No Masking Needed: Predicting Latent Semantics for Efficient Pre-training of Encoder-based Language Models

Gábor Berend
Institute of Informatics
University of Szeged
Szeged, H6720, Hungary
berendg@inf.u-szeged.hu

Abstract

In this paper, we introduce Latent Semantic Modeling (LSM), a sample-efficient alternative for pre-training encoder-based language models that does not require the use of a special mask symbol. This is beneficial, as it allows us to perform pre-training on naturally occurring sentences (as opposed to such ones that contain an unnatural mask symbol, which symbol is otherwise not included during fine-tuning). The main idea behind LSM is that instead of requiring the language model to return the exact identity of randomly selected (sub)tokens from an input sequence, it determines their latent semantic properties extracted from an auxiliary model using sparse coding over its activations. When fine-tuned for downstream applications, models pre-trained with LSM often perform as good or better as alternatively pre-trained models that use considerably more compute per update steps and which have gone through considerably more pre-training steps, illustrating the improved representation developed during LSM pre-training. We release our code base for replicating our results under https://github.com/szegedai/MLSM.

1 Introduction

Encoder-based pre-trained language models (PLMs) predominantly rely on the Masked Language Modeling (MLM) pre-training objective (Devlin et al., 2019; Liu et al., 2019; Lan et al., 2020, *inter alia*). MLM offers a form of self-supervised learning, during which one selects a subset of tokens from the input sentence and replaces them with a special [MASK] token. The objective then is to recover *the exact identity* of the tokens that stood at the locations of [MASK] tokens.

While convenient, this kind of pre-training bears a certain level of unnaturalness, as it assumes that only a single token can be located at a given token position (i.e., the exact one that has been replaced by the [MASK] token), which assumption seldom holds. Masking also introduces a distribution shift between the pre-training and fine-tuning phases, as [MASK] tokens are only included during pre-training, but not fine-tuning.

Meng et al. (2024) showed that masking-based pre-training wastes model capacity and results in *representation deficiency*. An additional drawback of masking-based pre-training is the introduction of *corrupted semantics* (Zheng et al., 2025), which arises from masking multiple potentially interfering tokens from an input sequence. *Surface form competition* (Holtzman et al., 2021) and the *sensitivity to masking ratio selection* (Wettig et al., 2023) are further phenomena that masking-based pre-training suffers from.

We alleviate all the above limitations of MLM by changing the learning objective to the reconstruction of the latent conceptual properties of the input tokens. Besides allowing the pre-training to be conducted over the intact input sequences without the need of any form of perturbation, this kind of pre-training inherently offers interpretability via the use of latent concepts. To test the effectiveness of the proposed pre-training paradigm, we pre-train ModernBERT models and compare them with strong baselines.

There exist alternative pre-training mechanisms that omit the use of the [MASK] token, however, they all make the pre-training data unnatural in one way or another. The mitigation strategy offered in (Meng et al., 2024), for instance omits the tokens that would have been masked otherwise, resulting in imputed pre-training sequences. Pre-training on imputed sentences, and fine-tuning on full sentences introduces a distribution shift, limiting the effectiveness of transfer learning.

Even though models using replaced token prediction (RTP) (Clark et al., 2020; Bajaj et al., 2022; He et al., 2023; Dong et al., 2023, 2024) do not rely on [MASK] token either, they still turn the

Findings of the Association for Computational Linguistics: IJCNLP-AACL 2026, pages 4271–4285
November 6-10, 2026 ©2026 Association for Computational Linguistics

pre-training data somewhat atypical by replacing certain tokens by tokens that are sampled according to the proposal distribution of an auxiliary model. Similar to the omission of those tokens that would have been masked otherwise, replacing them with a noisy token arguably makes the pre-training sentences differ from naturally occurring ones, which is unfortunate from a transfer learning perspective.

In contrast to the existing masking-free PLM objectives, we propose such an alternative that requires *no modification* to the pre-training sequences. By employing such a pre-training mechanism which operates over *intact input sequences*, we introduce no artificial distribution shifts between pre-training and fine-tuning, allowing for a more efficient transfer between pre-trained and fine-tuned models.

During LSM pre-training, we determine a distribution of latent semantic properties of the input tokens by performing sparse coding on their corresponding activations produced by an auxiliary model. In LSM , we make use of the crucial observation that the support of its output probability distribution differs from the vocabulary of the model. That is, instead of outputting a distribution over the vocabulary of the model, we output a distribution over the inventory of latent semantics that we determine efficiently in an unsupervised manner.

Since, the range of the output symbols for LSM differs from that of the input, we can safely process the intact input sequence of tokens, without the need of introducing artificial `[MASK]` tokens. The omission of the `[MASK]` tokens during MLM pre-training, however, would be problematic, as predicting the identity of randomly selected tokens from the intact input sequence (while the target tokens are still present in the sentence in their original form) would yield a trivial pre-training task.

2 Related work

Masking plays a central role in the typical pre-training of encoder-based PLMs, with the default masking strategy originating from BERT (Devlin et al., 2019). In BERT, 15% of the input tokens are selected for pre-training, and 80% of the selected tokens get replaced by the special `[MASK]` token, while the remaining selected tokens are either kept unchanged or replaced to any random token. The pre-training objective is then to recover the exact identity of the 15% of the input tokens selected for pre-training.

Wettig et al. (2023) explored the effects of masking different token ratios from the input, finding it beneficial to mask larger fractions of tokens for larger models. They also noted that replacing 10-10% of the originally masked tokens to a random token and the original one is not a crucial step during MLM pre-training. Recent studies have proposed a variety of alternative masking strategies for improving MLM pre-training as opposed to the random selection of pre-training tokens (Joshi et al., 2020; Levine et al., 2021; Sadeq et al., 2022; Yang et al., 2023), however, these studies are orthogonal to our research, as our main goal is to devise such a pre-training paradigm which does not require the insertion of artificial mask tokens into the input sequences.

The elimination of the special `[MASK]` tokens can be further motivated by the recent results from (Meng et al., 2024), showing that PLMs pre-trained with MLM suffer from representation deficiency. The deficiency is rooted in the fact that the `[MASK]` tokens are only included during pre-training, however, during task-specific fine-tuning and inference, such artificial tokens are no longer part of the input. (Meng et al., 2024) also proposed a masked autoencoder (MAE-LM) architecture to mitigate the representation deficiency issue: instead of replacing the randomly selected tokens with the `[MASK]` symbol, MAE-LM completely removes them and trains an auxiliary decoder model for recovering the removed tokens.

