2019), we again generate the output text given the input (or "prefix"), but we use a single transformer decoder (similar to language modeling) instead of an encoder-decoder model. Differing from the original language modeling objective, here the loss is only computed over the output text and does not include the prefix. One modification that we make to how the prefix-LM objective is typically used, is that we use a causal mask for the prefix tokens as well instead of having bi-directional attention over the prefix tokens, since we found the latter to perform subpar in our initial experiments (unstable in-distribution performance).

Sequence classification. In the sequence classification objective, 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 four-way classification task.

Cloze completion. In the cloze completion setting, 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. 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 these positions and <EMPTY> if an auxiliary is not present at a particular position. 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.[3]

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

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 and not just in terms of one or two tokens, which makes defining the masking strategy in this case non-trivial.

Please refer to §A.1.1 for full details of each objective for all of the five tasks.

3.2 Experimental Setup

We train transformer models from scratch for all of our experiments. We use transformer models with 8 heads and embedding dimension 512 for all datasets and objectives. Following Murty et al. (2023a), for question formation and tense reinflection, we train transformer models with 6 layers for the former (18M parameters) and 4 layers (12M parameters) for the latter task, for all objectives excluding seq2seq. For the remaining tasks, we use 6-layer transformer encoder/decoder layers (18M parameters) depending on the training objective. For the seq2seq objective, we use a 6-layer encoder/6-layer decoder model (25M parameters) for all tasks.[4] For all the tasks, tokenization is performed at the word level. We use the Adam optimizer (Kingma and Ba, 2015) for training the model with a learning rate of 0.0001, following Murty et al. (2023a). We use batch size of 8, and train for 300k steps (24 epochs) for all tasks excluding simple agreement, which we train for 200k steps (32 epochs), since the dataset is half the size of others (recall that we have 100k training examples for all the tasks except simple agreement for which we have 50k). We run each experiment with 5 seeds and report the average performance.

Baselines. By design of the test datasets, a model following the linear rule will obtain 100% in-distribution accuracy and 0% generalization accuracy. Only a model consistent with the hierarchical rule will obtain 100% accuracy on both test sets for all the 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 obtain close to 100% accuracy on the in-distribution test sets (except sequence classification which performs slightly worse for simple agreement), there is a lot of 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[5] and passivization respectively, they perform much worse on the other tasks.

We believe that the choice of objective might be the reason behind the discrepancy in the results of Murty et al. (2023a) showing that transformer models with the language modeling objective generalize hierarchically, and the result of Petty and Frank (2021), Mueller et al. (2023) showing that transformer models with seq2seq objective do not generalize hierarchically. 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).[6]

We also provide the training curves depicting the in-distribution and generalization performance of different objectives over the course of training in Figure 6 in Appendix. Consistent with the findings of Murty et al. (2023a), for all the tasks, we find a delay in generalization for the LMs – models typically obtain 100% in-distribution accuracy much earlier than achieving high generalization performance. One might also notice that while LM objective consistently achieves high generalization performance, it is not perfect, like 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 LM models achieve 100% generalization accuracy, and there are also others with lower accuracy. Apart from the two exceptions discussed above, this is not the case for other objectives, where none of the five seeds get 100% generalization accuracy.

[4]We also considered out other choices of number of layers for the seq2seq objectives and found results consistent with our findings (see Appendix §A.1.3).

[5]We suspect that the seq2seq model for tense reinflection might not actually be generalizing hierarchically; see discussion in Appendix §A.1.4.

[6]This difference 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.

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 more rich compared to the English grammar, as it also consists of both infinitival and past participle (depending on if the sentence consists of auxiliaries or modals) forms of the main verbs, while only infinitival forms are included in the English version. As noted in McCoy et al. (2018), presence of rich hierarchical cues in the data can aid in hierarchical generalization.

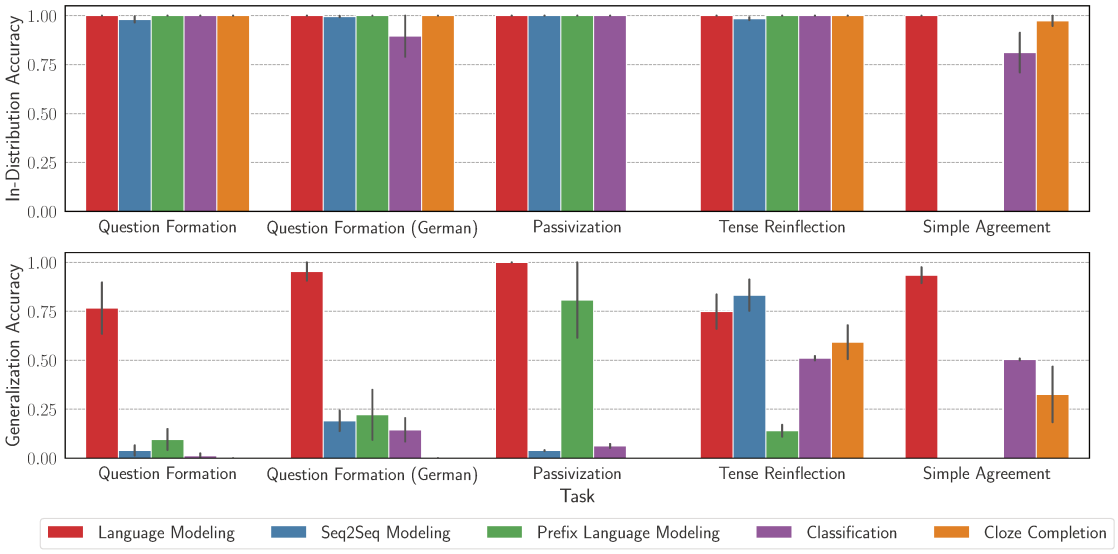


Figure 1: Effect of training objective on hierarchical generalization in transformers. The error bars correspond to the standard errors across 5 random seeds. Only the language modeling objective consistently obtains high generalization accuracy on all tasks.

Revisiting McCoy et al. (2020)'s results for RNNs Our results above suggest that the language modeling objective imposes bias towards hierarchical generalization in transformers. Based on this, we revisit hierarchical generalization in RNNs, as the experiments of McCoy et al. (2020) were conducted using the seq2seq objective (and found to *not* generalize when not using attention mechanism). We train 2-layer GRU (Cho et al., 2014) models using a learning rate of 0.001 using both LM and seq2seq objectives for the question formation task. As shown in Figure 2, like transformers, RNNs also generalize hierarchically on the question formation task when trained using the language modeling objective, but that's not the case for the seq2seq objective. This further strengthens the evidence that language modeling acts as a source of inductive bias for hierarchical generalization.

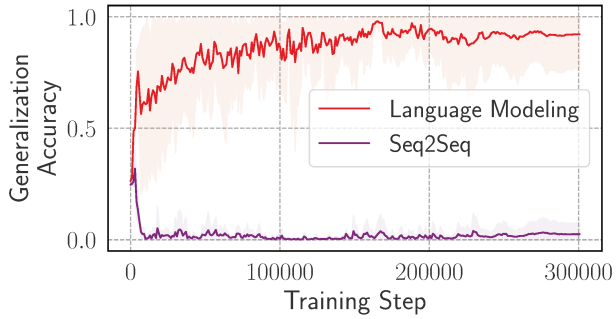


Figure 2: Training an RNN (GRU) using language modeling and seq2seq objectives on the question formation task. 300k training steps correspond to 24 epochs (or passes through the training data).

Takeaways. Overall, *our experiments implicate language modeling as a source of inductive bias for the neural models to generalise hierarchically*. We hypothesize that the reason LMs (approximately) learn the hierarchical rule rather than the linear rule is that, when viewing the full training objective as modeling the distribution of all the tokens in the

sentence pairs and not just moving of the auxiliary to the beginning, it is the combined simplicity of the hierarchical rule and the data is greater than the linear rule and the data. Perhaps modeling the hierarchical phrase structure is beneficial for modeling the distribution over full sequences. We will explore this hypothesis in more depth in §5.

