Hardness Masking via Auto-Regressive Language Model

Anonymous ACL submission

Abstract

Pre-trained masked language models have achieved tremendous success in natural language processing. Most of these methods rely on recovering randomly masked tokens, which is in general not as good as when tokens are masked based on how well the model can predict. However, it is costly for a large-scale model to self-identify tokens that it still struggles to predict. On the other hand, we observe that a smaller language model can often effectively find what a large model fails to learn. Inspired by this observation, we propose to leverage a compact bi-directional auto-regressive language model to dynamically discover tokens that a large language model has not learned well and guide its training via *hardness masking*. Comprehensive experiments demonstrate that our masking method can effectively boost the performance of pre-trained language models on general language understanding benchmarks.

1 Introduction

Language Modeling (LM) is one of the most fundamental natural language processing tasks. Recently, pre-trained models based on Masked Language Modeling (MLM) and large-scale unannotated corpus, e.g. BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), and T5 (Raffel et al., 2019), have achieved state-of-the-art performance on many NLP tasks. In MLM, the model aims to recover randomly masked input tokens. The masking can happen at token level (Devlin et al., 2018; Liu et al., 2019) or phrase level (Joshi et al., 2020; Raffel et al., 2019). However, most of these works sample tokens or phrases from a uniform distribution, which encourages models to learn more about frequent examples, but less on hard ones. As shown in (Levine et al., 2020), the model quality can significantly be increased if the model masks tokens such that they are harder to predict. One way to estimate the difficulty to recover a given token is to leverage the perplexity of prediction of this token when it's masked. However, it is computational expensive to have a large MLM to self-identify such tokens: in order to mask 15% of the tokens at a time, it takes 7 rounds to estimate the difficulty of all input tokens.

On the other hand, we find that it is often the case that a simple auto-regressive language model (ALM) and a large-scale MLM share a similar set of tokens that they find hard to predict. In other words, a small language model can quickly help to detect hard tokens for a large MLM. As an example, Figure 3 shows the predicted word perplexities by a small bi-directional ALM, where the darker color indicates a higher perplexity, i.e. tokens that are more difficult to recover. We find that the RoBERTa-large model (Liu et al., 2019) struggles to reconstruct these difficult tokens as well. If we randomly mask 15% of all tokens from a uniform distribution, the average perplexity of RoBERTa-large is 2.47. But this number jumps significantly to 35.43 if we selectively mask the same number of tokens based on the probabilities from auto-regressive language modeling. This result indicates that a small ALM can act as a good indicator to guide the training of a large MLM by providing an estimated perplexity.

Thus, we propose to use a compact bi-directional auto-regressive language model to estimate the difficulty of each token. Specifically, after collecting the average perplexities of this small LM from both directions, we normalize the values across the whole sequence and use these scores as the sampling distribution for masking. We apply this technique to further pre-train large language models based on MLM. We show that the proposed efficient masking technology can effectively improve the quality of pre-trained MLM. For example, on GLUE dataset, the average performance of RoBERTa-large under our hardness masking method is improved by 0.6%.

We summarize the main contributions of this

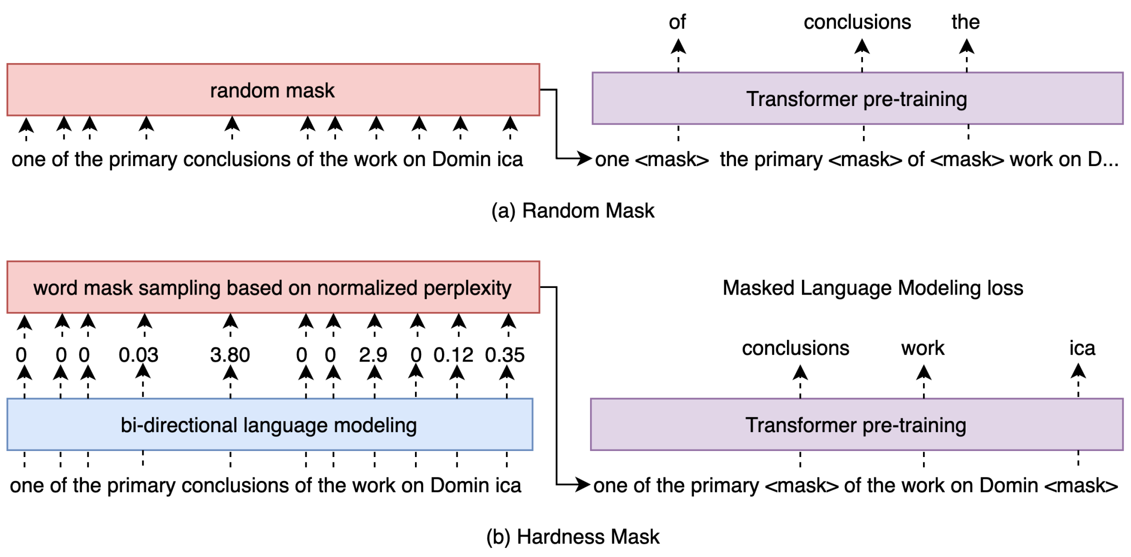
of
conclusions
the
random mask
Transformer pre-training
one of the primary conclusions of the work on Domin ica
one <mask> the primary <mask> of <mask> work on D...
(a) Random Mask
word mask sampling based on normalized perplexity
Masked Language Modeling loss
0 0 0 0.03 3.80 0 0 2.9 0 0.12 0.35
conclusions
work
ica
bi-directional language modeling
Transformer pre-training
one of the primary conclusions of the work on Domin ica
one of the primary <mask> of the work on Domin <mask>
(b) Hardness Mask

