McGill BabyLM Shared Task Submission: The Effects of Data Formatting and Structural Biases

Ziling Cheng[1,2] **Rahul Aralikatte**[1,2] **Ian Porada**[1,2]
Cesare Spinoso-Di Piano[1,2] **Jackie Chi Kit Cheung**[1,2,3]
[1]Mila – Quebec Artificial Intelligence Institute [2]McGill University
[3]Canada CIFAR AI Chair
{ziling.cheng, ian.porada, cesare.spinoso-dipiano}@mail.mcgill.ca,
rahul.aralikatte@mila.quebec, jackie.cheung@mcgill.ca

Abstract

In this study, we describe our submission to the 2023 BabyLM shared-task's *strict-small* track. Our findings demonstrate the feasibility of training high-performing models within the constraints of limited data, computational resources, and time. We provide evidence that the formatting of input can significantly impact downstream performance. Furthermore, the induction of structural biases into the models through the use of part-of-speech trees yields modest benefits. Our most successful model achieves 79% on the BLiMP evaluations and 72% on the SuperGLUE evaluations. All models trained during this study can be found at https://huggingface.co/mcgill-babylm.[1][2]

1 Introduction

The pretraining of large language models (LLMs) is a resource-intensive process, requiring substantial computational power, time, and particularly, data. Contemporary LLMs are trained on billions, if not trillions, of tokens to achieve satisfactory performance (Kaplan et al., 2020; Hoffmann et al., 2022). This approach is not ideal, given that humans can learn to perform more complex tasks with data that are smaller by orders of magnitude (Linzen, 2020). Consequently, there is a burgeoning interest within the NLP community to identify and implement more data-efficient pretraining regimes.

The 2023 BabyLM challenge (Warstadt et al., 2023) seeks to unify research in this domain by formalizing the constraints and providing common pretraining and evaluation corpora. This shared task involves pretraining LLMs from scratch using data at a scale comparable to what a thirteen-year-old human child would have been exposed to. This approach enables researchers to concentrate their efforts on developing data-efficient pretraining techniques, potentially drawing inspiration from human cognitive development. The "strict-small" track, which limits the amount of pretraining data to 10M words, is of particular interest, and will be the focus of this study. An additional constraint of not being able to use any tools trained on external data further increases the difficulty. The pretraining corpus comprises multiple datasets, primarily consisting of transcribed conversations and other forms of simple language text. The evaluation of pretrained models includes zero-shot linguistic benchmark (Warstadt et al., 2020a, BLiMP) as well as finetuning the models for both standard Natural Language Understanding (NLU) tasks (Wang et al., 2019a, SuperGLUE) and evaluations of linguistic generalization (Warstadt et al., 2020c, MSGS). Brief descriptions of these datasets are provided in Section 3.

In this study, we limit our experiments to a modest computational and time budget of one GPU and 24 hours, respectively. This constraint compels us to focus on incorporating better inductive biases into model pretraining, rather than resorting to the more straightforward, but costlier, approach of extensive hyperparameter tuning. We adhere to standard transformer architectures (Vaswani et al., 2017) and pretraining strategies: masked language modeling (Devlin et al., 2019, MLM) and left-to-right, causal language modeling (Radford et al., 2018, CLM). We first explore the formatting of the data. We also attempt to induce structural biases using part-of-speech (POS) tags. While we did not include our models that incorporate POS tags in our official submission, as this contravenes the rules of the two tracks we are interested in, we believe it represents a promising research direction.

Findings Our findings indicate that within our

[1] Corresponding author: Ziling Cheng (ziling.cheng@mail.mcgill.ca).

[2] The code is available at https://github.com/ziling-cheng/babylm.

(a) (b)

Figure 1: Visualization of sentence-level training examples employing different grouping strategies, with a maximum sequence length of 32 words. The numbers on the left denote the i-th training example. (a) **sentence-level ungrouped**: any portion exceeding 32 words will be truncated. (b) **sentence-level grouped**: different documents/sentences could be grouped into a single training example, each totaling 32 words.

constrained setting, *data formatting* has the most significant impact on downstream performance. By data formatting, we specifically mean the formats of individual examples (e.g. sentence, document) and the methods of configuring multiple examples into a training minibatch (e.g. data grouping, or truncation). We observe that models pretrained with grouped data perform considerably worse than models pretrained with ungrouped data (62% vs. 79% on BLiMP). We also discover that inducing structural biases using POS trees modestly improves the downstream performance of the models ($\sim$1% on BLiMP).

2 Related Work

Existing research, particularly that conducted before the advent of LLMs, has explored the training of language models on relatively small datasets. For instance, Bengio et al. (2003) trained a neural language model on a corpus of approximately 1 million words. Additionally, Penn Treebank (Marcus et al., 1993) and WikiText (Merity et al., 2017) have been commonly used datasets for training language models.

In recent work, Samuel et al. (2023) have examined architectural enhancements to BERT when training on 100 million words, focusing on aspects such as position embeddings or layer normalization. Other studies have trained standard model architectures on limited data and evaluated syntactic or linguistic competency (Warstadt et al., 2020c; Yedetore et al., 2023; Pérez-Mayos et al., 2021). However, these studies have not thoroughly examined the data formatting and syntactic biases that we consider in our experiments.

Previous research has proposed syntactically-motivated inductive biases in the training of language models to enhance performance. These include the ON-LSTM (Shen et al., 2019), Tree Transformer (Wang et al., 2019b), and StructFormer (Shen et al., 2021). These studies have aimed to induce syntactic dependency and constituency parses.

