Increasing The Performance of Cognitively Inspired Data-Efficient Language Models via Implicit Structure Building

Omar Momen, **David Arps** and **Laura Kallmeyer**
Heinrich Heine University
Düsseldorf, Germany
{omar.hassan,david.arps,laura.kallmeyer}@hhu.de

Abstract

In this paper, we describe our submission to the BabyLM Challenge 2023 shared task on data-efficient language model (LM) pretraining (Warstadt et al., 2023). We train transformer-based masked language models that incorporate unsupervised predictions about hierarchical sentence structure into the model architecture. Concretely, we use the Structformer architecture (Shen et al., 2021) and variants thereof. StructFormer models have been shown to perform well on unsupervised syntactic induction based on limited pretraining data and to yield performance improvements over a vanilla transformer architecture (Shen et al., 2021). Evaluation of our models on 39 tasks provided by the BabyLM challenge shows promising improvements of models that integrate a hierarchical bias into the architecture at some particular tasks, even though they fail to consistently outperform the baseline model on all tasks.[1]

1 Introduction

Transformer-based Language Model (LM) performance is heavily influenced by three scaling factors: the number of model parameters, the pretraining dataset size, and the amount of computing. For optimal performance, all three factors must be simultaneously scaled up (Kaplan et al., 2020). This scaling law has introduced several challenges in advancing research on neural language modeling. One major obstacle lies in the unequal distribution of resources across languages. Consequently, the current approach of transformer-based models falls short of achieving equally high-performance levels for models dedicated to different languages (Choudhury and Deshpande, 2021).

Moreover, we see a considerable difference when comparing the way LMs learn how humans acquire language. One difference concerns the data that is input to learning: LMs such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019b) or GPT-3 (Brown et al., 2020) are exposed to billions of tokens during training, far surpassing what an individual human is exposed to when learning a language (Warstadt and Bowman, 2022). This fundamental discrepancy raises important considerations when drawing parallels between language learning in machines and humans.

To improve the data-efficiency of LMs, one direction is to adapt the model architecture. An effective approach in this endeavor involves incorporating an inductive bias into the models' architectures, which could potentially facilitate acquiring more knowledge from the same amount of data compared to standard models. However, the specific type of inductive bias to be added is still under exploration. Recently, there have been efforts to investigate the use of syntactic hierarchical inductive biases as a potential improvement (Mulligan et al., 2021; Papadimitriou and Jurafsky, 2023).[2]

One of these potential solutions is the StructFormer architecture (Shen et al., 2021), a transformer that is trained on the masked language modeling task. An additional convolutional neural network (CNN) component produces unlabeled dependency and constituency trees as a byproduct and influences the self-attention mechanism of the transformer layers. The model has obtained demonstrated competitive results in structure induction evaluations and a decrease in perplexity over a vanilla transformer baseline (Vaswani et al., 2017). However, it is an open question whether the inductive bias learned in this architecture enhances performance on downstream NLP tasks.

We pretrain the StructFormer architecture on a dataset from a different domain that had not been tested on that model before. Moreover, we use a

---

[1] Implementation and models checkpoints can be found here: https://github.com/omarTronto/structformer-babylm

[2] Note that we don't want to claim that humans integrate such an inductive bias and therefore can learn language with less data, compared to large LMs.

more sophisticated tokenizer in comparison to the most frequent words dictionary used to train the models in the original experiment. Additionally, we modify the model architecture to investigate whether injecting a hierarchical bias in the middle layers of the transformer architecture (rather than after the embedding layer) leads to improved downstream performance. Eventually, we evaluate seven model variants through the evaluation pipeline of the shared task and submit our best-performing model to the shared task challenge.

1.1 The BabyLM Challenge

The BabyLM Challenge is a shared task with the aim of data-efficient language modeling for English. Participants pretrain a LM from scratch on data that corresponds to the amount of linguistic data available to a child. The task is a great setting for conducting our experiments. It provides us with a pretraining dataset, a thorough evaluation pipeline, and, furthermore, an environment where we can compare our models' performance to other interesting architectures from the systems participating in the shared task.

Dataset The shared task is conducted in two tracks with different dataset sizes: a 100M words corpus, and a 10M words corpus as a sample of the larger corpus. The size is inspired by the assumption that children are exposed to 2M-7M words per year (Gilkerson et al., 2017). To account for the fact that children mostly interact with spoken rather than written language data, the datasets include a high proportion of transcribed data from different domains. For more details regarding the source domains, please refer to Warstadt et al. (2023).

Evaluation A thorough evaluation pipeline that comprises 39 different tasks is used to evaluate every model participating in the shared task. These tasks are supposed to represent a model's performance with respect to efficiency and applied NLP, as well as cognitive science, and linguistics. A group of 17 tasks, named *BLiMP* (Warstadt et al., 2020a) are performed via zero-shot predictions, while the other two groups of tasks; *SuperGLUE* (11 tasks, Wang et al., 2019) and *MSGS* (11 tasks, Warstadt et al., 2020b) need finetuning of the submitted models for classification. Refer to Appendix A for the complete list of tasks.

