Should You Mask 15% in Masked Language Modeling?

Alexander Wettig* **Tianyu Gao*** **Zexuan Zhong** **Danqi Chen**
Department of Computer Science, Princeton University
{awettig,tianyug,zzhong,danqic}@cs.princeton.edu

Abstract

Masked language models conventionally use a masking rate of 15% due to the belief that more masking would provide insufficient context to learn good representations, and less masking would make training too expensive. Surprisingly, we find that masking up to 40% of input tokens can outperform the 15% baseline, and even masking 80% can preserve most of the performance, as measured by fine-tuning on downstream tasks. Increasing the masking rates has two distinct effects, which we investigate through careful ablations: (1) A larger proportion of input tokens are corrupted, reducing the context size and creating a harder task, and (2) models perform more predictions, which benefits training. We observe that larger models in particular favor higher masking rates, as they have more capacity to perform the harder task. We also connect our findings to sophisticated masking schemes such as span masking and PMI masking, as well as BERT's curious 80-10-10 corruption strategy, and find that simple uniform masking with `[MASK]` replacements can be competitive at higher masking rates. Our results contribute to a better understanding of masked language modeling and point to new avenues for efficient pre-training.

1 Introduction

Pre-trained language models have transformed the landscape of natural language processing (Devlin et al., 2019; Liu et al., 2019; Raffel et al., 2020; Brown et al., 2020, *inter alia*). Large language models are trained on vast quantities of text data and acquire rich and versatile language representations. Compared to autoregressive models, which always predict the next token in a sequence, masked language models (MLMs) like BERT (Devlin et al., 2019) predict a masked subset of input tokens based on the remaining context and are more effective due to their bidirectional nature.

This comes at the cost of confining models to learn from only a small subset of tokens—usually 15% per sequence—which are masked out and used for predictions. The choice of 15% reflects the assumption that models cannot learn good representations when too much text is masked, and it has been ubiquitously used by BERT's successors (Liu et al., 2019; Joshi et al., 2020; Lan et al., 2020; He et al., 2021b). Meanwhile, only predicting on 15% of the sequence has been viewed as a limitation for efficient pre-training of MLMs (Clark et al., 2020).

In this work, we uncover a surprising finding that under an efficient pre-training recipe, we can mask 40-50% of input text and achieve better downstream performance than the default 15%. Table 1 shows examples of masking 15%, 40%, and 80%, as well their downstream task performance.[1] With 80% masking, even though most context is corrupted, the model still learns good pre-trained representations and preserves more than 95% of the performance on downstream tasks compared to 15% masking. This challenges common intuitions about masking rates and poses the question of how models benefit from high masking rates.

To address this, we propose to decompose the masking rate into two factors: the *corruption rate*—how much of the context is masked—and the *prediction rate*—how much of the tokens the model predicts on. In MLM, both the corruption rate and the prediction rate are the same as the masking rate. However, these two factors have opposing effects: while higher prediction rates generate more training signals and benefit the optimization, higher corruption rates make the learning problem more challenging given less context. To study the two factors independently, we design ablation experiments to separate corruption and prediction. We verify that models benefit from higher prediction rates and suffer from more corruption. Whether the benefit from more prediction can outshine the down-

*The first two authors contributed equally.

[1] We invite our readers to try to recover the masked tokens.

	Pre-training			Downstream		
m	Example	PPL	MNLI	QNLI	SQuAD[2]
80%	We [M] high [M] [M] [M] [M] [M] [M] models [M]	1141.4	80.8	87.9	86.2
40%	We study high [M] [M] rates [M] pre- [M] [M] models .	69.4	**84.5**	**91.6**	**89.8**
15%	We study high [M] ing rates [M] pre-training language models .	17.7	84.2	90.9	88.0

Table 1: Masked examples, validation perplexity, and downstream task development performance under different masking rates. All models are `large` models trained with the efficient pre-training recipe (§3). m: masking rate.

sides of more corruption decides whether models perform better with higher masking rates. As an example, we find that larger models—which possess greater capacity to handle more corruption—exhibit a higher optimal masking rate.

Motivated by our results, we consider higher masking rates in the context of more sophisticated masking schemes, such as span masking (Joshi et al., 2020; Raffel et al., 2020) or PMI masking (Levine et al., 2021). These methods are shown to outperform simple uniform masking when evaluated at 15% masking. However, we find that uniform masking is competitive to sophisticated masking baselines at their respective optimal masking rates. Our framework of prediction and corruption also sheds new light on BERT's practice of predicting based on the original or random tokens (the 80-10-10 strategy)—and we find that models usually perform better without this.

We conclude by discussing how our findings about high masking rates give rise to new avenues in efficient pre-training: Adopting higher masking rates in MLM leads to better performance, especially in a limited resource setting; working on directions to remove the mask tokens from the input or to disentangle corruption and prediction has the potential to further accelerate the pre-training.

