Masked Latent Semantic Modeling: an Efficient Pre-training Alternative to Masked Language Modeling

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

Abstract

In this paper, we propose an alternative to the classic masked language modeling (MLM) pre-training paradigm, where we modify the objective from the reconstruction of the exact identity of randomly selected masked subwords to the prediction of their latent semantic properties. We coin the proposed pre-training technique *masked latent semantic modeling* (MLSM for short). In order to make the contextualized determination of the latent semantic properties of the masked subwords possible, we rely on an unsupervised technique using sparse coding. Our experimental results reveal that the fine-tuned performance of those models that we pre-trained via MLSM is consistently and significantly better compared to the use of vanilla MLM pre-training and other strong baselines.

1 Introduction

Recent successes in natural language processing are predominantly fueled by the use of large pre-trained language models (PLMs) that are constructed in a self-supervised manner over massive amounts of raw text. Autoencoder-style language models (Devlin et al. 2019; Liu et al. 2019; Lan et al. 2020; *inter alia*) are typically trained via masked language modeling (MLM).

PLMs pre-trained using MLM are capable of returning distributions over their vocabulary that peak at plausible substitutes of masked (sub)tokens given some sequence of input text. The individual updates performed during MLM pre-training, however, are not aligned to what we expect from the PLMs in the long run, i.e., that they output distributions of plausible substitutes for masked (sub)tokens.

As a motivating example, consider the sentence *"Alice is eating a cake."*, and suppose that we randomly select the token *cake* to be masked. For such a training example, we would obtain the masked input sentence *"Alice is eating a* `[MASK]`*."*.

The smallest possible training loss would incur *for this particular example* if our model allocated all its output probability mass to the word *"cake"* as being the *only* possible replacement of the `[MASK]` token, while assigning *precisely zero* probability to other alternatives, that are otherwise totally viable from a human cognitive perspective, including words such as *pear*, *croissant*, *soup*, *etc.*

What eventually provides PLMs trained with the MLM objective the ability to output token distributions that are plausible from a human perspective, is that they are trained over massive amounts of diverse batches, and the different possible substitutes even out in expectation. The hypothesis that we investigate in this paper is that we can train PLMs more efficiently if – instead of relying on the exact identity of the masked tokens during pre-training – we required our model to output such distributions that are not peaked at a single symbol (corresponding to the identity of the masked token).

A more natural and perhaps more sample efficient option to overcome the misalignment between the individual pre-training updates of PLMs and their long-term objective could compare the output distribution of the model to some desired distribution of substitutes, which would – instead of encoding the masked input token with a one-hot categorical distribution – assign nonzero probabilities to multiple viable tokens. Note, however, if we had access to such desired output token distributions for the masked tokens, then language modeling was already solved and the task of training PLMs would become obsolete.

An alternative approach for pre-training that mitigate the exact reliance on the identity of the masked tokens could rely on semantic resources, such as WordNet (Fellbaum, 1998) and ConceptNet (Speer et al., 2017). In this case, one might require the language model to output semantic properties of the masked tokens, i.e., in the previous example input sentence, instead of recovering the word *cake* for the [MASK] token, the goal of the PLM was to output its semantic properties, such that the masked word refers to a concept that is *edible* and is *a kind of dessert*.

One difficulty of such an approach is that obtaining semantic resources with sufficient coverage is notoriously arduous (Gale et al., 1992). In addition, such a pre-training paradigm would need the training corpus to be annotated with the possible properties of the words in their particular contexts. Few well-resourced languages have such sense-annotated corpora available, e.g., SEMCOR (Miller et al., 1994) for English, but their size is still several orders of magnitude smaller compared to the size of the corpora used for pre-training PLMs.

In this work, we propose masked latent semantic masking (MLSM), such an alternative of MLM, which attenuates the direct reliance on the masked tokens during pre-training. The main idea behind MLSM is that we no longer require the PLM to recover the exact identity of the masked tokens, but we predict their (latent) semantic properties.

Our pre-training procedure is based on the observation that sparse representations obtained via sparse coding on the hidden representations of neural models are well aligned with human interpretable properties (Balogh et al., 2020; Berend, 2020; Yun et al., 2021). The way we incorporate the above property into MLSM pre-training is that we derive context sensitive latent semantic information about the masked tokens by performing sparse coding on their hidden representation and we aim at predicting those as a pre-training task. Since we determine the sparse representations in an unsupervised way, our approach is not affected by the difficulties that would arose when relying on external semantic resources.

Our evaluation confirms our expectations, i.e., that PLMs pre-trained with MLSM can significantly outperform such models that are trained via vanilla MLM and other strong baselines. We release our code for performing MLSM pre-training at `https://github.com/szegedai/MLSM`.

2 Related work

Integrating external semantic knowledge into PLMs has gained increasing research interest (Mihaylov and Frank, 2018; Bauer et al., 2018; Peters et al., 2019; Ye et al., 2019; Yang et al., 2019; Qiu et al., 2019; Liu et al., 2020; Wang et al., 2021; Lu et al., 2021). These efforts cover a wide methodological spectrum, depending on how the external knowledge is incorporated into the PLMs. The different approaches can be distinguished for instance on the location where the external knowledge gets injected, i.e., either at the input, architectural or output level.