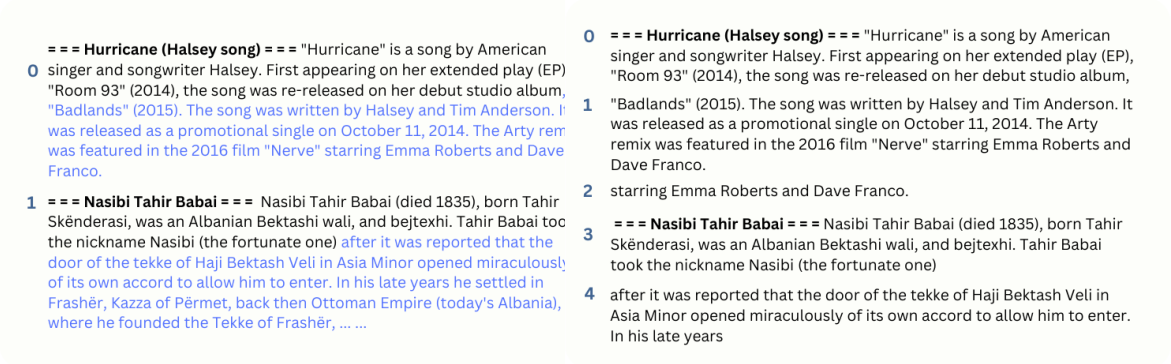
3 Data

In this section, we first introduce the pretraining and evaluation data used, we then explain how we preprocess them (Section 3.1), along with some analysis (Section 3.2).

Pretraining Corpus The pretraining corpus released by the organizers contains ten different carefully selected sub-datasets from different domains, inspired by the typical input children would receive (Warstadt et al., 2023). About 55% and 45% of the pretraining corpus is transcribed-spoken English and written English, respectively.

Evaluation Corpora The shared evaluation pipeline scores models on both syntactic evaluations and semantic (NLU) benchmarks. The benchmarks have been filtered according to the vocabulary of the STRICT-SMALL dataset such that each word in each example should appear in the training set at least twice.

Zero-shot linguistic abilities of the model is assessed mainly using the BLiMP dataset (Warstadt



(a) (b)

Figure 2: Visualization of document-level training examples employing different grouping strategies, with a maximum sequence length of 32 words. The numbers on the left denote the i-th training example. (a) **document-level ungrouped**: truncated text is shown in blue. (b) **document-level ungrouped without truncation**: document boundary is preserved. The **document-level grouping** strategy is omitted due to its similarity to sentence-level grouping strategy.

et al., 2020a). BLiMP consists of minimally different sentence pairs based on grammatical phenomenon where a model is expected to assign higher probability to the grammatically correct sentence. The sentence pairs were generated from expert-crafted grammars. This evaluation includes 12 phenomena from BLiMP, and also five supplemental phenomena not included in the original BLiMP dataset.

Fine-tuning evaluation is based on 11 canonical NLP tasks from the (Super)GLUE(Wang et al., 2018, 2019a) collections as well as MSGS (Warstadt et al., 2020c) which evaluates the extent to which a fine-tuned model favors linguistic generalizations as compared to spurious surface patterns.

3.1 Data Preprocessing

The pretraining corpus is a collection of subcorpora and was released as one file for each subcorpus. Each file consists of multiple documents from the sub-corpus which have been concatenated and then delimited by new line characters. We refer to each file as a sub-dataset, and we refer to each line in a given sub-dataset as a sentence. For each sub-dataset, in addition to training with these sentence-level examples, we also experiment with transforming the sub-dataset into "document-level" examples motivated by Liu et al. (2019) who have shown that formatting inputs as individual sentences negatively affects downstream task performance.

We thus consider two dataset formats in our experiments: *sentence-level* and *document-level*. More specifically, sentence-level refers to the case where each line in a corpus file is considered an independent training example. Document-level refers to the case where we approximately recover the original document boundaries (e.g. a chapter in the book corpus, a wikipedia article, a conversation) using heuristics and take each reconstructed document to be an independent training example. The heuristics we use to approximate document boundaries are based on corpus-specific, keyword-based rules (e.g. the keyword "Chapter" is a sign of a change of document for book corpora). In the case of speech corpora, it is impossible to reconstruct the conversation boundary because of the formatting of the released sub-dataset; therefore, for speech corpora we still consider each line (utterance) to be an independent training example, even for document-level experiments.

Both dataset variants are divided into train, validation, and test splits. We only use the train split to pretrain the models (80% of the original data).

3.2 Data Analysis

In this section, we provide some statistics of the pretraining data and BLiMP evaluation datasets, as well as some analysis of lexical overlap between the two.[3] We first compute the number of unique and total unigrams and bigrams in the data. To understand the extent of syntactic commonality between the datasets, we also examine the ratio of

[3] In this work, we mainly use BLiMP for all analyses, experimentations, and ablations, unless noted otherwise.

10M	Avg. Words			Avg. Tokens		
	train	test	dev	train	test	dev
sent.	9.41	9.74	9.16	14.41	14.99	14.28
doc.	47.22	48.10	46.76	64.33	66.13	64.66

Table 1: Average number of words (split on white space) and tokens (split using WordPiece tokenizer) in the sentences and documents of the BabyLM *strict-small* data.

the overlap of unigrams, bigrams, and linearized dependency graphs between the pretraining corpus and BLiMP data.[4] This allows us to reason about the diversity of the data and what fraction of linguistic structures present in the evaluation is seen by the model during pretraining.

Pretraining Data Table 1 shows that, on average, there are approximately 9 words per sentence and 48 words per document, in the STRICT-SMALL corpus. When measured with a WordPiece tokenizer (pretrained on the 10M data with a vocabulary size of 32,768), each sentence and document contain around 14 and 65 tokens, respectively. The training set of the pretraining corpus contains approximately 181.3K unique unigrams (words) and 2.07M unique bigrams.

