Frustratingly Simple Pretraining Alternatives to Masked Language Modeling

Atsuki Yamaguchi[1]*, **George Chrysostomou**[2], **Katerina Margatina**[2] and **Nikolaos Aletras**[2]
[1]Research and Development Group, Hitachi, Ltd., Japan
[2]Department of Computer Science, University of Sheffield, United Kingdom
[1]`atsuki.yamaguchi1@gmail.com`
[2]`{gchrysostomou1, k.margatina, n.aletras}@sheffield.ac.uk`

Abstract

Masked language modeling (MLM), a self-supervised pretraining objective, is widely used in natural language processing for learning text representations. MLM trains a model to predict a random sample of input tokens that have been replaced by a `[MASK]` placeholder in a multi-class setting over the entire vocabulary. When pretraining, it is common to use alongside MLM other auxiliary objectives on the token or sequence level to improve downstream performance (e.g. next sentence prediction). However, no previous work so far has attempted in examining whether other simpler linguistically intuitive or not objectives can be used standalone as main pretraining objectives. In this paper, we explore five simple pretraining objectives based on token-level classification tasks as replacements of MLM. Empirical results on GLUE and SQUAD show that our proposed methods achieve comparable or better performance to MLM using a BERT-BASE architecture. We further validate our methods using smaller models, showing that pretraining a model with 41% of the BERT-BASE's parameters, BERT-MEDIUM results in only a 1% drop in GLUE scores with our best objective.[1]

1 Introduction

Masked Language Modeling (MLM) pretraining (Devlin et al., 2019; Liu et al., 2019; Lan et al., 2020; Wang et al., 2020) is widely used in natural language processing (NLP) for self-supervised learning of text representations. MLM trains a model (typically a neural network) to predict a particular token that has been replaced with a `[MASK]` placeholder given its surrounding context. Devlin et al. (2019) first proposed MLM with an additional next sentence prediction (NSP) task (i.e. predicting whether two segments appear consecutively in the original text) to train BERT.

Recently several studies have extended MLM, by masking a contiguous segment of the input instead of treating each token independently (Song et al., 2019; Sun et al., 2020; Joshi et al., 2020). Yang et al. (2019) reformulated MLM in XLNET, to mask out attention weights rather than input tokens, such that the input sequence is auto-regressively generated in a random order. ELECTRIC (Clark et al., 2020a) addressed the expensive softmax issue of MLM using a binary classification task, where the task is to distinguish between words sampled from the original data distribution and a noise distribution, using noise-contrastive estimation. In a different direction, previous work has also developed methods to complement MLM for improving text representation learning. Aroca-Ouellette and Rudzicz (2020) have explored sentence and token-level auxiliary pretraining objectives, showing improvements over NSP. ALBERT (Lan et al., 2020) complemented MLM with a similar task that predicts whether two sentences are in correct order or swapped. ELECTRA (Clark et al., 2020b) introduced a two-stage token-level prediction task; using a MLM generator to replace input tokens and subsequently a discriminator trying to predict whether a token has been replaced or not.

Despite these advances, simpler linguistically motivated or not auxiliary objective tasks acting as primary pre-training objectives substituting completely MLM have not been explored. Motivated by this, we propose five frustratingly simple pretraining tasks, showing that they result into models that perform competitively to MLM when pretrained for the same duration (e.g. five days) and fine-tuned in downstream tasks in GLUE (Wang et al., 2019) and SQUAD (Rajpurkar et al., 2016) benchmarks.

Contributions: (1) To the best of our knowledge, this study is the first to investigate whether linguistically and non-linguistically intuitive tasks can effectively be used for pretraining (§2). (2) We empirically demonstrate that our proposed objec-

*Work was done while at the University of Sheffield.

[1]Our code is publicly available here: `https://github.com/gucci-j/light-transformer-emnlp2021`