Colon-Hernandez et al. (2021) provides a comprehensive overview of approaches aiming at the incorporation of external knowledge into PLMs. What all these approaches have in common is that they rely on some explicit knowledge representation, e.g., in the form of triplets of some knowledge graph. In contrast, our main research question was whether it is possible to increase the semantic awareness of PLMs *without* having access to explicitly stored knowledge during the pre-training.

SenseBERT (Levine et al., 2020) is such a modification of BERT that also aims at the integration of semantic knowledge, however, it also differs from our approach in multiple aspects. The most important difference is that MLSM does not require any external linguistic resources, whereas SenseBERT relies on WordNet, making it only available for languages where such a linguistic resource is accessible.

Our fully unsupervised approach for inducing implicit semantic information to words within their context builds on the observation that performing sparse coding over the contextual representations of PLMs results in such sparse contextualized representations that align well with the semantic categories of the words. Berend (2020) showed that these sparse vectors can be used for improving word sense disambiguation (WSD), whereas Yun et al. (2021) used it for creating visualizations that help understanding the inner workings of PLMs from a semantic perspective. (Berend, 2022) provided further evidence that sparse contextualized word representations can be successfully exploited in cross-lingual WSD as well.

3 Methodology

As opposed to vanilla MLM, being agnostic to the semantics of the masked tokens, we propose MLSM, an alternative pre-training formulation which gets enhanced via semantic information.

3.1 Determining semantic information

As mentioned earlier, we wish to incorporate context-sensitive semantic information of the masked tokens into the pre-training procedure.

That is, for some token t_i within an input sequence of tokens $\mathcal{I} = [t_1, t_2, \dots, t_i, \dots, t_{|\mathcal{I}|}]$, we need to be able to determine its semantic profile $s(t_i)$.

3.1.1 Using external semantic resources

Semantic information of words can be obtained from ontologies or knowledge graphs such as WordNet (Fellbaum, 1998) and ConcepNet (Speer et al., 2017), containing semantic information in the form of triplets. For instance, *(cake, HasProperty, sweet)* is one of the triplets included in ConceptNet, based on which $s(t_i)$ can be determined.

One difficulty arising when using knowledge resources for determining $s(t_i)$ originates from the potential ambiguity of words, as knowledge bases can provide multiple (often semantically conflicting) relations a word can partake. For instance, ConceptNet contains the triplet *(book, AtLocation, university)*, as well as *(book, SameAs, reserve)*, which refer to the noun and verb senses of the word *book*, respectively. When the knowledge base offers multiple semantic relations a word is involved, we choose one of the viable semantic information uniformly at random, which is akin to how Levine et al. (2020) handled ambiguity.

We consider such a modified pre-training procedure in which we extend the output vocabulary of our model by special symbols corresponding to the relations pertaining to some knowledge base $\mathcal{K}$, and the objective of pre-training is to output such a special symbol for a randomly masked input token, which is compatible with the knowledge in $\mathcal{K}$. We refer to this approach as MESM (Masked Explicit Semantic Modeling). When referring to a concrete realization of MESM, we shall suffix it with the abbreviated name of the knowledge base we rely on, with WN and CN denoting WordNet and ConceptNet, respectively.

3.1.2 Using sparse coding

We propose an efficient *unsupervised* method for determining the latent semantic description of *any* token given its context. Our approach requires a teacher PLM denoted by $\mathcal{T}$. During MLSM pre-training of a student model $\mathcal{S}$, we can use $\mathcal{T}$ for inferring latent semantic information that we require $\mathcal{S}$ to recover as its pre-training objective.

Our proposed way of determining $s(t_i)$ from $\mathcal{T}$ is based on the use of sparse coding (Mairal et al., 2009). We first perform a dictionary learning phase, during which we solve for

$$\min_{\boldsymbol{D},\boldsymbol{\alpha_j}\in\mathbb{R}^k_{\geq 0}} \sum_{i=1}^{N} \frac{1}{2}\|\boldsymbol{h_j^{(l)}} - \boldsymbol{D}\boldsymbol{\alpha_j}\|_2^2 + \lambda\|\boldsymbol{\alpha_j}\|_1, \quad (1)$$

where $\boldsymbol{D} \in \mathbb{R}^{d\times k}$ is a dictionary matrix, the norm of its column vectors not exceeding 1, $\boldsymbol{\alpha_j} \in \mathbb{R}^k$ is a sparse vector of coefficients indicating the extent to which the vectors from $\boldsymbol{D}$ are used for the reconstruction of $\boldsymbol{h_j^{(l)}} \in \mathbb{R}^d$, which is the hidden state of token j determined by $\mathcal{T}$ in layer l. λ is the regularization coefficient that control the sparsity of $\boldsymbol{\alpha_j}$.

Once the dictionary matrix $\boldsymbol{D}$ is determined, we can obtain sparse contextualized representation for any $\boldsymbol{h_i^{(l)}}$, i.e., a hidden state from layer l of $\mathcal{T}$ as

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

An important difference between (1) and (2) is that for the latter case, we do not optimize towards $\boldsymbol{D}$, hence the determination of the sparse coefficients for a fixed dictionary matrix $\boldsymbol{D}$ corresponds to solving a LASSO optimization on the hidden representation of the tokens for which we determine sparse contextual representations for.

