Towards more Human-like Language Models based on Contextualizer Pretraining Strategy

Chenghao Xiao G Thomas Hudson Noura Al Moubayed Department of Computer Science Durham University

Abstract

Taking inspiration from human children learning, we pose a question: can a "baby language model" gradually internalize a concept by exposing itself to the concept in unlimited, oftentimes irrelevant contexts, and what this means to limited pretraining resource (both data-wise and GPU-wise).

Throughout the study, we restrict our experiments to two data-limited settings, 10M and 100M tokens, which are respectively 1/3000 and 1/300 to what were available to the training of RoBERTa. Our best performing training recipe performs within 1.2% of RoBERTa, and on-par with BERT, on the BLiMP zero-shot linguistic knowledge benchmark, using 1/300 RoBERTa's pretraining data and can be trained on only 1 GPU in 4 days, trained for only 1 epoch.

1 Introduction

In recent years, the success of pretrained language models has relied on scaling up both parameter counts and the size of the datasets that models are exposed to, in order to improve performance. According to (Warstadt et al., 2023), the number of words that the modestly-sized language model, Chinchilla (Hoffmann et al., 2022) goes through (1.4 trillion words), is equivalent to over 10000 words for every one word a 13-year-old child has heard in their entire life.

In this work, we take a contextualization perspective to rethink, why can a human child build up their understanding of the world with an exposure to merely 2M-7M words per year (Gilkerson et al., 2017), and without largely changing the current pretraining techniques, how can we facilitate the learning of a language model to imitate such behaviors to the greatest extent.

With many nuanced experimental findings, our main findings can be summarized analogically as one trick:

"Learning to solve math problems in a history class".

Metaphorically, exposing a language model to datasets of different domains is like sending a kid to a kindergarden that teaches classes of diverse content. Taking the learning of math as an example, a child does not only do math in a math class, nor is their math capability only aroused when they see a math test paper. They do math in a math class, at home, and during playing time. If a child is good at math in a math class, they theoretically should be able to demonstrate their math abilities any time when presented with a real-life scenario that requires these skills.

We argue that such ability should also apply to language models, and find that, exposing a language model to knowledges of a domain surrounded by knowledges of that same domain, poses a **"contextualization trap"**. This induces overfitting to contexts, over-attendance to spuriously relevant tokens, and thus under-exploitation of semantics signals in the limited data available.

In fact, if a language model can recover masked Wikipedia texts surrounded by Wikipedia texts, it should be no worse at recovering them when it is "watching" cartoons.

We find that, designing training recipes solely based on this inspiration largely improves pretraining performance, enabling a baby language model to achieve similar performance to RoBERTa on zero-shot linguistic knowledge tasks, and competitive SuperGLUE performance, with less than 1/300 of its pretraining data.

2 Method: Contextualizer

To exploit the limited data available, we propose **Contextualizer**, a framework to create more (theoretically unlimited) contexts that a fixed input is surrounded by.



Figure 1: Concept of Contextualizer. Assume we build a training set with four datasets of different domains (say math, history, politics, and physics), each with one input. **Clean Padding** splits every input to different chunks, then pads the end of each input with padding tokens, and does not allow mixing components of different inputs to a same chunk. **Noisy Padding**, on the other hand, allows different inputs to be padded to the same chunk. **Context-fixed Padding** simply takes the chunks padded by Noisy Padding and does a round of shuffling. **Context-augmented Padding** shuffles the original inputs every time and re-pads them, allowing original inputs to be truncated at diverse positions and joint with different contexts. In our best training configuration, we do (4b) 40 times.

2.1 Recipes

We have designed two recipes to facilitate Contextualizer. 1) Context^{clean}. Aligning with the process of children learning, one would expect that teaching a concept to a baby for the first time requires exposing them to the concept in a "clean context", i.e., the context that the concept's supposed to be in. 2) Context^{noisy}. After a model/baby has attained certain level of knowledge, we consider augmenting this knowledge by practicing the knowledge in different contexts. As a further intuitive example, after a baby has remembered a quote from some cartoon character, they can repeat this sentence in a standalone manner in any context, and this does not require locating this sentence in a context with (max sequence length - quote length) of relevant cartoon dialogues any more.

