How does the task complexity of masked pretraining objectives affect downstream performance?

Atsuki Yamaguchi[1], **Hiroaki Ozaki**[1*], **Terufumi Morishita**[1*],
Gaku Morio[2*] and **Yasuhiro Sogawa**[1]
[1]Hitachi, Ltd., Kokubunji, Tokyo, Japan
[2]Hitachi America Ltd., Santa Clara, CA, USA
[1]{atsuki.yamaguchi.xn,hiroaki.ozaki.yu,
terufumi.morishita.wp,yasuhiro.sogawa.tp}@hitachi.com
[2]gaku.morio@hal.hitachi.com

Abstract

Masked language modeling (MLM) is a widely used self-supervised pretraining objective, where a model needs to predict an original token that is replaced with a mask given contexts. Although simpler and computationally efficient pretraining objectives, e.g., predicting the first character of a masked token, have recently shown comparable results to MLM, no objectives with a masking scheme actually outperform it in downstream tasks. Motivated by the assumption that their lack of complexity plays a vital role in the degradation, we validate whether more complex masked objectives can achieve better results and investigate how much complexity they should have to perform comparably to MLM. Our results using GLUE, SQuAD, and Universal Dependencies benchmarks demonstrate that more complicated objectives tend to show better downstream results with at least half of the MLM complexity needed to perform comparably to MLM. Finally, we discuss how we should pretrain a model using a masked objective from the task complexity perspective.[1]

1 Introduction

Masked language modeling (MLM) (Devlin et al., 2019), where a model needs to predict a particular token that is replaced with a mask placeholder given its surrounding context, is a widely used self-supervised pretraining objective in natural language processing. Recently, simpler pretraining objectives have shown promising results on downstream tasks. Aroca-Ouellette and Rudzicz (2020) have proposed various token-level and sentence-level auxiliary pretraining objectives, showing improvements over BERT (Devlin et al., 2019). Yamaguchi et al. (2021) and Alajrami and Aletras (2022) have demonstrated that such token-level objectives themselves, i.e., pretraining without MLM, perform comparably to MLM.

Although these simple token-level objectives themselves, e.g., predicting the first character of a masked token (First Char) (Yamaguchi et al., 2021), have exhibited competitive downstream performances to MLM with smaller computations, no objectives using mask tokens are not clearly comparable to MLM on downstream tasks. We conjecture that the main reason behind the performance difference lies in its lack of complexity, i.e., the number of classes to be predicted, and similar arguments have been made for auxiliary task ineffectiveness (Lan et al., 2020) and pretraining task design (Yamaguchi et al., 2021).

This paper sheds light on the task complexity of masked pretraining objectives and investigates **RQ1**: whether a more complex objective, becoming closer to MLM, can achieve a better downstream result and **RQ2**: how much complexity they need to obtain comparable results to MLM. To this end, we propose masked n character prediction as a control task, which requires us to predict the first or last n characters of a masked token, allowing us to empirically evaluate how the task complexity affects downstream performance by varying n. We pretrain 14 different types of models with the proposed control task in addition to MLM for reference and evaluate their downstream performance on the GLUE (Wang et al., 2019), SQuAD (Rajpurkar et al., 2016), and Universal Dependencies (UD) (Nivre et al., 2020) benchmarks. We also conduct a cost-benefit analysis of performance gains with respect to task complexity and analyze how to select an optimal complexity for a given task.

Contributions (1) We model the task complexity of a masked pretraining objective as masked n character prediction (§2) and revealed how it affects downstream performance (§4). (2) We conduct two analyses to provide insights into how we should

[*] Equal contribution
[1]Our code and pretrained models are available at https://github.com/hitachi-nlp/mlm-probe-acl2023.

Findings of the Association for Computational Linguistics: ACL 2023, pages 10527–10537

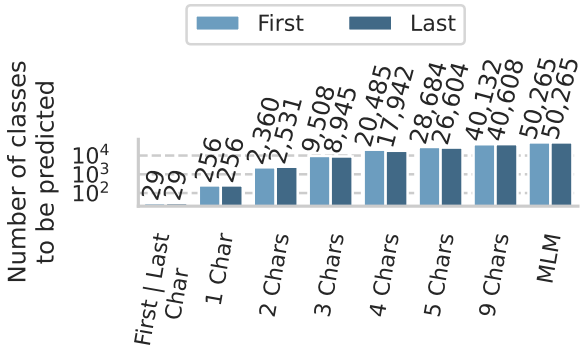

Figure 1: Number of classes to be predicted for each n Chars objective along with First Char, Last Char, and MLM.

pretrain a model by using a masked objective from the task complexity perspective (§5).