As both (1) and (2) include a non-negativity constraint towards $\boldsymbol{\alpha_i}$, a natural approach to convert its sparsity structure into a latent semantic distribution is to ℓ_1-normalize it. That way, we can handle the ℓ_1-normalized coefficients of $\boldsymbol{\alpha_i}$ as a probability with which the k latent semantic properties (expressed by the column vectors of $\boldsymbol{D}$) hold for token t_i in its particular context.

During MLSM pre-training, we consider these sparse normalized distributions obtained from (2) for each masked token as the latent semantic information describing them. Similar to the integration of explicit human-collected semantic information into pre-training described in Section 3.1.1, we introduce k new special symbols into the output vocabulary of the model.

The k new symbols correspond to the semantic atoms that comprise the columns of the dictionary matrix $\boldsymbol{D}$ determined by (1), and we consider the loss of the student model $\mathcal{S}$ regarding the masked input tokens by comparing its output distribution towards the k special symbols with the latent semantic distribution that we obtain from the teacher model $\mathcal{T}$ described above. Unless stated otherwise, the loss that we employ for comparing the similarity of the output distribution of $\mathcal{S}$ and the desired target distribution derived from $\mathcal{T}$ during MLSM is the KL divergence.

4 Experiments

We pre-trained several BERT language models of `medium` size relying on the `transformers` library (Wolf et al., 2020). The `medium` models (Turc et al., 2019; Bhargava et al., 2021) comprise of 8 transformer blocks and use a hidden representation of 512 dimensions. We considered the official pre-trained `bert-base-cased` model as the teacher model $\mathcal{T}$ for MLSM and we relied on its tokenizer for all the models that we pre-trained.

In order to assess the effects of the different pre-training variants, we evaluated the fine-tuning performance of the models over 10 diverse tasks. We repeated all fine-tuning experiments 10 times in order to account for the variability of the results that occur due to the random initialization of the task-specific classification head (Dodge et al., 2020).

We used 3 NVIDIA A6000 GPUs for performing the experiments. Each pre-training and the repeated fine-tuning experiments that followed them for a single checkpoint took approximately 2 GPU weeks and 1 GPU day, respectively. [1]

4.1 Details on pre-training

We used the preprocessing pipeline released as part of the WikiBERT models (Pyysalo et al., 2021)[2] for obtaining a recent Wikipedia dump for conducting pre-training. We also shuffled the preprocessed input sequences for ensuring the diversity of the pre-training batches. Our corpus consisted of approximately 125 million sentences and 2.7 billion whitespace delimited tokens.

As large batch-size has been demonstrated to be beneficial during pre-training (Liu et al., 2019), we set the batch size to 1024 (using gradient accumulation over 32 batches). For ensuring model comparability, we fixed the input contents (including the positions of the tokens being masked) and the order of the batches in which they followed each other across the different pre-trainings.

We performed 300,000 update steps, resulting in approximately 300 million input sequences. Similar to the original implementation of BERT (Devlin et al., 2019), in order to speed up pre-training, we used a maximal sequence length of 128 for 90% of the pre-training, then increased it to 512 for the remaining steps. We used the typical learning rate of $1\mathrm{e}{-4}$ with linear learning rate scheduling, a warm-up phrase of $3,000$ steps and the AdamW optimizer (Loshchilov and Hutter, 2019).

The way MLSM determines latent semantic distribution to tokens requires obtaining $\boldsymbol{D}$. For doing so, we followed (Berend, 2020), i.e., we processed the SEMCOR corpus (Miller et al., 1994), and collected the hidden states from the last layer of `bert-base-cased`, while setting the number of semantic atoms and the regularization coefficient to $k = 3000$ and $\lambda = 0.05$, respectively.

Besides using MLSM, we constructed four different PLMs that served as our baselines. One of our baseline was pre-trained using the standard MLM objective, where the goal is to reconstruct the exact identity of the masked tokens from the output distribution of the model.

We pre-trained two further PLMs, where we relied on the contents of explicit semantic information in the form of knowledge bases. For these models, that we refer to as MESM-WN and MESM-CN, we required the models to output semantic information about the masked words according to WordNet and ConceptNet, respectively. When relying on WordNet, we introduced 45 additional special symbols to the vocabulary of our model, each corresponding to one of the possible lexnames in WordNet, and our goal was to classify masked tokens to their correct supersense. When using ConceptNet, we first collected its 3000 most frequent relations (since we used that many semantic atoms for determining our latent semantic descriptions as well), and created a special symbol for each (e.g.,`IsA_food`).

Since MLSM utilizes a pre-trained teacher model $\mathcal{T}$, which allows us to determine the latent semantic distributions that is required by MLSM pre-training, a natural baseline to compare against is based on distillation from the same teacher model. We refer to the distilled pre-training as MLM-D. In this scenario, we first determine the token output distribution of $\mathcal{T}$ for the masked tokens, and calculate our pre-training loss as the KL divergence between that distribution and the one our model outputs.

4.2 Fine-tuning evaluations

We next investigate if the different pre-training regimes come with different fine-tuning capabil-

[1] We have since added support for AMP, which provides an approximate 2-fold speedup during pre-training.

