Integrating Quasi-symbolic Conceptual Knowledge into Language Model Pre-training

Gábor Berend
Institute of Informatics,
University of Szeged
2 Árpád tér Szeged, Hungary
berendg@inf.u-szeged.hu

Abstract

In this paper, we investigate the integration of latent conceptual knowledge into the pre-training of masked language models. Our solution is based on the use of an auxiliary model, from which we extract training signals for training a student model. We determine the training signals from the hidden representations of the student model in an unsupervised way, using sparse coding. Models trained on latent concepts alone have an improved fine-tunability on downstream tasks, however, they perform worse on traditional language modeling, i.e., when the goal is to output missing tokens as opposed to latent semantic classes of words. In order to preserve the improved fine-tuning capability of the models, while making them better at the task of language modeling, we propose a final stage of pre-training, during which we perform traditional masked language modeling. The final stage of pre-training is based on a model that has already been pre-trained on the task of modeling latent semantic properties, with the weights of the backbone model being frozen. During the final training phase, we only train a lightweight linear classifier layer on top of the logits that the model determines for the latent semantic properties. With this modification, we can obtain the benefits of both the traditional training paradigms and the one which is based on the use of latent semantic properties. We release our source code at github.com/SzegedAI/MLSM.

1 Introduction

Language acquisition involves forming a rich battery of concepts and the ability to use and manipulate those concepts. Even though human cognition is rooted in concepts, this is not reflected in the typical pre-training of neural language models. In contrast, standard pre-training techniques ignore the concept-oriented nature of language when they expect a single ground truth token to be predicted during pre-training time.

Shani et al. (2023) argues for the need of integrating conceptual information into language models, while (Berend, 2023) recommended a knowledge distillation approach for doing so. The Masked Latent Semantic Modeling (MLSM) approach (Berend, 2023) relies on an auxiliary teacher model that steers the pre-training of the student model by performing sparse coding on its hidden representation and requiring the student model to recover those instead of the actual tokens. As the location of the non-zero coefficients in the sparse contextualized word representations obtained that way can be viewed as quasi-symbolic latent semantic concepts (Berend, 2020), the pre-training becomes driven by concepts as opposed to tokens.

While the favorable properties of MLSM pre-trained models have been demonstrated in obtaining models with improved fine-tuning capabilities, models pre-trained with it struggle on tasks that require language modeling ability, i.e., predicting actual token substitutes for missing/masked token positions from an input sequence. This is a consequence of the modeling in MLSM being shifted from the actual tokens to the latent concepts determined in an unsupervised way.

In this work, we extend such a modification to MLSM modeling, which ensures that the final model does not only have improved fine-tuning capability, but it is also capable of performing regular language modeling on the token level. We achieve this goal by integrating a lightweight post pre-training phase, during which we keep the weights of the model determined via MLSM fixed, and add a small a final linear module to the network (while freezing the rest of it), such that the token predictions are made on the logits that the originally pre-trained model would return towards the latent concepts. This modification ensures that the positive properties of MLSM and traditional masked language modeling (MLM) pre-training can be integrated into a single final model.

2 Masked Latent Semantic Modeling

We first overview the MLSM pre-training technique, as it plays a central role in our modified model architecture. The way MLSM works is that it changes the domain of the output distribution of the model from its vocabulary of subword units (as in MLM) to the inventory of quasi-symbolic latent semantic properties that we determine in an unsupervised manner. In Figure 1, we provide a visual comparison between the MLM and MLSM pre-training techniques.

The way MLSM determines the latent semantic properties of some token is by relying on an already pre-trained auxiliary model $\mathcal{T}$. In a preparatory phase, a representative sample of hidden representations produced by $\mathcal{T}$ is collected from its layer l as $\{\boldsymbol{h}_1^{(l)}, \ldots, \boldsymbol{h}_N^{(l)}\}$. A dictionary learning problem (Mairal et al., 2009) is then solved of the form

$$\underset{\boldsymbol{D}^{(l)}, \boldsymbol{\alpha_j} \in \mathbb{R}^k_{\geq 0}}{\arg\min} \sum_{j=1}^{N} \frac{1}{2} \|\boldsymbol{h}_{\boldsymbol{j}}^{(l)} - \boldsymbol{D}^{(l)} \boldsymbol{\alpha_j}\|_2^2 + \lambda \|\boldsymbol{\alpha_j}\|_1, \tag{1}$$