Due to the removal of the masked tokens from the input sequence, MAE-LM also operates over out-of-distribution input sequences, i.e., such partially imputed sentences that are not coming from the true distribution of input sequences. Furthermore, unlike our proposed technique, obtaining direct pre-training signal from all the input tokens is not possible for MAE-LM, as it would mean that the network should recover entire sentences from the empty string.

Additionally, while MAE-LM mitigates the representation deficiency caused by the use of the `[MASK]` token, it is important to note that it does not fully eliminate the use of mask tokens, rather defers their use to a later layer of the transformer network. This is because the decoder of MAE-LM do involve `[MASK]` tokens, meaning that during backpropagation, `[MASK]` gradients still play a role in shaping the encoder of the model.

De-noising pre-training (Bajaj et al., 2022) is another alternative approach for eliminating the

need for [MASK] tokens during pre-training. A prototypical representative is ELECTRA (Clark et al., 2020) that uses the Replaced Token Detection (RTD) objective. RTD simultaneously relies on a generator (also called auxiliary) and a larger-capacity discriminator (also called main) model during pre-training. The generator suggests for perturbing the original input sequence, whereas the discriminator model has to decide which tokens of the input sequence are replaced by a generator-proposed alternative.

The RTD objective operates over all the input tokens, however, similar to MAE-LM, the pre-training input sequences do not follow the true distribution of the modeled language either. For MAE-LM, certain tokens get eliminated, whereas for RTD some of the original input tokens are replaced by model-generated alternatives. This causes an inherent distribution shift between pre-training and fine-tuning, something that our proposed approach is not affected by, since we process the exact same, intact input sequences during both pre-training and fine-tuning.

The training signal of our approach comes from sparse coding the activations of an auxiliary model. The sparse target signals that we construct are reminiscent to those representations produced by sparse autoencoders (SAE) (Huben et al., 2024) that have became a popular tool for identifying interpretable features in LLMs. Our use of sparse coding, however, does not rely on SAEs and our use of the sparse representations is also different from the mostly interpretability oriented work that use SAEs, as we use them for improving the efficiency of encoder-based PLMs. Sparse concept features have recently been proposed to enhance PLMs, however, these work have either been focusing on autoregressive models (Tack et al., 2025) or they have failed to address the representation deficiency issue (Berend, 2023).

3 Latent Semantic Modeling

In contrast to vanilla masked language modeling, models pre-trained with LSM change the domain of their output distribution from the subword units of the model vocabulary to the inventory of latent semantic properties determined in an unsupervised manner using sparse coding.

We determine the profile of latent semantic properties as introduced in (Berend, 2020) by relying on an already pre-trained auxiliary model $\mathcal{A}$. In a preparatory phase, a sample of N hidden representations produced by $\mathcal{A}$ is collected from its layer l as $\{\boldsymbol{h_1^{(l)}}, \ldots, \boldsymbol{h_N^{(l)}}\}$. This step is then followed by a preparatory phase, which can be formalized as a dictionary learning problem (Mairal et al., 2009) of the form

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

where $\boldsymbol{D}^{(l)} \in \mathbb{R}^{d \times k}$ is a dictionary matrix, with its column vectors upper bounded by unit norm, and the $\boldsymbol{\alpha_j} \in \mathbb{R}^k$ vectors contain 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 produced by layer l of $\mathcal{A}$, $\boldsymbol{h_j^{(l)}} \in \mathbb{R}^d$. λ is a regularization coefficient which affects the sparsity of $\boldsymbol{\alpha_j}$.

Solving (1) is a one time effort, performed before the actual pre-training phase, which results in a negligible ($\ll 1\%$) computational overhead compared to the typical costs of pre-training. Once the dictionary matrix $\boldsymbol{D}^{(l)}$ is determined, it is used for determining the sparse contextualized representation for any $\boldsymbol{h_i^{(l)}}$, i.e., a hidden state from layer l produced by $\mathcal{A}$ as

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

Objective (2) is computationally advantageous as it no longer requires optimizing towards $\boldsymbol{D}^{(l)}$, i.e., it relies on the one determined during the preparatory phase via (1). With $\boldsymbol{D}^{(l)}$ being fixed, obtaining the sparse linear coefficients of $\boldsymbol{\alpha_i}$ constitutes an efficiently solvable instance of a LASSO optimization problem.

Due to the non-negativity constraint for the coefficients of $\boldsymbol{\alpha_i}$ in (2), we can conveniently treat the ℓ_1-normalized version of $\boldsymbol{\alpha_i}$ as a probability distribution over the k latent semantic properties. LSM pre-training considers these sparse normalized distributions of latent semantic properties as the targeted output. Hence, the output space of LSM is *not* the entire vocabulary of the model, but it ranges over k special symbols, each corresponding to one of the latent semantic properties that are identified using (1).

If [MASK] tokens were not employed during traditional MLM, the pre-training task would become trivial, as the identity of the input tokens had to be predicted in their presence. The same,

however, does not hold for LSM pre-training as the input space (i.e., sequence of tokens) differs from the output space (i.e., distributions of latent semantic properties extracted from $\mathcal{A}$). This crucial observation makes the use of `[MASK]` tokens during pre-training obsolete, which allows us to conduct pre-training over the intact input sequences, alleviating the shortcomings of MLM pre-training mentioned previously.

During pre-training, we calculate the loss as the Kullback–Leibler divergence between the targeted distribution of latent semantic properties (that we determine using (2)) and the output distribution of our model. Since masking is not employed, we can increase the ratio of input tokens that directly contribute to the pre-training loss. As the input sequences do not get corrupted during pre-training in any way, making direct use of all the tokens of the pre-training corpora becomes feasible for LSM pre-training.

4 Experiments

Throughout our experiments, we created multiple PLMs that we evaluated over a comprehensive set of downstream benchmark evaluation tasks. In what follows, we describe the details of pre-training (§ 4.1) and fine-tuning (§ 4.2).

4.1 Pre-training experiments

We created ModernBERT models (Warner et al., 2025) from scratch for our experiments. The model consists of 22 transformer layers and employs 768 and 1152 dimensional hidden vectors and intermediate vectors in the FFNs of the transformer blocks.

We used the AdamW optimizer (Loshchilov and Hutter, 2019) with linear learning rate scheduling with a peak value of $1\mathrm{e}{-4}$, employing a warm-up phase that consisted of 2% of the $100,000$ total update steps that we performed. Throughout pre-training, we employed an effective batch size of 1024 (using gradient accumulation of 4 with a batch size of 256) and relied on the `transformers` library (Wolf et al., 2020).