Figure 1: Masking strategies. (a) Random masking is the most widely adopted method. As it does not depend on the input text, the most frequent tokens, such as stopwords, are more likely to be sampled. (2) Hardness masking samples tokens based on their difficulty computed by a small bi-directional auto-regressive language model. We use the normalized perplexity at each position as the sampling distribution for masking.

paper as follows. (i) We find that a simple auto-regressive language model and a large-scale pre-trained masked language model such as RoBERTa often share a similar set of words that are difficult to predict. (ii) We design a hardness masking strategy to further pre-train RoBERTa to recover words detected by a simple auto-regressive language model, (iii) Our masking strategy significantly improves pre-trained language models on multiple tasks from the GLUE benchmark (Wang et al., 2018).

2 Method

In this section, we will introduce our hardness masking framework which combines auto-regressive language modeling (ALM) and masked language modeling (MLM), as shown in Figure 1.

2.1 Auto-Regressive Language Modeling

Auto-regressive language modeling learns the probability of next word occurrence based on all the previous words. Given a sequence X with n tokens, language modeling can predict the probability distributions of each token given all of tokens to its left:

$$\overrightarrow{P}(X) = \overrightarrow{\text{Transformer-LM}}(X), \qquad (1)$$

where $\overrightarrow{P}(X) \in \mathbb{R}^n$ are the word level perplexities. A word with a higher perplexity indicates that it is harder to be predicted by the language model.

We compute the right-to-left perplexity of each token in a similar way:

$$\overleftarrow{P}(X) = \overleftarrow{\text{Transformer-LM}}(X). \qquad (2)$$

We then average the perplexities from both directions for each token, and normalize the values over all the tokens in the sequence.

$$P(X) = \frac{\overleftarrow{P}(X) + \overrightarrow{P}(X)}{2} \Big/ \left\| \frac{\overleftarrow{P}(X) + \overrightarrow{P}(X)}{2} \right\|_1 . \qquad (3)$$

We argue that $P(X) \in \mathbb{R}^n$ is a token-level hardness distribution over the whole sequence, and it can reflect which words are harder to predict. We provide a visualization of $P(X)$ in Appendix A.

2.2 Hardness Masking for MLM Pre-training

We find a simple auto-regressive language model, e.g. 6-layer Transformer, can already give the exact correct next tokens prediction for half of the tokens in 512-length sequences. Thus, $P(X)$ can be effectively computed with a small ALM model. These findings make it possible to make a pipeline based method for masked language model (MLM) pre-training: First train model with ALM and use it to efficiently mask tokens for MLM training. The model trained with ALM can be treated as a prior knowledge for model pre-training.

Most existing MLM methods adopt uniform random sampling for masking tokens/phrases. This

	SST	MRPC	CoLA	STS	MNLI	QNLI	QQP	RTE	Avg
Development Set									
RoBERTa (base)(Liu et al., 2019)	94.8	90.2	63.6	91.2	87.6/-	92.8	91.9	78.7	
Random Mask (base)	95.3	90.2	62.8	90.9	88.1/87.8	93.1	91.9	79.4	86.6
Hardness Mask (base)	95.9	91.2	64.7	91.1	88.1/87.8	93.3	91.9	81.9	87.3
RoBERTa (large)(Liu et al., 2019)	96.4	90.9	68.0	92.4	90.2/90.2	94.7	92.2	86.6	89.1
Random Mask (large)	97.0	91.7	69.3	92.4	90.6/90.2	95.0	92.2	87.7	89.6
Hardness Mask (large)	**97.0**	**92.6**	**71.4**	**92.6**	**91.0/90.6**	**95.1**	**92.3**	**88.8**	**90.2**
Test Set									
PMI-Masking (base)(Levine et al., 2020)	94.7	87.7	57.4	87.7	86.6/85.8	93.1	89.5	72.9	83.9
Random Mask (base)	95.6	87.1	59.0	88.1	87.0/87.2	93.2	89.4	72.7	84.4
Hardness Mask (base)	95.8	87.0	60.3	88.6	87.8/87.3	93.4	89.5	75.4	85.0
Random Mask (large)	95.9	87.4	63.4	90.6	**90.3**/89.7	94.5	89.8	**84.7**	87.3
Hardness Mask (large)	**96.5**	**88.7**	**65.8**	**91.2**	90.1/**89.8**	**94.9**	**89.9**	83.7	**87.8**