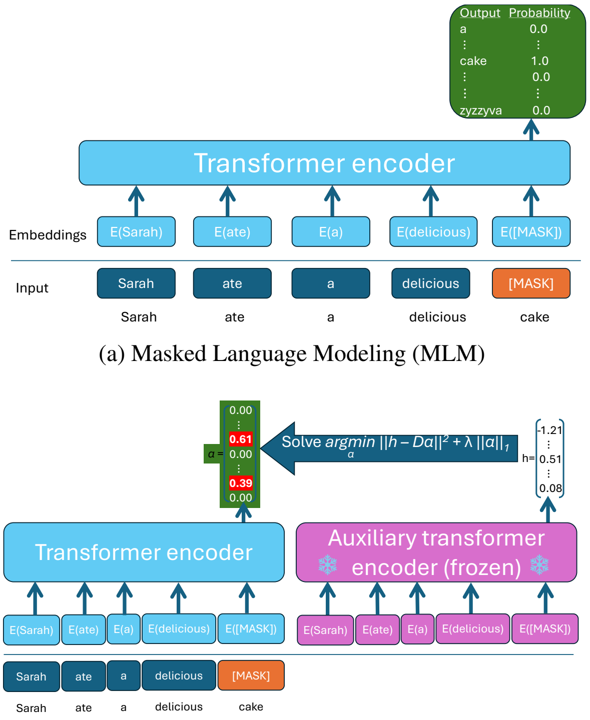
where $\boldsymbol{D}^{(l)} \in \mathbb{R}^{d \times k}$ is a dictionary matrix, with column vector norms bounded by 1, $\boldsymbol{\alpha_j} \in \mathbb{R}^k$ contains the sparse linear coefficients that indicate the extent to which the vectors from $\boldsymbol{D}^{(l)}$ are used in reconstructing the d-dimensional hidden representation from the l-th layer of $\mathcal{T}$, $\boldsymbol{h}_{\boldsymbol{j}}^{(l)} \in \mathbb{R}^d$. λ serves as a regularization coefficient, controlling for the level of sparsity in $\boldsymbol{\alpha_j}$.

Solving (1) is performed in advance to the actual pre-training, with a negligible ($\ll 1\%$) computational overhead compared to the costs of pre-training. Once the dictionary matrix $\boldsymbol{D}^{(l)}$ is determined, it is used for determining the sparse contextualized representation for any $\boldsymbol{h}_{\boldsymbol{i}}^{(l)}$, i.e., a hidden state from layer l of $\mathcal{T}$ as

$$\underset{\boldsymbol{\alpha_i} \in \mathbb{R}^k_{\geq 0}}{\arg\min} \frac{1}{2} \|\boldsymbol{h}_{\boldsymbol{i}}^{(l)} - \boldsymbol{D}^{(l)} \boldsymbol{\alpha_i}\|_2^2 + \lambda \|\boldsymbol{\alpha_i}\|_1. \tag{2}$$

Objective (2) is computationally convenient, as it does not require optimizing towards $\boldsymbol{D}^{(l)}$. With $\boldsymbol{D}^{(l)}$ being fixed from (1), obtaining the sparse linear coefficients of $\boldsymbol{\alpha_i}$ constitutes an efficiently solvable LASSO optimization problem.

Due to the non-negativity constraint imposed towards $\boldsymbol{\alpha_i}$ in (2), the ℓ_1-normalized sparse linear coefficients can be conveniently treated as probability distributions over the k latent semantic concepts.



(a) Masked Language Modeling (MLM)

(b) Masked Latent Semantic Modeling (MLSM)

Figure 1: Comparison of the MLM and MLSM pre-training paradigms. The distributions in the green boxes represent the expected output for the masked token.

MLSM pre-training then considers these sparse normalized distributions of latent semantic concepts of the masked tokens as the desired target outputs and computes the Kullback–Leibler divergence as the loss function.

3 Pre-training

Our proposed pre-training consists of three sequential steps. In the first step, we used classical masked language modeling for pre-training. The models pre-trained at this stage serve as the auxiliary teacher model for the subsequently trained model (see Figure 1b). As MLSM does not output subtokens, it is expected to have limited capabilities in performing tasks that require outputting distributions over the vocabulary of the model.