4 Discovering Subnetworks with Different Generalization Behaviors

The results from Murty et al. (2023a), and from §3.3 show that the transformer LM obtains perfect in-domain accuracy much earlier during the training, while generalization comes later. This implies that the model might be implementing something akin to the linear rule in the beginning of training and eventually generalizes to the hierarchical rule. In this section, we explore whether these rules are implemented as subnetworks in the model and ask how these subnetworks evolve over the course of training.

4.1 Finding Subnetworks

Following Merrill et al. (2023) we use pruning to find the existence of subnetworks or circuits 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. This introduces `number of heads` $*$ `number of layers` learnable parameters, which is typically equal to 48 in our experiments. 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 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 the 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. In the case that all heads in a layer are zeroed-out, that particular layer is skipped: inputs to the layer pass through unchanged to the next layer due to the residual connections.

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% or 100 examples) to prune the attention heads. If successful, this pruning would yield a subnetwork consistent with hierarchical generalization—obtaining close to 100% generalization accuracy.

3. `Train\Gen`**-prune** is 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 but obtains 100% in-distribution accuracy.

Experimental setup. Unless specified otherwise, for pruning, we use a learning rate of 0.05, the L_0 regularization penalty coefficient as 0.015, and train for 10k steps, which we found to work well across different pruning settings. Here we report the experiments for the question formation task and discuss the others in Appendix §A.2, for which we also obtain consistent results. Note that 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 make sure that the discovered subnetworks are not just a by-product of the pruning procedure, we also consider control groups, which are obtained by pruning randomly initialized networks.

4.2 Results

In Figure 3, we show the effect of different pruning methods on an intermediate model checkpoint, which does not yet generalize hierarchically (the model before pruning has a generalization accuracy of 30%). 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.

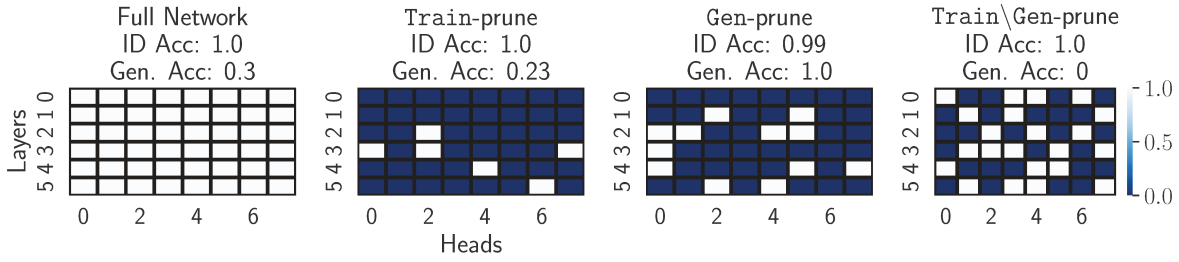


Figure 3: Pruning a transformer LM trained for 15000 steps using the three methods. Dark blocks mean that the head is pruned and light means it is kept.

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 4(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 4(a). Soon after, we see in Figure 4(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 4(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.

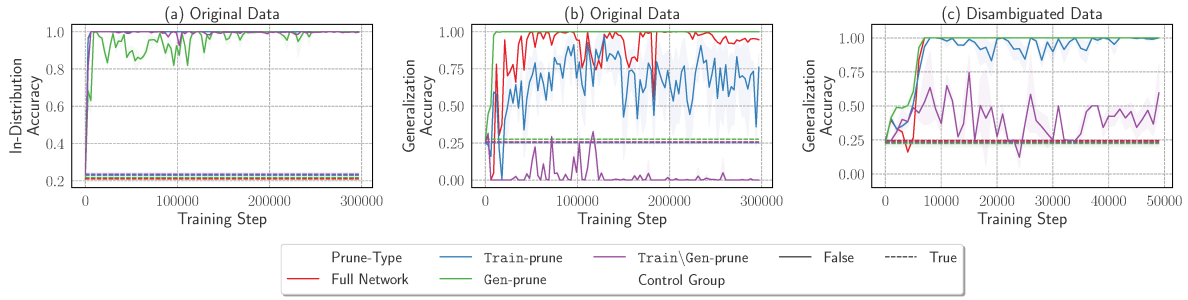


Figure 4: Tracking training dynamics with respect to the three pruning methods's subnetworks and the full network. (a) and (b): in-distribution and generalization accuracies of the LMs trained on the original ambiguous question formation data after pruning using the three methods, (c): generalization 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).

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.[7] We plot the training dynamics of the model trained with this disambiguated data in Figure 4(c). The full model without any pruning in this case, as expected, generalizes perfectly after a few thousand training steps (see 4(c)) without any noise, in contrast with generalization accuracy of the full-model trained on ambiguous data in Figure 4(b). More interestingly, Figure 4(c) shows that `Train\Gen`-prune fails

[7]We add 10k such examples to the existing dataset containing 100k examples.

to yield subnetworks that obtain 0% generalization accuracy, in contrast to the ambiguous data case in figure 4(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 12 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 (linear rule) 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 13 in Appendix). 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 "simplicity bias", where the inductive bias towards simpler functions (De Palma et al., 2019) has been shown to explain why neural networks tend to generalize instead of overfitting the training data (Valle-Perez et al., 2019, Bhattamishra et al., 2023). In our case, we are not interested in comparing the learned behavior of the language models (hierarchical rule) with the overfit solution, but instead with an alternate generalization (linear rule). Can we explain through "simplicity" the preference of the model towards hierarchical generalization? This might sound counterintuitive, because at least on surface it appears that the linear-rule should be simpler compared to the hierarchical rule. 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 (not, for instance, just the first auxiliary in the question formation task), modeling the dependencies between the tokens hierarchically as opposed to learning a linear rule for each dependency, might be simpler.[8] In this section, we present a study showing some evidence for the simplicity of the hierarchical generalization over the linear-rule based to explain the preference for the former by transformer LMs. We leverage the Bayesian framework of Perfors et al. (2011), utilizing generative grammars to model data-generation processes corresponding to the hierarchical and linear rules, and operationalize the notion of simplicity and goodness of fit using the posterior probabilities of the grammars given the observed data. We then show that there is a correlation between transformers' ability to generalize hierarchically and the training dataset being better explained using a hierarchical grammar than a regular one (which models the linear rule) according to the posterior criterion.

5.1 Background

Operationalizing the notion of simplicity. We make use of *Solomonoff's theory of inductive inference* (Solomonoff, 1964), which formalizes Occam's razor – 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 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. In other words, a more complex hypothesis will entail making more choices ("high program length") and hence have a lower prior probability. 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). This is closely related to "Bayesian Occam's razor" and the Minimum Description Length principle (Rissanen, 1978).

Probabilistic grammars. We mentioned in the previous paragraph that computing the posterior over the competing hypotheses can help us choose the one which better balances the trade-off between goodness of fit and simplicity. But for our problem, what form should these hypotheses or "programs" take to represent the linear and hierarchical rules? Note that since our training objective is language modeling, we need to consider the hypotheses that generate the entire sequence of tokens as represented in the training data. Following Perfors et al. (2011), we use probabilistic generative grammars to model the data-generation process.

For the purposes of this work we consider probabilistic context-free grammars (PCFGs) that can be represented using a 5-tuple i.e., $G = \{V, \Sigma, R, S, P\}$. Here, V denotes the set of nonterminal symbols that form phrases or constituents

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

in a sentence, Σ denotes the set of terminal symbols or words in the sentences, $R \in V \times \{V \cup \Sigma\}^*$ denotes the set of production rules mapping phrases to sub-phrases or words, $S \in V$ is the start symbol that represents the whole sentence, and P denotes the probabilities on the production rules. Specifically, for a given non-terminal when there are multiple productions possible, P assigns a probability to each possible production rule. To generate data from a PCFG, we start from the start symbol S and for each terminal that arises we apply a production rule sampled according to P and repeat the procedure till we are only left with terminals to generate sentences. 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.