Table 1: Experiment results on GLUE datasets. The results in the top half are the best performance on the dev sets of different tasks, and the bottom half are the corresponding results on test sets. Random-Mask is same as RoBERTa MLM training, but further train for more epochs to make a fair comparison with our method Hardness Mask.

strategy encourages models to learn more about frequently seen words, but less on hard examples. A wide experiment study (Joshi et al., 2020; Raffel et al., 2019) shows that the language model quality can significantly increase if the model chooses "hard" tokens to mask, i.e. the words it finds harder to predict.

We sample the indices of tokens to mask according to $P(X)$:

$$\text{Mask\_Index} = \text{Multi\_Sample}(\frac{P(X)+\alpha}{||P(X)+\alpha||_1}, M) \quad (4)$$

where $\alpha \geq 0$ is a scalar value for smoothing. When $\alpha = 0$, the tokens are sampled completely according to the perplexity $P(X)$; when $\alpha = \infty$, our strategy falls back to uniform random sampling. We sample M tokens to mask based on the smoothed probabilities.

In this paper, we hypothesis that even if a large model has been trained for a long time over a large corpus, it still cannot remember everything. And our method is to efficiently test the existing pre-training models and make the models focus on learning unseen knowledge. Thus, we will work on continue pre-training existing models, such as RoBERTa. This will also be cost and energy efficient.

3 Experiment

3.1 Experiment Setting

We are using same amount of data with RoBERTa for pre-training and evaluate models on GLUE benchmark (Wang et al., 2018). Our bi-directional auto-regressive language model is a Transformer with 6 layers. For model pre-training, We further pre-train RoBERTa for another 6 epochs on the 160g data with 10% warm-up steps.

3.2 Experiment Results

We report the best performance on dev sets after hyper-parameter search as shown in Table 1, and the best model will be evaluated on the hidden test sets.

Random Mask V.S. Hardness Mask To make a fair comparison with proposed "Hardness Mask" method, we apply "Random Mask" in the same pre-training pipeline, where the only difference is the masking strategy. Both methods are first initialized by RoBERTa checkpoints, and then further pre-train for several epochs. Based on the experiment results on development and test sets, our methods work for both base and large model structures.

PMI-Masking V.S. Hardness Mask We make a comparison with another strong baseline, PMI-Masking (Levine et al., 2020), which also makes use of prior knowledge to construct pre-training tasks. This knowledge is statistically computed by Pointwise Mutual Information (PMI). We can compare it with our "Hardness Mask (base)", and our method can achieve better performance on almost all the tasks, except MRPC dataset.

3.3 Hardness Analysis

ALM perplexity distribution First let us take a look at a distribution of word hardness, $P(x)$, as

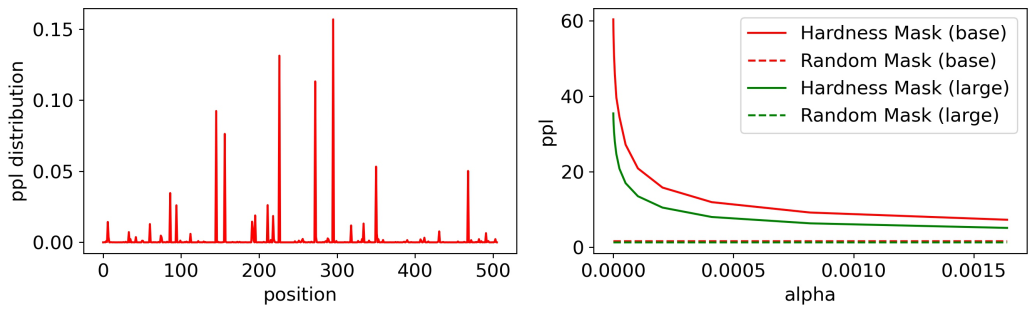

Figure 2: The left figure visualizes the distribution of $P(X)$. It is a case study on the token perplexities computed by auto-regressive language models. X-axis is the token positions. Y-axis is the hardness of the tokens over all the tokens at each position. The right figure is about MLM perplexities of RoBERTa-large given different degree of hardness defined by α. If α=0, it would be the hardest masking strategy, and if α is large enough, it would be same as random masking.