In summary, our contributions are the following:

- We show that we can successfully train masked language models with surprisingly high masking rates. For example, a large model with the efficient pre-training recipe performs better with a masking rate of 40% than 15%.
- We propose to disentangle the masking rate as corruption rate and prediction rate, two opposite factors that affect the task difficulty and training signals, respectively. We use this framework to show that larger models have higher optimal masking rates and masking only using `[MASK]` tokens outperforms the 80-10-10 strategy.
- We demonstrate that at high masking rates uniform masking becomes competitive with more advanced masking schemes, specifically span masking and PMI masking.

2 Background

Pre-trained language models are at the center of natural language processing today. These Transformer-based models (Vaswani et al., 2017) are trained on large amounts of unlabeled data with language modeling objectives, and the resulting contextualized representations can be further fine-tuned on a wide range of downstream tasks.

Language modeling objectives for pre-training mainly fall into two categories: (1) Autoregressive language modeling, where the model is trained to predict the next token based on the previous context (Radford et al., 2018; Brown et al., 2020):

$$L(\mathcal{C}) = \mathop{\mathbb{E}}_{x \in \mathcal{C}} \left[\sum_{x_i \in x} \log p(x_i | x_1, x_2, ..., x_{i-1}) \right],$$

where $\mathcal{C}$ is a pre-training corpus and x is a sampled sequence from $\mathcal{C}$. (2) De-noising auto-encoding, where the model is trained to restore a corrupted input sequence. In particular, masked language models (MLMs) (Devlin et al., 2019; Liu et al., 2019) mask a subset of input tokens and predict them based on the remaining context:

$$L(\mathcal{C}) = \mathop{\mathbb{E}}_{x \in \mathcal{C}} \mathop{\mathbb{E}}_{\substack{\mathcal{M} \subset x \\ |\mathcal{M}| = m|x|}} \left[\sum_{x_i \in \mathcal{M}} \log p(x_i | \tilde{x}) \right], \quad (1)$$

where one masks m (masking rate, typically 15%) percentage of tokens from the original sentence x and predicts the masked tokens $\mathcal{M}$ given the corrupted context $\tilde{x}$ (the masked version of x).

Different masking strategies have been proposed to sample $\mathcal{M}$: Devlin et al. (2019) randomly

[2]For our SQuAD experiments, we continue training the models with 512-token sequences for 2,300 steps and report F1. See Appendix A for more details.

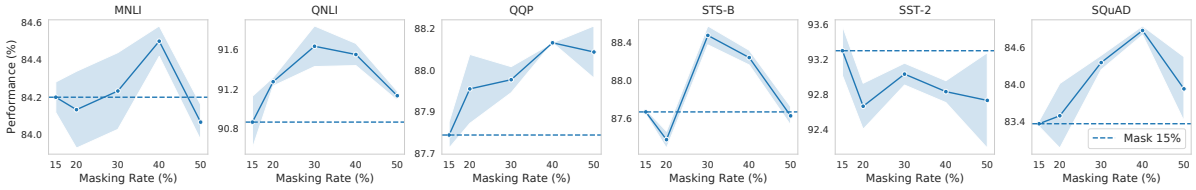

Figure 1: Impact of masking rates to `large` models with the efficient pre-training recipe. We see that on most tasks, higher masking rates outperform 15%; overall 40% is the optimal masking rate.

choose from the input tokens with a uniform distribution; Joshi et al. (2020) sample contiguous spans of text; Levine et al. (2021) sample words and spans with high Pointwise Mutual Information (PMI). These advanced sampling strategies prevent models from exploiting shallow local cues from uniform masking and lead to efficient pre-training.

MLMs can encode bidirectional context while autoregressive language models can only "look at the past", and thus MLMs are shown to be more effective at learning contextualized representations for downstream use (Devlin et al., 2019). On the other hand, MLMs suffer a significant computational cost because it only learns from 15% of the tokens per sequence, whereas autoregressive LMs predict every token in a sequence. In this work, we focus on MLMs and study the implications of masking rates for improving pre-training efficiency.

3 Experiment Setup

In this work, we build most of our experiments on a recent efficient pre-training recipe: the 24hBERT recipe from Izsak et al. (2021), by using which models can match `BERT-base` performance 6× faster. Since one major goal of this study is to improve training efficiency of MLM pre-training, we believe that our findings combined with this training recipe would lead to an overall better solution. We acknowledge that the optimal masking rate and strategies may depend on training recipes, hence we also explore training longer and the original RoBERTa recipe in Appendix B and find that our findings still generally hold.

Izsak et al. (2021) make the pre-training faster by using a `large` model, a larger learning rate (2e-3), a larger batch size (4,096), a shorter sequence length (128), and fewer training steps. We deviate from the 24hBERT with a few simple changes: (1) we adopt RoBERTa's tokenizer (Liu et al., 2019) for it performs better in our preliminary experiments; (2) instead of adopting BERT's 80-10-10% token corruption strategy, we simply replace all the masked tokens by `[MASK]` by default. We will discuss the impact of different corruption strategies detailedly in §5.3. We also do not use the next sentence prediction following previous work (Joshi et al., 2020; Liu et al., 2019; Izsak et al., 2021), which was shown to hurt performance. We show the detailed hyperparameters for pre-training as well as a comparison across different recipes (Devlin et al., 2019; Liu et al., 2019) in Appendix A.