2 Language Modeling and Hierarchical Information

Transformer LMs use syntactic information in their predictions. This has been shown by work on interpreting their internal representations as well as by investigating the grammatical correctness of their predictions (Mahowald et al., 2023; Kulmizev and Nivre, 2022). However, the vanilla transformer architecture that underlies both encoder and decoder-based LMs does not encode hierarchical information explicitly. Rather, objectives such as masked language modeling and next-token prediction are based on linear relationships between tokens. This has inspired two lines of work that incorporate hierarchical knowledge into LMs. The first group of papers introduces models in which the training objective involves syntactic labels explicitly (e.g. Dyer et al., 2016; Sartran et al., 2022), The second group introduces models in which hierarchical information is encoded implicitly as a byproduct of a language modeling task (Shen et al., 2018, 2021; Li et al., 2019; Kim et al., 2019; Choi et al., 2018; Williams et al., 2018). We consider the second group of models more relevant for this shared task since it allows us to train models with a hierarchical architecture bias on raw text data. In particular, we use the StructFormer model (Shen et al., 2021), a transformer in which one architecture component, the parser network, predicts the position of each token in the hierarchical structure of the sentence. The prediction of the parser network puts soft constraints on the attention mask of the transformer layers. The model is pretrained on the masked language modeling task, and we view two experimental contributions of Shen et al. (2021) as most relevant for using this model: First, they show that a StructFormer achieves lower perplexity on limited training data than a transformer that replaces the parser network with standard self-attention. Second, the induced hierarchical structure corresponds to unlabeled dependency trees. Concretely, evaluation on the Penn Treebank (PTB) shows that 61.6% of the undirected dependency edges are recovered. We further implement a variant of the model in which the parser network predicts hierarchical information based on hidden states that are contextualized with classical transformer layers, rather than using uncontextualized token embeddings as direct input to the parser network (Sec. 3.2.4).

3 Experiment

This section introduces the objectives of our experiment, a description of the model architectures, and the technical aspects of the pretraining and evaluation process.

3.1 Objectives

In this work, we aim to validate the claim that the performance of LMs, in particular on syntax-sensitive tasks, can be improved through the implicit integration of an inductive bias into the model's architecture that yields a hierarchical structure of the tokens. Concretely, we conduct experiments towards pursuing the following three primary objectives:

1. Assess the robustness of the finding that LM performance is enhanced through the utilization of a linguistically informed model architecture (Shen et al., 2021).

2. Investigate whether the claim that transformer architectures better represent syntactic information in their middle attention layers is supported in a practical use case (Vig and Belinkov, 2019; Arps et al., 2022; Müller-Eberstein et al., 2022).

3. Develop models that surpass the performance of the baseline models offered by the organizers of the shared task.

3.2 Methodology

In order to address the questions posed by the experiment's objectives, we train a tokenizer, develop several model variants, and perform iterations of model pretraining, finetuning, and evaluation. Due to limited resources, we only conducted our experiments on the 10M words dataset. Furthermore, from the model architectures provided by the shared task, we chose the encoder-type models due to their adaptability for integrating a hierarchical bias in the model architecture.

3.2.1 Tokenizer

We use the same tokenizer across all variations of our models. Specifically, we train a Byte Pair Encoding (BPE) tokenizer (Sennrich et al., 2016; Gage, 1994) from scratch on the 10M BabyLM corpus. Since BPE tokenizers require specifying the vocabulary size as a hyperparameter before training on the corpus, we carefully determined an appropriate size. Our goal was to obtain a tokenizer that

Vocabulary Size	Least Frequent Tokens	Frequency
8K	sought, arts, stolen, ATOR	230
10k	accounts, seated, lemn, feathers	165
12k	sailors, goss, reun, irlines	126
16k	sophisticated, olleyball, AMES, poorly	80
32k	jets, estus, iesselin, UCLA, mannik	26

Table 1: Tokenizer Vocabulary Size Experiments

accurately represents tokens in our relatively small dataset while adhering to best practices for LMs. To achieve this, we train the tokenizer on the same corpus with different vocabulary sizes. We then observed the resulting vocabularies and identified the least frequent tokens within each (Table 1).

Based on our analysis, a vocabulary size of 32K tokens provides a fair representation relative to the corpus size for the least frequent tokens. Additionally, Geiping and Goldstein (2022) found that a BPE tokenizer with 32K tokens yielded the best results.

3.2.2 Baseline model

To achieve objective 1, we pretrained a standard transformer architecture that we call *transformer-base*, using our custom-trained tokenizer and following the same model and training hyperparameters to minimize any effects due to uncontrolled variables.

3.2.3 Hyperparameters

Due to resource limitations, and to assure fair comparisons between models, we use one set of pretraining and finetuning hyperparameters: We chose the default hyperparameters settings that were used to pretrain the shared task baseline models (Warstadt et al., 2023). In order to speed up the evaluation of finetuning tasks, we made modifications to the finetuning hyperparameters that were used to evaluate the baseline models. Our main hyperparameters are reported in Appendix B. We pretrain all models with the same batch size and the same number of steps. We use the training pipeline that Warstadt et al. (2023) introduced to train their baseline modes to minimize any effects due to uncontrolled variables.