2 Methodology

Our main hypothesis is that the more complexity, i.e., the number of classes to be predicted, masked pretraining objectives have, the better the downstream performance they will achieve. This is because a more complex task should have a larger number of classes to be predicted, giving more informative signals to a model via training.

To verify the hypothesis empirically and answer the two research questions listed in §1, we extend First Char (Yamaguchi et al., 2021) and let a model predict the first or last $n \in \mathbb{N}$ characters of a masked token (n Chars)[2]. The task is trained with the token-level cross-entropy loss averaged over masked tokens. Our extension allows us to evaluate how the complexity affects downstream performance by varying n.

Figure 1 shows the number of classes to be predicted for n Chars when using a pretrained tokenizer of RoBERTa (Liu et al., 2019). We can see that the larger n, the closer the objective becomes to MLM. For 1 Char, we simply pick up the first character of each token in the vocabulary instead of casting it into 29 classes as in First Char[3], resulting in 256 types of characters.

[2]We also allow *last* n characters to be predicted because the prediction direction should not matter for a model to learn effective representations.

[3]In First Char, a model needs to predict the first character of a masked token as 29-way classification, including alphanumeric characters, punctuation marks, and any other characters.

3 Experimental Setup

Here, we describe our experimental setups for both pretraining and fine-tuning.[4]

Model We used the base configuration of BERT (Devlin et al., 2019). The model consists of 12 hidden layers and attention heads, and the dimensions of hidden layers and intermediate feed-forward layers are 768 and 3072, respectively. We simply put a linear layer on the BERT model for n Chars and First Char.

Baselines We pretrained MLM and First Char for reference to evaluate the influence of masked pretraining objective complexity. We also set up Last Char, where a model needs to predict the last character of a masked token from 29 categories the same as in First Char.

Pretraining Data Following Devlin et al. (2019), we pretrained all models on English Wikipedia and BookCorpus (Zhu et al., 2015) using the datasets (Lhoest et al., 2021) library. We set the maximum sequence length to 512. We tokenized texts using byte-level Byte-Pair-Encoding (Sennrich et al., 2016), and the resulting corpora consist of 10 million samples and 4.9 billion tokens in total.

Fine-tuning Data We used the GLUE, SQuAD v1.1, and UD v2.10 benchmarks to measure both semantic and syntactic downstream performances in detail. For UD, we used its English subset of Universal Dependencies English Web Treebank (EWT) (Silveira et al., 2014).

Evaluation Following previous work (Aroca-Ouellette and Rudzicz, 2020), we report matched accuracy for MNLI, Matthews correlation for CoLA, Spearman correlation for STS-B, accuracy for MRPC, F1 scores for QQP and SQuAD, and accuracy for all other tasks. For UD, we used labeled attachment score (LAS). For each task, we report a mean score over five runs with different random seeds. We excluded problematic WNLI following prior work (Aroca-Ouellette and Rudzicz, 2020).

Implementation Details We implemented our models using the PyTorch (Paszke et al., 2019) and Hugging Face Transformers (Wolf et al., 2020) libraries. For fine-tuning on UD, we trained a deep biaffine attention parser (BAP) (Dozat and Manning, 2017) built on top of pretrained language

[4]For more details, please refer to Appendix B.