We use downstream fine-tuning performance on GLUE (Wang et al., 2019) and SQuAD (Rajpurkar et al., 2016) as a measure of model performance. We fine-tune the models with 3 random seeds and grid search to obtain reliable results (more details are provided in Appendix A).

4 You Should Mask More than 15%

Devlin et al. (2019) choose the mysterious masking rate of 15%, for the belief that masking more leads to insufficient context to decode the tokens, and masking fewer makes the training inefficient. To understand how much one can mask in MLM and how masking rates affect the pre-trained model performance, we pre-trained a series of models with different masking rates, ranging from 15% to 80%. Figure 1 shows the downstream task performance change with respect to different masking rates (complete results are in Appendix C).

We see that surprisingly, masking as high as 50% can achieve comparable or even better results, compared to the default 15%-masking model. Masking 40% achieves overall the best downstream task performance (though the optimal masking rates for different downstream tasks vary). Our results suggest that only masking as little as 15% is not necessary for language model pre-training, and the optimal masking rate for a `large` model using the efficient pre-training recipe is as high as 40%. To further compare the 15% and the 40% masking rates, we show their GLUE test results in Table 2, and plot how the downstream task performance changes with different training steps in Figure 2.

	Time	MNLI-m/mm	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B	SQuAD
Masking 15%	24h	84.2/83.4	90.9	70.8	73.5	**92.8**	88.8	**51.8**	87.3	88.0
★ Masking 40%	24h	**84.7/84.0**	**91.3**	**70.9**	**75.5**	92.6	**89.8**	50.7	**87.6**	**89.8**
BERT`base`	140h	84.6/84.0	90.6	72.0	76.5	92.8	89.9	55.1	87.7	88.5
BERT`large`	632h	86.0/85.2	92.6	72.0	78.3	94.5	89.9	60.9	87.5	90.9

Table 2: The test results on the GLUE benchmark with `large` models, the efficient pre-training recipe, and with 15% or 40% masking rates.[3]

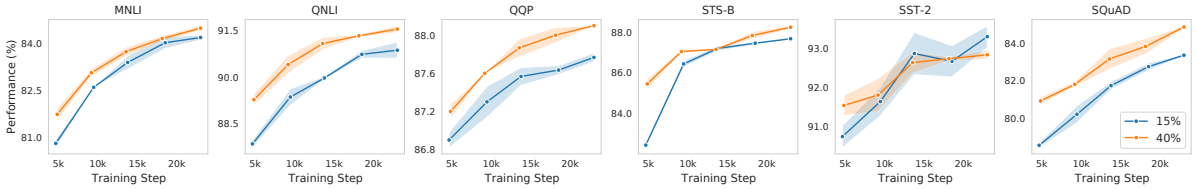

Figure 2: Downstream task development performance of `large` models trained with the efficient pre-training recipe, under masking rates of 15% and 40%.

Table 2 further verifies that masking 40% significantly outperforms 15%—with almost 2% improvement on SQuAD. We also see that 40% masking has a consistent advantage over 15% during the whole process of training in Figure 2.

To get a better understanding, we plot masked examples with different masking rates in Table 1, along with their validation perplexity and downstream task performance. We see that masking rates higher than 40% lead to a highly corrupted context that is even hard for human to reconstruct. When masked with 80%, most of the information from the context is erased and the perplexity becomes extremely high (>1,000). However, the fine-tuning performance of 40% outperforms 15% and even masking 80% can preserve more than 95% of the downstream task fine-tuning performance, compared to the 15%-masking baseline.

This changes our understanding about how masked language modeling works and how much context the model needs to learn good pre-training parameters. We hypothesize that even if it is impossible to reconstruct tokens because of the high masking rates, the models can still learn good contextual representations surprisingly.

5 Understanding the Masking Rate

In this section, we analyze how masking rates affect the pre-training process of MLM, through two distinct perspectives: task difficulty and optimization. Under this framework, we further discuss the relationship between masking rates, model sizes and different corruption strategies, as well as their impact on downstream task performance.

5.1 Masking as Corruption and Prediction

We identify that the masking rate m determines two import aspects of the pre-training problem: the *corruption rate* m_{corr} and the *prediction rate* m_{pred}. m_{corr} is the proportion of tokens that are erased from the input sequence—typically by substituting `[MASK]`. m_{pred} is the proportion of tokens that the models predicts, and each of those tokens contributes to the cross-entropy loss.