However, one variable that could not be fixed during the experiment is the number of trainable parameters in each model. When adding a convolution parser network to a particular model, the increase in the number of parameters in that model is inevitable (parameter counts are listed in Appendix B). We are aware that this can have misleading effects on the results and conclusions, however, we

still think that the experiment in its current setting can show interesting behaviors that may encourage further investigation in a fully controlled experiment.

3.2.4 Model Architectures

We develop two primary variants of model architectures for our experiment.

StructFormer This variant (Figure 1) closely follows the architecture in Shen et al. (2021). In brief, it incorporates a parser network that consists of 4 convolution layers. The input to the parser network is token embeddings, and the output is probability distributions for dependencies between tokens. These distributions are then integrated into the multi-head self-attention mechanism of a standard transformer model. For a complete description of the architecture, we refer readers to Shen et al. (2021). We name models of this variant by the prefix *structformer*.

StructRoBERTa The second variant (Figure 1) is similar to the StructFormer, but instead of employing a standard transformer, it utilizes a base RoBERTa encoder (Liu et al., 2019b). We modify the HuggingFace (Wolf et al., 2020) implementation, which has a few differences from the vanilla transformer implementation, mainly adding normalization and dropout layers after the embeddings layer, and also adding an additional intermediate block within each layer. The models following this architecture will be identified with the prefix *structroberta*.

Vanilla transformer For transformers without parser networks, we reuse the implementation by Shen et al. (2021) which follows the standard transformer introduced by Vaswani et al. (2017), except that a layer normalization is added in front of each layer.

Variants Subsequently, for each of the main variants, *structformer* and *structroberta*, we create two sub-variants to explore a different placement of the parser network within the architecture (Figure 2). This decision is based on insights from previous experiments, which indicate that syntactic information tends to be better represented in the middle layers of the transformer (Liu et al., 2019a; Vig and Belinkov, 2019; Arps et al., 2022).

In our approach, we divide the initial $n_{context}$ layers of either the transformer or RoBERTa component in *structformer* or *structroberta* respectively.
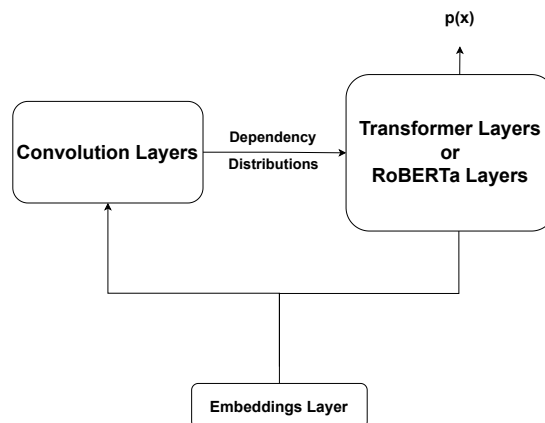

Figure 1: StructFormer and StructRoBERTa Architectures (s_1)

We label these $n_{context}$ layers as the Front Attention Layers, while the remaining attention layers are labeled as Rear Attention Layers. The input embeddings pass through the Front component, generating embeddings that are subsequently fed into the parser network. The parser network, in turn, outputs dependency distributions that are integrated into the Rear component of the architecture.
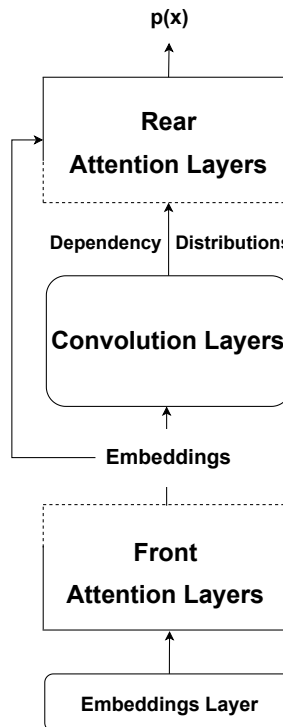

Figure 2: In-between Parser Architectures (s_2), dotted lines indicate intervening the encoder layers at two positions, where the parser network connects the two split parts of the encoder

To distinguish between the two sub-variants, we append the suffix s_1 to models with the parser network before the attention layers (Figure 1), and the suffix s_2 to models with the parser network in-between the middle attention layers (Figure 2).

To achieve objective 3, we introduce two additional models, *structroberta$_{s1'}$* and *structroberta$_{s2'}$*, to enhance the evaluation scores so we could submit the best attainable results to the shared task. These two models are basically an upgrade in the number of convolution layers (from 4 to 6) of the parser network in *structroberta$_{s1}$* and *structroberta$_{s2}$* respectively.