As for the pre-training corpus, we relied on RefinedWeb (Penedo et al., 2023), released as part of the Dolma dataset (Soldaini et al., 2024). We additionally ensured for better comparability between the different PLMs that the individual batches over the pre-training came with identical contents and order.

All together the number of input sequences that we used for pre-training was approximately 100 million, i.e., each update was based on 1024 input sequences and we made 100000 updates in total. Over the course of pre-training, we have processed approximately 10 billion tokens.

4.1.1 Hyperparameter choices

When performing LSM, we treated the official `bert-base-cased` as our auxiliary model $\mathcal{A}$. Regarding the hyperparameters affecting (2), we selected the number of semantic concepts to be identical to the dimension of the hidden size of $\mathcal{A}$, i.e., $k = 768$. According to our additional experiments (Figure 6), increasing k does not provide better models, however, it increases the cost of pre-training as the parameters in the final output layer are proportional to the choice of k.

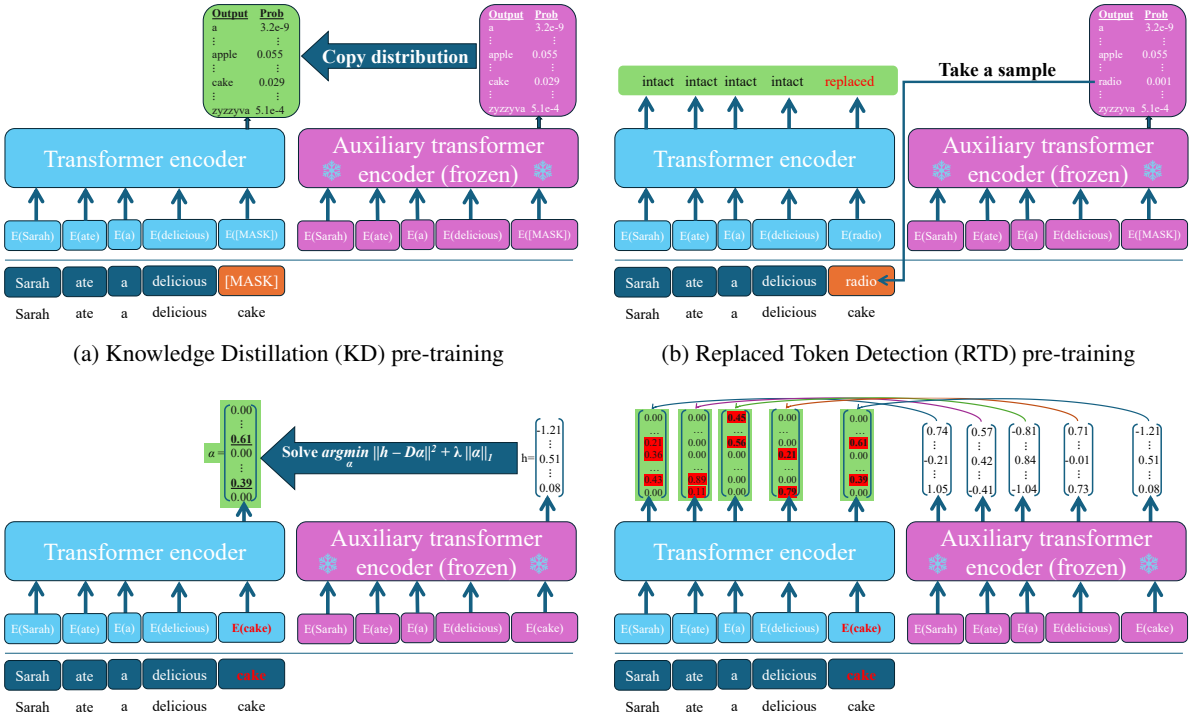
For obtaining $\boldsymbol{D}^{(l)}$ according to (1), we relied on 2,000,000 hidden vectors from $\mathcal{A}$. Finally, we determined the latent semantic distributions based on the tenth layer of the auxiliary model ($l = 10$).

This choice provided a good trade-off between performance and compute as the benefit of using earlier layers of $\mathcal{A}$ is that it allows us to reduce the costs of determining the target distribution of the main model via early exiting the inference of $\mathcal{A}$. By choosing the hyperparameters of LSM in the above stated manner, we could ensure that the pre-training costs of LSM were basically the same as that of MLM pre-training. More details on the computational aspects are included in Section 4.1.3. Additionally, for the sake of completeness, and to demonstrate that LSM is not particularly sensitive to the choice of hyperparameters, we report additional results on choosing different combinations of k and l in Appendix A.

4.1.2 Baseline models

Next, we introduce the baselines that we compare LSM with. We pre-trained all baseline models with identical hyperparameters (such as the optimizer, learning rate, batch size) as mentioned in Section 4.1. These hyperparameters were determined based on commonly used choices in the literature. Not only the hyperparameters, but also the contents of the batches, and the randomly selected tokens that directly contributed to the training were identical, meaning that when comparing the results of the different models, we can isolate the effect of using different pre-training approaches, without any conflating parameter.

Our first baseline is vanilla **MLM** pre-training,



(a) Knowledge Distillation (KD) pre-training

(b) Replaced Token Detection (RTD) pre-training

(c) MAE-LM pre-training

(d) Latent Semantic Modeling (LSM) pre-training

Figure 1: Overview of the different pre-training paradigms that rely on an auxiliary model. The auxiliary models are colored magenta, while the target outputs have a light green background. It is only the LSM pre-training (d) that is performed over the intact input sequence.

for which we employed dynamic masking strategy. Unless stated otherwise, we opted for the typical masking ratio of 15%.

As LSM uses an auxiliary model, it is reminiscent of knowledge distillation (Hinton et al., 2015; Aguilar et al., 2020). To this end, we conducted experiments using vanilla knowledge distillation (**KD**) pre-training. During KD (visually illustrated in Figure 1a), we also rely on the same pre-trained auxiliary model $\mathcal{A}$ that we use for determining the latent semantic distributions for LSM. In the KD setting, we mask the input sequences as for MLM, then query $\mathcal{A}$ for its prediction towards the distribution of substitute tokens, and calculate our pre-training loss based on how close can our student model approximate the output distribution of the auxiliary model in terms of KL-divergence.

ELECTRA-style replaced token detection (**RTD**) is also considered a sample efficient pre-training paradigm and it also resembles our approach as it does not apply an artificial `[MASK]` token (at least for its discriminator). Instead of using a `[MASK]` token, it replaces certain input tokens, which can also result in unnatural input sequences (as illustrated in Figure 1b).