We trained a separate Unigram tokenizer for the two corpora, with a 25,000 vocabulary size. During all three stages of pre-training, we used the AdamW optimizer with a peak learning rate of 0.0001 and an effective batch size of 1024 (that resulted from using gradient accumulation over 8 batches). When masking is involved, we employ the typically chosen 15% random masking rate selected dynamically from the batches.

3.1 Preliminary pre-training

The preliminary phase of pre-training was conducted using vanilla masked language modeling objective. At this stage, we trained a classical DeBERTa (He et al., 2021) model of the base size (i.e., 12 layers, 12 attention heads, 768 hidden dimensions). This model has roughly 100 million non-embedding parameters and approximately 20 million embedding parameters.

When pre-training the auxiliary models, we conducted 100,000 update steps, which roughly corresponds to 200 epochs on the 10 million token dataset, and 20 epochs on the 100 million training corpus. As it was the number of update steps that we kept constant, pre-training took roughly the same time on both corpora, i.e., approximately 2 days on a single NVIDIA A6000 GPU.

3.2 Pre-training involving latent concepts

Once the auxiliary model was created, we determined the dictionary matrix according to Eq. (1). We chose to extract $k = 1500$ quasi-symbolic latent properties based on the hidden representations originating from the last layer of the network ($l = 12$), using the regularization coefficient $\lambda = 0.05$. We selected the hidden representations from the auxiliary model for 1 million tokens from the respective corpora for determining the dictionary matrices.

The student models that we trained based on the dictionary matrices created in a preparatory phase were also DeBERTa base models. As the architecture of the student models are identical to the auxiliary model, it was possible to initialize the weights of the student models with those of the respective auxiliary model. Unless stated otherwise, we applied that kind of weight initialization of the student models.

For this phase, we went for an additional 20 epochs of pre-training. This corresponds to approximately 10,000 and 100,000 update steps for the 10 million and the 100 million pre-training corpora. This resulted in approximately 5 and 50 hours of additional GPU compute for the 10 million and the 100 million token corpora, respectively.

We also implemented such variants of MLSM that perform concept-driven pre-training without the need to employ special mask tokens. These variants are based on the observation that the range of input symbols in MLSM differs from that of the expected output symbols, i.e., the model receives subtoken units and outputs a distribution over k latent quasi-symbolic concepts, which renders masking during pre-training unnecessary.

The omission of masking has the benefit that we do not have to restrict ourselves to learning from only 15% of the input tokens (i.e., the ones that are masked otherwise), and it also makes the distribution of the sequences seen during pre-training more similar to the ones seen in either fine-tuning or inference time (due to the lack of a special mask token). Apart from not replacing 15% of the input symbols to a special mask token, this kind of pre-training is performed in the same way as MLSM, and we refer to this variant as Latent Semantic Modeling (LSM), reflecting the fact that no artificial masking token is involved during the pre-training.

We created two versions of LSM. One was such that it derived pre-training loss from all the input tokens, while the other version (the LSM15) is such that it omits the masking token during pre-training, but resembles typical pre-training which involves masking in that only a randomly selected 15% of the tokens is used for updating the model.

3.3 Language Modeling head training

The goal of this phase is to secure classical language modeling capabilities of the models that we pre-trained in the previous step using latent concepts. To achieve this goal, we take the resulting model from the second phase and add an extra linear module on top of it, the goal of which was to perform token predictions based on the logits that the model from the previous stage determined for the distinct latent semantic categories.

As we wanted to preserve the concept forming capabilities of the model and not alter its fine-tuning abilities, we froze all the weights of the backbone model, the only weights that were learned at this stage were the ones in the final, newly added linear layer, which transformed the k latent concepts to the vocabulary of the model. That is, we introduced an additional 1500×25000 parameters in order to improve the language modeling capability of our models, resulting in a final model of 158 million parameters (out of which 20 million were embedding parameters). As this phase of training only involved the calibration of a single linear layer, we opted for only 10 thousand update steps (corresponding to roughly 20 and 2 epochs on the 10M and the 100M token corpora, respectively).

corpus	Phase 1	Phase 2	Phase 3
strict-small	≈ 50 h	≈ 5 h	≈ 2 h
strict	≈ 50 h	≈ 50	≈ 2 h

(a) GPU hours (on an NVIDIA A6000)

corpus	Phase 1	Phase 2	Phase 3
stict-small	≈ 200	≈ 20	≈20
strict	≈20	≈ 20	≈2

(b) Epochs performed

Table 1: The amount of compute broken down at the individual phases.