4 Results

After completing the pretraining process of the 7 investigated models, a comprehensive linguistic evaluation is conducted for the seven models under study. The shared task evaluation pipeline is used for this purpose. Detailed evaluation results are presented in Tables 2 3, 4, and 5. We compare the scores of the following models: *transformer-base* (TF$_{base}$), *structformer$_{s1}$* (SF$_{s1}$), *structformer$_{s2}$* (SF$_{s2}$), *structroberta$_{s1}$* (SR$_{s1}$), *structroberta$_{s2}$* (SR$_{s2}$), *structroberta$_{s1'}$* (SR$_{s1'}$) and *structroberta$_{s2'}$* (SR$_{s2'}$). We are particularly interested in assessing to which extent the introduction of a hierarchical bias improves a model's performance on a specific task. Therefore, in addition to the scores of the individual models, we also report the differences in scores as follows:

- $\Delta_{SF_{s1}} = Score(SF_{s1}) - Score(TF_{base})$
- $\Delta_{SF_{s2}} = Score(SF_{s2}) - Score(TF_{base})$
- $\Delta_{SR_{s12}} = Score(SR_{s1}) - Score(SR_{s2})$
- $\Delta_{SR_{s1'}} = Score(SR_{s1'}) - Score(SR_{s1})$
- $\Delta_{SR_{s2'}} = Score(SR_{s2'}) - Score(SR_{s2})$

All numerical values in the result tables are measures of accuracy unless explicitly stated otherwise.

4.1 Pseudo-perplexity

We report the corpus-level pseudo-perplexity ($PPPL$, Salazar et al., 2020) on the test split of the BabyLM shared task dataset[3] (Table 2). $PPPL$ is computed by masking out each token in turn and collecting the log-likelihoods. This evaluation contributes to objective 1 in our experiment. Shen et al. (2021) found that *structformer*

---

[3]We use Kauf and Ivanova (2023)'s implementation for computing $PPPL$ scores and remove the 100 longest sentences from the dataset to reduce the computation time.

models incorporating hierarchical inductive bias achieve lower $PPPL$ than their baseline *transformer* model. We want to assess this finding on the BabyLM dataset and using our custom-trained tokenizer. SF$_{s1}$ shows lower $PPPL$ compared to TF$_{base}$, which follows the previous findings. However, the model with a parser network within the middle layers shows a higher $PPPL$ than the baseline TF$_{base}$. The addition of more convolution layers at the parser network shows an improvement at SR$_{s2'}$ but surprisingly shows a deterioration at SR$_{s1'}$.

4.2 BLiMP

BLiMP is a challenging benchmark comprising a set of tests designed to evaluate the linguistic knowledge of LMs with a specific focus on linguistic phenomena encompassing syntax, morphology, and semantics (Warstadt et al., 2020a). Originally, the benchmark consisted of 12 tasks (see Appendix A). Additionally, in the shared task (Warstadt et al., 2023), 5 more tasks were added to BLiMP as held-out tasks, aiming to assess the generalization capabilities of the submitted models. The random chance accuracy for all original BLiMP tasks is 50, while chance was not reported for the additional 5 supplement tasks.

According to the BLiMP scores in Table 3, within the *Set A* models, the models incorporating hierarchical inductive bias (SF$_{s1}$ and SF$_{s2}$) do not show consistent outperformance or underperformance in comparison to the baseline model TF$_{base}$.

However, on average, the SF$_{s1}$ model is on par with and occasionally outperforms the TF$_{base}$ model. In particular, SF$_{s1}$ excels in the following tests: Argument Structure, Determiner Noun Agreement, Filler Gap, Irregular Forms, Quantifiers, and Subj. Verb Agreement. Conversely, SF$_{s1}$ underperforms the TF$_{base}$ in the tasks of QA Congruence Easy, Subject Aux Inversion and Turn Taking. We hypothesize that this is because syntactic knowledge is helpful for the former list of tasks, but to a lesser degree for the latter, for example, Turn Taking, which focuses on knowledge of discourse and dialogue structure, in particular of referential properties of NPs, which is not reflected in the syntactic structure. A sample pair from this data set is *"Should you quit?" – "No, I shouldn't."* (good) versus *"Should she quit?" – "No, I shouldn't."* (bad). The negative and the positive data points have the same syntactic structure and the dependents are

	Set A			Set B			
	TF_{base}	SF_{s1}	SF_{s2}	SR_{s1}	SR_{s2}	$SR_{s1'}$	$SR_{s2'}$
Perplexity	32.84	26.48	38.26	**21.15**	23.15	37.11	22.48