[2] https://github.com/spyysalo/wiki-bert-pipeline

ities. During our evaluations, we relied on a wide range of tasks spanning both token classification in the form of named entity recognition (NER), and various sentence classification and regression tasks. More specifically, we performed evaluations on the Corpus of Linguistic Acceptability (CoLA; Warstadt et al., 2019), the CoNLL 2003 dataset for NER (Tjong Kim Sang and De Meulder, 2003), MNLIm and MNLImm, i.e., the matched and mismatched versions of the MNLI natural language inference dataset (Williams et al., 2018), the Microsoft Research Paraphrase Corpus (MRPC; Dolan and Brockett, 2005), the QNLI benchmark (Rajpurkar et al., 2016; Wang et al., 2019b) datasets, 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) and the Word-in-Context (WiC; Pilehvar and Camacho-Collados, 2019) datasets.

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. The hyperparameters were not tuned in any way to perform well on these sets, i.e., we used the same hyperparameters for all the tasks, and our choice for the selected values was purely driven by adapting commonly used values from earlier work and common best practices.

We accessed the above benchmarks and performed the evaluation of the fine-tuned models via the `datasets` and `evaluate` libraries (Lhoest et al., 2021; von Werra et al., 2022). We used the same frequently used hyperparameters for fine-tuning all the datasets. That is, we used a learning rate of $2\text{e}{-5}$ with linear learning rate scheduling and a batch size of 32, performing 3 epochs. As the evaluation metric, we always report the fine-tuning performance after the last epoch.

4.2.1 Results of the fine-tuning experiments

As mentioned before, evaluations were conducted 10 times for each task and differently pre-trained PLM. Table 1 summarizes the average performance of each PLM on each tasks, verifying that MLSM pre-training offers a competitive edge during fine-tuning compared to all other investigated alternatives of pre-training.

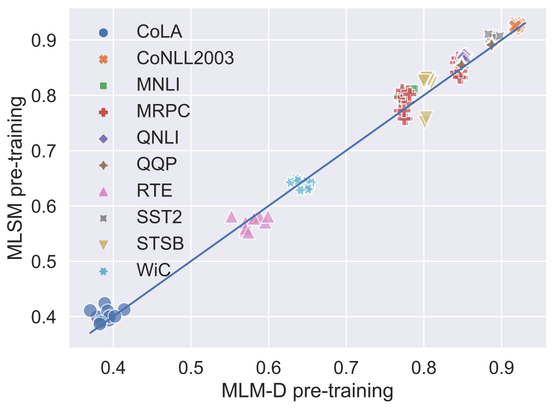

Figure 1: Scatterplot of the pairwise performances of the individual fine-tuning evaluations of the model pre-trained with MLSM and distillation (MLM-D). Markers above the diagonal indicate evaluations for which the MLSM pre-trained model scored better.

The largest performance gap is between MLSM and the approaches that try to integrate external human knowledge bases into the pre-training (the MESM-* paradigms). On average, the distilled model (MLM-D) was able to stay the closest in performance to the PLM pre-trained using MLSM, however, its performance still lags considerably behind that of our proposed model.

We performed the 10 repeated experiments for each dataset by ensuring their comparability across differently pre-trained models, i.e., we made sure that whenever a pair of PLMs was evaluated with the same seed and on the same task, their classification heads were initialized identically. Additionally, for each task, the batches included the same instances and got utilized in the same order during the fine-tuning experiments.

We depict the pairs of final task performances across all the fine-tuning experiments between the comparable trials of the two best performing models, i.e., the one pre-trained with MLSM and the one using distillation (MLM-D) in Figure 1. We can see that the MLSM pre-trained model resulted in better fine-tuning performances compared to those of the distilled model in most of the cases.

Indeed, the p-value of the Wilcoxon signed-rank test between their paired experimental results is $p < 2\text{e}{-12}$, which indicates that MLSM performs significantly better during fine-tuning compared to standard distillation. The p-value between the results of vanilla MLM and MLSM was even smaller, i.e., $p < 3\text{e}{-17}$.

Dataset	Metric	MLM	MESM-WN	MESM-CN	MLM-D	MLSM
CoLA	Matthews correlation	0.391	0.371	0.369	0.390	**0.403**
CoNLL2003	F1	0.919	0.905	0.909	0.920	**0.926**
MNLIm	Accuracy	0.768	0.719	0.735	0.771	**0.798**
MNLImm	Accuracy	0.771	0.731	0.749	0.782	**0.808**
MRPC	Accuracy	0.761	0.713	0.712	0.775	**0.786**
MRPC	F1	0.841	0.815	0.815	0.846	**0.851**
QNLI	Accuracy	0.850	0.775	0.779	0.852	**0.870**
QQP	Accuracy	0.886	0.840	0.850	0.888	**0.892**
QQP	F1	0.847	0.777	0.797	0.849	**0.855**
RTE	Accuracy	0.571	0.552	0.552	**0.578**	0.571
SST2	Accuracy	0.892	0.880	0.891	0.890	**0.905**
STSB	Pearson correlation	0.809	0.277	0.243	0.803	**0.818**
STSB	Spearman correlation	0.809	0.264	0.226	0.804	**0.820**
WiC	Accuracy	0.629	0.614	0.599	**0.644**	0.639
Average		0.7675	0.6596	0.6590	0.7709	**0.7815**

Table 1: Fine-tuning performances of the differently pre-trained PLMs, averaged over 10 independent initializations of their classification head.