Model	MNLI 393k	QQP 364k	QNLI 105k	SST 67k	CoLA 8.6k	STS 5.7k	MRPC 3.7k	RTE 2.5k	GLUE Avg.	SQuAD 88k	UD 13k
MLM	**82.3**	**86.9**	89.2	<u>91.8</u>	**58.0**	**87.0**	**86.7**	**64.8**	**80.8** (0.3)	<u>88.1</u> (0.6)	**88.8** (0.1)
First 9 Chars	81.6	86.4	89.2	**91.9**	53.0	85.6	85.2	58.2	78.9 (0.9)	87.4 (0.4)	88.5 (0.1)
First 5 Chars	82.0	86.6	89.3	91.1	51.8	85.6	85.5	59.2	78.9 (0.5)	87.9 (0.5)	88.5 (0.1)
First 4 Chars	82.0	86.6	**89.6**	91.3	54.2	85.5	85.7	57.3	79.0 (0.4)	87.9 (0.4)	**88.8** (0.1)
First 3 Chars	81.9	<u>86.8</u>	88.7	90.7	52.0	85.9	85.6	58.9	78.8 (0.5)	87.6 (0.3)	88.1 (0.1)
First 2 Chars	81.1	86.5	88.6	90.8	51.1	85.1	83.7	60.6	78.4 (0.7)	86.8 (0.7)	87.8 (0.1)
First 1 Char	80.5	86.3	88.5	90.4	48.6	84.7	83.3	60.0	77.8 (0.3)	86.1 (0.3)	87.6 (0.1)
First Char	80.7	86.3	88.2	90.6	50.0	85.1	85.4	59.5	78.2 (0.3)	85.6 (0.4)	87.8 (0.0)
Last 9 Chars	<u>82.1</u>	86.7	89.3	91.4	55.0	85.6	85.1	57.6	79.1 (0.3)	**88.4** (0.2)	<u>88.7</u> (0.1)
Last 5 Chars	81.8	86.4	89.1	91.3	54.8	<u>85.8</u>	85.4	58.7	<u>79.2</u> (0.8)	87.5 (0.7)	88.5 (0.1)
Last 4 Chars	81.6	86.6	<u>89.4</u>	90.2	<u>56.0</u>	85.6	<u>86.2</u>	56.9	79.1 (0.3)	87.6 (0.8)	88.4 (0.1)
Last 3 Chars	81.3	86.4	88.9	91.0	53.0	84.9	84.9	56.1	78.3 (0.6)	87.0 (0.5)	88.4 (0.1)
Last 2 Chars	81.0	86.3	88.0	90.7	50.7	84.5	85.7	58.6	78.2 (0.4)	87.0 (0.3)	88.0 (0.1)
Last 1 Char	80.2	86.3	87.9	90.4	54.8	84.6	84.8	<u>61.2</u>	78.8 (0.3)	86.0 (0.8)	87.8 (0.1)
Last Char	79.8	86.0	87.5	90.2	48.8	85.2	85.2	55.7	77.3 (0.5)	85.5 (0.1)	88.1 (0.1)

Correlation r	MNLI	QQP	QNLI	SST	CoLA	STS	MRPC	RTE	GLUE Avg.	SQuAD	UD
First Char/MLM	.787	.520	.659	.734	.705	.653	.486	.182	.721	.783	.873
Last Char/MLM	.942	.739	.835	.662	.675	.714	.322	.336	.789	.840	.877

Table 1: Results and their correlation values on GLUE, SQuAD, and UD (EN-EWT) with standard deviations over five runs in parentheses. Values under dataset names are the number of their corresponding training samples. We show test set results for UD and dev sets results for GLUE and SQuAD. **Bold** and <u>underlined</u> values denote best and second best scores for each dataset.

models. We used the SuPar library[5] to implement the parser and followed its default hyperparameter configurations.

4 Results

RQ1: Do more complex objectives achieve better results? Table 1 displays downstream task results on GLUE, SQuAD, and UD for our control tasks and their comparisons against MLM, First Char, and Last Char. Overall, we observe the larger n, the better downstream performance is. Looking at each dataset result closely, we see that the datasets with over 5k training samples exhibit moderate to high correlations with the lowest and highest correlation values of 0.520 (First Char to MLM) for QQP and 0.942 (Last Char to MLM) for MNLI, respectively. In contrast, the corpora with less than 5k samples tend to exhibit low to moderate correlations, ranging from 0.182 (First Char to MLM) for RTE to 0.486 (First Char to MLM) for MRPC. Therefore, we can see a general trend that a more complex masked pretraining objective yields better performance in downstream tasks especially under a high-resource scenario, whereas it does not always achieve a better result on a low-resource dataset, where in this case MLM tends to achieve a better result.

RQ2: How much complexity do we need to obtain comparable results to MLM? To answer the RQ2, we only compare results on high-resource corpora, given that results on low-resource corpora have large standard deviations of 1.0 or more (see Appendix C.2). We can observe from Table 1 that at least $n = 4$ complexity is necessary to achieve comparable results to MLM. For instance, First n Chars objectives require $n = 4$ (20k classes) to surpass the MLM performance on at least one of the target downstream tasks. Last n Chars also need $n = 4$ (18k classes) to beat MLM on one of the tasks. For MNLI and QQP, our control tasks did not yield better results.

5 Analysis and Discussion

On the basis of results from the RQ1 and RQ2, we discuss how we should pretrain a model in practice from the task complexity perspective.