Table 2: Perplexity Results

	Set A					Set B						
	TF_{base}	SF_{s1}	SF_{s2}	$\Delta_{SF_{s1}}$	$\Delta_{SF_{s2}}$	SR_{s1}	SR_{s2}	$SR_{s1'}$	$SR_{s2'}$	$\Delta_{SR_{s12}}$	$\Delta_{SR_{s1'}}$	$\Delta_{SR_{s2'}}$
Anaphor Agreement	88	88	74	0	-14	89	87	90	87	-2	1	0
Argument Structure	68	69	68	1	0	69	72	73	68	3	4	-4
Binding	68	68	66	0	-2	72	70	70	67	-2	-2	-3
Control Raising	66	66	64	0	-2	69	70	68	63	1	-1	-7
Det. Noun Agreement	87	90	86	3	-1	92	93	93	88	1	1	-5
Ellipsis	79	79	72	0	-7	70	71	77	70	1	7	-1
Filler Gap	63	70	63	7	0	69	67	74	64	-2	5	-3
Irregular Forms	76	90	86	14	10	83	92	85	84	9	2	-8
Island Effects	44	44	37	0	-7	49	45	52	43	-4	3	-2
NPI Licensing	58	58	55	0	-3	55	59	68	53	4	13	-6
Quantifiers	73	78	73	5	0	71	68	68	71	-3	-3	3
Subj. Verb Agreement	64	70	60	6	-4	75	75	76	66	0	1	-9
Hypernym	50	50	50	0	0	48	48	50	49	0	2	1
QA Congruence Easy	59	56	56	-3	-3	64	69	66	64	5	2	-5
QA Congruence Tricky	38	35	35	-3	-3	28	34	28	28	6	0	-6
Subject Aux Inversion	82	78	81	-4	-1	70	71	76	70	1	6	-1
Turn Taking	67	65	55	-2	-12	61	59	60	61	-2	-1	2
Average	66.5	67.9	63.6	1.4	-2.9	66.7	67.7	**69.1**	64.5	0.9	2.4	-3.2

Table 3: BLiMP Results

perfectly fine as argument fillers.

While the model with a parser network in-between the middle layers SF_{s2}, underperforms TF_{base} on average, but interestingly it demonstrates a noteworthy improvement in the specific task of Irregular Forms. Remarkably, similar to SF_{s1}, SF_{s2} significantly outperform TF_{base} in this particular task. The task of Irregular Forms involves aspects of lexical decisions but the syntax of course also plays a role.

Within the RoBERTa model variations in Set B, again the model with a parser network in-between the middle layers SR_{s2} fails to improve over the one with a parser network ahead of the encoder layers SR_{s1} in most of the tasks. It even gets worse with the upgrade in the number of convolution layers within the parser network at $SR_{s2'}$. On the other hand, the upgrade in the number of convolution layers at $SR_{s1'}$ shows also an upgrade in accuracies over SR_{s1}. Generally, $SR_{s1'}$ achieves the best results among all the investigated models on average.

Moreover, the Set B models exhibit improvements over Set A models in the tests of Binding, Det. Noun Agreement, Subject Verb Agreement, and QA Congruence Easy.

It is not so clear how to interpret the results of the two Question Answering (QA) Congruence tasks, where the baselines achieve only very low scores. For the QA Congruence Easy task, which tests for detecting selectional preference violations on ob-

ject fillers in answers (e.g., *"What did you sell? - A chair."* (good) versus *"What did you sell? - Sarah."* (bad)), knowing about the syntactic structure of the first sentence probably helps to apply selectional restrictions and thereby assessing the quality of the second as a possible reply. This might be the reason why we see an improvement in model performance in the SR models when adding implicit hierarchical information that reflects syntactic dependencies. The QA Congruence Tricky task is similar, except that the selectional preference that is violated in the negative data points does not refer to the direct object. Furthermore, the object is dropped in most examples and sometimes the (incorrect) argument filler would be a plausible direct object (e.g., *"Who ate? - Sarah ate."* (good) versus *"Who ate? - Pasta ate."* (bad)). This is why the task is tricky. In this context, it is important to keep in mind that our StructFormer models learn only unlabeled dependencies and therefore cannot distinguish between object and subject. This means that for *Pasta ate*, a structure would be implicitly predicted where *pasta* is a dependent of *ate*, which is perfectly fine semantically (as a direct object). This might be a reason why the structformer models struggle with this test and partly lead to a decrease in the performance, compared to our baseline, since the unlabeled dependency tree actually licenses the negative data points.

4.3 SuperGLUE

SuperGLUE consists of eleven diverse tasks (see Appendix A) which evaluate various performance aspects. These tasks include sentiment analysis, linguistic acceptability judgments, entailment detection, and semantic similarity evaluations of words within contexts, among others (Wang et al., 2019).

The scores (see Table 4) in most of the tasks fall in a narrow range across all the investigated models. The incorporation of hierarchical inductive bias does not show clear improvements in most of the tasks. A noticeable result that is observed for the models with a parser network within the middle layers *(s2)* is the result of the MRPC task, where *s2* models consistently outperform the *s1* models in both sets for this particular task. The upgrade in the number of convolution layers also does not show a clear improvement in most of the tasks for both $SR_{s1'}$ and $SR_{s2'}$ models.

Notably, in the case of the WSC task, we observe that all models' predictions heavily favored one specific class. This raises concerns about the success of the finetuning process for this particular task.

4.4 MSGS

The MSGS tasks, listed in Appendix A, were introduced by the shared task as held-out tests specifically designed to evaluate generalization capabilities. Detailed information and further insights about these tasks are expected to be disclosed in an upcoming publication. MSGS tasks are measured using the Matthews correlation coefficient (MCC). MCC is used in machine learning as a measure of the quality of binary (two-class) classifications, introduced by Matthews (1975)

The MSGS results (Table 5), resemble to the SuperGLUE results. The models incorporating hierarchical inductive bias show contradicting behavior across the different tasks. While for some tasks e.g Control Raising (Control), Relative Position (Control), and Syntactic Category (Relative Position), SF_{s1} and SF_{s2} are strengthening the correlation in comparison to the baseline model, but with other tasks e.g Lexical Content (Control), Main Verb (Lexical Content) and Syntactic Category (Lexical Content), SF_{s1} and SF_{s2} are shown weakening the correlation.