A Bayesian view of language generation. We can view the data-generation process that generates dataset D using the probabalistic grammar G Given the dataset D, we can compute the posterior $p(G \mid D) \propto p(D \mid G) \cdot p(G)$, where $p(D \mid G)$ is the likelihood of the data given the probabilistic grammar, and $p(G)$ measures the simplicity of G. To get an intuitive understanding of the prior probability of a grammar and how it encodes simplicity, recall that grammars that we consider are 5-tuples $\{V, \Sigma, R, S, P\}$. Hence choosing a grammar G involves making choices like the number of nonterminal and terminal symbols, number of production rules from each nonterminal, nature of the production rule etc. By assigning probability distributions to each of these choices, we can compute the prior probability of a given grammar. We can choose the prior distribution that favours simpler grammars, e.g., following Perfors et al. (2011), we use geometric distributions for the number of nonterminals and productions, hence a simple grammar with fewer nonterminals and productions will receive a higher probability compared to a more complex grammar. We can hence compute the posteriors for the grammars representing the competing generalization hypotheses (hierarchical and linear rule) to compare how each of these balances the tradeoff between goodness of fit and simplicity.

5.2 Method

We now discuss how we apply the Bayesian Occam's razor approach discussed above to explain why transformer language models generalize hierarchically. As an overview of our approach, we start by constructing a PCFG to model the hierarchical rule (denoted `CFG`) and a regular grammar (`Reg`) that generates data based on the linear rule. We then generate data using both the grammars – $\mathcal{D}_{\texttt{CFG}}$ from `CFG` and $\mathcal{D}_{\texttt{Reg}}$ from `Reg`. The intersection of the two datasets, $\mathcal{D}_{\texttt{CFG}} \cap \mathcal{D}_{\texttt{Reg}}$, is comprised of ambiguous examples consistent with both the linear rule and hierarchical rule. We will use this as our training corpus $\mathcal{D}_{\text{train}}$. We then compute the posterior probabilities for both `CFG` and `Reg` given $\mathcal{D}_{\text{train}}$ and select the one with the higher posterior: $G^* = \arg\max_{G \in \{\texttt{CFG},\texttt{Reg}\}} p(G \mid \mathcal{D}_{\text{train}})$. We then train a transformer language model on $\mathcal{D}_{\text{train}}$, and check if it generalizes according to G^*. Specifically, if $G^* =$ `CFG`, does the transformer follow the hierarchical rule, and if $G^* =$ `Reg`, does the transformer follow the linear rule? The selection of G^* is intended to simulate "idealized" Bayesian learning, and to the extent that the transformer's learning behavior matches G^* across different scenarios, we find support for a simplicity bias in the transformer's training setup. In what follows, we provide details about each of these steps.

Task. For the purposes of this study we consider the simple agreement task, as constructing hierarchical and linear grammars for its data is straightforward.[9]

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 16 and 17 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.

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

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 nonterminals 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 $\mathcal{D}^{\texttt{Lin}}_{\text{test}}$.

Computing the posterior for each grammar. Now that we have the four grammars constructed, we can compute the posteriors for the grammars given the corresponding training datasets. Note that, since we are only interested in comparing the posteriors of CFG and regular grammars, we can estimate the posterior by computing the likelihood and prior and taking product of the two, i.e., $p(G|D) \propto p(D \mid G)\, p(G)$. 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:

$$p(G) = p(|V|) \prod_{k=1}^{|V|} p(P_k)\, p(\theta_k) \prod_{i=1}^{P_k} p(R_{k,i}). \tag{2}$$

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, $p(R_{k,i}) = \frac{1}{2}\frac{1}{|V|^2}\mathbb{1}(|R_{k,i}| = 2) + \frac{1}{2}\frac{1}{|\Sigma|}\mathbb{1}(|R_{k,i}| = 1)$. Since the regular grammars are in the right linear form i.e.

productions of the form $A \to bC$ or $A \to a$, we can compute $p(R_{k,i}) = \frac{1}{2}\frac{1}{|\Sigma|}\frac{1}{|V|}\mathbb{1}(|R_{k,i}| = 2) + \frac{1}{2}\frac{1}{|\Sigma|}\mathbb{1}(|R_{k,i}| = 1)$. One might notice that we are missing the probability of number of terminal symbols $p(\Sigma)$ in the prior equation. We ignore this because both the CFG and regular grammars have the same number of terminals in our experiments, and since we are interested in just comparing the probabilities, the inclusion or exclusion of $p(\Sigma)$ doesn't make a difference.[10]

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

$$p(D \mid G) = \prod_{i=1}^{m} p(S_i \mid G), \tag{3}$$

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 $p(G \mid D)$.

Other choices of grammars. Given our generated training datasets ($\mathcal{D}_{\text{train}-\text{S}}, \mathcal{D}_{\text{train}-\text{L}}$), there can be grammars other than the four we constructed that can generate these datasets. In their analysis, Perfors et al. (2011) also consider two subsets of the regular grammars: Flat and One-state. Flat grammars have production rules which are the list of memorized sentences, i.e., of the form $S \to a_1 a_2 \cdots a_n$. Here $a_i's$ are terminal symbols and there are no nonterminals other than S. Hence, flat grammars can be used to model memorization without generalization. One-state grammars are equivalent to finite state automata with a single state and hence permit any terminal symbol to follow any other. We also include these two grammars in our analysis.

Further, even among the class of context-free and regular grammars, there might exist grammars with better posteriors on the training datasets $\mathcal{D}_{\text{train}-\text{S}}$ and $\mathcal{D}_{\text{train}-\text{L}}$ than the ones that we hand-construct. To remedy this, we also experiment with applying local search on our constructed grammars, using Bayesian model merging (Stolcke and Omohundro, 1994) to minimize the grammars while improving the posterior on the respective training datasets. While in the main text we discuss the results for hand-constructed grammars, we provide the details on the minimization algorithm and the corresponding results for minimized grammars in §A.3.

Explaining generalization in transformers. Recall that our goal has been to quantify the notion of simplicity of the two competing hypotheses (hierarchical and linear rule), which are consistent with the training data used to train transformer-based LMs. The posterior probabilities of the two types of grammars are a way to measure which grammar better balances the trade-off between the goodness of fit and simplicity. Our aim is to check whether the trained transformer LM exhibits generalization consistent with choosing the simpler (i.e., larger posterior) grammar. We evaluate this by comparing the negative log-likelihood (NLL) assigned by the transformer LM to the test sets corresponding to the two generalizations. E.g. for the transformer model trained using data derived from $\mathcal{D}_{\text{train}-\text{L}}$, we evaluate its NLL on the generalization test sets derived from the two grammars $\mathcal{D}^{\texttt{Hier}}_{\text{test}-\text{L}}$ and $\mathcal{D}^{\texttt{Lin}}_{\text{test}-\text{L}}$, and check if it assigns a lower NLL to the test data coming from the simpler grammar. Note that we compute the average NLL overall all examples in a test set for the final value. For a more intuitive metric, we also compute the main-verb accuracy – fraction of test examples for which the verb predicted by the model agrees with the hierarchically associated noun in the sentence. A main-verb accuracy of 1 indicates that the model's generalization is consistent with the hierarchical rule and an accuracy of 0 when it is consistent with the linear rule.

5.3 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. However, the one-state grammar also fits the data the worst which is indicated by the lowest log-likelihood (for both datasets). 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 can come at a cost of increased complexity, especially when the training data is diverse; and so the flat grammar has the lowest log-prior on the full dataset.