This final phase took less than two hours of GPU calculation. In Table 1, we summarize the amount computation performed for arriving to a final model both in terms of GPU hours (Table 1a) and the number of epochs (Table 1b).

4 Experimental results

We evaluate our models using the official evaluation framework of the shared task that was provided by the organizers (Warstadt et al., 2023). The evaluation involved model fine-tuning on various GLUE tasks (Wang et al., 2019) as well as zero-shot evaluations towards the BLiMP (Warstadt et al., 2020) and EWoK (Ivanova et al., 2024) benchmarks.

4.1 Fine-tuning experiments

We did not investigated in hyperparameter optimization, simply adopted the default fine-tuning hyperparameters recommended by the organizers. The only hyperparameter we modified was the random seed of the fine-tuning, and we only modified it, so that we can report performances that are statistically more robust by averaging the fine-tuning performances obtained on the different tasks.

We repeated fine-tuning on all dataset 5 times (with random seeds ranging from 12 to 16) and report the mean performance on each task. The performance metrics we include are accuracy, except for the CoLA, MRPC and QQP tasks, where it is the Matthew Correlation Coefficient for the former, and the F1 score for the latter two. The averaged performance metrics are presented in Table 2. Those models that were additionally pre-trained with the objective of being able to predict the latent semantic properties of the tokens show better fine-tunability when trained on any of the pre-training corpora.

	MLM	MLSM	LSM15	LSM
BoolQ	0.665	0.669	0.668	**0.673**
CoLA	0.398	**0.417**	0.384	0.400
MNLI	0.757	**0.761**	0.758	0.760
MNLI-mm	0.764	**0.769**	0.768	0.765
MRPC	0.822	0.819	0.820	**0.823**
MultiRC	**0.646**	0.636	0.629	0.633
QNLI	0.828	0.831	0.833	**0.836**
QQP	0.861	0.862	**0.864**	0.863
RTE	0.535	0.545	**0.566**	0.564
SST2	0.893	**0.900**	0.896	0.892
WSC	0.415	**0.485**	0.462	0.392
Avg.	0.689	**0.699**	0.695	0.691

(a) models pre-trained on the 10M corpus

	MLM	MLSM	LSM15	LSM
BoolQ	0.686	**0.697**	0.689	0.693
CoLA	0.509	0.484	0.511	**0.541**
MNLI	0.779	**0.789**	0.782	0.783
MNLI-mm	0.783	**0.791**	0.788	0.790
MRPC	0.906	0.905	0.913	**0.919**
MultiRC	0.629	**0.643**	0.639	0.635
QNLI	0.846	**0.853**	0.849	0.852
QQP	0.868	0.868	0.869	**0.869**
RTE	0.616	0.607	**0.645**	0.632
SST2	0.903	**0.905**	0.899	0.898
WSC	0.400	**0.419**	0.412	0.396
Avg.	0.720	0.724	0.727	**0.728**

(b) models pre-trained on the 100M corpus

Table 2: Fine-tuning results of models pre-trained with different strategies. Results are the average of 5 independent experiments using random seeds ranging between 12 and 16.

Based on the results in Table 2, there seems to be little difference in the fine-tuning ability of the models that integrate latent semantic information during their pre-training (*LSM*), however, our next experiment reveals the true strength of the masking-free variants of MLSM. For this experiment, we started the latent semantics-driven pre-training of DeBERTa models with randomly initialized weights. In our previous experiments, the reason for being able to warm start our student model for the second phase of pre-training, i.e., to initialize it with the weights of the auxiliary model, was that the student and teacher models matched in both their architecture and size.

	MLM	MLSM	LSM15	LSM
BoolQ	0.665	0.640	0.677	**0.674**
CoLA	**0.398**	0.000	0.176	0.291
MNLI	**0.757**	0.347	0.750	0.755
MNLI-mm	**0.764**	0.342	0.756	0.762
MRPC	0.822	0.811	0.822	**0.826**
MultiRC	**0.646**	0.576	0.625	0.613
QNLI	**0.828**	0.509	0.818	0.815
QQP	**0.861**	0.000	0.854	0.855
RTE	0.535	0.460	**0.594**	0.573
SST2	0.893	0.518	0.882	**0.894**
WSC	0.415	**0.523**	0.392	0.439
Avg.	**0.689**	0.430	0.668	0.681

Table 3: Fine-tuning results of models pre-trained on the 10 million token corpus. Results are the average of 5 independent experiments using random seeds ranging between 12 and 16. This time the weights of the student models were randomly initialized and the pre-training of the student models involved only 10 million updates, while the auxiliary model was created in 100 million update steps.