4.5 Aggregation

Indeed, analyzing the performance changes across 39 tasks for 7 different models is a complex process. To simplify the assessment and present a concise summary of each model's overall performance, we report an aggregate score of all the 39 scores for each model (Table 6). This aggregation approach was internally computed by the shared task submission platform to represent each model with a single score, providing a more straightforward evaluation of the overall performance. Subsequently, we select the model with the best aggregate score $SR_{s1'}$ to represent our submission in the shared task.

5 Discussion

Although the evaluation pipeline of the shared task was meticulously designed to encompass a comprehensive analysis of pretrained LMs, covering aspects of efficiency, applied NLP standards, cognitive science, linguistics, and language acquisition (Warstadt et al., 2023), it was discussed in Warstadt et al. (2020a) that some tasks that involve semantic phenomena such as Island Effects and NPI Licensing are very difficult for LMs in general. Consequently, the consistently low performance observed across all models on these tests can be attributed to this matter. As a result, we refrain from considering the aggregate score as a single definitive metric for representing how a model's performance compares to another. Instead, we advocate for a thorough investigation of individual tests while considering the test's objectives, dataset, and evaluation strategy.

Overall, the models incorporating hierarchical inductive bias did not show significant improvement in the scores of the BabyLM evaluation tasks, however, some exceptions of the evaluation tasks that show improvements in terms of scores when using the *structformer* and *structroberta* models, encourage a deeper investigation for patterns in the outputs predictions that might lead to a different conclusion. Namely, the tasks that we think are worth more investigation are: *Argument Structure, Determiner Noun Agreement, Filler Gap, Irregular Forms, Quantifiers, Subj. Verb Agreement, Control Raising (Control), Relative Position (Control) and Syntactic Category (Relative Position).*

Contrary to our expectations, the modification of placing the parser in-between the middle attention layers has not demonstrated notable improvements but rather a decline in performance compared to the models with the parser placed right after the input

	Set A					Set B						
	$\textbf{TF}_{base}$	$\textbf{SF}_{s1}$	$\textbf{SF}_{s2}$	$\Delta_{SF_{s1}}$	$\Delta_{SF_{s2}}$	$\textbf{SR}_{s1}$	$\textbf{SR}_{s2}$	$\textbf{SR}_{s1'}$	$\textbf{SR}_{s2'}$	$\Delta_{SR_{s12}}$	$\Delta_{SR_{s1'}}$	$\Delta_{SR_{s2'}}$
BoolQ	63	61	62	-2	-1	66	66	64	65	0	-2	-1
COLA (MCC)	0.16	0.19	0.14	—	—	0.23	0.23	0.19	0.26	—	—	—
MNLI	71	71	70	0	-1	72	72	69	72	0	-3	0
MNLI-MM	72	73	72	1	0	73	73	70	73	0	-3	0
MRPC (F1)	75	75	79	0	4	76	81	77	75	5	1	-6
MultiRC	61	58	62	-3	1	62	59	59	54	-3	-3	-5
QNLI	81	77	78	-4	-3	71	72	66	74	1	-5	2
QQP (F1)	81	82	81	1	0	82	82	80	81	0	-2	-1
RTE	48	42	47	-6	-1	46	57	53	56	11	7	-1
SST2	87	85	82	-2	-5	87	82	86	83	-5	-1	1
WSC	61	61	61	0	0	61	59	61	61	-2	0	2

Table 4: (Super)GLUE Results. Values are not aggregated across each model due to the presence of different metrics (Accuracy, F1 score, and MCC)

	Set A			Set B			
	$\textbf{TF}_{base}$	$\textbf{SF}_{s1}$	$\textbf{SF}_{s2}$	$\textbf{SR}_{s1}$	$\textbf{SR}_{s2}$	$\textbf{SR}_{s1'}$	$\textbf{SR}_{s2'}$
Control Raising (Control)	0.54	0.56	0.69	0.57	0.56	0.69	0.56
Control Raising (Lexical Content)	-0.45	-0.04	-0.02	-0.03	-0.07	-0.36	-0.14
Control Raising (Relative Position)	-0.94	-0.89	-0.92	-1.00	-0.98	-0.77	-0.98
Lexical Content (Control)	1.00	0.88	0.6	1.00	0.98	1.00	0.78
Main Verb (Control)	0.93	0.96	0.84	0.85	0.98	0.96	0.98
Main Verb (Lexical Content)	-1.00	-0.79	-0.84	-1.00	-1.00	-0.99	-1.00
Main Verb (Relative Position)	-0.87	-0.78	-0.89	-0.98	-0.93	-0.83	-0.95
Relative Position (Control)	0.67	0.81	0.78	0.86	0.95	0.97	1.00
Syntactic Category (Control)	0.62	0.23	0.47	0.80	0.73	0.66	0.87
Syntactic Category (Lexical Content)	-0.61	-0.17	-0.17	-0.42	-0.59	-0.26	-0.76
Syntactic Category (Relative Position)	-0.32	-0.57	-0.44	-0.47	-0.47	-0.63	-0.52