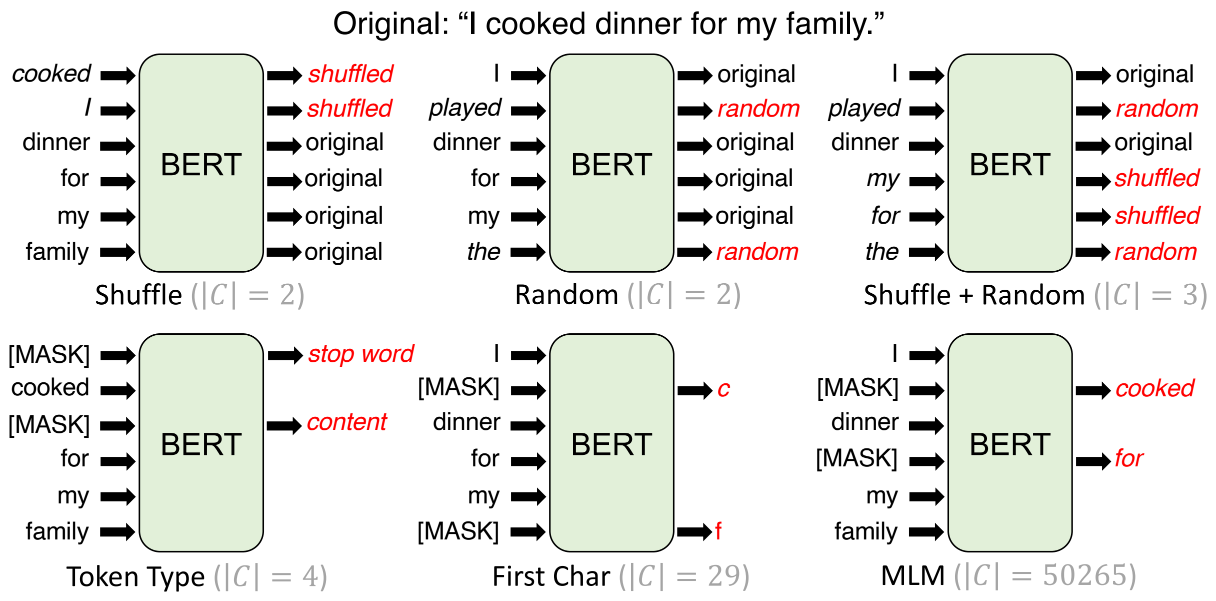

Figure 1: Overview of our five frustratingly simple pretraining tasks along with a comparison to MLM. $|C|$ denotes the number of classes for each task.

tives are often computationally cheaper and result in better or comparable performance to MLM across different sized models (§4).

2 Pretraining Tasks

Our methodology is based on two main hypotheses: (1) effective pretraining should be possible with standalone token-level prediction methods that are linguistically intuitive (e.g. predicting whether a token has been shuffled or not should help a model to learn semantic and syntactic relations between words in a sequence); and (2) the deep architecture of transformer models should allow them to learn associations between input tokens even if the pretraining objective is not linguistically intuitive (e.g. predicting the first character of a masked token should not matter for the model to learn that 'cat' and 'sat' usually appear in the same context). Figure 1 illustrates our five linguistically and non-linguistically intuitive pretraining tasks with a comparison to MLM.

Shuffled Word Detection (SHUFFLE): Motivated by the success of ELECTRA, our first pretraining objective is a token-level binary classification task, consisting of identifying whether a token in the input sequence has been shuffled or not. For each sample, we randomly shuffle 15% of the tokens. This task is trained with the *token-level* binary cross-entropy loss averaged over all input tokens (i.e. shuffled and original). The major difference between ours and ELECTRA is that we do not rely on MLM to replace tokens. Our intuition is that a model can acquire both syntactic and semantic knowledge by distinguishing shuffled tokens in context.

Random Word Detection (RANDOM): We now consider replacing tokens with out-of-sequence tokens. For this purpose we propose RANDOM, a pretraining objective which replaces 15% of tokens with random ones from the vocabulary. Similar to shuffling tokens in the input, we expect that replacing a token in the input with a random word from the vocabulary "forces" the model to acquire both syntactic and semantic knowledge from the context to base its decision on whether it has been replaced or not.

Manipulated Word Detection (SHUFFLE + RANDOM): For our third pretraining objective, we seek to increase the task difficulty and subsequently aim to improve the text representations learned by the model. We therefore propose an extension of SHUFFLE and RANDOM, which is a three-way token-level classification task for predicting whether a token is a shuffled token, a random token, or an original token. For each sample, we replace 10% of tokens with shuffled ones from the same sequence and another 10% of tokens with random ones from the vocabulary. This task can be considered as a more complex one, because the model must recognize the difference between tokens replaced in the same context and tokens replaced outside of the context. For this task we use the cross-entropy loss averaged over all input tokens.

Masked Token Type Classification (TOKEN TYPE): Our fourth objective is a four-way classification, aiming to predict whether a token is a stop word,[2] a digit, a punctuation mark, or a content word. Therefore, the task can be seen as a simplified version of POS tagging. We regard any tokens that are not included in the first three categories as content words. We mask 15% of tokens in each sample with a special `[MASK]` token and compute the cross-entropy loss over the masked ones only not to make the task trivial. For example, if we compute the token-level loss over unmasked tokens, a model can easily recognize the four categories as we only have a small number of non-content words in the vocabulary.

Masked First Character Prediction (FIRST CHAR): Finally to test our second hypothesis, we propose a simplified version of the MLM task, where the model has to predict only the first character of each masked token instead of performing a softmax over the entire vocabulary. We define a 29-way classification task, where 29 categories include the English alphabet (0 to 25), a digit (26), a punctuation mark (27), or any other character (28). We mask 15% of tokens in each sample and compute the cross-entropy loss over the masked tokens only.[3]

3 Experimental Setup

Models: We use BERT (Devlin et al., 2019) (BASE) as our basis model by replacing the MLM and NSP objectives with one of our five token-level pretraining tasks in all our experiments. We also consider two smaller models from Turc et al. (2019), MEDIUM and SMALL, where we reduce the size of the following components compared to the BASE model: (1) hidden layers; (2) hidden size; (3) feed-forward layer size; and (4) attention heads. More specifically, MEDIUM has eight hidden layers and attention heads, while SMALL has four hidden layers and eight attention heads. The size of feed-forward and hidden layers for both models are 2048 and 512, respectively.