It can, however, often be the case that the student model we train differs from the auxiliary in either of its size or architecture. In such cases, simply continuing the pre-training of the teacher model is not directly applicable. To this end, we conducted such experiments, where the student model – albeit remaining of the same size and architecture as the auxiliary model – was initialized with random weights, so that we can simulate a more general situation when continued pre-training is not an option to go for.

The results of this setting, when pre-training was conducted on the 10 million tokens strict-small dataset, is included in Table 3. We can see that the performance of MLSM degrades severely, whereas its masking-free counterparts do not degrade as much. In fact, the LSM pre-trained model manages to reach the performance of its teacher from a randomly initialized state in one tenth of the pre-training, as the second phase pre-training lasted only for 10 thousand update steps for the small-strict corpus, whereas we conducted 100 thousand training steps for obtaining the auxiliary model. We omit the results for the 100 million token corpus for brevity, but the general trends are the same in that case as well.

It is only the CoLA task, where the LSM model (initialized from scratch) lags behind the MLM pre-trained auxiliary model. This is not that surprising as the CoLA tasks is related to linguistic acceptability, for which task a model that was pre-trained to predict the correct word forms can offer better transfer compared to a model that was purely constructed to model latent semantic categories that arguably play a less important role when deciding linguistic acceptability.

4.2 Zero-shot experiments

We report next the results when evaluation is performed on the language modeling capabilities of the differently pre-trained models, i.e., the evaluation metrics on the BLiMP datasets (Warstadt et al., 2020) and the EWoK (Ivanova et al., 2024) benchmark. Table 4 contains the results of our auxiliary model, as well as our models prior going through the third phase of pre-training and after the final lightweight pre-training phase being completed.

It is not surprising that the models that were pre-trained with a focus on latent semantic categories are not performing well in language modeling prior to the final phase of pre-training. Table 4 reveals that once the final pre-training phase – which only involves training a single linear classification layer and is only conducted for 10K update steps – is finished, the models that were previously pre-trained with an emphasis on modeling latent semantic categories of tokens can perform just as well as the auxiliary model, which had a sole focus on being able to accurately predict masked word forms. As the weights of our backbone model were frozen during the last phase of pre-training, our models also preserved their ability to predict latent semantic categories to input tokens and the final token-level predictions are precisely made based on those categories determined by the models.

It is worth mentioning, that an alternative way to achieve that the trained models have a combined command of modeling latent semantic properties and concrete word forms would be the use of a multi-task objective, in which the MLM and MLSM objectives are combined together. Our preliminary experiments showed, however, that models pre-trained that way do not have better performance during fine-tuning. Moreover, this kind of multitask training objective would be incompatible with the masking-free variant of latent semantics based pre-training, as LSM does not replace any of the input tokens with a special mask token, something that is required by MLM pre-training.

		W/o third phase pre-training			With third phase pre-training		
	MLM	MLSM	LSM15	LSM	MLSM	LSM15	LSM
BLiMP	0.653	0.521	0.528	0.528	0.654	0.642	0.641
BLiMP supplement	0.603	0.511	0.484	0.494	0.590	0.580	0.591
EWoK	0.647	0.680	0.682	0.689	0.652	0.667	0.666
Average	0.634	0.571	0.565	0.570	0.632	0.630	0.633

(a) models pre-trained on the 10M corpus

		W/o third phase pre-training			With third phase pre-training		
	MLM	MLSM	LSM15	LSM	MLSM	LSM15	LSM
BLiMP	0.702	0.446	0.465	0.471	0.696	0.687	0.684
BLiMP supplement	0.623	0.495	0.529	0.533	0.654	0.613	0.608
EWoK	0.657	0.681	0.654	0.659	0.656	0.667	0.669
Average	0.661	0.541	0.549	0.554	0.669	0.656	0.654

(b) models pre-trained on the 100M corpus

Table 4: Zero-shot results of models pre-trained with different strategies.