Table 5: MSGS Results

	Set A			Set B			
	$\textbf{TF}_{base}$	$\textbf{SF}_{s1}$	$\textbf{SF}_{s2}$	$\textbf{SR}_{s1}$	$\textbf{SR}_{s2}$	$\textbf{SR}_{s1'}$	$\textbf{SR}_{s2'}$
Aggregate Score	0.52	0.53	0.52	0.53	0.54	**0.55**	0.52

Table 6: Shared Task Leaderboard Results

embedding layer. We can only speculate about why this is so. It might be that it is an advantage to push the model very early towards identifying structural relations between words. More precisely to do so at a stage where the contributions of the single tokens are still separated from each other. The parsing network placed between the middle layers acts at a moment where single token contributions are already blurred.

To understand the effect of placing the parser network within the middle layers, we propose probing the layers of the Front and Rear modules and comparing them to the corresponding layers in the model where the parser network is placed ahead of the attention layers. Such a comparative analysis can provide valuable insights and either support or contradict our hypothesis regarding the learning of syntactic features in the middle layers of transformer models.

Regarding the aim of achieving competitive scores on the shared task challenge, the best score we could get was from the model $structroberta_{s1'}$, this model is an upscaling of the $structroberta_{s1}$.

6 Conclusion

In this paper, we extend the work of Shen et al. (2021) to explore the capabilities of the Struct-Former architecture as an example of employing hierarchical bias in addressing the challenges posed by relatively small LLM pretraining datasets. Furthermore, we modify the StructFormer architecture to examine whether integrating the hierarchical bias within the middle attention layers leads to performance improvements. To accomplish these objectives, we pretrain seven model variants using the same dataset and configuration settings. We evaluate these models on 39 different tasks. The evaluation outcomes reveal varying behavior across the models, exhibiting inconsistencies in performance. We could not show strong evidence that models incorporating hierarchical bias are performing better in the context of this shared task, nor could we show practical evidence for the claim that syntactic information is better represented in the middle attention layers within the scope of our experiment. We have noted substantial enhancements in certain tasks when models incorporate hierarchical bias in their architectural designs. Nonetheless, to ensure the reliability of our findings and to eliminate potential confounding factors related to the varying number of parameters in each model, as well as the distinct objectives and complexities of individual tasks, we intend to carry out an in-depth analysis of each model's performance on a task-by-task basis.

References

David Arps, Younes Samih, Laura Kallmeyer, and Hassan Sajjad. 2022. Probing for constituency structure in neural language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6738–6757, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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. 2020. Language models are few-shot learners.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2018. Learning to compose task-specific tree structures. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press.

Monojit Choudhury and Amit Deshpande. 2021. How linguistically fair are multilingual pre-trained language models? *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12710–12718.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of*

the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.

Philip Gage. 1994. A new algorithm for data compression. *C Users J.*, 12(2):23–38.

Jonas Geiping and Tom Goldstein. 2022. Cramming: Training a language model on a single gpu in one day.

Jill Gilkerson, Jeffrey A. Richards, Steven F. Warren, Judith K. Montgomery, Charles R. Greenwood, D. Kimbrough Oller, John H. L. Hansen, and Terrance D. Paul. 2017. Mapping the early language environment using all-day recordings and automated analysis. *American Journal of Speech-Language Pathology*, 26(2):248–265.

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

Carina Kauf and Anna Ivanova. 2023. A better way to do masked language model scoring. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 925–935, Toronto, Canada. Association for Computational Linguistics.

Yoon Kim, Alexander Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer, and Gábor Melis. 2019. Unsupervised recurrent neural network grammars. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1105–1117, Minneapolis, Minnesota. Association for Computational Linguistics.

Artur Kulmizev and Joakim Nivre. 2022. Schrödinger's tree—on syntax and neural language models. *Frontiers in Artificial Intelligence*, 5.

Bowen Li, Lili Mou, and Frank Keller. 2019. An imitation learning approach to unsupervised parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3485–3492, Florence, Italy. Association for Computational Linguistics.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach.

Kyle Mahowald, Anna A. Ivanova, Idan A. Blank, Nancy Kanwisher, Joshua B. Tenenbaum, and Evelina Fedorenko. 2023. Dissociating language and thought in large language models: a cognitive perspective.

B.W. Matthews. 1975. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 405(2):442–451.

Max Müller-Eberstein, Rob van der Goot, and Barbara Plank. 2022. Probing for labeled dependency trees. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7711–7726, Dublin, Ireland. Association for Computational Linguistics.

Karl Mulligan, Robert Frank, and Tal Linzen. 2021. Structure here, bias there: Hierarchical generalization by jointly learning syntactic transformations. In *Proceedings of the Society for Computation in Linguistics 2021*, pages 125–135, Online. Association for Computational Linguistics.