	SST	MRPC	CoLA	STS	MNLI	QNLI	QQP	RTE	Avg
Random Mask (α=∞)	95.3	90.2	62.8	90.9	88.1/87.8	93.1	91.9	79.4	86.6
Hardness Mask (α=0)	95.2	89.5	62.1	89.7	88.0/87.4	93.0	89.9	78.7	85.9
Hardness Mask (α=0.001)	95.9	91.2	64.7	91.1	88.1/87.8	93.3	91.9	81.9	87.3
Hardness Mask (α=0.005)	95.6	90.9	65.3	90.8	88.0/87.8	93.4	91.9	81.6	87.3

Table 2: GLUE dev fine-tuning performance with checkpoints pre-trained in different MLM tasks. The hardness of MLM task is tuned with α value. α=0 is the hardest MLM and α=∞ is the simplest MLM with random mask.

shown in the left part of Figure 2. We can see that a small faction of words are much harder to predict than others, which means ALM can efficiently pick up the most important tokens but not uniformly assign the weights.

Relation between ALM and MLM perplexities We use α in Eqn.(4) to define The hardness of different masking strategies. Based on the right side of Figure 2, we can see that when we set α to 0, the perplexity boosts to over ten times larger. With the α value increasing, some words that are easy to recover get more chance to be sampled. Thus the perplexities of both RoBERTa-base and RoBERTa-large decreases quite fast.

Hardness mask pre-training As shown in Table 2, we can see that when we tune the MLM task to be the hardest one with α=0, the model will get worse performance. According to our experiments in Table 2, selecting an appropriate pre-training task with certain hardness can lead to better fine-tunig performance.

4 Related Work

Mask language modeling has dominated the pre-training task. It achieves better performance on most downstream tasks than auto-regressive pre-trained language models (Lan et al., 2019). Later on, how to mask tokens for MLM has drawn much attention. The tokens are masked in sub-word level in earlier works (Devlin et al., 2018; Liu et al., 2019). Later on, whole word masking and span masking are proposed to further improve the pre-training quality (Joshi et al., 2020) Instead of uniformly selecting words to mask based on positions only, some simple statistical information can be treated as the prior knowledge which can be used to help the large-scale pre-training, such as PMI-Masking (Levine et al., 2020) and ELECTRA (Clark et al., 2020). Our work is under this direction.

5 Conclusions

In this paper, we propose an efficient hardness masking method for pre-training masked language models. We propose to use the word level perplexity predicted by a lightweight bi-directional auto-regressive language model to guide a large MLM to mask and recover difficult tokens. Experiments show that our method can effectively improve the quality of language models, e.g. it improves the average performance of RoBERTa-large by 0.6% on the GLUE benchmark.

References

Eneko Agirre. 2007. Lluıs marquez, and richard wicentowski, editors. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007). Association for Computational Linguistics, Prague, Czech Republic*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

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

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

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus. *urlhttp://Skylion007. github. io/OpenWebTextCorpus*.

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.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Yoav Levine, Barak Lenz, Opher Lieber, Omri Abend, Kevin Leyton-Brown, Moshe Tennenholtz, and Yoav Shoham. 2020. Pmi-masking: Principled masking of correlated spans. *arXiv preprint arXiv:2010.01825*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Sebastian Nagel. 2016. Cc-news.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Iyer Shankar, Dandekar Nikhil, and Csernai Kornl. 2016. First quora dataset release: Question pairs.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y 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.

Trieu H Trinh and Quoc V Le. 2018. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

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 R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

A Perplexity Visualization

We visualize the perplexity of each word from a bi-directional language model in Figure 3.