BLiMP As we primarily use zero-shot BLiMP task performance to evaluate the model quality, we report the counts of unique unigrams and bigrams in the BLiMP task datasets in Table 2. The total unique vocabulary size is small: 2334 is only around 15% of the simple sum of unigrams of each task dataset, which suggests a considerable vocabulary overlap across different task datasets. Conversely, sentence structures, as characterized by dependency graphs, are remarkably diverse, with 97.6% of the BLiMP data points across tasks featuring unique linearized dependency graphs.

Lexical Overlap 98.67% of the unigrams in BLiMP are seen by models during pretraining. This is expected as the organizers filter the evaluation data based on the vocabulary in the training set. However, only 19% of the bigrams are found in the pretraining data, and more importantly, there is just 2% overlap of linearized dependency trees, suggesting that BLiMP tasks are really 'zero-shot' for BabyLM-trained models.

[4] We use SpaCy for dependency parsing and NLTK to linearize the trees.

BLiMP Phenomenon	Unigram	Bigram	Dep. G.
Anaphor Agreement	0.64K	3.52K	0.06K
Argument Structure	1.80K	19.50K	1.67K
Binding	1.07K	20.16K	2.01K
Control Raising	1.79K	14.10K	4.20K
D-N Agreement	1.17K	13.74K	0.70K
Ellipsis	1.16K	11.51K	3.28K
Filler Gap	1.44K	20.83K	8.36K
Irregular Forms	0.70K	4.29K	0.20K
Island Effects	1.01k	14.52K	3.52K
Npi Licensing	1.82K	19.20K	4.34K
Quantifiers	1.28K	10.08K	1.33K
Subject Verb Agreement	1.82K	17.60K	1.84K
Sum	15.68K	169.06K	31.52K
Total (unique)	2.33K	106.38K	30.78K

Table 2: BLiMP task statistics: number of unique and total unigrams, bigrams, and linearized dependency graphs are reported with respect to the dataset of each task. (Dep. G. stands for Dependency Graphs)

4 Methods

We experiment with two kinds of pretraining: (i) Vanilla pretraining: where we use standard processes as described in the original works that introduced the models (Liu et al., 2019; Radford et al., 2019), and where we only ablate on the way we format the input data. (ii) Structurally biased pretraining: where we induce some syntactic structure into the model either by explicitly augmenting the inputs with POS tags, or by allowing the models to implicitly induce dependency and constituency structures in an unsupervised manner (Shen et al., 2021).

4.1 Input Formatting

To efficiently utilize available compute resources during pretraining, multiple input examples can be 'grouped' together to form a bigger single example. By grouping, we mean that multiple sentences/documents are first concatenated, and then divided into training examples of maximum sequence length supported by the model. If the grouped input examples are not related to each other, the learning might be sub-optimal since the model attends to unrelated tokens. There are two ways to solve this problem: (i) do not group examples – this will require us to generally pad the examples, which brings the compute efficiency down, or (ii) build a dynamic mask such that each token only attends to other relevant tokens – this is harder to implement. We choose to continue with method (i) since the size of the pretraining data is small and the loss of efficiency is manageable, and

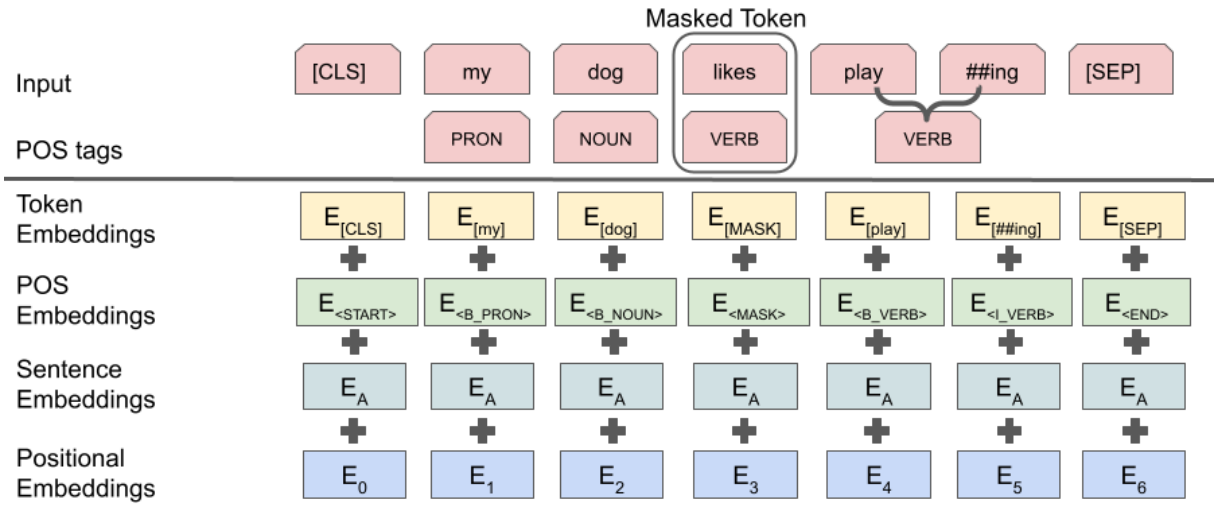


Figure 3: Part-of-speech augmentation: the input embeddings are the sum of the token embeddings, the sentence type embeddings, the positional embeddings, and the POS embeddings. In Sentence Embeddings, E_A denotes the embedding for the token type A. When a sentence B follows sentence A, the tokens in sentence B will have token type B. In Positional Embeddings, E_i refers to the absolute positional embeddings for a token at position i.

refer to this strategy as 'ungrouping'.