In Eq. (1), m_{corr} controls how much content is corrupted in $\tilde{x}$ compared to the original sentence x, and m_{pred} controls the number of predictions in the set $\mathcal{M}$. Usually, both the corruption and the prediction rates are tied to the masking rate i.e., $m_{\text{corr}} = m_{\text{pred}} = m$, but they may impact representation quality differently.

m_{corr} controls task difficulty. Masked language modeling attempts to learn a conditional probability distribution over the vocabulary given the corrupted context $p(\cdot|\tilde{x})$ during pre-training. If a larger proportion of the input is corrupted, a token prediction is conditioned on less context tokens, making predictions harder and more uncertain.

m_{pred} affects optimization. Predicting more means the model learns from more training signals, so higher prediction rates boost the model performance. From another perspective, each prediction

[4]For RTE, MRPC, and STS-B we fine-tune from the MNLI model following convention set by Phang et al. (2018). For SQuAD we take the same setting as Table 1. Original BERT numbers and the training time are from Izsak et al. (2021).

m_{corr}	m_{pred}	MNLI	QNLI	QQP	STS-B	SST-2
40%	40%	84.5	91.6	88.1	88.2	92.8
40%	20%	83.7↓	90.6↓	87.8↓	87.5↓	92.9
20%	20%	84.1↓	91.3↓	87.9↓	87.4↓	92.7↓
20%	40%	85.7↑	92.0↑	87.9↓	88.6↑	93.4↑
10%	40%	86.3↑	92.3↑	88.3↑	88.9↑	93.2↑
5%	40%	86.9↑	92.2↑	88.5↑	88.6↑	93.9↑

Table 3: Corruption vs prediction. We take 40% masking as the baseline, disentangle m_{corr} and m_{pred}, and manipulate each independently. The trend is clear: more prediction helps and more corruption hurts.

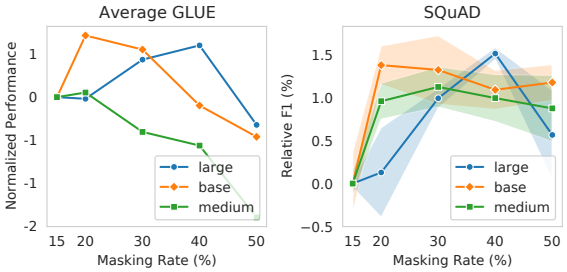

Figure 3: Impact of masking rates on different model sizes (`large>base>medium`).[4] We see that larger models favor larger optimal masking rates.

at each masked token leads to a loss gradient, which is averaged to optimize the weights of the model. Averaging across more predictions has a similar effect to increasing the batch size, which is proved to be beneficial for pre-training (Liu et al., 2019).

Experiments. In MLM pre-training, it is always the case that $m = m_{corrupt} = m_{predict}$. To study how m_{corr} and m_{pred} affect models' downstream performance independently, we use a simple ablation experiment to disentangle m_{corr} and m_{pred}:

1. If $m_{\text{pred}} < m_{\text{corr}}$, we mask m_{corr} of tokens and only make predictions on m_{pred} of the tokens. This can be easily implemented without any additional cost. For example, with $m_{\text{corr}} = 40\%$ and $m_{\text{pred}} = 20\%$, we mask 40% and only predict on a subset of 20% tokens.

2. If $m_{\text{pred}} > m_{\text{corr}}$, we duplicate each sequence $\lceil \frac{m_{\text{pred}}}{m_{\text{corr}}} \rceil$ times and mask disjoint sets of m_{corr} of the tokens in different sequences. For example, with $m_{\text{corr}} = 20\%$ and $m_{\text{pred}} = 40\%$, for each sentence, we do twice 20% masking on different tokens and predict on all the masked tokens—this leads to a 20% corruption but a 40% prediction on each sequence. Note that this ablation takes $\lceil \frac{m_{\text{pred}}}{m_{\text{corr}}} \rceil$ times longer because we do multiple passes on every sequence, and is not efficient in practice.

Table 3 shows the ablation results with disentangled m_{corr} and m_{pred}. We see that (1) fixing the m_{corr} as 40%, lowering the m_{pred} from 40% to 20% results in a consistent drop on downstream tasks, showing that more predictions lead to better performance; (2) fixing the m_{pred} as 40%, lowering the m_{corr} leads to consistently better performance, suggesting that lower corruption rates make the pre-training task easier to learn and are better for pre-training. Though we see that the performance gain by lowering m_{corr} from 10% to 5% is much smaller than that by lowering m_{corr} from 40% to 20%, suggesting a diminishing marginal return of reducing the corruption rate. (3) comparing $m_{\text{corr}} = 20\%, m_{\text{pred}} = 20\%$ and $m_{\text{corr}} = 40\%, m_{\text{pred}} = 40\%$, we see that the gain brought by more predictions transcends the drawback of more corruption, leading to better performance.

The ablation shows that when we tune the masking rate, we are tuning the corruption rate and the prediction rate together, which have antagonist effects. The final outcome is decided by which rate weighs more—the model benefits from higher masking rates if the hindrance brought by high corruption is surpassed by the advantage from predicting more. Many factors can affect the balance, and in the following we study the model sizes.

5.2 Large Models Favor More Masking

Given the harder task difficulty, we hypothesize that larger models benefit more from higher masking rates, as they have larger capacity to handle the harder task. Indeed, we find that *larger models possess higher optimal masking rates.* Figure 3 demonstrates the impact of masking rates on `large` (354M parameters), `base` (124M parameters), and `medium` (51M parameters) models. We see that under the efficient pre-training recipe, on average, `large` models take 40% as the optimal masking rate; `base` models and `medium` models take roughly 20% as the optimal masking rate. This clearly shows that models with larger capacity benefits more from higher masking rates.