4.3 Submitted results

In Table 5, we summarize the results that our submitted models achieved, along with the baseline scores provided by the shared task organizers, the BabyLlama (Timiryasov and Tastet, 2023) and the LTG-BERT (Samuel et al., 2023) models being the best performing decoder and encoder-based submissions in last years evaluation campaign.

	GLUE	BLiMP	BLiMP suppl.	EWoK
BabyLlama	0.633	0.698	0.595	0.507
LTG-BERT	0.603	0.606	0.608	0.489
MLSM	0.733	0.654	0.590	0.508
LSM15	0.721	0.642	0.580	0.508
LSM	0.708	0.641	0.591	0.507

(a) Using the 10M token strict-small pre-training corpus

	GLUE	BLiMP	BLiMP suppl.	EWoK
BabyLlama	0.690	0.731	0.606	0.521
LTG-BERT	0.684	0.692	0.665	0.519
MLSM	0.748	0.696	0.654	0.523
LSM15	0.747	0.687	0.613	0.527
LSM	0.741	0.684	0.608	0.522

(b) Using the 100M token strict pre-training corpus

Table 5: The baseline performances provided by the organizers and our final submitted scores, the results above the horizontal bars are the baselines provided by the organizers.

We can see a drop in the EWoK performances between Table 4 and Table 5. The reason behind this is that in Table 4, we reported evaluation metrics that we obtained using the official evaluation scripts during the development phase. The organizers, however, discovered that those scripts produced inflated scores on EWoK (which were caused by the way the evaluation framework handled ties in the probabilities produced by a model). The results in Table 5 are the ones that contain the EWoK scores after this issue has been fixed.

5 Conclusions

In this paper, we investigated the integration of latent concepts extracted from an auxiliary model into the sample efficient pre-training of neural language models. We gave multiple modifications to existing approaches, including a masking-free variant of the originally proposed approach and the inclusion of a final, lightweight pre-training phase into the pre-training procedure, which ensures that the final model is not only capable of modeling semantic properties of tokens, but it can also accurately predict the identity of masked word form based on the latent semantic properties that the backbone model determines. Finally, we make the models that we pre-trained openly accessible from `https://huggingface.co/SzegedAI` (the models named with prefix `babylm24`).

Acknowledgments

This paper was supported by the János Bolyai Research Scholarship of the Hungarian Academy of

Sciences. The research received additional support from the European Union project RRF-2.3.1-21-2022-00004 within the framework of the Artificial Intelligence National Laboratory.

References

Gábor Berend. 2020. Sparsity makes sense: Word sense disambiguation using sparse contextualized word representations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8498–8508, Online. Association for Computational Linguistics.

Gábor Berend. 2023. Masked latent semantic modeling: an efficient pre-training alternative to masked language modeling. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13949–13962, Toronto, Canada. Association for Computational Linguistics.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTa: Decoding-enhanced BERT with disentangled attention. In *International Conference on Learning Representations*.

Anna Ivanova, Aalok Sathe, Benjamin Lipkin, Unnathi Kumar, Setayesh Radkani, Thomas H Clark, Carina Kauf, Jennifer Hu, Pramod RT, Gabriel Grand, Vivian Paulun, Maria Ryskina, Ekin Akyurek, Ethan Wilcox, Nafisa Rashid, Leshem Choshen, Roger Levy, Evelina Fedorenko, Josh Tenenbaum, and Jacob Andreas. 2024. Elements of world knowledge (ewok): A cognition-inspired framework for evaluating basic world knowledge in language models. *arXiv*.

Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. 2009. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 689–696, New York, NY, USA. ACM.

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.

Chen Shani, Jilles Vreeken, and Dafna Shahaf. 2023. Towards concept-aware large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13158–13170, Singapore. Association for Computational Linguistics.

Inar Timiryasov and Jean-Loup Tastet. 2023. Baby llama: knowledge distillation from an ensemble of teachers trained on a small dataset with no performance penalty. In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 279–289, Singapore. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In the Proceedings of ICLR.

Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Bhargavi Paranjabe, Adina Williams, Tal Linzen, and Ryan Cotterell. 2023. Findings of the BabyLM challenge: Sample-efficient pretraining on developmentally plausible corpora. In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 1–34, Singapore. Association for Computational Linguistics.

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