[10] 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.

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

We obtain consistent findings with some subtle differences for the grammars minimized using the Bayesian model merging algorithm, which we detail in §A.3.

Sensitivity to prior. Note that choosing the prior is subjective and can influence these results. Hence, to be extra careful, we conduct a sensitivity analysis by varying the values of the geometric distribution parameter p. We experiment with $p \in \{0.01, 0.1, 0.2, 0.3, \cdots, 0.9, 0.99\}$ for the probability distribution on the nonterminals ($p(|V|)$) and number of productions ($p(P_k)$), and obtain findings consistent with those in Table 2 (see Figure 14 in Appendix). We also experiment with having different values of p parameter for $p(|V|)$ and $p(P_k)$, and try out 49 combinations ($\{0.01, 0.1, 0.3, 0.5, 0.7, 0.9, 0.99\} \times \{0.01, 0.1, 0.3, 0.5, 0.7, 0.9, 0.99\}$) . For each of these combinations, we find that for $\mathcal{D}_{\text{train}-\text{S}}$ case, consistent with Table 2 the `CFG-S` always obtain a lower posterior compared to the `One-State` grammar. Similarly for the `CFG-L` and `Reg-L`, the findings are also consistent across all 49 combinations i.e. `CFG-L` always obtain a higher posterior than `Reg-L`.

Table 2: Comparing the log-probabilities for each of the 4 grammars given the training datasets $\mathcal{D}_{\text{train}-\text{L}}$ and $\mathcal{D}_{\text{train}-\text{S}}$.

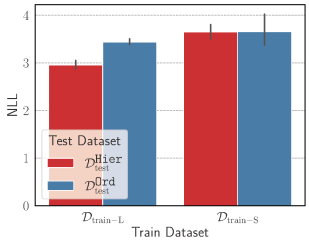
Grammar	$\mathcal{D}_{\text{train}-\text{L}}$ (120 types)			$\mathcal{D}_{\text{train}-\text{S}}$ (12 types)		
	log-Prior	log-Likelihood	log-Posterior	log-Prior	log-Likelihood	log-Posterior
`CFG`	-367	-639	**-1006**	-169	-34	-203
`Reg`	-619	-616	-1235	-190	**-30**	-220
`Flat`	-4567	**-574**	-5141	-281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

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. Note that for both datasets, 50k training examples are used. Recall that the two training datasets differ in their diversity (120 types in $\mathcal{D}_{\text{train}-\text{L}}$ vs. 12 in $\mathcal{D}_{\text{train}-\text{S}}$). We use the same experimental setup as discussed in §3.2. In Figure 5a, 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 set being significantly lower than that on the $\mathcal{D}^{\texttt{Lin}}_{\text{test}}$ test set.

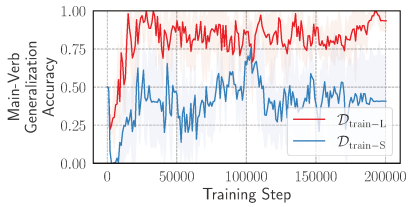
Besides NLL, we also compute the main-verb accuracy, where we check whether the model assigns a higher probability to the main verb agreeing with the correct inflection form than the incorrect one. As we can see in Figure 5b, the model trained on the $\mathcal{D}_{\text{train}-\text{L}}$ dataset obtains close to 100% accuracy when evaluated on the $\mathcal{D}^{\texttt{Hier}}_{\text{test}}$ test set. However, the model 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 (the latter would lead to 0% accuracy). We also verify that these results are not just a by-product of the choice of hyperparameters, and train transformer models with different layers (2, 4, 6, 8, 12), on the $\mathcal{D}_{\text{train}-\text{S}}$ dataset, and in none of the cases did we observe the models exhibiting preference for hierarchical generalization (see results in Appendix Figure 18).

We also consider a stricter metric: all-verb generalization accuracy which is obtained by checking whether *all* predicted verbs (and not just the main verb) in the sentence have the correct inflection. The reason for considering this metric is that, for the agreement task, 100% main-verb generalization accuracy can also be obtained without learning the hierarchical rule and simply agreeing with the first noun in the sentence. Note that the all-verb accuracy is computed by feeding prefixes preceding each verb in the sentence and obtaining the model's predictions. We provide the results with all-verb generalization accuracy in Figure 5c, where we show that a baseline that always selects the verb to agree with

the first noun in the sentence obtains 33% accuracy according to this metric, but as we can see transformers perform substantially better than this baseline, indicating that they do not trivially learn this heuristic to obtain high main-verb accuracy.



(a) Negative log-likelihood evaluated on the hierarchical ($\mathcal{D}^{\texttt{Hier}}_{\text{test}}$) and linear ($\mathcal{D}^{\texttt{Lin}}_{\text{test}}$) generalization test sets.



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

(c) All-verb generalization accuracy on $\mathcal{D}^{\texttt{Hier}}_{\text{test}}$ test for the model trained on the 120-types dataset.

Figure 5: Performance of transformer models trained on the $\mathcal{D}_{\text{train}-\text{L}}$ and $\mathcal{D}_{\text{train}-\text{S}}$ datasets.

Takeaways. Results from this study 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.

Limitations. First, it may be possible to find better grammars, either context-free or regular, in terms of posterior, on our two datasets. While local search using Bayesian model merging aims to find strong candidates, the possibility of better grammars cannot be eliminated as searching the space of all possible grammars is intractable. Secondly, our study treats transformers as black boxes and we do not answer whether or how these models implement hierarchical structure internally. The results from Murty et al. (2023a), which show that the representations of the transformer models become more and more tree-structured over the course of training, do provide some relevant evidence. Future work might focus on obtaining mechanistic explanations of this phenomenon.

6 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 and Elman, 2001, Reali and 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 and 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 and 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 and 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 and 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. (2023a), 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 and 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. 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.

7 Conclusion

We showed that language modeling training objective can act as a source of inductive bias towards hierarchical generalization, by comparing different training objectives on five tasks and finding the LM objective to be the only one that consistently generalizes hierarchically across all of them. We also find that when the training data is consistent with two rules, we can find subnetworks in the transformer LM trained on this data corresponding to each of these rules, which continue to coexist over the course of training. Finally, we provided a Bayesian interpretation to explain why transformer LMs generalize hierarchically: hierarchical grammars that fit sufficient diverse language data as well as regular grammars are often, in a sense, simpler.

There are multiple directions that can be explored in the future. While our results indicate language modeling as a source of hierarchical bias, it still remains unclear why hierarchical generalization is delayed. Further, Murty et al. (2023a) showed that deeper transformer LMs often fail to generalize hierarchically, which remains unexplored in our setting. 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. In our work, we used the Bayesian interpretation to understand hierarchical generalization in transformers. However, the Bayesian interpretation has been useful to study other forms of generalization in humans as well, including (among others) word learning (Xu and Tenenbaum, 2007), concept learning (Goodman et al., 2008, Lake et al., 2015), pragmatics (Frank and Goodman, 2012), and theory of mind (Baker et al., 2011) , and these capabilities have also been observed to some

extent in transformer based LMs as well Patel et al. (2023), Hu et al. (2023), Shapira et al. (2023) . How well these interpretations can be applied to explain such capabilities in transformers is another potentially interesting direction.

References

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

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes, 2017. URL https://openreview.net/forum?id=ryF7rTqgl.

Chris Baker, Rebecca Saxe, and Joshua Tenenbaum. Bayesian theory of mind: Modeling joint belief-desire attribution. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011.

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

Adithya Bhaskar, Dan Friedman, and Danqi Chen. The heuristic core: Understanding subnetwork generalization in pretrained language models, 2024.