Apart from intuitions from children learning, the spirit of the recipes has also seen its empirical ground in previous research. For one, research on shortcut learning (Geirhos et al., 2020) has attributed vulnerability of a model's prediction partly to its overfitting towards spurious correlation. Taking Tweet toxicity classification as an example, a model can easily learn its over-reliance on @ as an indicator for toxicity, because of the frequent appearance of @user in toxic tweets. Such vulnerability has hindered a real understanding towards the semantics of a large amount of tokens. Our contextualization method has largely removed learning shortcuts, by truncating complete inputs at diverse positions, making tokens in the input unseen from one another from time to time, while co-occurring with unlimited contexts from other inputs (be them relevant or irrelevant). This improves the model's robustness against irrelevant noise, while training its intra-input attentions to be activated by real relevant tokens.

2.2 Implementation Details

As discussed, we process data into Context^{clean} and Context^{noisy} on a high level. However, under the category of Context^{noisy}, we have designed three settings, namely, Noisy Padding, followed by either Context-fixed Padding and Contextaugmented Padding (Figure 1). As we will show in later experiments, these techniques show large behavioral and performance gap. **Context**^{clean} The concept of Context^{clean} is very straightforward and only facilitated by one setting: Clean Padding (Figure 1). Taking the original inputs, **Clean Padding** truncates and allocates each original input to different chunks, and extends each input with [pad] tokens, if the last portion of the input is shorter than the max chunk length by itself.

Context^{noisy} Context^{noisy}, on the other hand, is facilitated by three settings. As opposed to Clean Padding, techniques in Context^{noisy} allow text from different original inputs to appear in a same chunk. 1) "Noisy Padding: First Portion" is used to create the first portion of noisy training set, in a dataset curriculum order (will be discussed later). In this setting, datasets are first concatenated in a pre-defined easy-to-difficult order. At the end of each input, the next input will follow immediately, instead of starting a new chunk. 2) Later portions of the training set are created by two options: 2a) Context-fixed Padding directly takes the first portion created by noisy padding, and shuffles them on a chunk level. This will only enable different chunks to appear in different batches in later training, but will not re-pad different contexts in a same chunk. In other words, content in a chunk always stays the same, but is just shuffled to different indexes in the training set portion. 2b) **Context-augmented Padding** is the most noisy setting (and most beneficial, as will be shown). At each operation, it shuffles the original input order again, and conducts Noisy Padding. In other words, Context-augmented Padding is in essence Noisy Padding without the dataset curriculum scheme. Using Context-augmented Padding, we can theoretically create n! training data examples by exhausting the order permutation of the original inputs, where n is the number of original inputs. As we will show, this technique leads to the most performance gain, by allowing the model/baby to revisit the same knowledge in many contexts, with different amounts of clean context available, and to develop different 'perspectives' to understand the same knowledge.

Datasets Our 10M and 100M datasets come from the two tracks of the BabyLM Challenge (Warstadt et al., 2023), including data sampled from CHILDES, Switchboard, OpenSubtitles, BNC, QED, CBT, Children Stories, Gutenberg, Simple Wikipedia, and Wikipedia, covering children speech, transcribed text, children stories, and Wikipedia data. In order to further imitate human learning, we apply a rough dataset-level curriculum for clean padding and the first portion of noisy padding, by manually arranging the order of importing the 10 datasets.

Notably, we did not apply any annotations or ordering to input-level data, but only arranged the training set at the dataset level following commonsense understanding (such as children speech datasets at the beginning, followed by children stories, and lastly Wikipedia datasets), further confirmed by manual inspection of linguistic statistics.

We leverage the TCT toolkit (Simig et al., 2022) to generate these statistics. For every dataset, we first compute the statistics on sentence level-inputs, and then average all outputs on the dataset level. As a further note, sentence-level statistics are just used to compute dataset-level statistics to roughly confirm our dataset order, and **none of these statistics in any form**. Also, there is no evidence that applying dataset-level curriculum in the first portion is useful to our method, but we would like to provide a starting point for future studies to combine our method with more human-like data orders.