As discussed in §5.1, if the downside from high corruption is smaller than the benefit from more

[4] For each task and each model size, *normalized performance* is calculated by $\frac{x - x_{15\%}}{\sigma}$ where $x_{15\%}$ is the performance of 15% masking rate and σ is the standard deviation across all masking rates. *Relative F1* is the F1 score subtracted by the 15% F1.

prediction, higher masking rates can lead to better performance. We hypothesize that larger models with more capacity can "handle" the harder tasks better and suffer less from the high corruption rates, and thus have a higher optimal masking rate.

This aligns well with our goal of training efficiency: Li et al. (2020) suggest it's more efficient to train larger models for fewer steps, as opposed to training smaller models for longer, since computational costs are offset by faster convergence.

5.3 Demystifying the 80-10-10 Rule

Devlin et al. (2019) suggest that it is beneficial to replace 10% of `[MASK]` tokens with the original token (keeping the word unchanged) and 10% with random tokens (substituting a random word). Since then, the 80-10-10 rule has been widely adopted in almost all the MLM pre-training work (Liu et al., 2019; Joshi et al., 2020; He et al., 2021b). The motivation is that masking tokens create a mismatch between pre-training and downstream fine-tuning, and using original or random tokens as an alternative to `[MASK]` may mitigate the gap. Based on this reasoning, masking even more of the context (e.g., masking 40%) should further increase the discrepancy, and yet we observe greater performance in downstream tasks. This begs the question whether we need the 80-10-10 strategy at all. First, we revisit the two kinds of mask replacements in the 80-10-10 rule and relate them to our framework of corruption and prediction.

Same token predictions. Predicting the same token is a very easy task—the model can simply copy the input to the output. The loss from same token predictions is very small and this objective should be regarded as an auxiliary regularization, which ensures that token information is propagated from the embeddings to the final layer. Thus, same token predictions should neither count towards the corruption nor to the prediction rate—they do not corrupt the input and contribute little to learning.

Random token corruptions. Replacing with random tokens contribute to corruption and prediction rate, as the input is corrupted and the prediction task is non-trivial. In fact, we find that the loss is slightly higher on random tokens compared to `[MASK]`, as (1) the model needs to decide for all tokens whether the information at the input is from a corruption or not, and (2) predictions need to be invariant to large changes in the input embeddings.

	MNLI	QNLI	QQP	STS-B	SST-2
40% mask	84.5	91.6	88.1	88.2	92.9
+ 5% same	84.2↓	91.0↓	87.8↓	88.0↓	93.3↑
w/ 5% rand	84.5	91.3↓	87.9↓	87.7↓	92.6↓
w/ 80-10-10	84.3↓	91.2↓	87.9↓	87.8↓	93.0↑

Table 4: Impact of substituting masks with random/same tokens. "+5% same": do extra 5% same token predictions. "w/ 5% rand": use mask for 35% mask tokens and random tokens for 5% . "w/ 80-10-10": for the 40% masked tokens, 10% are same token predictions and 10% are random token corruptions.

Ablation experiments. We adopt the $m = 40\%$ model using only `[MASK]` replacements as the baseline, on top of which we add three models:

1. "+ 5% same": we mask 40% of tokens but predict on 45% of tokens. As discussed above, adding same token predictions does not change m_{corr} or m_{pred}.

2. "w/ 5% random": we mask 35% of tokens and randomly replace another 5% of tokens, predicting on 40% in total.

3. "80-10-10": the BERT recipe where among all masked tokens, 80% are replaced with `[MASK]`, 10% are replaced by the original token, and 10% are replaced by random tokens. Note that for this model, $m_{\text{corr}} = m_{\text{pred}} = 36\%$ only due to same token predictions.

Our results are shown in Table 4. We observe that same token predictions and random token corruptions deteriorate performance on most downstream tasks. The 80-10-10 rule performs worse than simply using all `[MASK]`. This suggests that in the fine-tuning paradigm, the model can quickly adapt to full, uncorrupted sentences, regardless of the use of alternative corruption strategies. Given our results, we suggest to use only `[MASK]` for MLM pre-training.

Information flow. To visualize the effect of these corruption strategies, we follow Voita et al. (2019)'s analysis of measuring mutual information between an input token and its intermediate representations. Figure 4 shows that each model initially loses some information about the source token while acquiring information from the surrounding context. Using same token predictions during pre-training leads to a "reconstruction" stage in the last few layers, as observed by Voita et al. (2019),

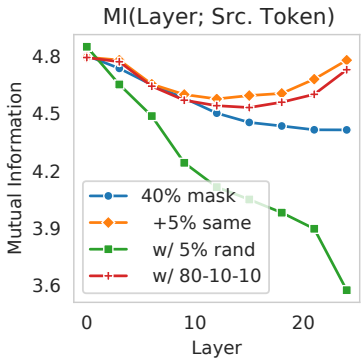

Figure 4: Mutual information between an input token and its intermediate representations for four different corruption strategies. See Table 4 for details on models.