[2]We use the Natural Language Toolkit's stop word list: `https://www.nltk.org/`.

[3]For more details on the pretraining tasks, including equations, see Appendix A.

Pretraining Data: We pretrain all models on the English Wikipedia and BookCorpus (Zhu et al., 2015) (WikiBooks) using the `datasets` library.[4]

Implementation Details: We pretrain and fine-tune our models with two NVIDIA Tesla V100 (SXM2 - 32GB) with a batch size of 32 for BASE and 64 for MEDIUM and SMALL. We pretrain all our models for up to five days each due to limited access to computational resources and funds for running experiments. We save a checkpoint of each model every 24 hours.[5]

Evaluation: We evaluate our approaches on GLUE (Wang et al., 2019) and SQUAD (Rajpurkar et al., 2016) benchmarks. To measure performance in downstream tasks, we fine-tune all models for five times each with a different random seed.

Baseline: For comparison, we also pretrain models with MLM. Following BERT and ROBERTA, we mask 15% of tokens in each training instance, where 80% of the tokens are replaced with `[MASK]`, 10% of the tokens are replaced with a random word and the rest of tokens remain unchanged. We compute the cross-entropy loss averaged over the masked tokens only.

4 Results

Performance Comparison: Table 1 presents results on GLUE and SQUAD, for our five pretraining tasks compared to MLM across all model configurations (§3). We also include for reference our replicated downstream performance by fine-tuning BERT-BASE (MLM + NSP) pretrained[6] for 40 epochs (Upper Bound).

We first observe that our best objective, Shuffle + Random, outperforms MLM on GLUE Avg. and SQUAD in the majority of model settings (BASE, MEDIUM and SMALL) with five days pretraining. For example in GLUE, we obtain an average of 79.2 using Shuffle + Random with BERT-BASE compared to 77.6 using MLM. This suggests that Shuffle + Random can be a competitive alternative to MLM. Although Shuffle + Random does not outperform MLM in SQUAD only with BERT-BASE, it remains competitive (83.5 compared to 84.8). The

[4]`https://github.com/huggingface/datasets`

[5]For more details on model setup, implementation, and data preprocessing, see Appendix C.

[6]We used an already pretrained model provided by Wolf et al. (2020).

Pretraining task	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	GLUE Avg.	SQuAD v1.1
	BASE - 40 Epochs Pretraining (Upper Bound)									
MLM + NSP	83.8	90.8	87.8	69.9	91.9	85.0	58.9	89.3	82.1 (0.4)	87.4 (0.6)
Ours	BASE - Five Days Pretraining									
MLM	**80.1**	**88.2**	85.9	61.4	**89.6**	81.6	49.6	84.7	77.6 (0.2)	**84.8 (0.2)**
Shuffle	73.3	81.6	82.1	57.5	82.4	79.1	33.4	79.9	71.2 (0.3)	74.8 (0.2)
Random	78.6	87.0	85.5	60.5	87.4	81.6	47.0	84.0	76.4 (0.2)	81.6 (0.4)
Shuffle + Random	78.6	87.7	**86.1**	**65.1**	87.8	**87.0**	**54.9**	**86.7**	**79.2 (0.3)**	83.5 (0.2)
Token Type	75.1	84.2	83.9	56.8	86.7	75.5	40.3	77.4	72.5 (0.2)	78.6 (0.7)
First Char	78.2	87.1	85.5	60.7	89.5	83.6	43.9	84.6	76.7 (0.5)	82.0 (0.1)
	MEDIUM - Five Days Pretraining									
MLM	**78.7**	85.3	85.4	61.7	**89.9**	80.6	43.1	84.5	76.1 (0.4)	**81.8 (0.5)**
Shuffle	77.3	86.4	85.3	64.0	87.9	83.4	**53.8**	84.1	77.8 (0.2)	81.3 (0.2)
Random	77.7	86.2	85.6	**64.3**	87.8	81.7	44.3	84.8	76.6 (0.3)	79.5 (0.1)
Shuffle + Random	78.3	**87.0**	**85.7**	63.3	87.8	**85.9**	52.4	**85.4**	**78.2 (0.2)**	**81.8 (0.2)**
Token Type	76.0	84.7	84.4	59.7	87.6	81.4	45.8	80.7	75.0 (0.4)	79.8 (0.4)
First Char	77.4	85.6	85.1	59.4	88.8	83.9	42.4	83.0	75.7 (0.3)	79.5 (0.2)
	SMALL - Five Days Pretraining									
MLM	76.2	84.2	84.8	57.5	**88.6**	**82.9**	36.3	83.0	74.2 (0.4)	77.1 (0.3)
Shuffle	74.9	84.0	84.2	59.8	86.4	80.0	**47.1**	81.1	74.7 (0.3)	76.1 (0.6)
Random	75.6	84.7	84.8	58.3	86.7	80.0	39.6	83.5	74.1 (0.4)	76.7 (0.5)
Shuffle + Random	**76.9**	**85.7**	**85.3**	**60.3**	87.1	81.8	41.7	**84.6**	**75.4 (0.4)**	**77.5 (0.3)**
Token Type	73.2	83.0	83.7	58.8	86.4	77.1	37.1	77.8	72.1 (0.4)	74.2 (0.3)
First Char	75.3	84.0	84.9	55.6	87.2	79.8	33.1	83.3	72.9 (0.8)	77.4 (0.2)