Table 1 presents statistics of four of the representative properties: Age of Acquisition (Mean), Age of Acquisition (Max), Flesch Reading Ease, and Flesch-Kincaid Grade Level. We also ran computations on Word per Sentence, Average Word Length - syllables, Average Word Length - letters, type-token ratio computed over all words, lexical diversity, mean meaningfulness, etc. We find that, albeit we cannot find a dataset order that makes all linguistic statistics monotonically decrease or increase, statistics computed on all linguistic properties display a strong correlation. These statistics also align with one's common sense.

Following these inspections, the final data order is determined to be: CHILDES, Switchboard, OpenSubtitles, BNC, QED, CBT, Children Stories, Gutenberg, Simple Wikipedia, and Wikipedia. For Context^{clean} and the first portion of Context^{noisy}, all datasets are concatenated in this order before processing. They are then shuffled before creating later portions of Context^{noisy}.

2.3 Other technical Details

Tokenizer For 10M and 100M settings, we train separate BPE tokenizers (Sennrich et al., 2016)

	CHILDES	Switchboard	OpenSub.	BNC	QED	СВТ	Child. Stories	Gutenberg	Sim. Wiki.	Wiki.
Age of Acquisition (Mean)	4.38	4.69	4.76	4.72	5.01	4.89	4.84	5.49	5.84	5.79
Age of Acquisition (Max)	5.43	6.80	6.66	6.94	7.88	8.62	9.58	9.38	9.99	11.37
Flesch Reading Ease	105.41	101.19	94.83	96.88	85.61	84.51	83.00	79.07	58.51	62.68
Flesch-Kincaid Grade Level	-0.28	1.03	1.53	2.15	3.86	6.19	6.80	4.35	7.67	9.30

Table 1: Dataset-level statistics of selected linguistic features, computed with TCT toolkit (Simig et al., 2022). These statistics align well with common sense, and confirm our manual dataset order.

from scratch with a fixed vocabulary size of 50k, in line with the original RoBERTa. We find that a vocabulary size of 10k and 30k degrades the performance of most BLiMP tasks (except on Irregular Forms, noticeably) in inital experiments with 10M datasets.

Arch./Size/Init. We use the architecture and parameter size of RoBERTa-base (Liu et al., 2019), and initialize the models with random weights.

Training Cost We fix the computation cost for models under the same track to be roughly the same. For 10M track, every model takes around 6-8 hours on a single RTX 3090; and for 100M track, every model takes around 3-4 days. The only factor that brings this around 10% - 20% training time difference is whether we add a round or two of Context^{clean} before or after the training with Context^{noisy}. We will explain how we decide the computation cost in experiment setting section.

Chunk Length For 10M track, we set the max sequence length of each padded input to be 64 (i.e., max length of input chunks), and for 100M track, we set it to 128. We find that a max chunk length of 128 degrades the performance of models trained on 10M corpus on BLiMP tasks. Notably, in initial tokenization before post-processing with Contextualizer, we do not impose any max sequence length, and keep every token available before context augmentation padding with Contextualizer, i.e., the "max sequence length" only applies to padding complete inputs to chunks.

We hypothesize that there exists a training stability-oriented scaling law between corpus size and max sequence length to be padded to, due to the difficulty of learning robust long-range dependencies with limited amount of training examples.

Training Objectives We stick to MLM objective with 15% masking rate, and use dynamic masking (Liu et al., 2019). For all stratigies we use, we conduct random masking on the tokenized inputs on the fly (in training loop instead of before).

Interestingly, in initial experiments, we find that using reconstruction loss instead of MLM loss improves performance of checkpoints in early phase of training, and the isotropy of the embeddings encoded (Xiao et al., 2023) (also better zero-shot performance on sts-b). However, the gap could be bridged in later training. We leave further exploration of this phenomenon for future work.