We ablate vanilla pretraining methods in both grouped and ungrouped setups and assess how they impact BabyLM pretraining on both sentence-level and document-level formats. Examples of different grouping strategies with sentence-level and document-level data are shown in Fig. 1 and Fig. 2, respectively. There is a general consensus that the benefits of grouping outweighs its disadvantages, but since the lengths of our pretraining data is small (because a large fraction of them are conversation data), the general consensus might not hold. Note that as the document distribution's extreme tail significantly exceeds the model's context size, we also explore an 'ungrouping without truncation' approach specifically for document-level data. This allows a single lengthy document to be divided into multiple examples without discarding extensive data, ensuring a fair comparison between different strategies. We test these strategies with three maximum sequence lengths: 32, 128, and 512.

4.2 Structurally Biased Pretraining

Part-of-speech Augmentation We first study the effect of introducing POS tags as additional inputs during pretraining. We embed POS tags of each token in the input and combine them with the token and positional embeddings to form the initial token representation, as illustrated in Fig.3. We first use NLTK's POS tagger[5] to automatically label the inputs using the universal tagset[6]. Since this tagging is done at the word-level, if a word is split into multiple subtokens by WordPiece tokenizers, we further process the label and decompose them into BIO style token-level tags.

This introduction of POS tags results in a slight change in the model architecture: a new embedding matrix for BIO style POS tags is added, and therefore the number of learnable parameters increases. During pretraining, when an input token is masked, we also mask its corresponding POS token to avoid any signal leakage.

StructFormer In contrast with the previous method, Structformer (Shen et al., 2021) allows us to induce structure implicitly. This encoder-only transformer uses dependency-constrained self-attention. This type of self-attention derives from unsupervised induction of constituency and dependency structures, allowing tokens to only attend to other tokens which are part of these structures. More concretely, it utilizes a parser network which learns to predict the syntactic distance between two tokens and the syntatic height of a token in an unsupervised manner, to generate dependency distributions. For more details, please see

[5] https://www.nltk.org/api/nltk.tag.html#nltk.tag.pos_tag

[6] NOUN (nouns), VERB (verbs), ADJ (adjectives), ADV (adverbs), PRON (pronouns), DET (determiners and articles), ADP (prepositions and postpositions), NUM (numerals), CONJ (conjunctions), PRT (particles), . (punctuation marks), X (other)

Model	BLiMP												
	Avg.	Ana. agr	Arg. str	Bind.	Ctrl. rais.	D-n agr	Ellip.	Filler gap	Irreg. form	Island effect	NPI	Quant.	S-v agr
GPT-2	**84.59**	99.70	84.00	79.00	80.00	95.90	85.10	80.90	95.80	78.30	76.50	71.90	88.00
RoBERTa	**86.03**	97.70	83.05	79.22	81.93	97.28	92.15	89.39	95.67	79.71	82.60	70.84	91.47
Human (Warstadt et al., 2020b)	**88.60**	97.00	90.00	87.30	83.90	92.20	85.00	86.90	97.00	84.90	88.10	86.60	90.90
OPT-125m (Warstadt et al., 2023)	**62.60**	63.80	70.60	67.10	66.50	78.50	62.00	63.80	67.50	48.60	46.70	59.60	56.90
RoBERTa-base (Warstadt et al., 2023)	**69.50**	81.50	67.10	67.30	67.90	90.80	76.40	63.50	87.40	39.90	55.90	70.50	65.40
T5-base (Warstadt et al., 2023)	**58.80**	68.90	63.80	60.40	60.90	72.20	34.40	48.20	77.60	45.60	47.80	61.20	65.00

Table 3: Ceiling and pre-released baseline model performance on BLiMP: the first three rows compare strong models with human performance, while the last three rows are pre-released BabyLM baselines.

BERT Base			BLiMP											
L	Format	Strategy	Avg.	Ana. agr	Arg. str	Bind.	Ctrl. rais.	D-n agr	Ellip.	Filler gap	Irreg. form	Island effect	NPI	Quant.	S-v agr
128	sent.	group	**62.57**	80.06	60.35	61.09	62.31	74.49	62.64	62.11	78.17	40.84	45.08	66.23	57.43
128	sent.	ungroup	**79.08**	94.68	74.03	72.10	73.66	94.27	77.77	78.63	89.72	59.90	74.05	74.65	85.47
128	doc.	group	**62.04**	84.00	57.52	66.84	60.30	58.68	56.76	64.61	71.96	53.21	46.25	71.02	53.37
128	doc.	ungroup	**69.38**	85.48	64.86	67.38	63.50	88.49	74.48	67.96	85.95	47.50	47.78	71.23	67.97
128	doc.	ungr. w/o trun.	**75.39**	92.28	71.02	68.83	69.02	95.01	83.89	75.16	84.78	49.66	63.36	70.09	81.52
32	sent.	group	**77.18**	92.23	72.41	70.21	70.97	94.05	84.76	74.68	91.50	53.18	72.12	67.80	82.26
32	sent.	ungroup	**78.38**	93.61	74.26	70.24	73.64	95.00	73.67	77.68	84.38	61.66	76.18	74.78	85.40
32	doc.	group	**74.90**	92.38	71.42	71.15	69.73	93.54	81.47	72.02	86.92	45.40	68.22	65.12	81.48
32	doc.	ungroup	**67.92**	76.28	63.62	64.37	64.47	88.66	71.02	68.60	83.16	44.73	55.44	64.99	69.68
32	doc.	ungr. w/o trun.	**76.75**	91.67	72.10	68.74	70.06	94.86	80.72	76.28	80.10	53.33	72.61	76.97	83.54
512	sent.	ungroup	**78.00**	94.02	73.90	72.35	72.80	94.97	76.50	77.76	87.48	57.25	71.00	73.03	84.99
512	doc.	ungr. w/o trun.	**73.04**	88.70	70.00	70.26	66.86	94.31	81.18	70.65	84.12	45.59	58.82	69.89	76.04

Table 4: Vanilla pretraining: effects of grouping strategies (Strategy), input formats (Format) and maximum sequence length (L) on BLiMP tasks using the BERT-base model. *Ungr. w/o trun.*, *sent.* and *doc.* denote ungrouped without truncation, sentence-level data, and document-level data, respectively. The metric for all tasks is accuracy.

the original paper.