Table 1: Results on GLUE and SQuAD dev sets with standard deviations over five runs in parentheses. For MNLI, we report matched accuracy, for CoLA Matthews correlation, for STS-B Spearman correlation, for MRPC accuracy, for QQP and SQuAD F1 scores; accuracy for all other tasks. **Bold** values denote best performing across each dataset and Avg. for each model setting.

remainder of our proposed tasks perform well, with First Char and Random being close to MLM across all model configurations confirming our two hypotheses. Finally, Shuffle with BERT-BASE records the lowest performance on GLUE (71.2 points), but it performs best when combined with Random (i.e. Shuffle + Random).

Computational Efficiency Comparison: Figure 2 presents the performance of our proposed methods across (a) epochs and (b) days in GLUE (SQUAD results available in Appendix E). Results suggest that our methods are, in general, more computationally efficient compared to MLM. Shuffle + Random trains for the largest number of epochs (i.e. faster forward-backward passes) in five days for the SMALL and MEDIUM settings, with Random outperforming the rest in the BASE model setting (Figure 2 (a)). If we take a closer look, we can also see that Shuffle + Random obtains higher performance to MLM across all model configurations when training for a similar number of epochs, suggesting that our approach is a more data efficient task. Finally, we can also assume that Shuffle + Random is more challenging than MLM as in all settings it results in lower GLUE scores after the first day of pretraining (Figure 2 (b)). However, with more iterations it is clear that it results in learning better text representations and quickly outperforms MLM. For example, it achieves a performance of 78.2 compared to 76.1 for MLM with MEDIUM on the fifth day. Regarding the remainder of our proposed objectives, we can see that they perform comparably and sometimes better than the MLM under SMALL and MEDIUM model settings. However, MLM on average outperforms them in the BASE setting where the models are more highly parameterized.

Lastly, we observe that for the majority of GLUE tasks, we obtain better or comparable performance to MLM with a maximum of approximately three epochs of training with a BASE model. This demonstrates that excessively long and computationally inefficient pretraining strategies do not add a lot in downstream performance.

5 Discussion

Based on our results, there are mainly two key elements that should be considered for designing

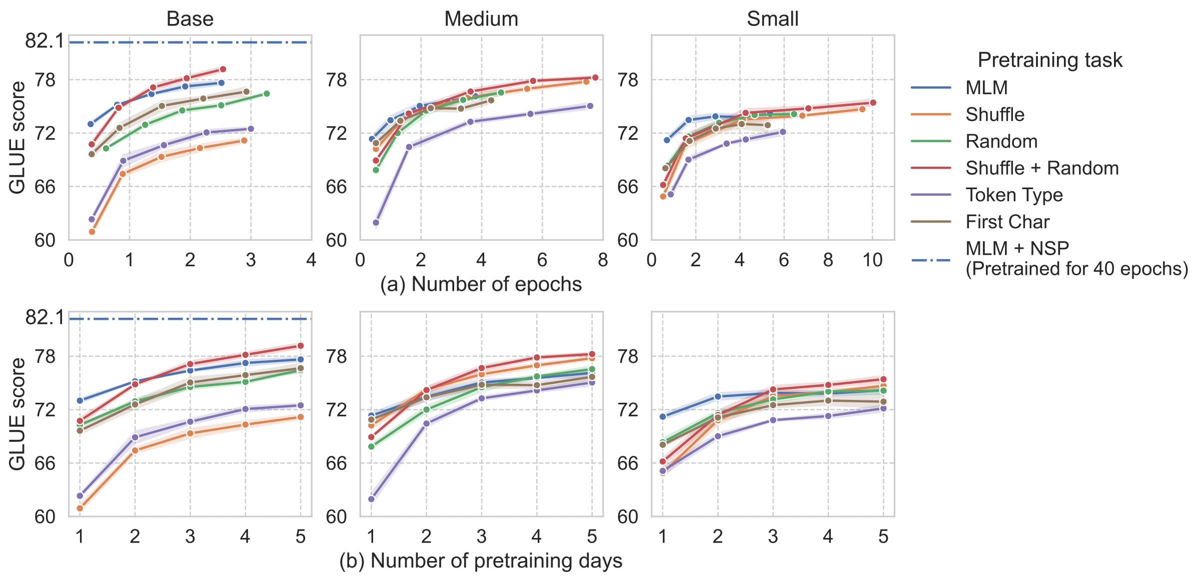

Figure 2: Results on GLUE dev sets across (a) epochs and (b) days. Each point is a checkpoint pretrained for $1 \leq n \leq 5$ day(s).

pretraining objectives.