We have also tried combining mlm loss and unsupervised contrastive loss (Gao et al., 2021b), and find unstable improvements (better on tasks related to representation - such as QQP and NLI tasks, and opposite otherwise). We have also tried masking rate curriculum and contrastive loss weighting curriculum, and find unstable improvements as well.

Therefore, we decide to only use a MLM loss with a static masking rate to focus on the study of contextualization.

Other Dataset Pre-processing We include a few extra pre-processing steps for all experiments. For Context^{clean}, we filter all original tokenized inputs that have only 2 tokens ([cls] and [sep] tokens) to make sure that the processed chunks later are not empty strings with only [cls], [sep] and the rest being all [pad] tokens. For Context^{noisy}, we only keep original tokenized inputs with at least 5 tokens before conducting noisy padding. This is because inputs with too few tokens are not self-contained. For instance, predicting "[cls] Hello! [sep]" with "hello" masked would only provide signals for the model's prediction to converge to token frequencybased probability distribution of the corpus (Chang and Bergen, 2022), and it is not useful for our noisycontext strategy. In terms of the datasets, we find that the Gutenberg dataset provided officially by BabyLM contains nextline splits in every paragraph once each line reaches certain length, and it is not ideal - because after tokenization, this would give unwanted [sep] tokens within a complete sentence that is not supposed to be split. Thus, we remove all nextline splits within same paragraphs. We find performance gains in initial experiments for all pre-

$\begin{array}{l} \text{Task} \rightarrow \\ \text{Model} \downarrow \end{array}$	Anaph. Agr.	Agr. Struct.	Bndg.	Ctrl./ Raise.	D-N Agr.	Ell.	F-G.	Irreg. Forms	Island Effects	NPI Lic.	Qnts.	S-V Agr.	Main Avg.
Baseline	89.50	71.30	71.00	67.10	93.10	83.80	68.00	89.60	54.50	66.30	70.30	76.20	75.06
1-40 n (ours)	96.01	78.84	76.68	74.52	96.45	91.97	76.13	90.48	70.67	71.99	65.20	83.41	81.03
40-1 n (ours)	97.49*	79.58	79.98*	78.26	96.80	92.73**	83.94*	94.50*	78.18	81.22	73.31**	90.35	<u>85.53</u>
40-1 cnc (ours)	97.55*	80.15*	<u>77.06</u>	80.11	<u>96.57</u>	92.26**	84.97*	<u>90.53</u>	80.12**	83.71*	73.80**	<u>89.63</u>	85.54
BERT	97.03	79.62	81.23	81.02	96.83	89.03	81.85	94.30	79.56	84.97	69.91	91.80	85.60
RoBERTa	97.70	83.04	79.21	81.90	97.28	92.15	89.39	95.67	79.67	82.58	70.40	91.47	86.70

Table 2: BLiMP Results of 100M recipes. **Bold Numbers** represent the best performance among our training recipes. <u>Underlined Numbers</u> represent second best. * denotes that the performance outperforms either BERT or RoBERTa. ** denotes that the performance outperforms both BERT and RoBERTa. Notably, our performances on Ellipsis, Island Effects, and Quantifiers outperform both BERT and RoBERTa, using respectively under 1/40 and 1/300 of their training data.

processing steps stated above.

Experiment Settings We conduct experiments with combinations of the above described Contextualizer data processing settings.

As stated, we fix the computation cost of experiments in the same track to be roughly the same. The exact cost is decided to align with the epoch number used in RoBERTa. We calculated that RoBERTa was roughly trained for 40 epochs on their training set. Therefore, for the noisy training set created by Context-fixed Padding, we train the model for 40 epochs (in result tables, we call this setting "1-40"). Then to align with this computation cost for context-augmented experiments, we perform Context-augmented Padding for 39 times on top of Noisy Padding first portion, creating a noisy training set 40 times larger than the contextfixed training set, and train it for only 1 epoch (we refer to this as "40-1" in result tables). Furthermore, we consider adding a round or two of Clean Padding data before or after the noisy data. This typically brings around 10% to 20% computation cost difference, since clean padding data has more examples (For instance, if we have 10 original inputs, each with 10 tokens, they could fit in one single chunk using Noisy Padding under a max chunk length of 128, but would create 10 chunks, using Clean Padding).