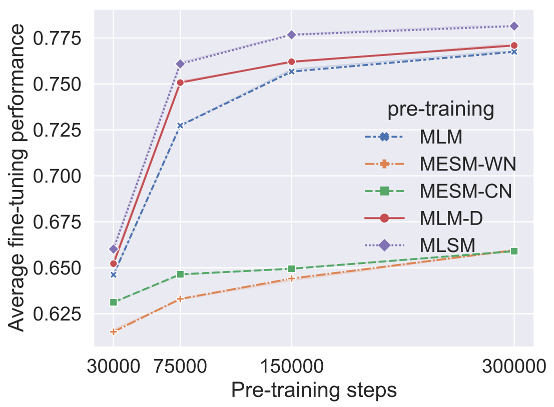

Figure 2: Average performance of the differently pre-trained models as a function of the number of pre-training update steps performed.

4.2.2 Investigating pre-training dynamics

Besides investigating the fine-tuning performance of the differently pre-trained models at the end of pre-training, we additionally evaluated them at different checkpoints, i.e., after processing 10% (30K), 25% (75K), 50% (150K) and 100% (300K) of all the update steps performed during pre-training.

Figure 2 illustrates that the fine-tuning performances of the PLM that was pre-trained using MLSM is substantially higher than any of the PLMs pre-trained in an alternative fashion. This observation does not only hold at the end of pre-training, but also for the earlier checkpoints.

Furthermore, the performance of the PLM relying on MLSM already surpasses the end-of-training performance of all alternatively pre-trained PLMs at its 25% checkpoint (at 75K pre-training steps), supporting our earlier hypothesis that MLSM converges faster and provides a more sample efficient form of pre-training.

4.2.3 The role of using semantic distributions

As mentioned in Section 3.1.2, the loss function employed by MLSM is the KL divergence between the latent semantic distributions determined for the masked tokens using the teacher model $\mathcal{T}$ and the output of the student model $\mathcal{S}$. We experimented with such a variant of MLSM that does not utilize the entire latent semantic distribution of the masked tokens, but only considers the index of the semantic category with the highest probability mass it is assigned to.

Under this variant of MLSM, the loss function we employed was no longer the KL divergence, but the cross entropy of the output of $\mathcal{S}$ and the most probable latent semantic category determined from $\mathcal{T}$. We refer to this variant of masked latent semantic modeling as MLSM-CE (owing to the use of cross entropy as the loss function).

Figure 3 compares the evaluation scores of both MLSM and MLSM-CE, revealing that the use of the entire latent semantic distribution determined by our approach together with KL divergence loss

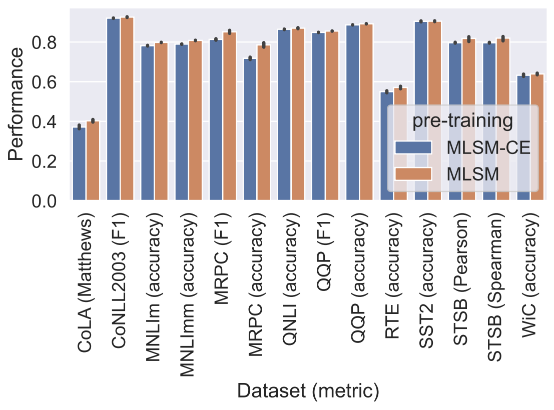

Figure 3: The fine-tuning performance of the PLMs pre-trained with MLSM-CE and MLSM.

Dataset	Metric	base w/ MLM		medium w/ MLSM	
CoLA	Matthews corr.	**0.430**	(0.014)	0.403	(0.012)
CoNLL2003	F1	**0.940**	(0.002)	0.926	(0.003)
MNLIm	Accuracy	**0.803**	(0.002)	0.798	(0.001)
MNLImm	Accuracy	0.807	(0.002)	**0.808**	(0.002)
MRPC	Accuracy	0.763	(0.056)	**0.786**	(0.020)
MRPC	F1	0.837	(0.034)	**0.851**	(0.013)
QNLI	Accuracy	**0.882**	(0.003)	0.870	(0.004)
QQP	Accuracy	**0.898**	(0.001)	0.892	(0.001)
QQP	F1	**0.862**	(0.001)	0.855	(0.001)
RTE	Accuracy	0.538	(0.019)	**0.571**	(0.011)
SST2	Accuracy	0.903	(0.006)	**0.905**	(0.004)
STSB	Pearson corr.	**0.828**	(0.004)	0.818	(0.024)
STSB	Spearman corr.	**0.824**	(0.003)	0.820	(0.021)
WiC	Accuracy	0.630	(0.025)	**0.639**	(0.007)
Average		0.7818		0.7815	

Table 2: Breakdown of the average performance of the base-sized BERT pre-trained with vanilla MLM and the medium-sized BERT pre-trained with MLSM. The standard deviation of the scores are put in parenthesis.

	Teacher model $\mathcal{T}$		
	BERT base	BERT large	RoBERTa large
MLM-D	0.7709	**0.7698**	0.7563
MLSM	**0.7815**	0.7696	**0.7783**

Table 3: The results of vanilla distillation and MLSM when using different teacher models.

is more beneficial compared to the use of the cross entropy-based loss. The overall average performance scores of the MLSM and MLSM-CE pre-trained models are 0.781 and 0.762, respectively.

4.2.4 Comparing model sizes

As a final quantitative experiment, we pre-trained another BERT model with vanilla MLM training, but this time, the model was of the `base` version. Apart from its larger capacity, pre-training was conducted entirely identically to the previously discussed smaller PLMs. The `base` model differs from the `medium` models in that it consists of 12 transformer blocks and it has a hidden dimension of size 768 (as opposed to the 8 blocks and 512-dimensional hidden states for the `medium` model). These differences make the size of the `base` model more than 2.5 times that of the `medium` (there are $\approx$ 110M and $\approx$ 42M parameters for the `base` and `medium` models, respectively).