Task Difficulty: A pretraining task should be moderately difficult to learn in order to induce rich text representations. For example, we can assume from the results that Token Type was somewhat easy for a model to learn as it is a four-way classification of identifying token properties. Besides, in our preliminary experiments, predicting whether a masked token is a stop word or not (Masked Stop Word Detection) also did not exhibit competitive downstream performance to MLM as the task is a lot simpler than Token Type.

Robustness: A model should always learn useful representations from "every" training sample to solve a pretraining task, regardless of the task difficulty. For instance, Figures 3 to 5 in Appendix D demonstrate that Shuffle needs some time to start converging across all model configurations, which means the model struggled to acquire useful representations at first. In contrast, the loss for Shuffle + Random consistently decreases. Because Shuffle + Random is a multi-class classification, unlike Shuffle or Random, we assume that it can convey richer signals to the model and help stabilize pretraining. Finally, we can also assume that MLM satisfies both elements as it is a multi-class setting over the entire vocabulary and its loss consistently decreases.

6 Conclusions

We have proposed five simple self-supervised pretraining objectives and tested their effectiveness against MLM under various model settings. We show that our best performing, manipulated word detection task, results in comparable performance to MLM in GLUE and SQuAD, whilst also being significantly faster in smaller model settings. We also show that our tasks result in higher performance trained for the same number of epochs as MLM, suggesting higher data efficiency. For future work, we are interested in exploring *which has the most impact in pretraining: the data or the pretraining objective?*

Acknowledgments

NA is supported by EPSRC grant EP/V055712/1, part of the European Commission CHIST-ERA programme, call 2019 XAI: Explainable Machine Learning-based Artificial Intelligence. KM is supported by Amazon through the Alexa Fellowship scheme.

References

Stéphane Aroca-Ouellette and Frank Rudzicz. 2020. On Losses for Modern Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4970–4981, Online. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc Le, and Christopher D. Manning. 2020a. Pre-training transformers as energy-based cloze models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 285–294, Online. Association for Computational Linguistics.

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

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.

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.

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*.

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.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

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.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. In *International Conference on Machine Learning*, pages 5926–5936.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie 2.0: A continual pre-training framework for language understanding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8968–8975.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models.

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*.

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Jiangnan Xia, Zuyi Bao, Liwei Peng, and Luo Si. 2020. Structbert: Incorporating language structures into pre-training for deep language understanding. In *International Conference on Learning Representations*.

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.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32, pages 5753–5763. Curran Associates, Inc.

Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the 2015 IEEE International Conference on Computer Vision*, pages 19–27.

Appendices

A Task Details

Here, we detail our frustratingly simple pretraining objectives, which are based on token-level classification tasks and can be used on any unlabeled corpora without laborious preprocessing to obtain labels for self-supervision.

Shuffled Word Detection (SHUFFLE): Our first pretraining task is a token-level binary classification task, which consists of identifying whether a token in the input sequence has been shuffled or not. For each sample, we randomly shuffle 15% of the tokens. This task is trained with the *token-level* binary cross-entropy loss averaged over all input tokens:

$$\mathcal{L}_{\text{shuffle}} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log p(x_i) + (1-y_i)\log(1-p(x_i)) \tag{1}$$

where N is the number of tokens in a sample, and $p(x_i)$ represents the probability of the i-th input token x_i predicted as *shuffled* by a model. y_i is the corresponding target label.

This task is motivated by the success of ELECTRA, whose pretraining task is to let a discriminator to predict whether a given token is original or replaced (replaced word detection) in addition to MLM. The major difference between ours and ELECTRA is that we do not rely on MLM, whereas ELECTRA utilizes it as its generator. Here, our intuition is that a model should acquire both syntactic and semantic knowledge to detect shuffled tokens in contexts.

Random Word Detection (RANDOM): We also consider replacing tokens with out-of-sequence tokens. For this purpose we propose RANDOM, a pretraining objective which replaces 15% of tokens with random ones from the vocabulary. Similar to shuffling tokens in the input, we expect that replacing a token in the input with a random word from the vocabulary "forces" the model to acquire both syntactic and semantic knowledge from the context to base its decision on whether it has been replaced or not. This task is trained with the token-level binary cross-entropy loss averaged over all input tokens (Eq. (1)).

Manipulated Word Detection (SHUFFLE + RANDOM): Our third task is a three-way token-level classification of whether a token is a shuffled token, a random token, or an original token. For each sample, we replace 10% of tokens with shuffled ones and another 10% of tokens with random ones. This task is an extension of SHUFFLE and RANDOM and can be regarded as a more complex one because the model must recognize the difference between a token replaced in the same context and a token replaced outside of the context. For this task we employ the cross-entropy loss averaged over all input tokens:

$$\mathcal{L}_{\text{manipulated}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{3} y_{ij}\log p_{ij}(x_i) \tag{2}$$

where $p_{ij}(x_i)$ represents the probability of the i-th input token x_i predicted as *shuffled* ($j = 1$), *randomized* ($j = 2$), or *original* ($j = 3$) by a model. y_{ij} is the corresponding target label.