Despite of its similarity, the ELECTRA formulation differs from our proposed pre-training paradigm, as it requires the training not only of the main model but the auxiliary model as well, whereas in LSM , the weights of the auxiliary model can be frozen. This difference not only results in a computational overhead for vanilla ELECTRA, but also makes its pre-training less stable (Dong et al., 2023, 2024; He et al., 2023).

To this end, we pre-trained a FastELECTRA (Dong et al., 2024) baseline, which performs such RTD pre-training that renders the training of the auxiliary model unnecessary. This not only reduces the pre-training cost (as the auxiliary model requires forward passes only), but it also makes the approach more comparable to ours.

We also implemented **MAE-LM** pre-training (illustrated in Figure 1c). Following (Meng et al., 2024), we opted for a shallow, 4-layer auxiliary decoder model for predicting the deleted tokens.

Finally, we also consider such a variant of LSM that masks the randomly selected tokens for pre-training (even though it was not necessary, as argued earlier). We refer to this version of LSM as masked latent semantic modeling (**MLSM**).

4.1.3 Computational costs of pre-training

We calculated the FLOPs related to processing a single batch of pre-training using the `calflops` package. In our calculations, we focused on the costs that incur over a single batch of pre-training our newly created models.

One might intuitively think that when using the same backbone architecture, self-supervised pre-training that exploits latent semantic information must require strictly more compute than vanilla MLM pre-training, which is caused by the use of the auxiliary model $\mathcal{A}$ for determining the target output distribution for LSM.

This sounds plausible at first, however, it does not hold in our case. The reason behind it is that for MLM pre-training, a non-negligible fraction of the computation is related to un-embedding, i.e., transforming the final hidden representations into the output space of the model. For MLM, the output space is the entire vocabulary V, whereas for LSM, it is the inventory of latent concepts, the size of which is tied to the number of columns of the dictionary matrix $D \in \mathbf{R}^{h \times k}$, with $k \ll |V|$. As such, the un-embedding operation is much cheaper (as it involves a multiplication with orders of magnitude smaller matrix of size $h \times k$ for LSM, as opposed to the use of an $h \times |V|$ for MLM, with h denoting the hidden size of the main model).

Another thing to note, is that even though LSM and KD both relies on an auxiliary model $\mathcal{A}$, LSM only requires its hidden states from some intermediate layer l (allowing to stop inference early), whereas KD requires computing a full forward pass over the entire $\mathcal{A}$. For smaller models, computing the output distribution can make up a reasonable fraction of the computation, something that can be spared in the case of LSM, but not for KD.

We report the costs of the different pre-training paradigms that we investigated throughout our experiments in a decreasing order in Table 1. We can see that no alternative pre-training technique had a more favorable cost than LSM. Table 1 reveals that the costs of LSM can be practically considered identical to that of MLM, whereas alternative approaches require up to +27% more compute.

	KD	RTD	MAE	(M)LSM	**MLM**
FLOPs	3.34E+13	2.9E+13	2.67E+13	2.67E+13	**2.63E+13**
Relative FLOP	127%	110%	103%	101.5%	**100%**

Table 1: FLOPs per pre-training batch with the pre-training approaches being ordered decreasingly. The costs of vanilla MLM pre-training (**in bold**) are treated as the baseline.

4.2 Fine-tuning experiments

We measure the utility of the PLMs by quantifying their fine-tuning performance over a wide range of benchmark tasks. During our evaluation, we relied on the following comprehensive list of datasets: CoNLL 2003 dataset for named entity recognition (Tjong Kim Sang and De Meulder, 2003), Corpus of Linguistic Acceptability (CoLA; Warstadt et al., 2019), MNLI natural language inference dataset (Williams et al., 2018), Microsoft Research Paraphrase Corpus (MRPC; Dolan and Brockett, 2005), QNLI benchmark (Rajpurkar et al., 2016; Wang et al., 2019b), Quora Question Pairs (QQP; Iyer et al., 2017), Recognizing Texutal Entailment (RTE; Dagan et al., 2006; Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009), Stanford Sentiment Treebank (SST2; Socher et al., 2013), Semantic Textual Similarity (STSB; Cer et al., 2017), Word-in-Context (WiC; Pilehvar and Camacho-Collados, 2019) datasets.

We accessed the above benchmarks and performed the evaluation of the fine-tuned models via the `datasets` (Lhoest et al., 2021) and `evaluate` (Von Werra et al., 2022) packages. As many of the datasets are part of the GLUE (Wang et al., 2019b) and SuperGLUE benchmarks (Wang et al., 2019a), where the labels of the test set are not available, we performed our evaluation on the development sets. We used the same frequently used hyperparameters for fine-tuning all the datasets. That is, we used a learning rate of $2\mathrm{e}{-5}$ with linear learning rate scheduling and a batch size of 32, performing 3 epochs. As the evaluation metric, we report the fine-tuning scores after the last epoch.

As we were interested to see how sample efficient the different pre-training alternatives are, we also evaluated the intermediate checkpoints through the course of training. In total, we evaluated 4 checkpoints for each differently pre-trained model which correspond to their 10%, 25%, 50% and 100% pre-training readiness levels relative to the total number of pre-training update steps.

For each investigated checkpoint, we performed 5 independent fine-tuning experiments on each downstream task, and report the averaged experimental results, in order to obtain a more robust approximation on the utility of the different pre-training paradigms.

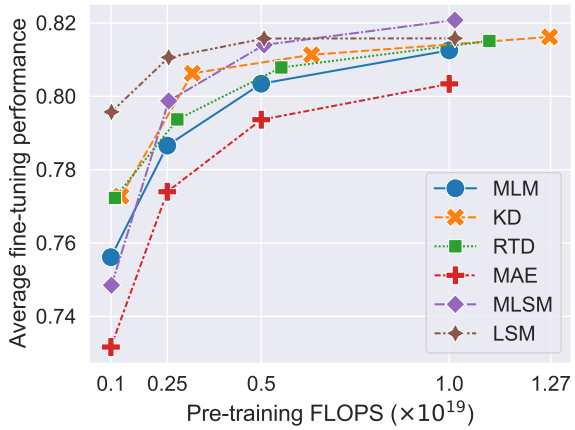


Figure 2: Overall fine-tuning scores obtained by differently pre-trained models at various pre-training stages. Markers indicate 10%, 25%, 50% and 100% pre-trained models. The location of the markers also reflects the different costs of the different pre-training objectives.

This means that we conducted $4 \times 5 = 20$ fine-tuning trials per task for each differently trained PLM. Performing the fine-tuning experiments over all the investigated dataset took approximately 2 GPU days per model checkpoints.