Concretely, "1-40 n" in result tables means: we train the model only on 1 portion of noisy data for 40 epochs. This is achieved by doing Noisy Padding to create one portion of data, and just shuf-fle this portion in the rest of the 39 epochs (essentially 39 times of "context-fixed padding"). On the other hand, "40-1 n" means that we create the first portion of data, and create 39 more portions with context-augmented padding, training on this 40-times larger training set for 1 epoch. "c" in the

result tables denotes the number of clean data concatenated before and after noisy data. For instance, "1-40 ccn" denotes first training on clean data twice, then 1 portion of noisy data for 40 epochs.

3 Results

We evaluate our models on BLiMP, SuperGLUE and MSGS tasks (Warstadt et al., 2020a; Wang et al., 2019; Warstadt et al., 2020b; Gao et al., 2021a). Notably, we use the versions processed by BabyLM, where each word has appeared in the 10M training set at least twice.

3.1 BLiMP Results

100M Track For the 100M track (Table 2), we can clearly see the benefits brought by Contextaugmented Padding (40-1 n), outperforming its Context-fixed Padding counterpart (1-40 n) by a large margin on the zero-shot BLiMP benchmark, and outperforming BabyLM official baseline for over 10 absolute percentage points. The model trained with Context-augmented Padding outperforms Context-fixed Padding on all BLiMP tasks, showing no trade-offs in introducing more noise from mixing contexts in the same inputs in the 100M setting.

Adding a round of clean data before and after noisy data (40-1 cnc) improves tasks like Agreement Structure, Control/Raising, Island Effects, and NPI Licensing, but degrades the model's performance largely on Irregular Forms, leading to only a small gain on average performance of all tasks. We hypothesize that there might exist a better data shuffling strategy when combining noisy and clean data, such as doing another round of training setlevel shuffling after concatenating clean and noisy data. We leave this for future work.

Notably, our best performing models are on-

Model	CoLA	SST-2	MRPC	QQP	MNLI	MNLI-mm	QNLI	RTE	BoolQ	MuiltiRC	WSC	Main Avg.
					Str	ict-Small (101	M Track)					
Baseline	25.80	87.00	79.20	73.70	73.20	74.00	77.00	61.60	66.30	61.40	61.40	67.33
1-40 cnc (Ours)	38.70	89.76	79.55	84.19	73.61	74.89	83.01	52.53	66.39	61.23	61.45	69.58
					5	Strict (100M	Frack)					
Baseline	45.30	88.60	80.50	78.50	68.70	78.00	82.30	51.50	59.90	61.30	61.40	68.73
40-1 cnc (Ours)	56.09	90.55	83.74	85.63	77.92	78.36	83.60	53.54	68.46	64.40	59.04	72.85
Model	CR	LC	MV	RP	SC	CR-LC	CR-RTP	MV-LC	MV-RTP	SC-LC	SC-RP	Main Avg.
					Str	ict-Small (10)	M Track)					
Baseline	43.10	100.0	97.70	76.70	86.20	-28.30	-77.70	-99.30	-79.40	16.30	-45.00	8.21
1-40 cnc (Ours)	75.68	100.00	99.93	99.96	85.67	-46.28	-89.12	-100.00	-62.37	13.40	-37.24	12.69
					1	Strict (100M	Frack)					
Baseline	74.7	100.0	99.9	100.0	59.2	-89.0	-91.2	-99.8	-15.3	-57.7	-39.2	3.78
40-1 cnc (Ours)	96.48	100.00	100.00	100.00	96.68	88.03	71.76	-32.02	30.91	21.97	-35.93	57.99

Table 3: SuperGLUE and MSGS results for 10M track and 100M track, comparing our selected models and baselines. Except CoLA (MCC), MRPC (F1) and QQP (F1), all other scores are Accuracy.

par with BERT, and is within a 1.2% gap with RoBERTa, using 1/40 and 1/300 of their training data respectively. This validates that, using Context-augmented Padding, we can create more pseudo-data that behaves closely like the real data. In our case, by running Context-augmented Padding 39 times, we actually create a 4B-token dataset using the 100M-token dataset. This is onpar with the training set with BERT, and actually gives us a model that behaves on-par with BERT on BLiMP. Given enough compute resource, we would expect running the augmentation 299 times would give us a model that performs more on-par with RoBERTa.