Satwik Bhattamishra, Arkil Patel, Varun Kanade, and Phil Blunsom. 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)*, pages 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.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

Angelica Chen, Ravid Shwartz-Ziv, Kyunghyun Cho, Matthew L Leavitt, and Naomi Saphra. 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.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. Improved neural machine translation with a syntax-aware encoder and decoder. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1936–1945, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi:10.18653/v1/P17-1177. URL https://aclanthology.org/P17-1177.

Xinyun Chen, Chang Liu, and Dawn Song. Tree-to-tree neural networks for program translation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/d759175de8ea5b1d9a2660e45554894f-Paper.pdf.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. doi:10.3115/v1/D14-1179. URL https://aclanthology.org/D14-1179.

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

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

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi:10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *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.

Michael C. Frank and Noah D. Goodman. Predicting pragmatic reasoning in language games. *Science*, 336(6084): 998–998, 2012. doi:10.1126/science.1218633. URL https://www.science.org/doi/abs/10.1126/science.1218633.

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

Noah D Goodman, Joshua B Tenenbaum, Jacob Feldman, and Thomas L Griffiths. A rational analysis of rule-based concept learning. *Cognitive science*, 32(1):108–154, 2008.

Jennifer Hu, Sammy Floyd, Olessia Jouravlev, Evelina Fedorenko, and Edward Gibson. A fine-grained comparison of pragmatic language understanding in humans and language models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4194–4213, Toronto, Canada, July 2023. Association for Computational Linguistics. doi:10.18653/v1/2023.acl-long.230. URL https://aclanthology.org/2023.acl-long.230.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *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.

Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. doi:10.1126/science.aab3050. URL https://www.science.org/doi/abs/10.1126/science.aab3050.

John D Lewis and Jeffrey L Elman. 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, pages 359–370. Citeseer, 2001.

Yongjie Lin, Yi Chern Tan, and Robert Frank. Open sesame: Getting inside BERT's linguistic knowledge. In Tal Linzen, Grzegorz Chrupała, Yonatan Belinkov, and Dieuwke Hupkes, editors, *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 241–253, Florence, Italy, August 2019. Association for Computational Linguistics. doi:10.18653/v1/W19-4825. URL https://aclanthology.org/W19-4825.

Ziming Liu, Ouail Kitouni, Niklas S Nolte, Eric Michaud, Max Tegmark, and Mike Williams. Towards understanding grokking: An effective theory of representation learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 34651–34663. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/dfc310e81992d2e4cedc09ac47eff13e-Paper-Conference.pdf.

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

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

R. Thomas McCoy, Roberta Frank, and Tal Linzen. 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.

R. Thomas McCoy, Robert Frank, and Tal Linzen. 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.

William Merrill, Nikolaos Tsilivis, and Aman Shukla. 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.

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

Aaron Mueller and Tal Linzen. How to plant trees in language models: Data and architectural effects on the emergence of syntactic inductive biases. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 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.

Aaron Mueller, Robert Frank, Tal Linzen, Luheng Wang, and Sebastian Schuster. Coloring the blank slate: Pre-training imparts a hierarchical inductive bias to sequence-to-sequence models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 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.

Aaron Mueller, Albert Webson, Jackson Petty, and Tal Linzen. 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].

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KSugKcbNf9.

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

Shikhar Murty, Pratyusha Sharma, Jacob Andreas, and Christopher D Manning. Characterizing intrinsic compositionality in transformers with tree projections. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=sAOOeI878Ns.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability, January 2023. URL https://arxiv.org/abs/2301.05217v2.

Arkil Patel, Satwik Bhattamishra, Siva Reddy, and Dzmitry Bahdanau. Magnifico: Evaluating the in-context learning ability of large language models to generalize to novel interpretations. *arXiv preprint arXiv:2310.11634*, 2023.

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

Amy Perfors, Joshua B. Tenenbaum, and Terry Regier. 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.

Matthew E. Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. Dissecting contextual word embeddings: Architecture and representation. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi:10.18653/v1/D18-1179. URL https://aclanthology.org/D18-1179.

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

Alethea Power, Yuri Burda, Harrison Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *ArXiv*, abs/2201.02177, 2022. URL https://api.semanticscholar.org/CorpusID:245769834.

Florencia Reali and Morten H Christiansen. 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.

Martin Redington, Nick Chater, and Steven Finch. 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.

J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978. ISSN 0005-1098. doi:https://doi.org/10.1016/0005-1098(78)90005-5. URL https://www.sciencedirect.com/science/article/pii/0005109878900055.

Rohin Shah. [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.

Natalie Shapira, Mosh Levy, Seyed Hossein Alavi, Xuhui Zhou, Yejin Choi, Yoav Goldberg, Maarten Sap, and Vered Shwartz. Clever hans or neural theory of mind? stress testing social reasoning in large language models. *arXiv preprint arXiv:2305.14763*, 2023.

R.J. Solomonoff. 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.

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

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. 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.

Vimal Thilak, Etai Littwin, Shuangfei Zhai, Omid Saremi, Roni Paiss, and Joshua Susskind. The slingshot mechanism: An empirical study of adaptive optimizers and the grokking phenomenon, 2022.

Guillermo Valle-Perez, Chico Q. Camargo, and Ard A. Louis. 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.

Vikrant Varma, Rohin Shah, Zachary Kenton, János Kramár, and Ramana Kumar. Explaining grokking through circuit efficiency, September 2023. URL https://arxiv.org/abs/2309.02390v1.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 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.

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

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

Fei Xu and Joshua B Tenenbaum. Word learning as bayesian inference. *Psychological review*, 114(2):245, 2007.

A Appendix

A.1 Training Objectives and Hierarchical Generalization

A.1.1 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.

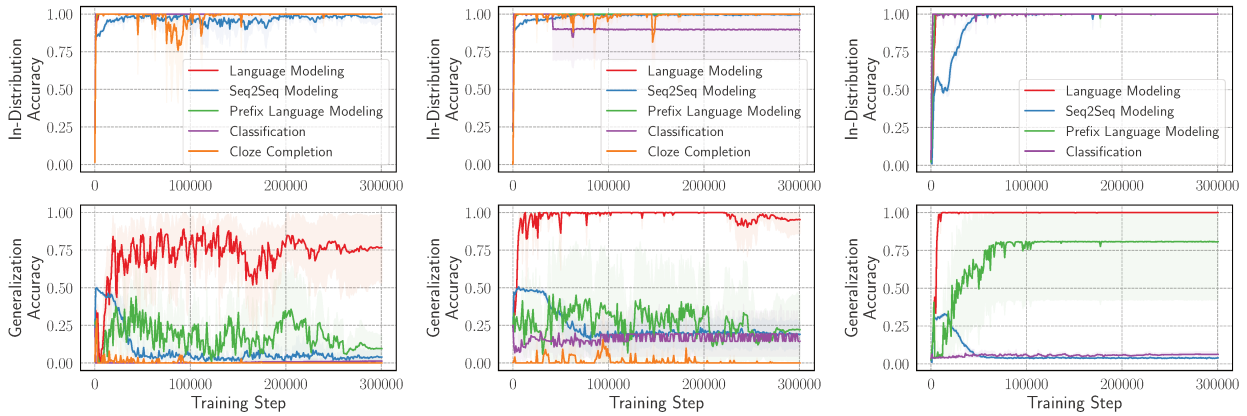
Sequence classification. For question formation, the input is the declarative sentence, and the output is the four possible auxiliary tokens, $\{\text{do}, \text{does}, \text{don}'\text{t}, \text{doesn}'\text{t}\}$ for English and $\{\text{können}, \text{kann}, \text{haben}, \text{hat}\}$ for German. 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 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 $<\texttt{EMPTY}>$ 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.