In this comparison, we investigated if the substantially larger capacity of the `base` model that we pre-trained using standard MLM would make it a clearly better choice for fine-tuning compared to the considerably smaller medium-sized model that we trained with MLSM. Fine-tuning experiments using the `base` model was also conducted 10 times for each task. The average performances of the two pre-trained models of different capacities and pre-training protocol are included in Table 2.

We can observe that even though the `medium` model has approximately only 40% of the parameters of the `base` model, when trained with the proposed MLSM approach, it is still capable of performing close to the more than $2.5\times$ sized `base` model. Indeed, the `medium` model pre-trained with MLSM is capable of preserving 99.96% of the average performance of the `base` model trained with vanilla MLM. In contrast, the `medium` model pre-trained with vanilla MLM had an average performance of 0.7675 (see Table 1), which corresponds to only roughly 98.17% of the 0.7818 average performance of the `base` model.

Our computational budget prevented us from using student models $\mathcal{S}$ larger than `medium` size, however, we managed to pre-train models with larger teacher models $\mathcal{T}$ (as for those backpropagation did not have to be performed). The two larger $\mathcal{T}$ that we relied on were `bert-large-cased` and `roberta-large`. Following the observations in (Berend, 2020), we determined the contextualized latent semantic profiles based on the hidden representations from layer 21 of the large models.

The fine-tuning results of the fully-trained PLMs averaged over all evaluation scenarios are reported in Table 3, illustrating that without increasing the capacity of $\mathcal{S}$, we were not able to improve the fine-tuning scores of the PLMs. A likely cause for this is the capacity gap between $\mathcal{S}$ and $\mathcal{T}$ (nearly a factor of $\times 10$). It is worth mentioning that this phenomenon not only affected MLSM, but vanilla distillation as well. In fact, when using BERT large as $\mathcal{T}$, the performance of MLM-D and MLSM are

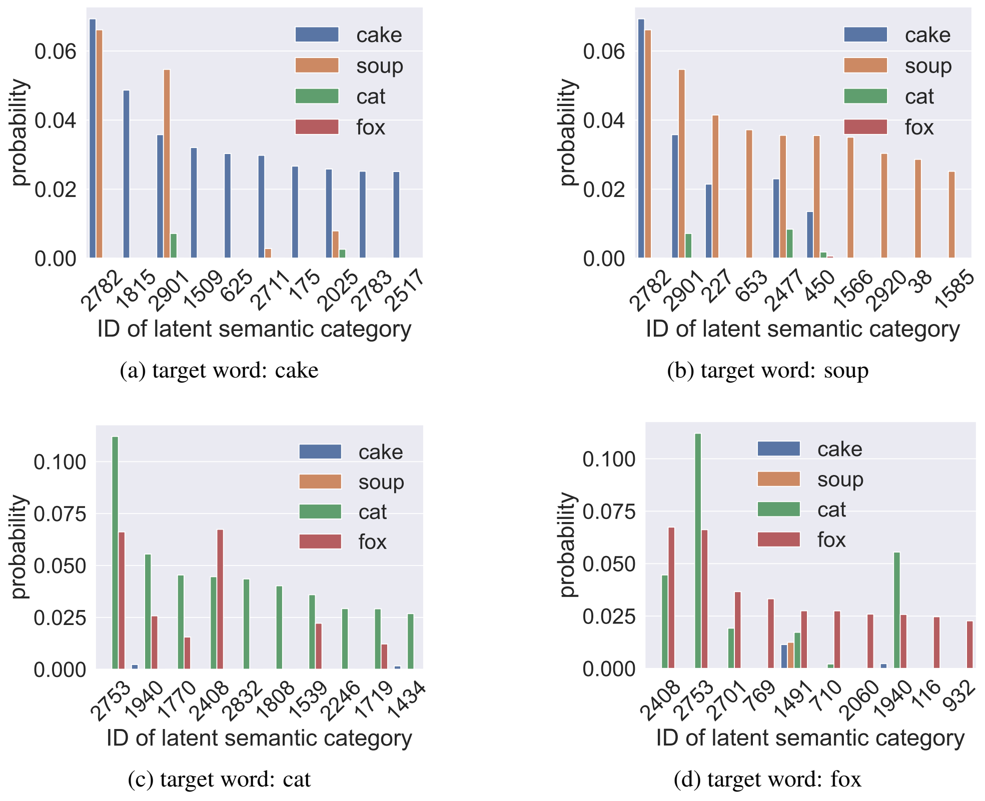

Figure 4: Illustration of the 10 most prominent latent semantic categories of the example words and the corresponding probabilities our approach assigned to them.

on par, whereas MLSM performs noticeably better compared to MLM-D when using RoBERTa large as $\mathcal{T}$. That is, using the same $\mathcal{T}$, MLSM always performed at least as good as vanilla distillation.

4.3 Illustrating latent semantic distributions

We next illustrate the latent semantic distributions qualitatively over a small example. Consider the following sentences, each containing a token of interest written in boldface:

(i) Alice is eating a **cake**.
(ii) Bob is cooking a **soup**.
(iii) The **cat** sits on the mat.
(iv) The **fox** is chasing a rabbit.