5 Experimental Setup

Model Architecture Language models usually come in three flavours: encoder-only, decoder-only, and encoder-decoder architectures. Since all the BabyLM downstream tasks are classification-based, we mainly focus our experiments on encoder-only models (BERT (Devlin et al., 2018) and Structformer), whose bidirectionality is more suitable for such tasks (Devlin et al., 2018; Tay et al., 2022). We train decoder-only models (GPT-2) (Radford et al., 2018) only for data grouping experiments, and do not consider encoder-decoder models in this work. Unless otherwise stated, all experiments will use BERT-base as the standard encoder-only model, and GPT-2 as the standard decoder-only model.

Tokenizer We use the same tokenizer for both encoder-only and decoder-only models: an uncased WordPiece tokenizer (Wu et al., 2016) with a vocabulary size of 32,768 (i.e., 2^{15}), trained on *strict-small* pretraining data. For the StructFormer, we follow the original work and train a word-level tokenizer with a vocabulary size of 184,192.

Training Objective We do not make changes to the training objective of any models. We use MLM for encoder-only (including StructFormer) models with a masking rate of 15%, and next token prediction for decoder-only models.

Implementation All models are optimized with AdamW (Loshchilov and Hutter, 2017), with a peak learning rate of 1e-4, a warmup of 2000 steps, and linear decay. All models are trained with a dropout rate of 0.1, and with GELU activations (Hendrycks and Gimpel, 2016). GPT-2 and BERT models are trained with bfloat16[7] on a single NVIDIA A100-SXM4-80GB GPU, and StructFormer is trained in half precision on a single RTX 8000 GPU. All models are pretrained for 30 epochs with a maximum sequence length $L \in \{32, 128, 512\}$. Unless otherwise mentioned, the effective batch size is 128 examples for all experiments.

[7] https://cloud.google.com/tpu/docs/bfloat16

GPT-2 Base		BLiMP												
L	**Strategy**	AVG.	ANA. AGR	ARG. STR	BIND.	CTRL. RAIS.	D-N AGR	ELLIP.	FILLER GAP	IRREG. FORM	ISLAND EFFECT	NPI	QUANT.	S-V AGR
128	group	**71.30**	92.38	72.49	70.85	65.69	86.99	65.99	67.99	84.68	43.95	57.96	79.52	67.06
128	ungroup	**72.71**	94.33	74.09	69.15	67.06	91.98	60.85	70.53	89.57	46.97	63.68	68.11	76.24
512	group	**67.37**	88.80	70.64	66.90	64.60	83.02	60.85	63.34	88.70	45.48	43.74	68.57	63.83
512	ungroup	**73.18**	92.94	73.33	69.95	68.36	90.92	64.55	69.39	91.09	44.77	63.06	71.92	77.89

Table 5: Vanilla pretraining: effects of grouping strategies (Strategy) and maximum sequence length (L) on BLiMP tasks using the GPT-2 base model with sentence-level data. The metric for all tasks is accuracy.

		BLiMP												
Model	**#H**	AVG.	ANA. AGR	ARG. STR	BIND.	CTRL. RAIS.	D-N AGR	ELLIP.	FILLER GAP	IRREG. FORM	ISLAND EFFECT	NPI	QUANT.	S-V AGR
VANILLA	768	**77.29**	91.56	73.91	69.72	70.92	94.63	79.62	76.74	89.41	52.43	72.75	72.44	83.29
POS	768	**78.07**	93.76	73.33	71.36	70.97	94.11	83.08	77.23	89.82	53.44	70.97	73.57	85.19
RANDPOS	768	**73.92**	91.00	71.46	69.65	69.77	93.24	82.16	66.54	86.51	41.89	67.99	62.52	84.34
VANILLA	1152	**77.40**	93.10	74.64	70.24	71.39	95.60	78.00	78.14	88.35	52.20	70.06	72.41	84.72
POS	1152	**78.87**	93.66	74.90	68.76	71.96	95.00	84.93	77.90	89.77	55.53	73.69	74.37	86.00
RANDPOS	1152	**74.34**	91.46	71.59	70.39	70.00	94.39	80.37	66.15	86.87	41.70	69.91	64.27	84.97

Table 6: Part-of-speech augmented pre-training: effect of POS augmentation on BLiMP tasks using BERT models. VANILLA, POS and RANDPOS denote vanilla BERT model, BERT model augmented with POS tags, and BERT model augmented with random POS tags. #H denotes the hidden size of the model. Metric for all tasks is accuracy.

6 Experiments & Results

In this section, we describe the various experiments we conducted, and their results obtained from the BabyLM evaluation pipeline (Warstadt et al., 2023; Gao et al., 2021). All models are evaluated on BLiMP and the best models from each category are further evaluated on SuperGLUE and MSGS tasks.

6.1 BLiMP

To get the ceiling performance of the models, we use the publicly available checkpoints (GPT-2, RoBERTa) which are pretrained on much larger datasets. These results along with the human-level performance is shown in Table 3. We see that the performance of these two models is only 2-3 points below human performance. In addition, we include the pre-released OPT, RoBERTa and T5 baselines, which were trained on the BabyLM data in Table 3. Unlike the publicly available GPT-2 and RoBERTa models, these baselines display a substantial gap with human performance.

6.1.1 Input Formatting

To investigate the impact of grouping strategies, we pretrain the standard BERT model on a variety of combinations.