We include the overall fine-tuning results in Figure 2 for all the differently pre-trained model variants throughout the course of pre-training at different checkpoints. The markers of Figure 2 indicate the checkpoints at different readiness levels of the individual models, while also reflecting the amount of FLOPs that the different pre-training paradigms require.

We can see a clear advantage for LSM, especially at the early stages of pre-training. Our LSM pre-trained model has the remarkable property that it delivers comparable or even better fine-tuning performance already at its 25% readiness level compared to the end of pre-training fine-tuning performance of the alternatively pre-trained models. This, together with the fact that the LSM model recovers 97.5% of its final fine-tuning capabilities already at its 10% readiness level illustrates its pre-training sample efficiency. For comparison, we summarize the fraction of fine-tuning scores that the differently pre-trained models recover at their 10% readiness level in Table 2.

The individual task performances can be found in Figure 8 of Appendix C. When evaluated for the same checkpoints, i.e., at the same readiness level, LSM pre-trained models almost always outperform all the alternatively pre-trained models, often by a large margin. This not only holds for the results

MLM	KD	RTD	MAE	MLSM	LSM
93.4	94.7	94.7	91.1	91.2	**97.5**

Table 2: The percentage of end of pre-training model fine-tuning performance preserved by the model at its 10% readiness level.

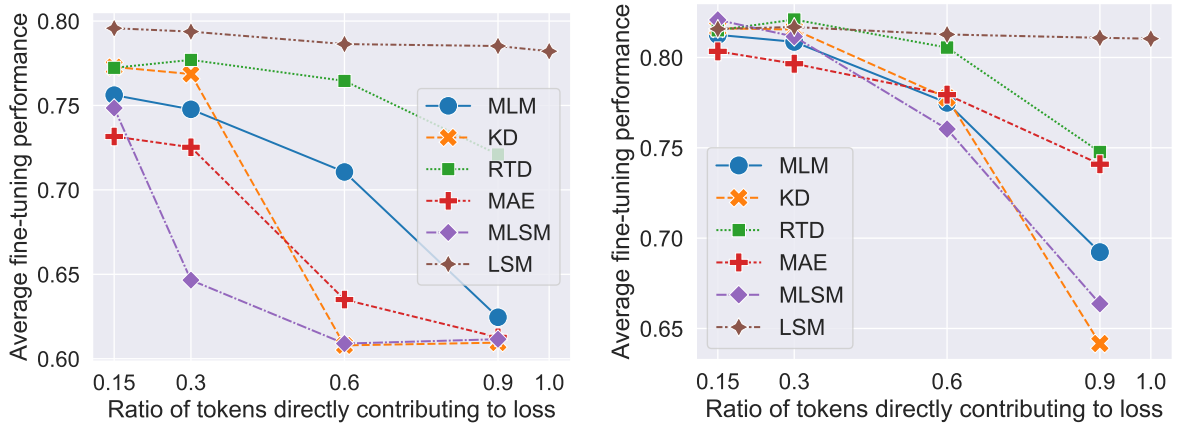
averaged over the different tasks, but it is also the case for almost all the individual evaluations, indicating that the improved performance of LSM is stable across various tasks and evaluations.

4.3 Investigating the ratio of tokens directly contributing to the loss

As LSM does not employ masking during, pre-training, we can safely increase the fraction of tokens that we compute our pre-training loss from. This corresponds to increasing the `[MASK]` token ratio for classical masking based pre-training. In an analogous manner, for MAE, increasing the ratio of tokens that we derive pre-training signal from means that we need to eliminate an increasing fraction of the original tokens from the pre-training input sequences.

Similar to Wettig et al. (2023), who investigated the effects of masking ratio during pre-training, we pre-trained alternative models with the fraction of tokens that we computed pre-training loss spanned $\{0.15, 0.3, 0.6, 0.9\}$. For LSM we additionally experimented a ratio of 1.0 that we did not test for the rest of the pre-training paradigms, as that would constitute a meaningless task. For instance in the case of MLM, the original input sequence had to be recovered from a sequence of tokens that consist of nothing else but `[MASK]` tokens.

Figure 3 displays the average fine-tuning performances of the differently pre-trained models at their 10% (Figure 3a) and 100% (Figure 3b) readiness levels as a function of the ratio of tokens that directly contribute to the pre-training loss. We can see that LSM shows practically no sensitivity to the ratio of tokens that the pre-training loss gets calculated neither at the early stage of the pre-training, nor at the end. All alternative models, however, showed a strong sensitivity to the fraction of tokens that directly contribute to the pre-training loss. The robustness of LSM pre-training to the ratio of tokens to compute loss from is another advantageous property that is caused by the fact that LSM makes it able to train from intact input sequences.

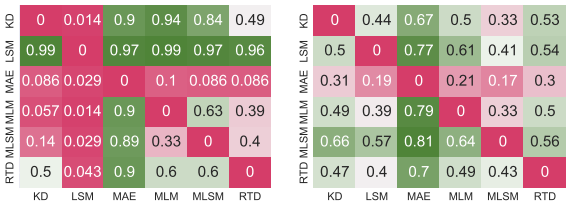


(a) Models after 10K pre-training steps

(b) Models after 100K pre-training steps

Figure 3: The average fine-tuning performance of differently pre-trained PLMs for varying ratio of tokens that directly constitute to the pre-training loss.

4.4 Significance testing



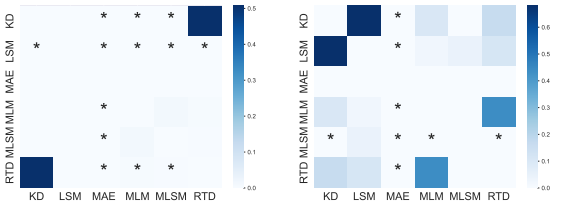
(a) 10% model readiness

(b) 100% model readiness

Figure 4: The fraction of fine-tuning experiments such that a model pre-trained with the technique along a given row performed better than the model pre-trained with the technique along a given column (when being initialized with identical classification specific weights).

As we can naturally pair fine-tuning experiments (when the classification heads were initialized identically at the beginning of a fine-tuning experiment), we measured the percentage of such experiments (out of 70 experiments in total for each pair of PLMs) that a particularly pre-trained model resulted in better fine-tuning score than an alternatively pre-trained model variant. We summarize these statistics in Figure 4.