Masked Token Type Classification (TOKEN TYPE): Our fourth task is a four-way classification task that identifies whether a token is a stop word[7], a digit, a punctuation mark, or a content word. We regard any tokens that are not included in the first three categories as content words. We mask 15% of tokens in each sample with a special `[MASK]` token and compute the cross-entropy loss over the masked ones only not to make the task trivial: if we compute the token-level loss, including unmasked tokens, a model can easily recognize the four categories of tokens as we have a small number of tokens for non-content words. In this task, a model should be able to identify the distinction between different types of tokens; therefore, the task can be seen as a simplified version of POS tagging.

Masked First Character Prediction (FIRST CHAR): Our last task is a 29-way classification task, where a model needs to predict the first character of a masked token. The 29 categories include the English alphabet (0 to 25), a digit (26), a punctuation mark (27), or any other character (28). We mask 15% of tokens in each sample and compute the cross-entropy loss over the masked ones only. This task can be seen as a simplified version of MLM as the model just need to predict the first

[7]A stop word category is based on the Natural Language Toolkit's stop word list: https://www.nltk.org/.

character of each masked token. Besides, it is also similar to masked character-level language modeling, in that the output of both tasks is in characters.

B Non-linguistically Intuitive Task

As we have described in Section 2, a non-linguistically intuitive task should not be "explicitly" related to an input sequence to solve, unlike linguistically intuitive tasks, such as Shuffle and Random. For example, predicting the first character of a masked token should not matter for a model to learn that 'cat' and 'sat' usually appear in the same context. However, because accurately predicting the first character requires the model to guess its whole word "implicitly" given its surrounding tokens, the first character of each masked token should be related to the context. The deep architecture of transformer-based models should allow them to learn such "implicit" associations between input tokens by solving the non-linguistically intuitive task, which leads to helping them to learn syntactic and semantic relations between tokens.

C Experimental Setup

C.1 Model Architecture

For all our experiments, we use BERT (Devlin et al., 2019) as our basis model by replacing the MLM and NSP objectives with one of our five token-level pretraining tasks. More specifically, we employ BERT-BASE (12 hidden layers and attention heads, $Dim_{\text{hidden}} = 768$, $Dim_{\text{intermediate}} = 3072$, Total parameters $= 125M$) (BASE), MEDIUM (eight hidden layers and attention heads, $Dim_{\text{hidden}} = 512$, $Dim_{\text{intermediate}} = 2048$, Total parameters $= 51.5M$), and SMALL (four hidden layers and eight attention heads, $Dim_{\text{hidden}} = 512$, $Dim_{\text{intermediate}} = 2048$ Total parameters $= 38.9M$).

C.2 Data

Following Devlin et al. (2019), we use the English Wikipedia and BookCorpus (Zhu et al., 2015) data (WikiBooks) downloaded from the `datasets` library[8]. We remove headers for the English Wikipedia and extract training samples with a maximum length of 512. For the BookCorpus, we concatenate sentences such that the total number of tokens is less than 512. For the English Wikipedia, we extract one sample from articles whose length is less than 512. We tokenize text using byte-level Byte-Pair-Encoding (Sennrich et al., 2016). The resulting corpus consists of 8.1 million samples and 2.7 billion tokens in total.

C.3 Implementation Details

We implement our models using PyTorch (Paszke et al., 2019) and the `transformers` library (Wolf et al., 2020). We pretrain our models with two NVIDIA Tesla V100 (SXM2 - 32GB) and use one for fine-tuning. Our code is publicly available on GitHub: https://github.com/gucci-j/light-transformer-emnlp2021.

Pretraining: We set the batch size to 32 for the BASE models and 64 for the MEDIUM and SMALL models. We pretrain models for five days and optimized them with an Adam optimizer (Kingma and Ba, 2014). We apply automatic mixed precision and distributed training during pretraining. Note that we generate labels dynamically during pretraining.

Finetuning: We fine-tune models for up to 10 and 20 epochs with early stopping for SQUAD and GLUE, respectively. To minimize the effect of random seeds, we test five different random seeds for each task. We omitted the problematic WNLI task for GLUE, following Aroca-Ouellette and Rudzicz (2020).

C.4 Hyperparameter Details

As explained in Section 3, we entirely followed the BERT architecture and only modified its output layer depending on the task employed. Table 2 shows the hyperparameter settings for pretraining and fine-tuning. Note that we utilized neither any parameter sharing tricks nor any techniques that did not appear in Devlin et al. (2019).

D Pretraining Behavior

Figures 3, 4 and 5 show the loss curves for our pretraining tasks in each model setting: BASE, MEDIUM and SMALL.

E Performance in SQUAD

Figure 6 demonstrates the performance of our proposed methods across (a) epochs and (b) days in SQUAD.