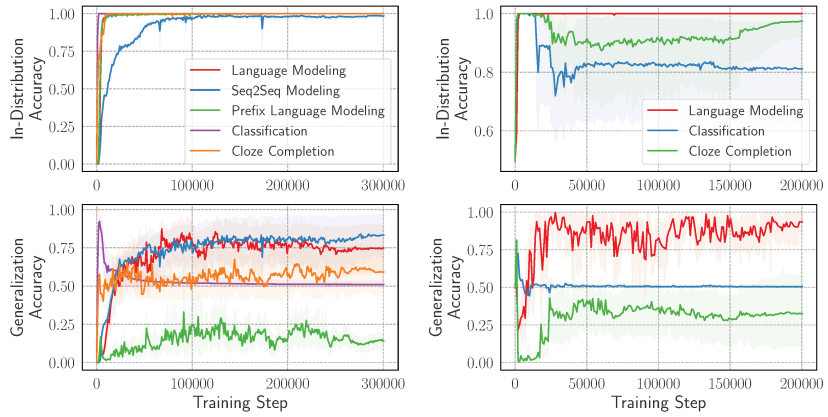
A.1.2 Training curves

The performance of the five objectives on the five datasets across model training is provided in Figure 6.

The effect of identity pairs on hierarchical generalization. As noted in §3.1, the training datasets for tasks like the question formation and tense reinflection tasks from McCoy et al. (2020), also used by Murty et al. (2023a), include identity pairs of declaratives (for question formation) and past tense sentences (for tense reinflection) for the auxiliary copy task. This data also includes input sentences with the same syntax as the inputs in the generalization set, though the outputs are still unseen. Since the model is exposed to the form of sentences in the generalization set, we would like to explore if this auxiliary data (identity pairs) contributes to hierarchical generalization: Do models learn hierarchical rule without having seen the input-types from the generalization set? As can be seen in Figure 7, for the question formation task removal of identity pairs results in a significant drop in generalization accuracy for transformers trained with language modeling objective. For tense reinflection, however, the generalization performance remains more or less similar. We suspect this might be due to the gap between the sentence types in the in-distribution and generalization set is arguably greater for question formation than tense reinflection. In question formation the generalization set has sentences with relative clauses attached to the subject and without the identity pairs, such sentences will be completely unseen during the training. On the other hand for tense reinflection, the training dataset has all nouns of the same plurality (i.e. either all singular or all plural), while in generalization the nouns have different plurality to break the



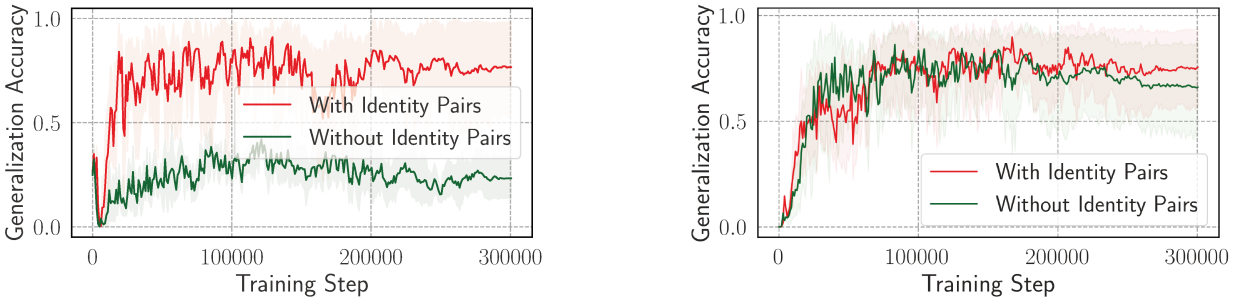
(a) Question formation (b) Question formation (German) (c) Passivization



(d) Tense reinflection (e) Simple agreement

Figure 6: Training curves for different objectives on the five tasks.

ambiguity. The overall structure of the sentence does remain same for tense-reinflection, while that's not the case for question formation.



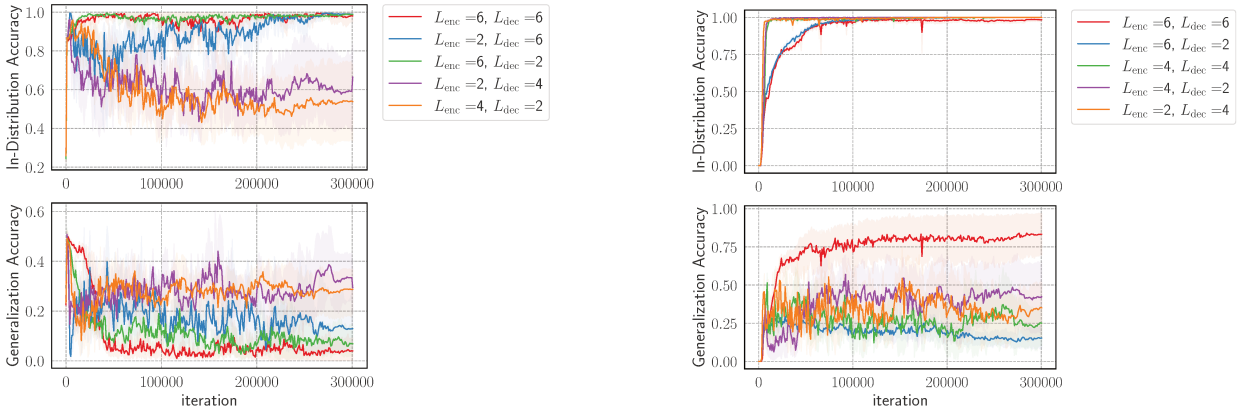
(a) Question formation (b) Tense reinflection

Figure 7: Effect of the presence of identity pairs (the auxiliary copy task) in training data on hierarchical generalization.

A.1.3 Effect of depth in hierarchical generalization in seq2models.

Murty et al. (2023a) performed a systematic study on the effect of model depth towards hierarchical generalization in language models and found that only the models with intermediate depths like 6 or 4 performed well, while shallower as well as deeper networks failed to generalize. Since, seq2seq models use both an encoder and decoder, the same depths found appropriate for language modeling might not be so for seq2seq models. While in the paper we presented

results with 6 encoder and 6 decoder layers, we also try different combinations of the two on the question formation and tense reinflection tasks and report the results in Figure 8. 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) Question formation

(b) Tense reinflection

Figure 8: Effect of the depth towards hierarchical generalization in seq2seq models on question formation and tense inflection tasks.

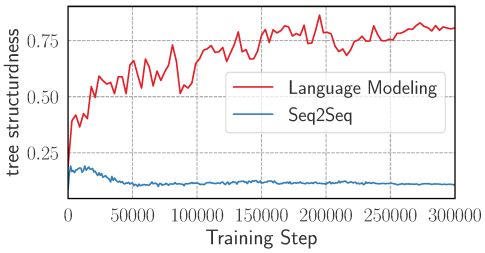
A.1.4 Do seq2seq trained transformers really generalize hierarchically for tense reinflection?

For tense reinflection, in Figure 1 we observe that seq2seq objective performs close to language modeling in terms of generalization accuracy. We believe this might be due to the fact that it is possible to achieve high generalization accuracy on tense reinflection without learning the hierarchical rule. Notice that in both examples for tense reinflection in Table 1, there exists an alternate non-hierarchical rule: The main-verb of the sentence should agree with the first noun in the sentence, since the way McCoy et al. (2020)'s dataset is constructed ensures that the first noun in the sentence is always the subject hierarchically connected to the main-verb. To validate if these models truly learn the hierarchical rule and not a shortcut, we evaluate the tree-structuredness of the learned representations of the trained models using the tree projection method of Murty et al. (2023b). Their method characterizes the extent to which computations in the transformer can be explained by an equivalent tree-structured neural network.