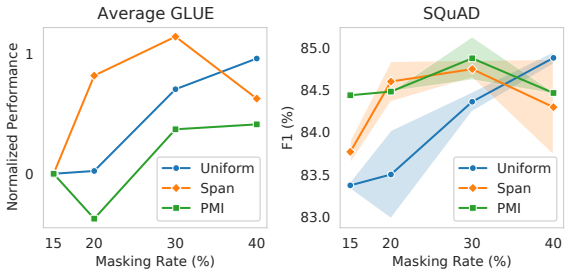

Figure 5: Performance of different masking strategies trained with different masking rates (efficient pre-training recipe, `large` models).

whereby information about the source token is restored from the context. However, this second stage is not present when same token predictions tokens are ablated: the `[MASK]`-only baseline propagates contextual features only—and no reconstruction occurs. One consequence is that information about the input tokens can be more easily extracted when pre-training with same token predictions.

6 Uniform Masking is Competitive at Higher Masking Rates

Devlin et al. (2019) and Liu et al. (2019) use uniform sampling for selecting positions to mask. Subsequent work showed that adopting more sophisticated masking strategies—such as span masking or PMI masking—can outperform uniform masking on a range of downstream tasks (Joshi et al., 2020; Raffel et al., 2020; Levine et al., 2021). The argument for adopting advanced masking is that uniform masking enables models to minimize the objective by using shallow local cues (Levine et al., 2021). An example is given by "`[MASK]` Kong": the model can easily predict "Hong" without using more context. However, all the previous studies used a fixed 15% masking rate regardless of masking strategies, which poses the question of whether the conclusions still hold with more masking.

To understand the interplay between masking rates and masking strategies, we experiment with multiple masking strategies under different masking rates, and find that uniform sampling—when at its optimal masking rate—performs better than or on par with more sophisticated masking strategies.

Figure 5 shows the results of uniform masking, T5-style span masking (Raffel et al., 2020), and PMI masking (Levine et al., 2021) under masking rates from 15% to 40%. We see that (1) for all masking strategies, the optimal masking rates are higher than 15%; (2) the optimal masking rates for span masking and PMI masking are lower than that of uniform masking; (3) when all strategies adopting the optimal masking rates, the uniform masking achieves comparable or even better results than the advanced strategies.

To understand the relation between higher masking rates and advanced masking strategies, we show in the following that more uniform masking essentially increases the chance of masking highly-correlated tokens, which reduces trivial mask tokens and potentially forces the model to learn more robustly (Levine et al., 2021). We note that even for uniform masking, higher masking rates should increase the chance of "accidentally" covering an entire PMI token span. By sampling masks over the corpus, we compute this probability in Figure 6, and find an 8-fold increase in the odds when raising the masking rate from 15% to 40%. Similarly, higher masking rates make the masked tokens form longer spans. It gives us an example of how the increased masking rate can have a similar effect to the advanced masking strategies and induce learning better representations.

7 Discussion and Future Work

Masking rates in other language models. In this paper, we study the masking rates of MLMs. Besides MLMs, there are other pre-training schemes that are widely adopted on NLP tasks, namely autoregressive language models (Radford et al., 2018; Brown et al., 2020) and sequence-to-sequence language models (Raffel et al., 2020; Lewis et al., 2020). Similarly, sequence-to-sequence models corrupt text with a certain masking rate and predict the masked text in an autoregressive manner. T5 (Raffel et al., 2020) also adopts a 15% masking

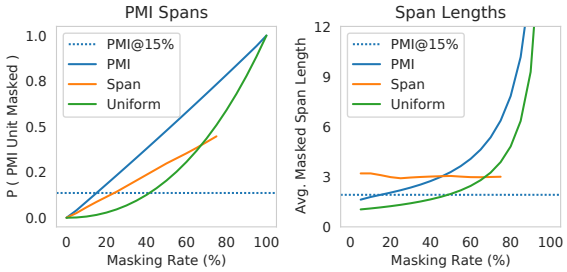

Figure 6: Higher masking rates increase the probability that an entire PMI span is masked (left) under different masking schemes (uniform, T5-style span masking and PMI). Uniform masking with a 40% masking rate masks as many PMI spans as regular PMI masking at 15% masking. Masks form longer spans for higher masking rates in uniform sampling, while the average span length was fixed at 3 for T5-style span masking.

rate and we plan to extend our study to text-to-text models and study the interplay between masking rates and different types of decoders.

We note that ELECTRA (Clark et al., 2020) is of particular interest to our study. ELECTRA uses a smaller MLM to fill in 15% of the blanks and trains a model to distinguish whether a token was generated by the MLM or not. Despite the complicated training procedure, the main motivation of ELECTRA is to improve the training efficiency by predicting on 100% of tokens. Interestingly, we find that the corruption rate becomes very low towards the end of training—the average corruption rate is roughly only 7% for ELECTRA, but the replacements are "hard" negatives generated by the small MLM. We leave the study of its inner workings and its connection to corruption and prediction rates as future work.