We additionally calculated a Wilcoxon signed rank test over the paired fine-tuning experiments. In Figure 5, asterisks mark those cases for which the fine-tuning performance of the model pre-trained by the approach indicated by the row labels was significantly better (at $p < 0.01$) than that of the model which was pre-trained by the technique indicated by the column labels of the figure. We can see that at the early stages of pre-training (Figure 5a) the fine-tuning performance of the LSM model is significantly better than that of the alternatively pre-trained model variants. By the end of their pre-training most models perform similarly from the perspective of fine-tuning (Figure 5b). This suggests that in the limit, the quality of the models do not differ substantially, but in cases when the amount of compute allocated for pre-training matters (which happens often in practice), LSM could be a better pre-training alternative.



(a) 10% readiness

(b) 100% readiness

Figure 5: The p-values of the Wilcoxon signed rank test of the paired fine-tuning experiments between pairs of differently pre-trained models. Asterisks indicate statistically significant difference with $p < 0.01$.

4.5 Semantic probing

As the proposed objective of LSM relies on the (latent) semantic properties of tokens,it is instructive to test and compare how good semantic representations the differently pre-trained models develop during pre-training.

In order to see how different pre-training paradigms influence the internal representations of the models, we evaluated the semantic capabilities of our models using Word Sense Disambiguation (WSD). WSD requires models to distinguish between different meanings of a word based on contextual information, thereby directly testing the quality of semantic representations encoded within the model. For training our probing model, we used the sense-annotated data from SemCor (Miller et al., 1994) and WordNet Gloss Tagged corpora (Fellbaum, 1998). These datasets provide contextualized examples labeled with WordNet senses and are widely used in supervised WSD research.

Rather than fine-tuning the encoder models on the WSD task, we opted for a simple 1-nearest neighbor (1-NN) classifier. For each target word occurrence, contextual embeddings were extracted from the final layer of the encoder models and compared against embeddings derived from the sense-annotated training instances. The predicted sense corresponded to the nearest example in the embedding space according to cosine distance.

We deliberately use a non-parametric classifier, as fine-tuning-based approaches can conflate the semantic quality of pre-trained representations with the learning capacity of the downstream classifier. In contrast, the 1-NN setup allows the evaluation to focus directly on how well semantic information is inherently encoded within the model representations themselves. Consequently, the resulting performance more accurately reflects the intrinsic semantic expressiveness of the encoder language models rather than the effectiveness of task-specific optimization.

For our WSD experiments, we relied on the commonly used benchmark dataset from (Maru et al., 2022). We include our WSD results in Table 3 as micro F1 scores over multiple evaluation datasets. Our WSD results indicate that the semantic information encoded in LSM models is of higher quality than any of the alternative models. These results indirectly suggest that LSM models develop better representations from a semantic perspective, as we could solve a semantics-related task without fine-tuning more effectively compared to all the alternatives.

	softEN	42D	S10	ALL	Avg.
MLM	67.9	36.0	70.5	65.1	59.9
KD	70.9	**46.2**	70.5	68.3	64.0
RTD	67.5	34.1	66.6	65.1	58.3
MAE	64.2	37.0	63.8	62.2	56.8
MLSM	68.9	40.8	66.6	67.1	60,9
LSM	**73.9**	42.7	**71.5**	**71.2**	**64.8**

Table 3: Micro F1 results of the WSD experiments using a nearest neighbor classifier over the final hidden states of the differently pre-trained models.

5 Conclusions

In this paper, we proposed latent semantic modeling, a sample-efficient pre-training technique for encoder-based language models that – different from other techniques – does not require *any* form of input perturbation. As illustrated by our rigorous experiments, LSM pre-trained models offer better transfer capabilities to downstream tasks via fine-tuning. This can be attributed to the fact, that LSM pre-training is not affected by the representation deficiency typical of traditional masking based pre-training paradigms, which arises from the fact that the distribution of pre-training sentences mismatch the one encountered in real world texts (that do not include `[MASK]` symbols). We also demonstrated that changing the output distribution of our model can make the pre-training more efficient as the output space is reduced substantially. Additionally, since no mask symbol is used, it potentially allows us to obtain pre-training signal from every *intact* input tokens of the pre-training corpus, which could further improve the sample efficiency of masking-free latent semantic modeling. This is something that we plan to investigate further in our future research. We release our source code and pre-trained models under `https://huggingface.co/collections/SzegedAI/latent-semantic-modeling`.

Limitations One limitation is that we only trained English models, even though the unsupervised nature of determining the pre-training signal would make it possible to perform pre-training in different languages. The fact that our approach relies on an auxiliary model can be viewed as a limitation. We should add, however, that due to the sheer amount readily available pre-trained models, this is a rather mild requirement. As encoder-based PLMs are still popular, and newer models need to be trained from time to time, e.g., to prevent model staling, it is an important question if pre-training

can be made more efficient.

The use of the auxiliary model can be looked as a limitation from a resource efficiency point of view, however, as we showed it in Section 4.1.3, this is actually not the case. The LSM models we trained required practically the same amount of resources as vanilla MLM-trained PLMs (see Table 1), since in LSM we perform un-embedding towards a much smaller space, i.e., the space of latent semantic concepts as opposed to the entire vocabulary of the model.

One limitation arises from this efficiency, i.e., as the pre-training is not performed towards actual subtokens, LSM pre-trained models are not capable of providing a probability distribution over token substitutes of the model vocabulary as an output. This is an issue, for which we have identified a simple solution in the form of a lightweight post-training phase following LSM . We have conducted experiments by freezing the weights of an LSM pre-trained model, then adding a single classification head which uses the pre-trained model with the frozen weights as a feature extractor, and trained that single classification head with the standard MLM objective. We have noticed that by training the single matrix of parameters for as little as half an hour results in reasonable masked token prediction capabilities, suggesting that the latent concepts that the LSM pre-trained backbone model produces are meaningful. We consider a more thorough investigation of this approach as a potential follow-up work.

The official ModernBERT was trained for approximately 2,000 NVIDIA H100 GPU hours. In contrast, our models were trained for less than 100 GPU hours on a single NVIDIA A4000. This can be viewed as a limitation, however, we consider it as a deliberate design choice, as our primarily interest was the investigation of data efficiency for the different pre-training approaches. As we show it in Figure 7, despite being trained for orders of magnitude shorter, we still managed to recover more than 97% of the fine-tuning performance of the official ModernBERT checkpoint.

Acknowledgments

This paper was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences. The research received additional support from project no. 2024-1.2.3-HU-RIZONT-2024-00017, which has been implemented with the support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation Fund, financed under the 2024-1.2.3-HU-RIZONT funding scheme. Additionally, we are grateful for the possibility to use ELKH Cloud (see Héder et al., 2022; https://science-cloud.hu/) which helped us achieve the results published in this paper.