[8] https://github.com/huggingface/datasets

Hyperparameter	Pretraining	Fine-tuning
Maximum train epochs	10 epochs for BASE and MEDIUM 15 epochs for SMALL	Up to 20 epochs for GLUE Up to 10 epochs for SQuAD
Batch size (per GPU)	16 for BASE 32 for HALF and QUARTER	32 for GLUE 16 for SQuAD
Adam ϵ	1e-8	
Adam β_1	0.9	
Adam β_2	0.999	
Sequence length	512	128 for GLUE 384 for SQuAD
Peak learning rate	**BASE**: 1e-4 for MLM and Token Type, 1e-5 for Shuffle. 5e-5 for First Char, Random and Shuffle + Random. **MEDIUM & SMALL**: 1e-4 for MLM, Token Type and First Char. 5e-5 for Shuffle, Random and Shuffle + Random.	3e-5
Warmup steps	10000	First 6% of steps
Weight decay	0.01	0
Attention Dropout	0.1	
Dropout	0.1	
Early stopping criterion		GLUE: No improvements over 5% of steps. SQuAD: No improvements over 2.5% of steps.

Table 2: Hyperparameters in our experiments. If not shown, the hyperparameters for fine-tuning are the same as the pretraining ones.

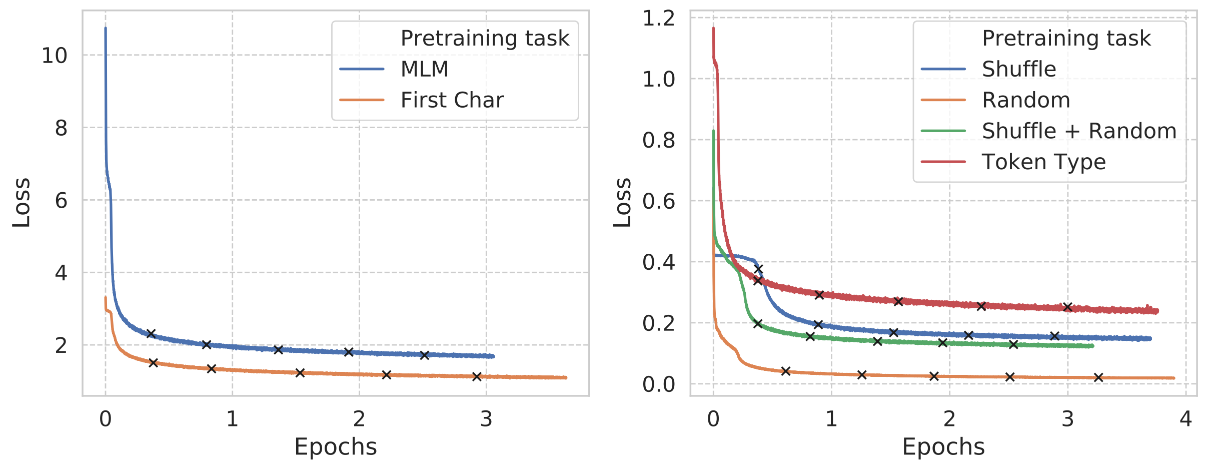

Figure 3: The loss curves for BERT-BASE models. Each $\times$ denotes a checkpoint pretrained for $1 \leq n \leq 5$ day(s).

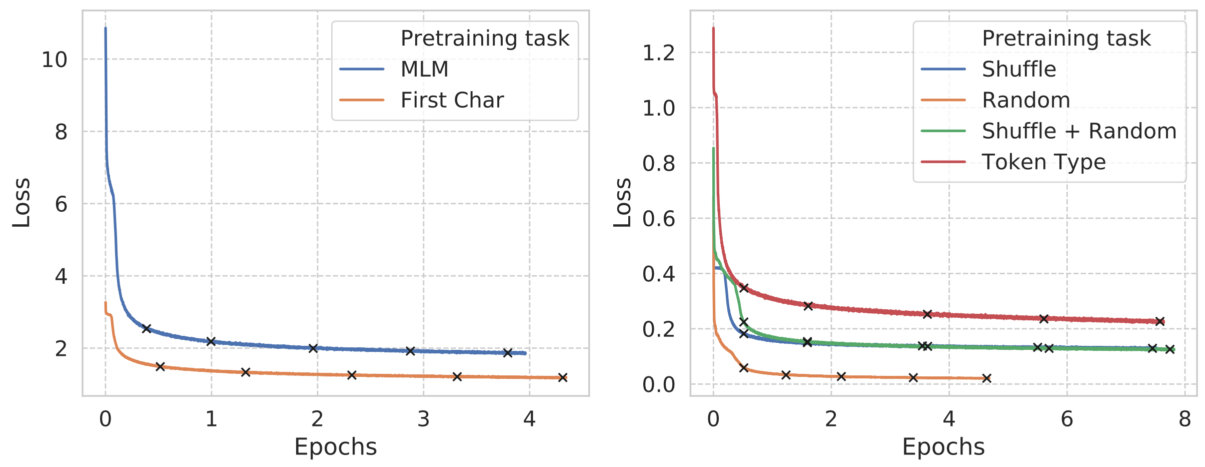

Figure 4: The loss curves for BERT-MEDIUM models. Each $\times$ denotes a checkpoint pretrained for $1 \leq n \leq 5$ day(s).