10M Track We find that, the best strategy for 10M deviates from the best strategy for 100M. We suggest this is because Context-Augmented padding dilutes the impact of informative datasets such as Wikipedia with noise (children mumbling, onomatopoeia data) from datasets such as CHILDES. Therefore, the decision for the optimal 10M strategy has been more difficult and nuanced. We leave the full 10M BLiMP results of 7 strategies that we have explored in Appendix A, and only present the SuperGLUE and MSGS results for one representative strategy (1-40 cnc, created by Context-fixed Padding) in the next section.

3.2 SuperGLUE and MSGS Results

Table 3 presents the results of SuperGLUE and MSGS tasks. Due to compute constraints, we only compare one of our models in each track with the BabyLM baselines. Our hyperparameter search space only concerns learning rate and batch

size, with the rest of hyperparameters following BabyLM's offical repo. With limited compute resource (evaluating all fine-tuning tasks once takes around 13-15 hours on 2 RTX 3090s), we have only explored the combinations of $\{5e-5, 64\}$, $\{3e-5,$ $32\}$ and $\{2e-5, 16\}$, instead of exhaustive permutations of them. This follows the empirical intuition that, smaller batch sizes lead to more unstable optimization, and should be paired with small learning rates. We find that smaller learning rates and batch sizes generally work better for small datasets like CoLA, MultiRC and RTE.

For 100M track, our model outperforms baseline models on all SuperGLUE and MSGS tasks. On average, our model outperforms baselines by 4.1 and 54.2 absolute percentage points on SuperGLUE and MSGS respectively.

For 10M track, our methods also provide competitive results, outperforming baselines by 2.3 and 4.5 absolute percentage points on SuperGLUE and MSGS respectively.

This again confirms the universal effectiveness of our recipe pool, and also provides support for our hypothesis when evaluating BLiMP that, 10M datasets seem to work less well with our method compared to 100M dataset, because of the less selfcontained original inputs provided by informative datasets like Wikipedia.

3.3 BabyLM challenge

The resultant models are submitted as part of the BabyLM challenge. Considering all results, we submitted the 1-40 cnc model for 10M track, and 40-1 cnc model for 100M track, and named

$\begin{array}{c} \textbf{Tasks} \rightarrow \\ \textbf{Models} \downarrow \end{array}$	BLiMP	BLiMP-sup	SuperGLUE	MSGS	Weighted Avg.
Contextualizer-RoBERTa-base-10M-v1	79.24	62.30	69.58	12.69	60.54
Contextualizer-RoBERTa-base-100	85.54	63.35	72.85	57.99	72.96

Table 4: Results on Dynabench Leaderboard. Notably, the BabyLM official evaluation has further included 5 BLiMP supplementary tasks, denoted as BLiMP-sup here.

them Contextualizer-RoBERTa-base- $10M-v1^1$ and Contextualizer-RoBERTa-base- $100M^2$.

Table 4 presents the official BabyLM Challenge results of our two models on the Dynabench Leaderboard.

4 Inner-workings Analysis

As simple as learning the same things repeatedly in different, oftentimes irrelevant contexts is, our method achieves surprising results without changing other technical details. It is a natural question to wonder how the method has facilitated better learning.

As partly discussed in the Recipe section, we hypothesize that the inner-working of this method is largely relevant to mitigating shortcut learning, and spurious correlation. For instance, if the same data keeps being displayed to the models throughout all epochs, the model might tend to overfit to the co-occurrence of words in certain inputs, or even simply remember the sequence.

For instance, without our method, if a padded input "[cls] Figure 1 is an example figure for the concept. [cls] This is a completely irrelevant sentence." is seen by the model 40 times, the model might incorrectly learn a rule that "example" is two tokens before "for", or even depends on "irrelevant" in the irrelevant chunk padded to the same input, due to stochasticity in optimization, instead of relying "example" on the information in "Figure 1" and "for the concept".