Figure 5 illustrates the pairwise cosine similarities between the token representations of the target words extracted from `bert-base-cased` and its sparse counterparts that we obtained by solving (2).

We can see in Figure 5a that the pairwise similarities are rather homogeneous for the dense representations, which can be explained by their anisotropy (Ethayarajh, 2019). Figure 5b, in contrast, reveals that the pairwise similarities of the inspected to-

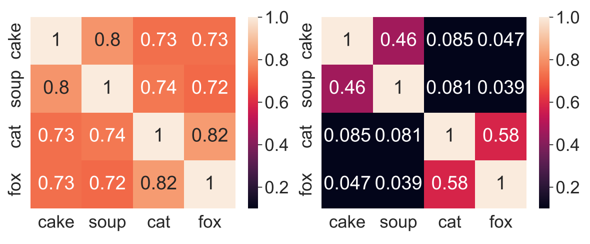

Figure 5: The pairwise cosine similarity between the words in the example sentences using the original dense contextualized representations and the sparse ones.

kens behave more plausibly for the sparse representations.

Figure 4 helps in better assessing the semantic distributions induced by our approach. Each subfigure considers one of the target words (disclosed in their captions), and contains their top-10 semantic categories (referenced by their indices) that the particular target word got assigned, ordered by their decreasing order of probability mass.

The subplots also include those probabilities that the other 3 non-target words received for the most dominant latent semantic categories of the target word. We can see in Figure 4a and Figure 4b that words referring to edible things (*cake* and *soup*) have an overlapping set of dominant semantic categories (with category 2782 being the most important for both of them), whereas their most prominent latent semantic categories overlap little (if any) with the other two example words that are related to animals. A similar tendency can be noticed between the animal related words in Figure 4c and Figure 4d.

5 Conclusion

In this paper, we proposed masked latent semantic modeling (MLSM), such a modification of the classical masked language modeling task, where the goal is changed from the prediction of the exact identity of the masked tokens to that of their latent semantic categories. We suggested a context-sensitive unsupervised approach for determining the latent semantic categories of tokens by performing sparse coding of their hidden representations from a pre-trained teacher model. The reliance on a teacher model makes our approach similar to model distillation in the sense that we can transfer the capabilities of a (larger) pre-trained model into a newly trained one. Comparison of the fine-tuning capabilities of the PLM pre-trained via classical model distillation and MLSM revealed a clear benefit towards the latter approach.

Our experiments also corroborate that MLSM pre-training behaves more sample efficient compared to other alternatives, as the fine-tuning performance of the pre-trained model at its 25% completeness level was capable of achieving better performances than any of the alternatively trained models at their end of pre-training stage (see Figure 2). More importantly, our experiments revealed that by relying on MLSM pre-training, it was possible to cram the fine-tuning capabilities of a PLM with $2.5\times$ parameter into a smaller one (see Table 2). Finally, in order to foster reproducibility of our proposed approach, we share all our code and pre-trained models at `https://github.com/szegedai/MLSM` and `https://huggingface.co/SzegedAI/bert-medium-mlsm`, respectively.

Acknowledgments

The research received support from the European Union project RRF-2.3.1-21-2022-00004 within the framework of the Artificial Intelligence National Laboratory. Additionally, we are grateful for the possibility to use ELKH Cloud (see Héder et al., 2022; https://science-cloud.hu/) which helped us in achieving the results published in this paper.

Risks and Limitations

Our pre-training relies on distributions of latent semantic properties of masked tokens that we determine in a purely unsupervised manner by performing sparse coding of the hidden states from an already pre-trained PLM. This property would make our approach in principle available to be used in pre-training PLMs for any language with an already pre-trained PLM and a pre-training corpus of raw, unannotated text available. Despite of this fact, we only considered English in our experiments, causing the potential risk of reinforcing the community bias in mainly focusing on the English language only.

The fact that our proposed approach requires an already pre-trained PLM can be deemed as a limitation. Hence for languages, where only a pre-training corpora exist, but no PLM has been pre-trained with vanilla MLM, a classical PLM needs to be pre-trained first, which increases the costs of performing MLSM. We should add, however, that once the MLSM pre-training ends, it has the same fine-tuning costs as a PLM pre-trained with traditional MLM.

It would be interesting to see if the proposed pre-training paradigm was applicable for autoregressive models as well. Currently we tested our proposed approach in autoencoder-style masked language models. Extending our work to autoregressive models is something we regard as a potential future work.

Finally, we considered the pre-training of `medium`-sized model, with the knowledge being distilled from larger models. This decision was driven by our computation budget, but it would be definitely instructive to see the effects of the same procedure employed for student models of larger capacity.

References

Vanda Balogh, Gábor Berend, Dimitrios I. Diochnos, and György Turán. 2020. Understanding the semantic content of sparse word embeddings using a commonsense knowledge base. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances* *in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7399–7406. AAAI Press.

Lisa Bauer, Yicheng Wang, and Mohit Bansal. 2018. Commonsense for generative multi-hop question answering tasks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4220–4230, Brussels, Belgium. Association for Computational Linguistics.

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. 2022. Combating the curse of multilinguality in cross-lingual WSD by aligning sparse contextualized word representations. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2459–2471, Seattle, United States. Association for Computational Linguistics.