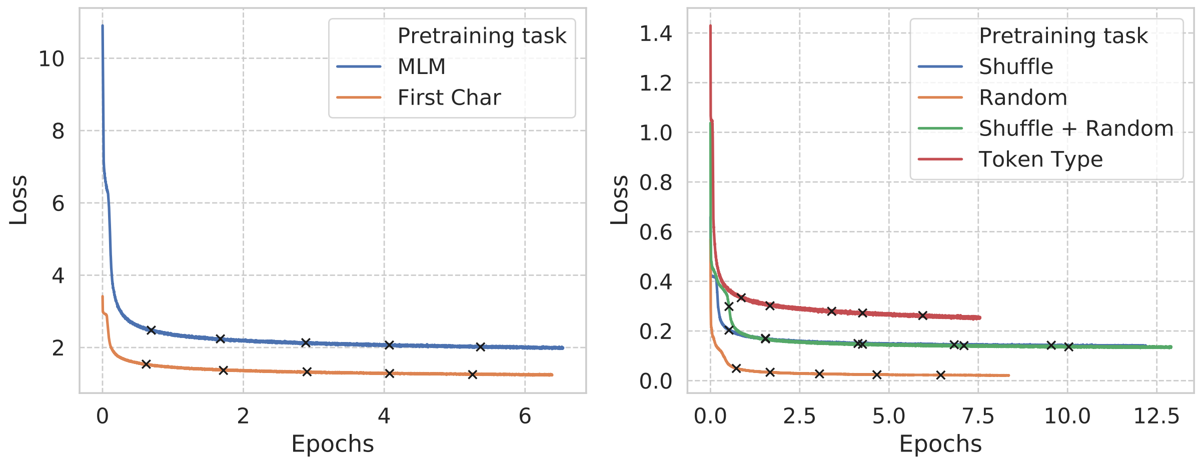

Figure 5: The loss curves for BERT-SMALL models. Each $\times$ denotes a checkpoint pretrained for $1 \leq n \leq 5$ day(s).

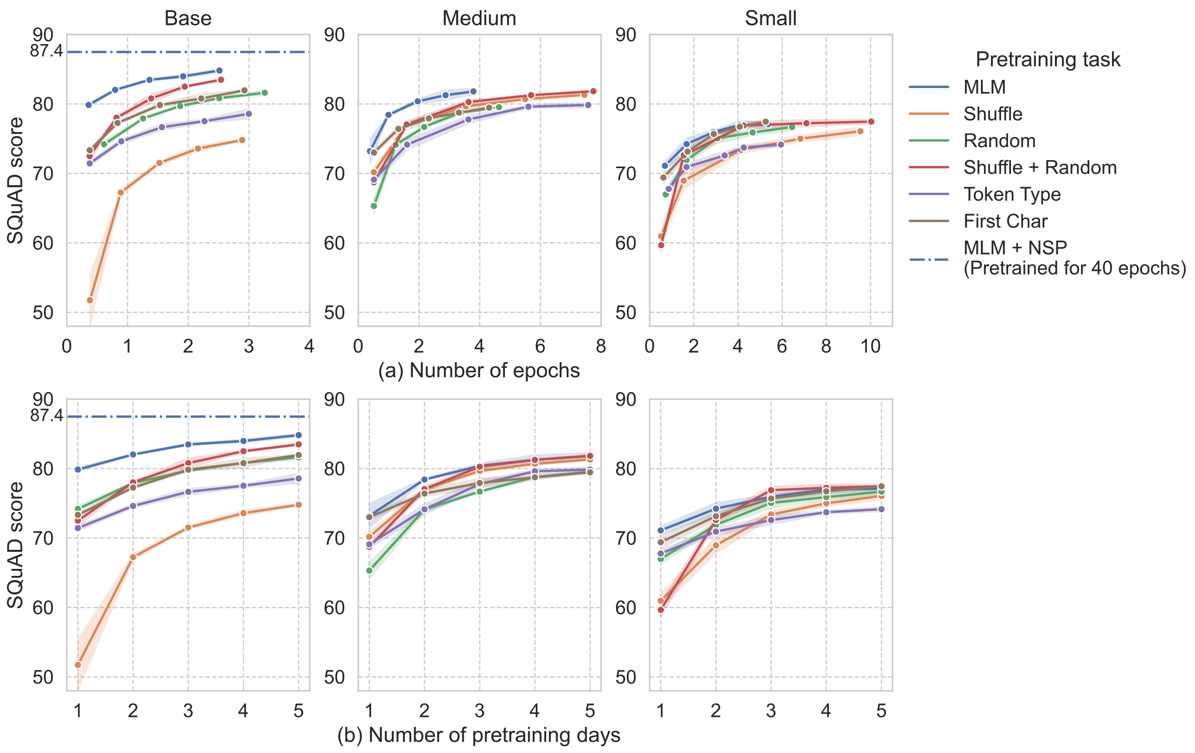

Figure 6: Results on SQUAD dev sets across (a) epochs and (b) days. Each point is a checkpoint pretrained for $1 \leq n \leq 5$ day(s).