Processing context and masks separately. We have shown that with high masking rates, models benefit from more predictions, and learn better representations. Given that the model can be trained with high masking rates (e.g., 40-50%), we expect that it opens up new horizons for speeding up MLM pre-training. For example, in an encoder-decoder architecture, it is possible to encode only unmasked tokens and use a small decoder to recover corrupted tokens. This can significantly reduce the training cost due to the shorter input to the encoder (i.e., we can half the input length if the masking rate is 50%). A similar approach has been explored in masked auto-encoders in computer vision (He et al., 2021a), where 75% of the input patches are masked and removed from the encoder input to achieve a 4.1× speedup. We expect similar efficiency improvements on MLM and leave it as future work.

Disentangling corruption and prediction. Models perform better when trained with lower corruption rates and higher prediction rates. However, in standard MLM, those two factors are always tied to the masking rate. Methods which can encode a sequence once and then efficiently predict many small disjoint sets of masks could substantially accelerate masked language modeling pre-training.

8 Conclusion

In this paper, we conduct a comprehensive study on the masking rates of masked language models and discover that 40% masking consistently outperforms 15%—the conventional masking rate—on downstream tasks. We gain a better understanding of masking rates by disentangling them as a corruption rate and a prediction rate, and show that larger models can benefit more from the higher masking rates. We also demonstrate that the 80-10-10 rule is largely not needed and that simple uniform masking is competitive at higher masking rates with sophisticated masking schemes like span masking. Based on our findings, we discuss the future directions of efficient MLM pre-training enabled by the higher masking rates.

Acknowledgements

We thank Sadhika Malladi and the members of the Princeton NLP group for helpful discussion and valuable feedback. Alexander Wettig is supported by a Graduate Fellowship at Princeton University. This work is also supported by a Google Research Scholar Award.

References

Roy Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second PASCAL recognising textual entailment challenge.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth PASCAL recognizing textual entailment challenge. In *TAC*.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*.

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 *the 11th International Workshop on Semantic Evaluation (SemEval-2017)*.

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 (ICLR)*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL recognising textual entailment challenge. In *the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional Transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

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

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2021a. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*.

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

Peter Izsak, Moshe Berchansky, and Omer Levy. 2021. How to train BERT with an academic budget. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 10644–10652.

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 of Computational Linguistics (TACL)*, 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 (ICLR)*.

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 (ICLR)*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Association for Computational Linguistics (ACL)*, pages 7871–7880.

Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joey Gonzalez. 2020. Train big, then compress: Rethinking model size for efficient training and inference of transformers. In *International Conference on Machine Learning (ICML)*, pages 5958–5968.

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*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Technical report, OpenAI.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text Transformer. *The Journal of Machine Learning Research (JMLR)*, 21(140).

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 3505–3506.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

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 *Empirical Methods in Natural Language Processing (EMNLP)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems (NIPS)*, 30.

Elena Voita, Rico Sennrich, and Ivan Titov. 2019. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. In *Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4396–4406.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations (ICLR)*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association of Computational Linguistics (TACL)*, 7.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

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 Experiment Setup

A.1 Pre-training

We implement our pre-training work based on `fairseq` (Ott et al., 2019). To further speed up pre-training, we integrate the `DeepSpeed` (Rasley et al., 2020) Transformer kernel for speedup.

We keep the other setting the same as the 24hBERT (Izsak et al., 2021), except that we use the RoBERTa tokenizer (Liu et al., 2019) and we do not adopt the 80-10-10 rule. We train our model on the English Wikipedia and BookCorpus (Zhu et al., 2015). We want to emphasize that using pre-layernorm (Shoeybi et al., 2019) is essential for the high learning rate in Izsak et al. (2021) to work. The hyperparameters for the efficient pre-training recipe are shown in Table 5.

Hyperparameter	Efficient pre-training recipe
Peak learning rate	2e-3
Warmup proportion	6%
Batch size	4,096
Training steps	23,000
Sequence length	128
Architecture	`large`

Table 5: Our pre-training hyperparameter settings.

A.2 Downstream Task Evaluation

We fine-tune our model on the GLUE benchmark (Wang et al., 2019), including SST-2 (Socher et al., 2013), CoLA (Warstadt et al., 2019), MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016), RTE (Dagan et al., 2005; Bar Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009), MRPC (Dolan and Brockett, 2005), QQP[5] and STS-B (Cer et al., 2017), and the SQuAD (Rajpurkar et al., 2016) dataset. For each dataset we run three random seeds and average the results. We apply grid search for the GLUE datasets, as shown in Table 6. For SQuAD, we use a learning rate of 1e-4, a batch size of 16, and train for 2 epochs. For both GLUE and SQuAD we use a linear scheduling for learning rates.

For all the results in the paper, we report accuracy for MNLI, QNLI, RTE, SST-2; we report F1 score for QQP, MRPC, and SQuAD; we report Matthew's correlation for CoLA and Spearman's correlation for STS-B.

For the SQuAD results in Table 1 and Table 2, we further train the models for 2300 steps (10% of the training) with a sequence length of 512, a learning rate of 5e-4, and a warmup rate of 10%. For other tables and figures, we present the SQuAD results without further pre-training, and the absolute numbers are lower because of the short pre-training sequence length. For some of the figures in the paper, we only show the results of MNLI, QNLI, QQP, STS-B, SST-2, and SQuAD due to limited space. Those tasks are selected because they have larger training set and the results are more reliable. We always show the development results except in Table 2, where we report the test numbers for GLUE tasks.