Task complexity affects computational efficiency. The task complexity is closely related to computational costs with more complex objectives larger costs. Table 2 shows the number of floating-point

[5] https://github.com/yzhangcs/parser

Model	FLOPs $\times 10^{19}$	GLUE Avg.	SQuAD F1	UD LAS
MLM	2.44	80.8	88.1	88.8
First				
9 Chars	2.29 (-6)	78.9 (-2.4)	87.4 (-0.9)	88.5 (-0.3)
5 Chars	2.12 (-13)	78.9 (-2.4)	87.9 (-0.3)	88.5 (-0.4)
4 Chars	1.99 (-18)	79.0 (-2.3)	87.9 (-0.2)	88.8 (0.0)
3 Chars	1.83 (-25)	78.8 (-2.5)	87.6 (-0.6)	88.1 (-0.8)
2 Chars	1.72 (-30)	78.4 (-3.0)	86.8 (-1.5)	87.8 (-1.1)
1 Char	1.69 (-31)	77.8 (-3.8)	86.1 (-2.3)	87.6 (-1.4)
First Char	1.68 (-31)	78.2 (-3.2)	85.6 (-2.9)	87.8 (-0.2)
Last				
9 Chars	2.30 (-6)	79.1 (-2.2)	88.4 (+0.3)	88.7 (-0.2)
5 Chars	2.09 (-15)	79.2 (-2.1)	87.5 (-0.7)	88.5 (-0.4)
4 Chars	1.96 (-20)	79.1 (-2.2)	87.6 (-0.6)	88.4 (-0.4)
3 Chars	1.82 (-26)	78.3 (-3.1)	87.0 (-1.3)	88.4 (-0.5)
2 Chars	1.72 (-30)	78.2 (-3.3)	87.0 (-1.3)	88.0 (-1.0)
1 Char	1.69 (-31)	78.8 (-2.5)	86.0 (-2.4)	87.8 (-1.1)
Last Char	1.68 (-31)	77.3 (-4.4)	85.5 (-3.0)	88.1 (-0.8)

Table 2: Computational efficiency comparison. Values in parentheses are in percent and show relative performance differences from MLM results.

Model	F1↑ w/ Miss	F1↑ w/o Miss	Miss↓
MLM	88.1	94.4	6.6
First			
9 Chars	87.4 (-0.9)	94.2 (-0.2)	7.2 (+9.8)
5 Chars	87.9 (-0.3)	94.0 (-0.3)	6.5 (-0.8)
4 Chars	87.9 (-0.2)	94.3 (0.0)	6.8 (+3.1)
3 Chars	87.6 (-0.6)	94.3 (0.0)	7.1 (+8.1)
2 Chars	86.8 (-1.5)	93.8 (-0.6)	7.4 (+12.4)
1 Char	86.1 (-2.3)	93.9 (-0.5)	8.2 (+25.0)
First Char	85.6 (-2.9)	93.9 (-0.5)	8.8 (+34.0)
Last			
9 Chars	88.4 (+0.3)	94.3 (-0.1)	6.3 (-5.1)
5 Chars	87.5 (-0.7)	94.2 (-0.2)	7.0 (+6.7)
4 Chars	87.6 (-0.6)	94.1 (-0.3)	6.8 (+3.7)
3 Chars	87.0 (-1.3)	94.0 (-0.4)	7.5 (+13.1)
2 Chars	87.0 (-1.3)	94.0 (-0.4)	7.5 (+13.7)
1 Char	86.0 (-2.4)	93.7 (-0.7)	8.2 (+24.2)
Last Char	85.5 (-3.0)	94.1 (-0.3)	9.1 (+38.2)

Table 3: Results on the SQuAD dev set. "Miss" denotes the percentage of samples with no overlap between predicted and gold spans. Values in parentheses are in percent and show relative performance differences from MLM results.

operations (FLOPs)[6] required for n Chars, First Char, Last Char, and MLM along with its downstream performance. We can see a clear trade-off between computational efficiency and downstream performance. Smaller n drastically reduces FLOPs with the maximum relative reduction of 31% for First Char and Last Char, while larger n has small reduction rates with the minimum value of 6% for First and Last 9 Chars. To obtain comparable results to MLM, as we discussed in the RQ2, we need at least $n = 4$ (First and Last) complexity, which can only reduce 18% and 20% of FLOPs, respectively. These results suggest that we need careful cost-benefit consideration when using a masked pretraining objective on the basis of target performance and computational costs.