B Dataset

Both our language modeling and masked language modeling are trained on large-scale unlabeled text data. For a fair comparison, We are using same amount of data with RoBERTa for training, which consists of English Wikipedia, Book Corpus (Zhu et al., 2015), CommonCrawl News (Nagel, 2016), OpenWebText (Gokaslan and Cohen, 2019), and CommonCrawl-Stories (Trinh and Le, 2018). Wikipedia, Book Corpus, and OpenWebText have not widely adopted for pre-training,

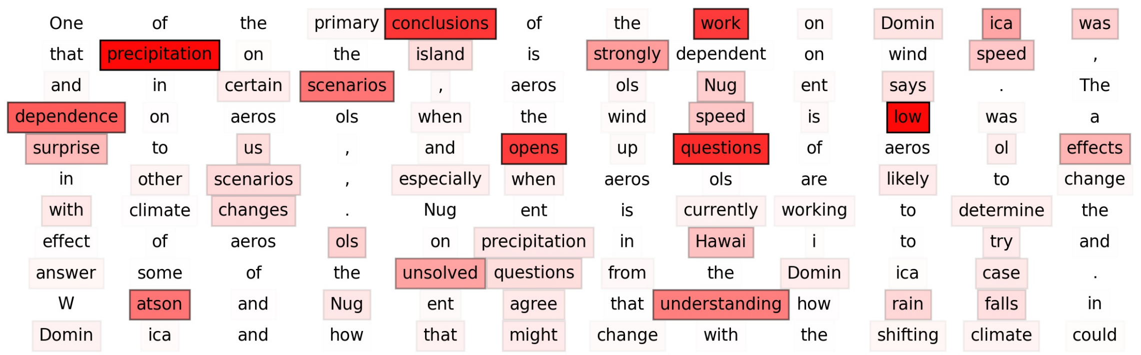

Figure 3: Perplexity of each word (inverse of probability of being predicted) in an example text given by a simple bi-directional language model. Words with a darker color has a higher perplexity, i.e. higher uncertainty under the model. We find that the RoBERTa-large model struggles to recover the words with darker color too: If we uniformly randomly mask 15% of all tokens, the average perplexity of RoBERTa-large model is 2.47. But if we randomly mask 15% of the words according to the color information, i.e. darker words are more likely to be masked, the average perplexity of RoBERTa-large significantly increases to 35.43.

and we follows RoBERTa (Liu et al., 2019) to collect data from CommonCrawl for pre-training. After all, we use the same datasets as RoBERTa to make a fair comparison with ours.

We evaluate our pre-trained models on the tasks from General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018), which consists of 9 datasets in total. The tasks can be divided into single sentence classification (regression) tasks, which includes Stanford Sentiment Treebank (SST) (Socher et al., 2013), Corpus of Linguistic Acceptability (CoLA) (Warstadt et al., 2019), Microsoft Research Paraphrase Corpus (MRPS) (Dolan and Brockett, 2005) and Semantic Textual Similarity Benchmark (STS) (Agirre, 2007); or sentence-pair classification tasks, which includes MultiGenre Natural Language Inference (MNLI-Matched and MNLI-Mismatched) (Williams et al., 2017), Question Natural Language Inference (QNLI) (Rajpurkar et al., 2016), Recognizing Textual Entailment (RTE) (Dagan et al., 2005) and Quora Question Pairs (QQP) (Shankar et al., 2016). Note that only the task STS is a regression problem. Based on official GLUE benchmark evaluation metrics, we use Spearman Correlation for STS, Matthews correlation for CoLA, and for rest of the tasks we report accuracy.

C Experiment Setting

Our hard masking strategy is based on a bi-directional auto-regressive language model, a Transformer with 6 layers and 512 dimensional hidden state. The language models in different directions are trained separately without sharing parameters. It is trained for 2 epochs over all of the 160g data. All the texts are cut into sequences with 512 tokens.

We use Byte-Pair Encoding tokenizer which is the same as RoBERTa (Liu et al., 2019). We set learning rate as 5e-04, batch size as 2560, dropout as 0.1, and use Adam (Kingma and Ba, 2014) as the optimizer. Besides word masking, we also adopt random word replacement (10% of the masked words) and original word replacement (10% of the masked words) from BERT and RoBERTa, which have been proven to be important for pre-training.

For model pre-training, We further pre-train RoBERTa for another 6 epochs on the 160g data with 10% warm-up steps. We run MLM pre-training by two mask strategies: random mask and proposed hardness mask. For hardness mask, we choose α in Eqn.(4) from $\{0, 0.001, 0.005\}$. The texts are also cut into sequences with 512 tokens. For experiments, We set learning rate as 1e-6, batch size as 2560, dropout as 0.1, and Adam optimizer.

For model finetuning on GLUE (Wang et al., 2018), we tune hyper-parameters on all tasks from learning rate {7e-06, 1e-05, 2e-05}, batch size {16, 32} and dropout {0.1, 0.2}. We report the best performance on dev sets of GLUE and also the corresponding performance on test sets.

Our pre-training and finetuning code is mainly based on fairseq[1], and all the experiments are done with 8 Nvidia V100 GPUs.

[1]https://github.com/pytorch/fairseq