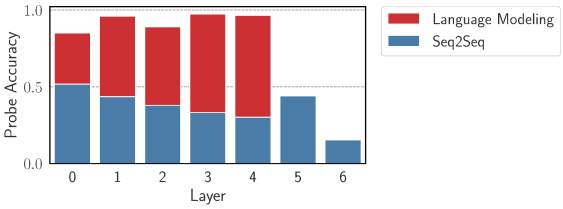
In Figure 9a, we compare the tree-structuredness scores for the transformer models trained using language modeling and seq2seq objectives. Consistent with Murty et al. (2023a), we observe that for the language modeling objective the transformer representations continue to become more tree-structured as training progresses. However, for seq2seq model the tree-structuredness score remains low throughout the training. This indicates that while the seq2seq model obtains high generalization accuracy, it might not actually be generalizing hierarchically. To verify this further, we also performed linear-probing (Alain and Bengio, 2017) on the final layer representations of the two models, and found for language modeling-trained model, the syntactic categories of the words (e.g. singular noun is the category for *walrus*) in the sentence can be recovered with a very high accuracy. By contrast, linear probes trained on seq2seq model are unable to extract the syntactic categories (50% probing accuracy compared to 99% for language model) (see figure 9b).

A.2 Subnetworks with Different Generalization Behaviors

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



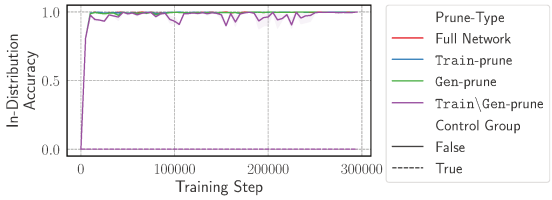
(a) Tree-structuredness scores for models trained on tense reinflection task.



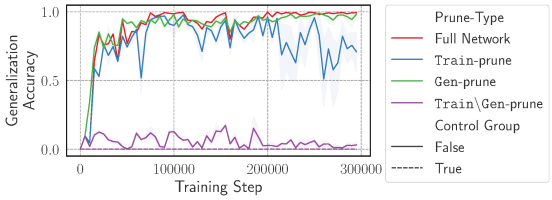
(b) Linear probing on transformers trained for tense reinflection tasks using language modeling and Seq2Seq objectives. The missing red bars for layer 5 and 6 are because the transformer trained with langauge modeling objective only has 4 layers, while the Seq2Seq model has 6.

Figure 9: Investigating if Seq2Seq models really generalize hierarchically for tense reinflection.

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 10 and 11. 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.

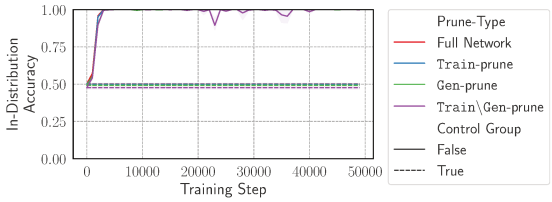


(a) In-distribution accuracy for different pruning methods across training (original question formation training data)

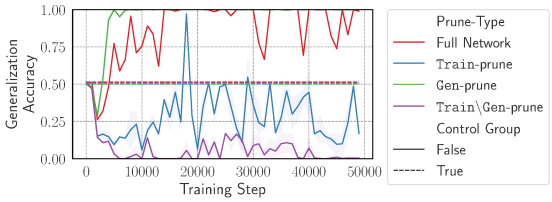


(b) Generalization accuracy for different pruning methods across training (original question formation training data)

Figure 10: 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 11: Tracking training dynamics w.r.t. to the three pruning methods for simple agreement task

A.3 Grammar details

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 and 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

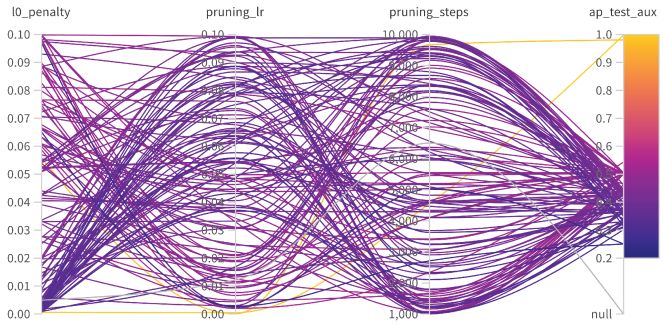


Figure 12: Performing Train\Gen-prune using different pruning hyperparameters, L_0 regularization penalty, pruning learning rate, and pruning steps. ap_test_aux denotes the generalization accuracy after pruning. We try 128 combinations of these parameters using Bayesian optimization and in no case we 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).

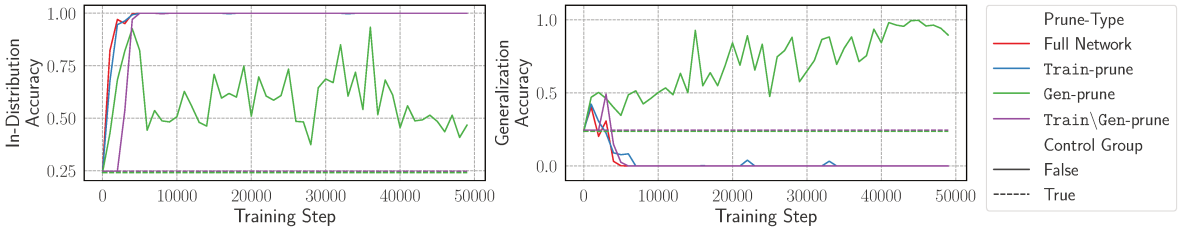


Figure 13: Training dynamics of transformer LM trained with question-formation data which is disambiguated with examples consistent only with the linear rule (by augmenting 10k such examples to the original 100k ambiguous examples). As can be seen, the full network in this case after a few thousand steps 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.

$F \to D$. 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 CFG* (CFG-S* and CFG-L*) and regular grammars as Reg* (Reg-S* and Reg-L*).

An important detail to note here is that while performing the merging algorithm, we use the ambiguous corpus $\mathcal{D}_{\text{train}-\text{L}}$ or $\mathcal{D}_{\text{train}-\text{S}}$ 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.

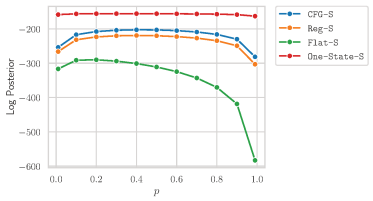
The log-probabilities after applying BMM algorithm are provided in Table 3. For the $\mathcal{D}_{\text{train}-\text{L}}$ dataset, we find that our results remain consistent with those for hand-constructed grammars in Table 2: CFG-L* obtains a lower posterior than Reg-L*. On the other hand for the $\mathcal{D}_{\text{train}-\text{S}}$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

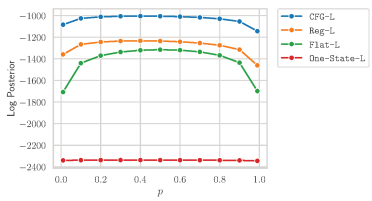
Table 3: Comparing the log-probabilities for each of the 4 grammars after performing BMM on the `CFG` and `Reg` grammars given the training datasets $\mathcal{D}_{\text{train}-\text{L}}$ and $\mathcal{D}_{\text{train}-\text{S}}$. The super-script $*$ symbol on log-posterior for `CFG`* on $\mathcal{D}_{\text{train}-\text{S}}$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

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`*	-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

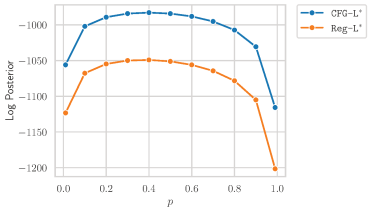
Sensitivity Analysis. We provide the plots for the sensitivity analysis o the choice of p parameter of the geometric distributions used to define the prior in Figures 14 and 15. We provide the sensitivity analysis for both hand-constructed and BMM minimized grammars. For the latter case, we only provide the analysis on $\mathcal{D}_{\text{train}-\text{L}}$ dataset case, since after minimization on the smaller grammars (`CFG-S` and `Reg-S`), we are left with no grammar obeying the hierarchical rule (see discussion in the previous paragraph for details).