Grouping Strategy From Table 4, we see that *ungrouped* [8]models consistently outperform the *grouped* models, across all sequence lengths. We postulate that this happens due to the nature of pretraining data. Since a large fraction of the data is conversation-based, the sentence lengths are generally short, and each utterance need not always be a logical continuation of the previous ones. This might cause confusion while learning grouped data since we do not impose any attention masking to stop the model from attending to unrelated tokens. This finding is not limited to encoder-only models. We see a similar pattern in decoder-only models as well, in Table 5. Also, encoder-only models demonstrate superior zero-shot generalization on BLiMP tasks in comparison to decoder-only models. This strengthens our hypothesis in Section 5 that bidirectionality is helpful for classification tasks. Therefore, all other experiments are conducted on encoder-only models.

Truncation on Documents As expected, the performance of *ungrouped* model lags behind the *grouped* model when using document-level data. This discrepancy is primarily due to the truncation, which discards extreme tails of the document distribution. However, when truncation is disabled, the performance of the model improves by 6-13

[8] Here, on document-level data, 'ungrouped' refers to 'ungrouped without truncation' for fair comparison.

points on average. It even surpasses the *grouped* model by approximately 2 points, confirming our conclusion drawn in the preceding paragraph.

Maximum Sequence Length Despite the additional parameters introduced, extending the maximum sequence length does not yield additional performance boost. Interestingly, there seems to be a negative correlation between the two. To make this point clear, we reduce the maximum sequence length to an extremely low value of 32. We see that there is no significant drop in performance among the models. In fact, even the document-level models perform well in this setting. This is because the smaller inputs are now similar to the sentence-level inputs. We also see that the difference between the *grouped* and *ungrouped* models also reduce from 13 points to 1 point, which further shows that sentence-level inputs provide better performance for BabyLM pretraining.

In summary, we see that the sentence-level *ungrouped* model with a sequence length of 128 performs the best with an average BLiMP score of 79.08. This is around 10 points higher than the pre-released Roberta-base baseline. Furthermore, this is only 5-6 points behind the ceiling performance of GPT-2 and RoBERTa-base models trained on much larger datasets. However it is difficult to conclude that these models learn efficiently since we have not yet evaluated them on semantic downstream tasks which require the models to capture long-range dependencies. But we can safely say that 10M words and sentence-level training is enough for models to learn simple linguistic phenomena as tested by BLiMP. Henceforth, we will perform subsequent experiments using only the most effective configurations identified, i.e., sentence-level *ungrouped* models.

6.1.2 POS Augmentation

To test whether explicitly inducing POS tree structures during pretraining improves downstream performance, we embed POS tags and add them to the input representations. To make sure that any improvement is not only due to the increase in the number of parameters,[9] we run two ablations with an effective batch size of 512:[10] (i) randomly shuffle the POS tags of a sentence before adding them to the input – this will make sure that the model gets no signal from the POS tags, and (ii) increase the hidden size of the models – this will contain signals from POS and further increase the number of learnable parameters.

Table 6 illustrates that the encoder-only models, when augmented with POS tags, exhibit a marginal performance improvement of approximately one point compared to the vanilla models, regardless of the hidden size. However, models with shuffled POS tags lag behind by approximately 4-5 points, suggesting that it is indeed beneficial to induce structures during pretraining. Next, we see that boosting the hidden size enhances model performance across all settings. However, on closer inspection we see that the benefit to the standard model is minor, $\sim$0.1 points. This is surprising since the number of learnable parameters in the model with the expanded hidden size is almost an order of magnitude larger than the model with just the additional POS embedding matrix. This result further underscores the benefits of inducing POS tree structures into the pretraining process.

6.1.3 StructFormer

All StructFormer experiments are all conducted using sentence-level ungrouped data, with a maximum sequence length of 512.

Though StructFormer outperforms vanilla BERT when the models are scaled down (Tiny), it fails to do so on larger model sizes. In fact, we see in Table 7 that BERT-Mini outperforms a StructFormer-Base model. We hypothesize: (i) that 10M words are not enough to learn good representations of 180K words present in the StructFormer vocabulary, and (ii) that 10M words are not big enough to fully train the unsupervised parsing network which in-turn affects the downstream performance. This model undertraining is evident from the fact that BERT performance jumps up by 14 points when its size is increased from Tiny to Base, whereas StructFormer's performance only increases by 5 points.

6.2 Other Evaluations

We select the best performing models of each setting mentioned in the previous section and perform a full evaluation on SuperGLUE, BLiMP

[9] This model has an additional embedding matrix for POS tags.

[10] We increase the batch size to improve the runtime of the experiments. But this causes slight discrepancies in the result of our vanilla models between Tables 4 and 6

Model	Size	BLiMP												
		Avg.	Ana. agr	Arg. str	Bind.	Ctrl. rais.	D-n agr	Ellip.	Filler gap	Irreg. form	Island effect	NPI	Quant.	S-v agr
BERT	tiny	**63.92**	73.52	64.05	64.16	62.84	80.35	53.29	62.18	91.96	42.53	49.42	63.14	59.62
StructFormer	tiny	**64.89**	68.51	63.07	61.61	63.52	81.42	48.38	63.26	88.80	51.64	53.31	72.62	62.53
BERT	base	**78.00**	94.02	73.90	72.35	72.80	94.97	76.50	77.76	87.48	57.25	71.00	73.03	84.99
StructFormer	base	**69.79**	79.09	67.01	67.81	67.26	92.50	62.64	64.47	89.72	46.11	57.41	78.80	64.66
BERT	mini	**70.27**	88.09	69.63	68.76	65.20	91.49	74.65	68.07	92.67	34.87	56.18	67.31	66.29

Table 7: StructFormer: comparison between BERT and StructFormer architectures on tiny and base sizes. Metric for all tasks is accuracy.