References

Gustavo Aguilar, Yuan Ling, Yu Zhang, Benjamin Yao, Xing Fan, and Chenlei Guo. 2020. Knowledge distillation from internal representations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7350–7357.

Payal Bajaj, Chenyan Xiong, Guolin Ke, Xiaodong Liu, Di He, Saurabh Tiwary, Tie-Yan Liu, Paul Bennett, Xia Song, and Jianfeng Gao. 2022. METRO: efficient denoising pretraining of large scale autoencoding language models with model generated signals. *CoRR*, abs/2204.06644.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The fifth PASCAL recognizing textual entailment challenge. In *In Proc Text Analysis Conference (TAC'09)*.

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.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages

177–190, Berlin, Heidelberg. Springer Berlin Heidelberg.

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.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Chengyu Dong, Liyuan Liu, Hao Cheng, Jingbo Shang, Jianfeng Gao, and Xiaodong Liu. 2023. Understand and modularize generator optimization in ELECTRA-style pretraining. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 8244–8259. PMLR.

Chengyu Dong, Liyuan Liu, Hao Cheng, Jingbo Shang, Jianfeng Gao, and Xiaodong Liu. 2024. Fast-ELECTRA for efficient pre-training. In *The Twelfth International Conference on Learning Representations*.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9, Prague. Association for Computational Linguistics.

R Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second PASCAL recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, volume 7.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. DeBERTav3: Improving deBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations*.

Mihály Héder, Ernő Rigó, Dorottya Medgyesi, Róbert Lovas, Szabolcs Tenczer, Ferenc Török, Attila Farkas, Márk Emődi, József Kadlecsik, György Mező, Ádám Pintér, and Péter Kacsuk. 2022. The past, present and future of the ELKH cloud. *Információs Társadalom*, 22(2):128.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Ari Holtzman, Peter West, Vered Shwartz, Yejin Choi, and Luke Zettlemoyer. 2021. Surface form competition: Why the highest probability answer isn't always right. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7038–7051, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. 2024. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations*.

Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. 2017. First quora dataset release: Question pairs.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Yoav Levine, Barak Lenz, Opher Lieber, Omri Abend, Kevin Leyton-Brown, Moshe Tennenholtz, and Yoav Shoham. 2021. PMI-masking: Principled masking of correlated spans. In *International Conference on Learning Representations*.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, and 13 others. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. 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.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

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.

Marco Maru, Simone Conia, Michele Bevilacqua, and Roberto Navigli. 2022. Nibbling at the hard core of Word Sense Disambiguation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4724–4737, Dublin, Ireland. Association for Computational Linguistics.

Yu Meng, Jitin Krishnan, Sinong Wang, Qifan Wang, Yuning Mao, Han Fang, Marjan Ghazvininejad, Jiawei Han, and Luke Zettlemoyer. 2024. Representation deficiency in masked language modeling. In *The Twelfth International Conference on Learning Representations*.

George A. Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G. Thomas. 1994. Using a semantic concordance for sense identification. In *HUMAN LANGUAGE TECHNOLOGY: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Hamza Alobeidli, Alessandro Cappelli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The refinedweb dataset for falcon LLM: Outperforming curated corpora with web data only. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Nafis Sadeq, Canwen Xu, and Julian McAuley. 2022. InforMask: Unsupervised informative masking for language model pretraining. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5866–5878, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, and 17 others. 2024. Dolma: an open corpus of three trillion tokens for language model pretraining research. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15725–15788, Bangkok, Thailand. Association for Computational Linguistics.

Jihoon Tack, Jack Lanchantin, Jane Yu, Andrew Cohen, Ilia Kulikov, Janice Lan, Shibo Hao, Yuandong Tian, Jason Weston, and Xian Li. 2025. LLM pretraining with continuous concepts. *arXiv preprint arXiv:2502.08524*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Leandro Von Werra, Lewis Tunstall, Abhishek Thakur, Sasha Luccioni, Tristan Thrush, Aleksandra Piktus, Felix Marty, Nazneen Rajani, Victor Mustar, and Helen Ngo. 2022. Evaluate & evaluation on the hub: Better best practices for data and model measurements. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 128–136, Abu Dhabi, UAE. Association for Computational Linguistics.

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. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

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

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Griffin Thomas Adams, Jeremy Howard, and Iacopo Poli. 2025. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2526–2547, Vienna, Austria. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Alexander Wettig, Tianyu Gao, Zexuan Zhong, and Danqi Chen. 2023. Should you mask 15% in masked language modeling? In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2985–3000, Dubrovnik, Croatia. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Dongjie Yang, Zhuosheng Zhang, and Hai Zhao. 2023. Learning better masking for better language model pre-training. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7255–7267, Toronto, Canada. Association for Computational Linguistics.

Kangjie Zheng, Junwei Yang, Siyue Liang, Bin Feng, Zequn Liu, Wei Ju, Zhiping Xiao, and Ming Zhang. 2025. ExLM: Rethinking the impact of [MASK] tokens in masked language models. In *Forty-second International Conference on Machine Learning*.

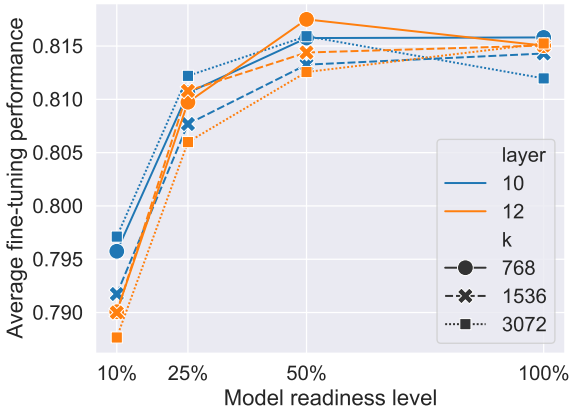


Figure 6: Comparing the fine-tuning performance of LSM models trained with different choices of l and k.

A Ablations on the sparse coding step

The hyperparameters of LSM relate to applying sparse coding. The two key hyperparameters of Equation (1) are the number of semantic concepts to be identified (k) and the layer of the auxiliary model $\mathcal{A}$ that we use for determining the latent concepts (l). Besides the hyperparameter choices included in the main paper ($k = 768$, $l = 10$), we also investigated the use of additional hyperparameters for k and l.

Regarding the number of the latent semantic properties, we also relied on either two or four times the original dimensionality of the hidden vectors of $\mathcal{A}$, i.e., we trained LSM models with $k \in \{1536, 3072\}$. Additionally, we also conducted experiments with the use of the very last layer from $\mathcal{A}$, i.e., $l = 12$.