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



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

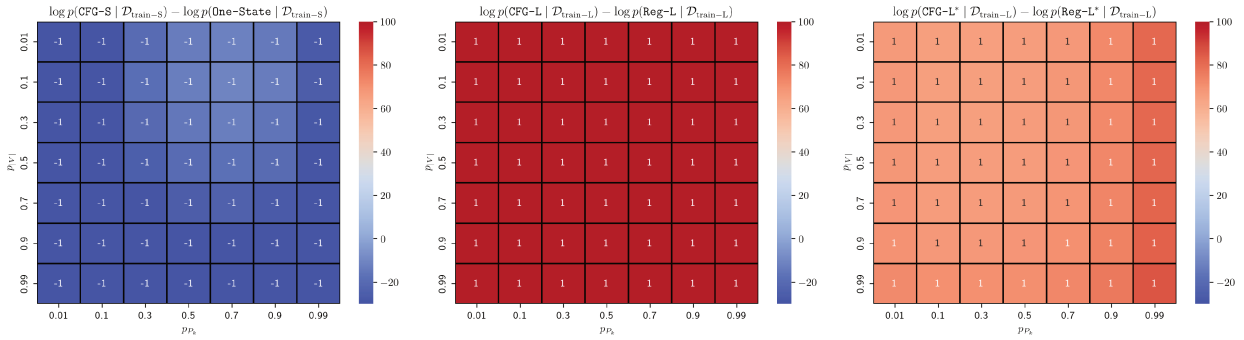


(c) High diversity data case – $\mathcal{D}_{\text{train}-\text{L}}$ for BMM minimized grammars

Figure 14: Sensitivity analysis on varying the geometric distribution parameter p. Note that the same p is used for both $p(|V|)$ and $p(P_k)$ here. For the BMM minimized case (*right*), we omit the Flat and one-state grammar to show the difference between the posteriors for `CFG-L`* and `Reg-L`* more clearly, as the other two grammars obtain much worse posteriors compared to the two (see the *middle* figure)

.

Example parses Example parses from the CFG and regular grammars are provided in Figures 16 and 17 respectively.



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

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

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

Figure 15: Sensitivity analysis on varying the geometric distribution parameter $p_{|V|}$ for $p(|V|)$ and p_{P_k} for $p(P_k)$. We plot the difference between the log-posterior of the CFG and the other grammar with the highest posterior, which is `One-State` for $\mathcal{D}_{\text{train}-\text{S}}$ and `Reg-L` (or `Reg-L`* for BMM minimized case) for $\mathcal{D}_{\text{train}-\text{L}}$. The values in the heatmaps correspond to the sign of the difference between the posteriors (1 for positive and -1 for negative). A positive sign implies that the CFG has the higher posterior than the alternate grammar and negative sign implies otherwise.

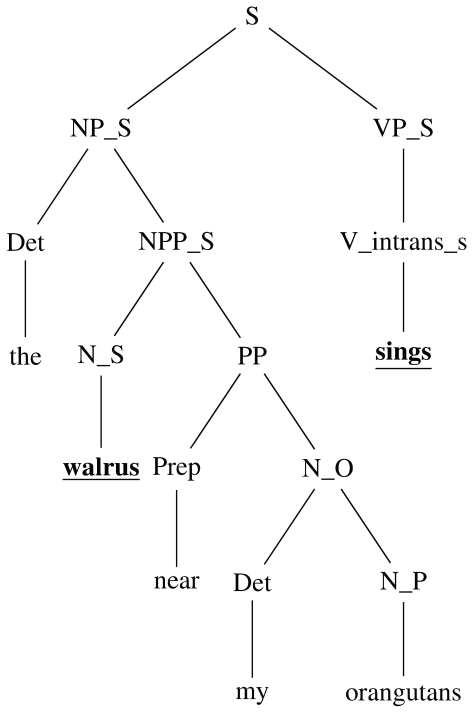


Figure 16: Example of production from the CFG with agreement following hierarchical rule

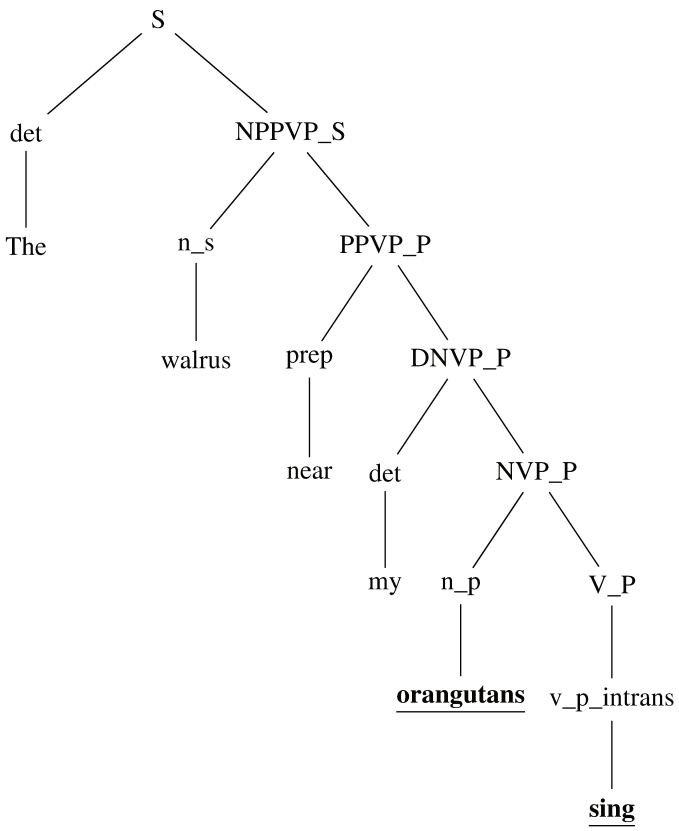
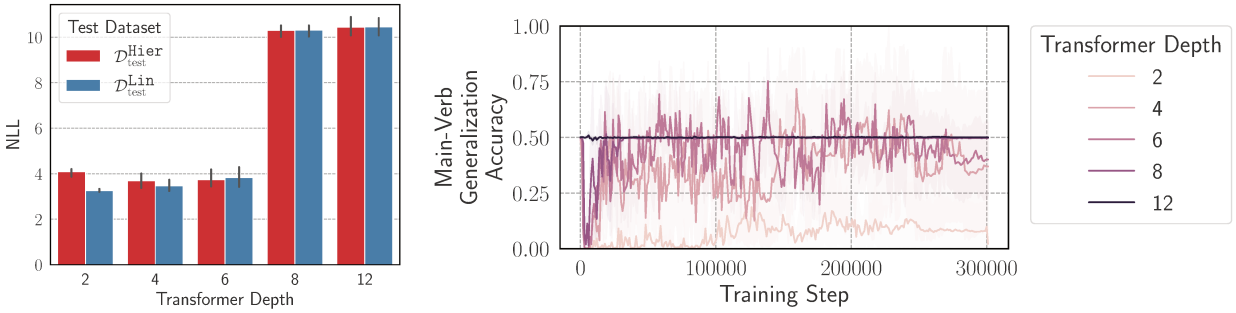


Figure 17: Example of production from the linear grammar with agreement following order rule



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

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

Figure 18: Do transformer models trained on $\mathcal{D}_{\text{train}-\text{S}}$ ever show hierarchical generalization? We vary the depth of the transformer-LM (number of decoder layers) and find in no case, transformer exhibiting hierarchical generalization. Interestingly, for smaller depths, we see the models generalizing according order rule, indicated by lower NLL on $\mathcal{D}_{\text{test}}^{\texttt{Lin}}$ than $\mathcal{D}_{\text{test}}^{\texttt{Hier}}$ 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.