Model	L	Format	Strategy	Dynabench	BLiMP	BLiMP Suppl.	SuperGLUE	MSGS
BERT	128	sent.	ungroup	**69**	79.08	58.19	72.37	81.51
BERT	128	doc.	ungr. w/o trun.	**68**	75.39	61.33	72.28	81.00
BERT-POS	512	sent.	ungroup	**68**	79.67	56.81	71.85	79.64
GPT-2	512	sent.	ungroup	**67**	73.18	55.47	69.23	82.14

Table 8: Results of other benchmarks for the top-performing models evaluated by BLiMP tasks: the score for each benchmark is reported as an average, detailed scores are in Appendix. Dynabench score aggregates all benchmarks and is provided by the model submission platform. *Ungr. w/o trun.*, *sent.* and *doc.* denote ungrouped without truncation, sentence-level data and doc. denotes document-level data, respectively. Metric for all tasks is accuracy.

supplement, and MSGS benchmarks[11]. In Table 8, we report average scores for each benchmark, and the final score computed by the model submission platform (Kiela et al., 2021, Dynabench), for each model. Detailed performance of each task for all benchmarks, as well as the pre-released baselines is given in Table 9, 10, 11, 12, 13, 14, and 15 in Appendix A.

We see in Table 8, which summarizes the results and provides the aggregated scores, that only slight differences exist among the models. The BERT-Base model, trained on sentence-level data, demonstrates superior performance overall, surpassing other models by 1-2 points, consistent with our BLiMP evaluations. Remarkably, the model trained with document-level inputs displays a substantial superiority in BLiMP supplement tasks, achieving a lead of nearly 3 points over models trained on sentences. Surprisingly, the GPT-2 model, despite underperforming in all other tasks, exhibits a robust performance on the MSGS tasks. The BERT model augmented with POS trees, despite its best performance on BLiMP tasks, fails to replicate the success across other benchmarks which suggests that it might have learned some specific structural patterns helpful only in certain cases as pointed out in Warstadt et al. (2020c).

[11] The last two benchmarks were released towards the end of the shared task.

7 Conclusion

In this work, we investigate the effects of data formatting and the induction of structural biases in data-efficient pretraining settings. These experiments were performed under the constrains of limited data, computational resources, and training time. Our findings indicate data grouping is the most significant factor affecting downstream performance due to the nature of the pretraining data. We also see that when the best data format considered is employed, inducing structural biases into the models enhances their downstream performance on BLiMP performance by approximately 1%.

Acknowledgements

We would like to thank the anonymous reviewers for their comments and suggestions. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), and in part by the McGill's Science Undergraduate Research Awards (SURA 2023) during Summer 2023. The authors acknowledge the material support of NVIDIA in the form of computational resources.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural proba-

bilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2021. A framework for few-shot language model evaluation.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack William Rae, and Laurent Sifre. 2022. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*.

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.

Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. 2021. Dynabench: Rethinking benchmarking in NLP. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4110–4124, Online. Association for Computational Linguistics.

Tal Linzen. 2020. How can we accelerate progress towards human-like linguistic generalization? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5210–5217, Online. 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. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations*.

Laura Pérez-Mayos, Miguel Ballesteros, and Leo Wanner. 2021. How much pretraining data do language models need to learn syntax? In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1571–1582, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

David Samuel, Andrey Kutuzov, Lilja Øvrelid, and Erik Velldal. 2023. Trained on 100 million words and still in shape: BERT meets British National Corpus. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1954–1974, Dubrovnik, Croatia. Association for Computational Linguistics.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. 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.

Yi Tay, Mostafa Dehghani, Vinh Q Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Dara Bahri, Tal Schuster, Steven Zheng, et al. 2022. Ul2: Unifying language learning paradigms. In *The Eleventh International Conference on Learning Representations*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Yaushian Wang, Hung-Yi Lee, and Yun-Nung Chen. 2019b. Tree transformer: Integrating tree structures into self-attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1061–1070, Hong Kong, China. Association for Computational Linguistics.

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, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R Bowman. 2020b. 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. 2020c. 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.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Aditya Yedetore, Tal Linzen, Robert Frank, and R. Thomas McCoy. 2023. How poor is the stimulus? evaluating hierarchical generalization in neural networks trained on child-directed speech. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9370–9393, Toronto, Canada. Association for Computational Linguistics.

A Appendix

In this section, we provide detailed performance of the top-performing models mentioned in Section 6.2 on each evaluation benchmarks. BLiMP, BLiMP supplement, SuperGLUE, and MSGS results are in Table 9, 11, 13 , and 15, respectively. Pre-released baseline performance for BLiMP supplement, SuperGLUE, and MSGS tasks are in Table 10, 12 , and 14, respectively.

				BLiMP												
Model	**L**	**Format**	**Strategy**	AVG.	ANA. AGR	ARG. STR	BIND.	CTRL. RAIS.	D-N AGR	ELLIP.	FILLER GAP	IRREG. FORM	ISLAND EFFECT	NPI	QUANT.	S-V AGR
BERT	128	sent.	ungroup	**79.08**	94.68	74.03	72.1	73.66	94.27	77.77	78.63	89.72	59.9	74.05	74.65	85.47
BERT	128	doc.	ungr. w/o trun.	**75.39**	92.28	71.02	68.83	69.02	95.01	83.89	75.16	84.78	49.66	63.36	70.09	81.52
BERT-POS	512	ungroup	sent.	**79.67**	94.73	75.36	72.32	73.95	96.15	82.04	78.51	89.21	59.57	71.50	74.57	88.06
GPT	512	sent.	ungroup	**73.18**	92.94	73.33	69.95	68.36	90.92	64.55	69.39	91.09	44.77	63.06	71.92	77.89

Table 9: BLiMP results: *Ungr. w/o trun.*, *sent.* and *doc.* denote ungrouped without truncation, sentence-level data and doc. denotes document-level data, respectively. Metric for all tasks is accuracy.