[5] https://www.quora.com/q/quoradata/

Hyperparameter	MNLI, QNLI, QQP
Peak learning rate	{5e-5, 8e-5}
Batch size	32
Max epochs	{3, 5}
	RTE, SST-2, MRPC, CoLA, STS-B
Peak learning rate	{1e-5, 3e-5, 5e-5, 8e-5}
Batch size	{16, 32}
Max epochs	{3, 5, 10}

Table 6: Grid search hyperparameters for GLUE tasks.

B Train Longer

To see that how the different masking rates perform with longer training, we modify the efficient pre-training recipe for longer steps to match the RoBERTa recipe. Since the final RoBERTa models use more training data, we refer to RoBERTa's ablation in Table 3 for adjusting the training steps. Table 8 shows the hyperparameters for the longer training, as well as a comparison to RoBERTa's recipe. The major difference is that we train with much larger learning rate and only a sequence length of 128.

We train the models with 15% and 40% masking rates longer and evaluate them on downstream tasks. Figure 7 shows the results. We see that on most of the tasks, the trend that 40% is better than 15% still holds, though the 40% has a larger advantage when the training steps are limited.

We also train the model using the original RoBERTa recipe and present the results in Table 7. We see that (1) on most tasks 40% achieves comparable or better results compared to 15%; (2) our "train longer" results, which uses shorter sequences and larger learning rates, are comparable to the original RoBERTa recipe results.

	MNLI-m/mm	QNLI	RTE	QQP	MRPC	STS-B	SST-2	MR	CoLA	SQuAD
Original RoBERTa recipe										
Masking 15%	87.40/87.23	93.04	67.53	88.43	80.80	90.05	94.13	89.73	59.80	90.72
Masking 40%	87.30/87.03	92.90	67.63	88.83	63.90	87.94	94.10	89.27	56.07	91.23
Train longer with the efficient pre-training recipe										
Masking 15%	87.47/87.02	92.95	69.93	88.40	82.50	88.89	94.07	89.60	61.00	87.29
Masking 40%	86.63/86.83	93.13	68.87	88.40	79.50	89.60	94.67	89.70	61.23	87.16

Table 7: Development set results with 15% masking vs 40% masking using the recipe from Liu et al. (2019), Table 3, and the efficient pre-training recipe (but longer).

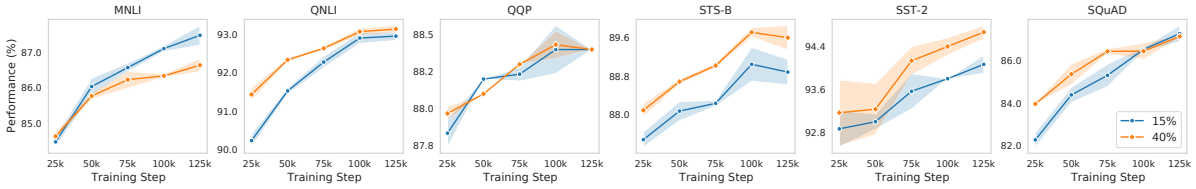

Figure 7: 15% vs 40% masking rates with `large` models and the efficient pre-training recipe but trained longer.

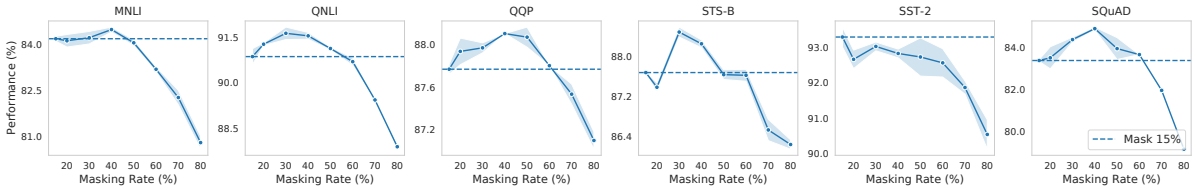

Figure 8: Impact of masking rates (ranging from 15% to 80%) on the `large` models, trained with the efficient pre-training recipe.

Hyperparameter	Train longer	RoBERTa
Peak learning rate	2e-3	7e-4
Warmup proportion	6%	6%
Batch size	4,096	2,048
Training steps	125,000	125,000
Sequence length	128	512

Table 8: Our training longer recipe.

C More Masking Rates

In Figure 8 we show the impact of different masking rates, from 15% to 80%. We see that even at a masking rate as high as 80%, the fine-tuning performance on downstream tasks are mostly preserved compared to 15%. Overall, 40% is the optimal masking rates for the `large` model.

D Model Details

We show the configuration comparison of different model sizes in Table 9.

	`medium`	`base`	`large`
#Layers	8	12	24
#Attention heads	8	12	16
Hidden size	512	768	1024

Table 9: Configurations of different model sizes.