By contrast, our method makes sure 1) an input is padded with a different input in every portion, so it will not be padded with the same "This is a completely random sentence." and seen by the model multiple times. This way, the model learns to focus attention within one document, instead of "peeking" tokens in other text chunks that happen to be padded into the same input with them. 2) an input is cropped at different positions in different portions of the data, making sure the model utilize information in a flexible way, instead of building over-reliance on certain shortcuts. As an example, in one portion, it might be "an example figure for the concept. [cls] sentences from dataset 1"; and in a different portion, it might be "sentences from dataset 5 [cls] Figure 1 is an example".

We have conducted a proof-of-concept experiment to support this hypothesis. We take the 100M 1-40 n and 40-1 n models respectively. Note that the 1-40n model sees the same padded inputs 40 times, and is theoretically prone to shortcut learning; while the 40-1 n model is expected to learn more actual dependencies among tokens, as it keeps seeing the different combinations of inputs. Note that they are both trained with 15%masking probability. We take the padded training set that is exposed to the 1-40 model, and mask $\{50\%, 85\%, 95\%\}$ tokens in every input respectively, we then compare the mlm loss produced by both models. Masking more than 50% of tokens should have already made most documents lose its semantics. If the mlm loss produced by 1-40 n model is much lower than 40-1 n model, we can conclude that, it presents certain levels of overfitting and shortcut learning, shown by its ability to recover more tokens, with very broken evidence.

$\begin{array}{l} \textbf{Mask Prob.} \rightarrow \\ \textbf{Model} \downarrow \end{array}$	50%	85%	95%
1-40 n	2.20	4.53	6.23 5.81
40-1 n	2.67	4.69	

Table 5: Memory Analysis. Both models are trained with 15% mask probability. We get the mlm losses with different mask probabilities on data exposed to 1-40 n model in training.

As Table 5 shows, this pattern clearly holds. For 50% and 85%, 1-40n model produces much lower mlm losses, showing its imposed memory on the corpus.

However, when the masking rate is increased to 95%, the 40-1 n model produces a lower loss. We speculate that this is because with 95% tokens

¹Dynabench ID: 1450

²Dynabench ID: 1343

masked (leaving around 6 tokens in every 128token input), the documents are extremely broken, and knowledge about basic grammar depending on these ~ 6 tokens are beneficial to recover parts of the tokens. Therefore, 40-1 n shows its robust grammar understanding towards more ubiquitous grammar phenomena.

5 Discussions

Due to limited time and compute resources, we position this work as a humble first step in studying contextualization as a data augmentation method for more human-like learning. We have proved the effectiveness of this context augmentation framework with its strong zero-shot linguistic knowledge performance gain, and leave design of other details such as the optimal way to tokenize, process and train the [cls] token for better performance on downstream fine-tuning tasks, as future work.

Moreover, while this work is largely restricted to studying pretraining of encoder models because of limited compute resources, we envision that the general findings are transferable to, or at least worth attention in several settings, including:

1) **Multilingual Models.** In the training of multilingual models, does practicing multiple languages in the same input chunk improve performance not only in code-switching scenarios, but also reflect in all languages individually? Does this performance manifest differently in low-resource languages, than in high-resource languages?

2) **Generative LLMs.** Does doing multiple instructions at the same time not only improve a LLM's multi-tasking abilities, but also improve abilities of individual tasks? How will this affect hallucinations?

Limitations

As discussed, our work is restricted to the study of encoder models. However, we envision certain transferability of our conclusions to encoderdecoder models and decoder-only models, and leave these for future work.

Moreover, again due to compute constraints, we have not been able to study the scaling law between model size and the number of times to augment the data with our Context-augmented Padding operation. We envision this to be very interesting, and SOTA-result-promising, as discussed in § 3.1.