	BLiMP Supplement					
MODEL	AVG.	HYPERNYM	QA CONGR. (EASY)	QA CONGR. (TRICKY)	SUBJ.-AUX. INVERSION	TURN TAKING
OPT-125M (Warstadt et al., 2023)	54.72	50.00	54.70	31.50	80.30	57.10
ROBERTA-BASE (Warstadt et al., 2023)	47.54	49.40	31.30	32.10	71.70	53.20
T5-BASE (Warstadt et al., 2023)	43.94	48.00	40.60	21.20	64.90	45.00

Table 10: BLiMP supplement pre-released baseline results: Metric for all tasks is accuracy.

				BLiMP Supplement					
Model	**L**	**Format**	**Strategy**	AVG.	HYPERNYM	QA CONGR. (EASY)	QA CONGR. (TRICKY)	SUBJ.-AUX. INVERSION	TURN TAKING
BERT	128	sent.	ungroup	**58.19**	49.07	70.31	29.70	79.39	62.50
BERT	128	doc.	ungr. w/o trun.	**61.33**	50.23	73.44	36.36	77.70	68.93
BERT-POS	512	ungroup	sent.	**56.81**	49.42	64.06	29.09	80.41	61.07
GPT	512	sent.	ungroup	**55.47**	50	53.12	29.7	85.95	58.57

Table 11: BLiMP supplement results: *Ungr. w/o trun.*, *sent.* and *doc.* denote ungrouped without truncation, sentence-level data and doc. denotes document-level data, respectively. Metric for all tasks is accuracy.

	SuperGLUE											
Model	AVG.	COLA (MCC)	SST-2	MRPC (F1)	QQP (F1)	MNLI	MNLI-MM	QNLI	RTE	BOOLQ	MULTIRC	WSC
Majority label (Warstadt et al., 2023)	46.3	0.0	50.2	82.0	53.1	35.7	35.7	35.4	53.1	50.5	59.9	53.2
OPT-125m (Warstadt et al., 2023)	58.9	15.2	81.9	72.5	60.4	57.6	60.0	61.5	60.0	63.3	55.2	60.2
RoBERTa-base (Warstadt et al., 2023)	67.3	25.8	87.0	79.2	73.7	73.2	74.0	77.0	61.6	66.3	61.4	61.4
T5-base (Warstadt et al., 2023)	56.4	11.3	78.1	80.5	66.2	48.0	50.3	62.0	49.4	66.0	47.1	61.4

Table 12: GLUE pre-released baseline results: Metric for all tasks unless otherwise stated.

				SuperGLUE											
Model	**L**	**Format**	**Strategy**	AVG.	BOOLQ	COLA	MNLI	MNLI-MM	MRPC (F1)	MULTIRC	QNLI	QQP (F1)	RTE	SST-2	WSC
BERT	128	sent.	ungroup	**72.37**	66.39	74.78	74.15	74.79	80.29	63.20	78.74	81.79	51.52	88.98	61.45
BERT	128	doc.	ungr. w/o trun.	**72.28**	66.11	72.33	75.36	76.29	77.78	59.58	82.50	84.13	51.52	87.99	61.45
BERT-POS	512	sent.	ungroup	**71.85**	67.22	75.76	73.80	75.03	77.22	60.24	78.70	83.10	49.49	88.39	61.45
GPT	512	sent.	ungroup	**69.23**	64.87	71.44	72.14	72.69	71.84	62.43	64.22	81.71	50.51	88.19	61.45

Table 13: GLUE results: *Ungr. w/o trun.*, *sent.* and *doc.* denote ungrouped without truncation, sentence-level data and doc. denotes document-level data, respectively. Metric for all tasks except MRPC and QQP is accuracy.

	MSGS											
Model	AVG.	CR CTRL.	LC CTRL.	MV CTRL.	RP CTRL.	SC CTRL.	CR LC	CR RTP	MV LC	MV RTP	SC LC	SC RP
OPT-125m (Warstadt et al., 2023)	80.9	97.2	82.6	100.0	99.8	88.1	75.3	67.1	66.3	66.8	84.8	62.0
RoBERTa-base (Warstadt et al., 2023)	81.7	93.0	100.0	100.0	100.0	89.0	68.3	66.8	66.6	80.2	67.4	67.4
T5-base (Warstadt et al., 2023)	82.3	95.1	100.0	100.0	99.8	88.7	76.7	69.4	67.0	67.7	72.7	68.0

Table 14: MSGS pre-released baseline results: Metric for all tasks is accuracy.

				MSGS											
Model	**L**	**Format**	**Strategy**	AVG.	CR CTRL.	LC CTRL.	MV CTRL.	RP CTRL.	SC CTRL.	CR LC	CR RTP	MV LC	MV RTP	SC LC	SC RP
BERT	128	sent.	ungroup	**81.51**	96.30	100.00	99.94	100.00	83.47	72.67	72.52	66.61	68.65	68.32	68.08
BERT	128	doc.	ungr. w/o trun.	**81.00**	92.32	100.00	99.89	98.26	95.59	67.17	67.14	66.61	68.04	69.73	66.22
BERT-POS	512	sent.	ungroup	**79.64**	91.08	100.00	99.87	99.99	89.70	71.79	67.05	66.77	68.80	63.30	57.64
GPT	512	sent.	ungroup	**82.14**	92.11	100.00	99.94	100.00	95.51	70.23	69.86	66.61	67.78	75.62	65.91

Table 15: MSGS results: *Ungr. w/o trun.*, *sent.* and *doc.* denote ungrouped without truncation, sentence-level data and doc. denotes document-level data, respectively. Metric for all tasks is accuracy.