Figure 6 shows the fine-tuning performance of the LSM model checkpoints obtained with the alternatively determined hyperparameters as a function of pre-training update steps performed. These results suggest that LSM pre-training is robust to the choice of its hyperparameters. It is worth noting that increasing the values for k and l also increases the computational need of pre-training. As such, it is reassuring that even the smaller values for these hyperparameters work as good as larger ones.

B Comparing with official ModernBERT

For the sake of completeness, we report a comparison of the average fine-tuning scores with the official ModernBERT checkpoint. It is important to note, however, that our goal was not to challenge the performance of the official checkpoint (which was trained orders of magnitude more com-

pute), but to show via rigorous experiments the benefits of LSM when all other things apart from pre-training are kept equal. Figure 7 reveals that our LSM model already recovered 95% and 97% of the fine-tuning performance of the official ModernBERT checkpoint at its 10% and 25% complete checkpoints, respectively.

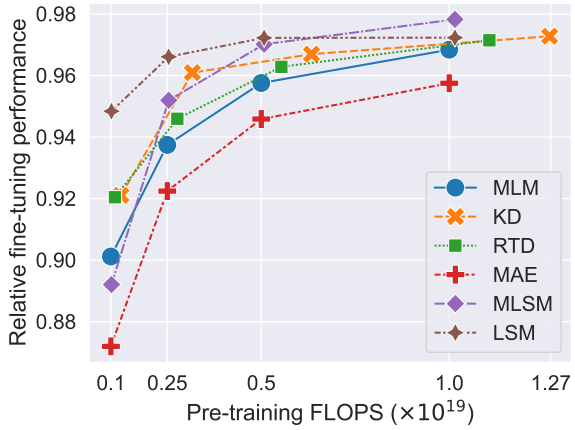
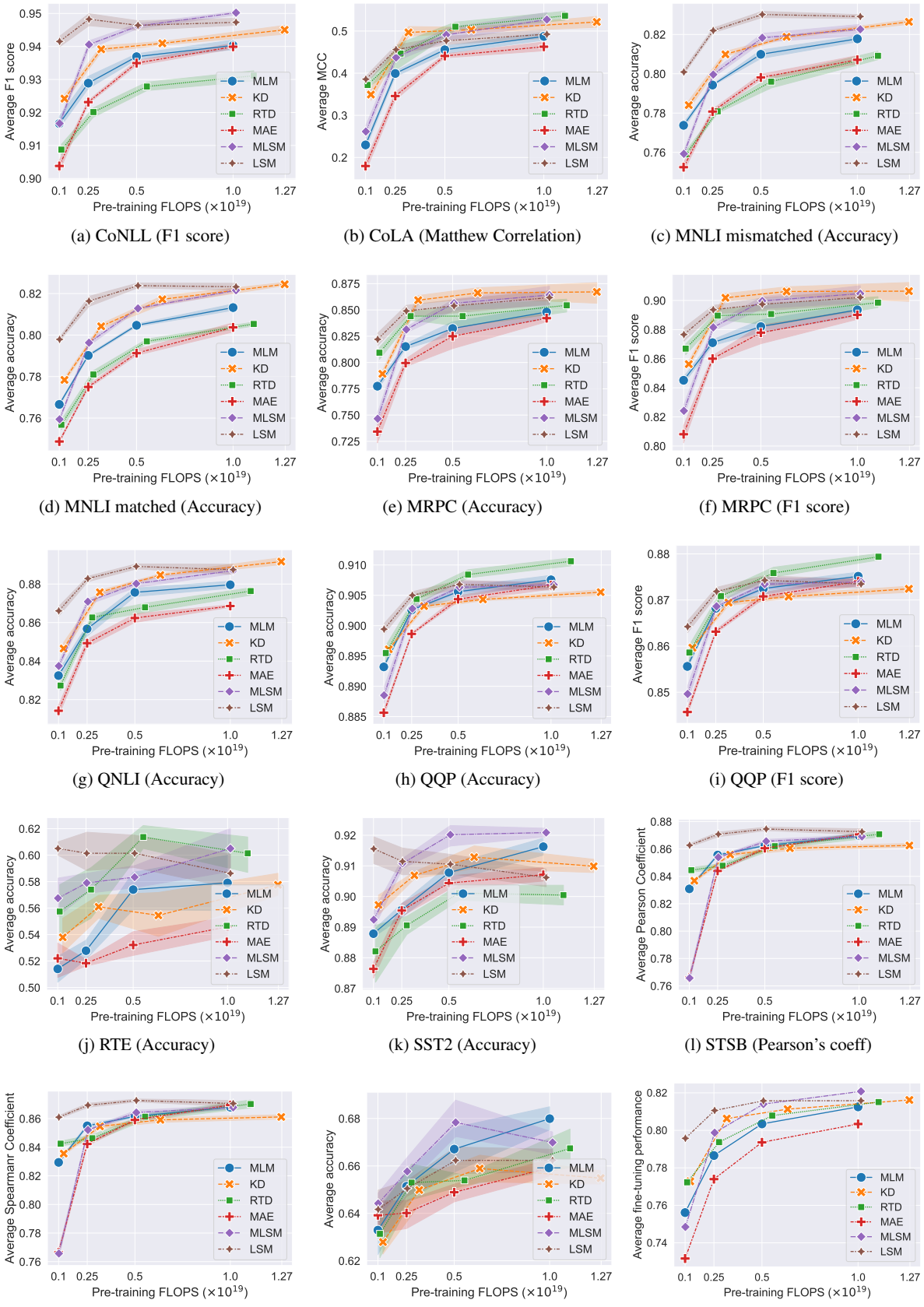


Figure 7: Relative performance of the differently trained PLMs compared to the fine-tuning results obtained by relying on the official ModernBERT model.

C Task-specific breakdown of the results

In Figure 8, we provide the fine-tuning performance of the differently pre-trained models on a task-specific basis as a function of the amount of compute invested into the given pre-trained model checkpoint that served as the input to fine-tuning.



(a) CoNLL (F1 score) (b) CoLA (Matthew Correlation) (c) MNLI mismatched (Accuracy)

(d) MNLI matched (Accuracy) (e) MRPC (Accuracy) (f) MRPC (F1 score)

(g) QNLI (Accuracy) (h) QQP (Accuracy) (i) QQP (F1 score)

(j) RTE (Accuracy) (k) SST2 (Accuracy) (l) STSB (Pearson's coeff)

(m) STSB (Spearman coeff) (n) WiC (Accuracy) (o) Average

Figure 8: Fine-tuning results broken down for the individual evaluation metrics and differently pre-trained models at various readiness levels as indicated along the x-axis.