Prajjwal Bhargava, Aleksandr Drozd, and Anna Rogers. 2021. Generalization in NLI: Ways (not) to go beyond simple heuristics. In *Proceedings of the Second Workshop on Insights from Negative Results in NLP*, pages 125–135, Online and Punta Cana, Dominican Republic. 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.

Pedro Colon-Hernandez, Catherine Havasi, Jason B. Alonso, Matthew Huggins, and Cynthia Breazeal. 2021. Combining pre-trained language models and structured knowledge. *CoRR*, abs/2101.12294.

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.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah A. Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *CoRR*, abs/2002.06305.

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

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65, Hong Kong, China. Association for Computational Linguistics.

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

William A. Gale, Kenneth W. Church, and David Yarowsky. 1992. A method for disambiguating word senses in a large corpus. *Computers and the Humanities*, 26(5):415–439.

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.

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.

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

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, Or Dagan, Ori Ram, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. 2020. SenseBERT: Driving some sense into BERT. In *Proceedings of the*

58th Annual Meeting of the Association for Computational Linguistics, pages 4656–4667, Online. Association for Computational Linguistics.

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, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 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.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-BERT: Enabling language representation with knowledge graph. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(03):2901–2908.

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*.

Yinquan Lu, Haonan Lu, Guirong Fu, and Qun Liu. 2021. KELM: Knowledge enhanced pre-trained language representations with message passing on hierarchical relational graphs. Anonymous preprint under review.

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.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 821–832, Melbourne, Australia. Association for Computational Linguistics.

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*.

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 43–54, Hong Kong, China. Association for Computational Linguistics.

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.

Sampo Pyysalo, Jenna Kanerva, Antti Virtanen, and Filip Ginter. 2021. WikiBERT models: Deep transfer learning for many languages. In *Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 1–10, Reykjavik, Iceland (Online). Linköping University Electronic Press, Sweden.

Delai Qiu, Yuanzhe Zhang, Xinwei Feng, Xiangwen Liao, Wenbin Jiang, Yajuan Lyu, Kang Liu, and Jun Zhao. 2019. Machine reading comprehension using structural knowledge graph-aware network. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5896–5901, Hong Kong, China. 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.

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.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 4444–4451. AAAI Press.

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.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: The impact of student initialization on knowledge distillation. *CoRR*, abs/1908.08962.

Leandro von Werra, Lewis Tunstall, Abhishek Thakur, Alexandra Sasha Luccioni, Tristan Thrush, Aleksandra Piktus, Felix Marty, Nazneen Rajani, Victor Mustar, Helen Ngo, Omar Sanseviero, Mario Sasko, Albert Villanova del Moral, Quentin Lhoest, Julien Chaumond, Margaret Mitchell, Alexander M. Rush, Thomas Wolf, and Douwe Kiela. 2022. Evaluate & evaluation on the hub: Better best practices for data and model measurements. *CoRR*, abs/2210.01970.

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.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

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

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, Mariama Drame, Quentin Lhoest, and Alexander Rush. 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.

An Yang, Quan Wang, Jing Liu, Kai Liu, Yajuan Lyu, Hua Wu, Qiaoqiao She, and Sujian Li. 2019. Enhancing pre-trained language representations with rich knowledge for machine reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2346–2357, Florence, Italy. Association for Computational Linguistics.

Zhi-Xiu Ye, Qian Chen, Wen Wang, and Zhen-Hua Ling. 2019. Align, mask and select: A simple method for incorporating commonsense knowledge into language representation models. *CoRR*, abs/1908.06725.

Zeyu Yun, Yubei Chen, Bruno Olshausen, and Yann LeCun. 2021. Transformer visualization via dictionary learning: contextualized embedding as a linear superposition of transformer factors. In *Proceedings of Deep Learning Inside Out (DeeLIO): The 2nd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 1–10, Online. Association for Computational Linguistics.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Both risks and limitation are discussed in the (unnumbered) section with the same title. This section is located after Section 5 (Conclusions).

☑ A2. Did you discuss any potential risks of your work?
Potential risks are discussed in the Risks and Limitations section as well. (to be found between Section 5 and the References).

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Introduction is Section 1 of the paper.

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

The datasets our experiments are based on are mentioned in Section 4.2, while the software stack is introduced at the beginning of Section 4.

☑ B1. Did you cite the creators of artifacts you used?
Section 4.2 mentions all the datasets used, and the software libraries are also cited at the beginning of Section 4.

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
We open source our source code. It is not mentioned in the paper itself, but our GitHub repository includes that the source code is released under the MIT License.

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
We used standard benchmark datasets for evaluation purposes. We believe that such a use complies with their intended use, and does not require a discussion.

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
We used a public Wikipedia dump for pre-training language models. Due to its encyclopedic and controlled nature, the above issues are atypical of that resource.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Information about the size of the pre-training corpus is provided in Section 4.1.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

C ☑ Did you run computational experiments?

Experiments are described across Section 4.

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
These aspects are covered at the beginning of Section 4 and additionally, in Section 4.2.4.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
For pre-training and fine-tuning, the relevant information are included in Section 4.1 and Section 4.2, respectively.

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Experimental results together with the descriptive statistics used are included in Section 4.2.

☒ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
These details are not included in the paper itself for conciseness, but the GitHub repository contains these information in the form of a requirements.txt file.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Not applicable. Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.