Isabel Papadimitriou and Dan Jurafsky. 2023. Pretrain on just structure: Understanding linguistic inductive biases using transfer learning.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. 2020. Masked language model scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2699–2712, Online. Association for Computational Linguistics.

Laurent Sartran, Samuel Barrett, Adhiguna Kuncoro, Miloš Stanojević, Phil Blunsom, and Chris Dyer. 2022. Transformer grammars: Augmenting transformer language models with syntactic inductive biases at scale. *Transactions of the Association for Computational Linguistics*, 10:1423–1439.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Yikang Shen, Zhouhan Lin, Chin wei Huang, and Aaron Courville. 2018. Neural language modeling by jointly learning syntax and lexicon. In *International Conference on Learning Representations*.

Yikang Shen, Yi Tay, Che Zheng, Dara Bahri, Donald Metzler, and Aaron Courville. 2021. StructFormer: Joint unsupervised induction of dependency and constituency structure from masked language modeling. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7196–7209, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Jesse Vig and Yonatan Belinkov. 2019. Analyzing the structure of attention in a transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, Florence, Italy. Association for Computational Linguistics.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. *SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems*. Curran Associates Inc., Red Hook, NY, USA.

Alex Warstadt and Samuel R. Bowman. 2022. What artificial neural networks can tell us about human language acquisition.

Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Gotlieb Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Adina Williams, Bhargavi Paranjabe, Tal Linzen, and Ryan Cotterell. 2023. Findings of the 2023 BabyLM Challenge: Sample-efficient pretraining on developmentally plausible corpora. In *Proceedings of the 2023 BabyLM Challenge*. Association for Computational Linguistics (ACL).

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020a. BLiMP: The benchmark of linguistic minimal pairs for English. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Alex Warstadt, Yian Zhang, Xiaocheng Li, Haokun Liu, and Samuel R. Bowman. 2020b. Learning which features matter: RoBERTa acquires a preference for linguistic generalizations (eventually). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 217–235, Online. Association for Computational Linguistics.

Adina Williams, Andrew Drozdov, and Samuel R. Bowman. 2018. Do latent tree learning models identify meaningful structure in sentences? *Transactions of the Association for Computational Linguistics*, 6:253–267.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

Appendix

A Evaluation Tasks

BLiMP

1. Anaphor Agreement

2. Argument Structure

3. Binding

4. Control Raising

5. Determiner Noun Agreement

6. Ellipsis

7. Filler Gap

8. Irregular Forms

9. Island Effects

10. Negative Polarity Items NPI Licensing

11. Quantifiers

12. Subject Verb Agreement

13. Hypernym

14. QA Congruence Easy

15. QA Congruence Tricky

16. Subject Aux Inversion

17. Turn Taking

SuperGLUE

18. Corpus of Linguistic Acceptability CoLA (MMC)

19. Stanford Sentiment Treebank SST-2

20. Microsoft Research Paraphrase Corpus MRPC (F1)

21. Quora Question Pairs QQP (F1)

22. MultiNLI Matched MNLI

23. MultiNLI Mismatched MNLI-mm

24. Question NLI QNLI

25. Recognizing Textual Entailment RTE

26. Boolean Questions BoolQ

27. Multi-Sentence Reading Comprehension MultiRC

28. Winograd Schema Challenge WSC

MSGS

29. Main Verb (Control)

30. Control Raising (Control)

31. Syntactic Category (Control)

32. Relative Position (Control)

33. Lexical Content The (Control)

34. Main Verb Lexical Content The

35. Main Verb Relative Token Position

36. Control Raising Lexical Content The

37. Control Raising Relative Token Position

38. Syntactic Category Lexical Content The

39. Syntactic Category Relative Position

B Hyperparameters and Models Configurations

In Table 7, we report the number of trainable parameters per model. In Table 8, we report all the important hyperparameters values for all our pretraining and finetuning experiments. Also, we report the main configuration settings for all our models. Unless specified otherwise, these values were used across all models.

Model	# of trainable parameters
transformer-base	110M
structformer$_{s1}$	133M
structformer$_{s2}$	133M
structroberta$_{s1}$	133M
structroberta$_{s2}$	133M
structroberta$_{s1'}$	144M
structroberta$_{s2'}$	144M

Table 7: Number of trainable parameters per model

Training Hyperparameters	
Batch size	96
Sequence Length	128
Optimizer	AdamW
Weight Decay	0.1
Learning Rate (Linear)	1e-4
Max Steps	62K
Masking Probability	0.15
Finetuning Hyperparameters	
Initial learning rate	5e-5
Batch size	120
Maximum epochs	10
Evaluate every (steps)	400
Patience	5
Random seed	12
Models configurations	
Number of Attention Heads	12
Number of Attention Layers	12
Embeddings (Hidden) Size	768
FFN inner hidden size	3072
Attention Dropout	0.1
Front Attention Layers (where applicable)	4
Rear Attention Layers (where applicable)	8
Parser Convolution Layers (where applicable)	4
Convolution Kernel Size	9

Table 8: Pretraining and Finetuning Hyperparameters, and Models Configurations Settings