References

- Tyler A Chang and Benjamin K Bergen. 2022. Word acquisition in neural language models. *Transactions of the Association for Computational Linguistics*, 10:1– 16.
- 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. 2021a. A framework for few-shot language model evaluation.
- Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021b. Simcse: Simple contrastive learning of sentence embeddings. In 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, pages 6894–6910. Association for Computational Linguistics (ACL).
- Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. 2020. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673.
- Jill Gilkerson, Jeffrey A Richards, Steven F Warren, Judith K Montgomery, Charles R Greenwood, D Kimbrough Oller, John HL 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.
- Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.
- 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.
- 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.
- Daniel Simig, Tianlu Wang, Verna Dankers, Peter Henderson, Khuyagbaatar Batsuren, Dieuwke Hupkes, and Mona Diab. 2022. Text characterization toolkit (tct). In Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing: System Demonstrations, pages 72–87.
- Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy,

and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

- 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, Haau Sing Li, Haokun Liu, and Samuel R Bowman. 2020b. Learning which features matter: Roberta acquires a preference for linguistic generalizations (eventually). In 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, pages 217–235. Association for Computational Linguistics (ACL).
- Chenghao Xiao, Yang Long, and Noura Al Moubayed. 2023. On isotropy, contextualization and learning dynamics of contrastive-based sentence representation learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12266–12283.

A 10M BLiMP results

The decision for which strategy to use for 10M track has been more nuanced and difficult. As shown in Table 6, we could see that Context-augmented Padding models (40-1) still outperform their Context-fixed Padding counterparts (1-40). However, this gap is not as large as the results in 100M, which as we discussed in main sections, is probably because of that, there exists much less self-contained data to resist against the noise brought by less semantic data (like children mumbling).

As shown in Table 2, using pure noisy data (entries with only "n" but no "c"), Context-augmented Padding (40-1 n) still outperforms Context-fixed Padding (1-40 n). However, when combining Clean Padding data, it seems to be detrimental to Contextaugmented Padding data, while is consistently contributing to Context-fixed Padding data. We hypothesize that, mixing Clean data and Noisy data in essence is a context-augmenting operation itself. And as we discussed, for 10M track, a moderate amount of noisy context, instead of too much, is better. We have also run some experiments on finetuning tasks on all model entries, and decided to use "1-40 cnc" for our final submission. Even though "40-1 n" provides the best BLiMP scores, it seems to rely on tasks with unstable behaviours such as irregular forms and quantifiers.

$\begin{array}{c} \text{Task} \rightarrow \\ \text{Model} \downarrow \end{array}$	Anaph. Agr.	Agr. Struct.	Bndg.	Ctrl./ Raise.	D-N Agr.	Ell.	F-G.	Irreg. Forms	Island Effects	NPI Lic.	Qnts.	S-V Agr.	Main Avg.
Baseline	81.50	67.10	67.30	67.90	90.80	76.40	63.50	87.40	39.90	55.90	70.50	65.40	69.47
1-40 n	90.18	74.73	69.07	71.90	94.59	86.49	72.74	88.40	59.60	67.70	69.45	84.17	77.42
1-40 cn	91.62	76.13	69.83	71.23	94.51	89.03	77.30	82.29	58.67	67.04	72.21	83.96	77.82
40-1 n	92.02	76.02	70.66	73.07	95.66	82.39	77.48	93.13	61.47	66.75	81.43	85.56	79.64
40-1 cn	93.15	76.33	70.73	73.93	96.96	82.91	77.90	87.74	63.23	66.50	77.77	87.35	79.54
40-1 ccn	92.38	77.30	70.97	74.61	95.48	82.56	78.20	86.77	62.03	62.92	71.90	87.05	78.51
1-40 cnc	92.79	76.33	71.64	72.56	95.65	89.15	75 4.99	87.63	61.47	69.78	72.46	86.43	79.24
40-1 cnc	94.12	74.48	67.60	71.45	95.12	82.79	76.38	91.55	58.67	74.05	82.17	83.34	79.31

Table 6: BLiMP Results of 10M recipes. Clearly, while all of our strategies outperform the baseline by a large
margin, results are more nuanced and it is not that straightforward to see which strategy is the best.