How can we select an optimal complexity? Finally, we discuss how we can decide an optimal complexity for a specific downstream task by analyzing how the task complexity affects the performance in detail. Here, we take SQuAD as an example case, where we observed a large relative difference of 3.0% in Table 2. Table 3 lists the results on SQuAD, including the ratio of mis-detection, i.e., the percentage of samples with no overlap between predicted and gold spans, and F1 scores calculated without mis-detected cases. We found that simple masked objectives were likely to suffer from mis-detection with the worst performance degradation of 38.2% for Last Char, which is far larger than those observed in other metrics and corpora (see Tables 2 and 7 in Appendix). In contrast, the relative performance difference values of F1 scores computed without mis-detected samples show quite similar trends to other high-resource corpora, which are typically less than 2% at a maximum. These results imply that the task complexity mainly contributes to an increase/decrease in the number of mis-detections in SQuAD, and selecting a complex masked objective (e.g., First 5 Chars) is a safeguard option to minimize the effect of misditection. Therefore, different downstream tasks might have different optimal complexities due to their characteristics and evaluation metrics, as observed in the example case above. We leave thorough investigation of these effects as future work.

6 Conclusion

This paper analyzed the impact of masked objective complexities over downstream performance, motivated by the assumption that the lack of task complexity in simple masked pretraining objectives (e.g., First Char) affects the performance degradation compared to MLM. Experiments using the GLUE, SQuAD, and UD datasets revealed that the

[6]We use FLOPs to evaluate computational costs following Clark et al. (2020).

task complexity significantly affected downstream performance with at least 35.7% of the MLM prediction classes needed to perform comparably to MLM on at least one of the high-resource corpora. Our analysis also showed that there exists a trade-off between downstream performance and computational efficiency, and different downstream tasks might have different optimal complexities. Future work includes analyzing other properties (e.g., fairness) with respect to task complexity.

Limitations

Model Architecture Due to our computational resource constraints, we only used the BERT base architecture. We cannot confirm whether our results and observations are transferable to any other Transformer-based architectures, especially for larger ones.

Randomness We did not run pretraining for multiple times with different random seeds due to the limited computational resources and research budgets, though we fine-tuned models five times each with different random seeds in any downstream tasks. This might affect the overall results shown in the paper.

Languages Other than English It is uncertain whether any results and conclusions presented in this paper are applicable to any other languages other than English, as our experiments are solely on English data. We may need further experiments especially for languages that do not belong to the same language family as English, such as Chinese and Japanese.

Ethics Statement

This work does not involve any sensitive data but only uses publicly available data, including Wikipedia, GLUE, SQuAD, and UD as explained in the paper. Although we plan to release the resulting models, they might perform unfairly in some circumstances, as reported in Baldini et al. (2022). We highly recommend users to refer to studies on debiasing pretrained language models, such as Guo et al. (2022).

Acknowledgements

We would like to thank anonymous ACL 2023 reviewers and Yuichi Sasazawa for their insightful comments. We also would like to thank Dr. Masaaki Shimizu for the maintenance and management of the large computational resources used in this paper.

References

Ahmed Alajrami and Nikolaos Aletras. 2022. How does the pre-training objective affect what large language models learn about linguistic properties? In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 131–147, Dublin, Ireland. Association for Computational Linguistics.

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.

Ioana Baldini, Dennis Wei, Karthikeyan Natesan Ramamurthy, Moninder Singh, and Mikhail Yurochkin. 2022. Your fairness may vary: Pretrained language model fairness in toxic text classification. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2245–2262, Dublin, Ireland. Association for Computational Linguistics.

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.

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

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

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, MLCW'05, page 177–190, Berlin, Heidelberg. Springer-Verlag.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of

deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *International Conference on Learning Representations*.

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

Yue Guo, Yi Yang, and Ahmed Abbasi. 2022. Autodebias: Debiasing masked language models with automated biased prompts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1012–1023, Dublin, Ireland. Association for Computational Linguistics.

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*.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

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

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.

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 H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *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.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Chris Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 2897–2904, Reykjavik, Iceland. European Language Resources Association (ELRA).

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

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*.

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

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

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

Atsuki Yamaguchi, George Chrysostomou, Katerina Margatina, and Nikolaos Aletras. 2021. Frustratingly simple pretraining alternatives to masked language modeling. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3116–3125, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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 *The IEEE International Conference on Computer Vision (ICCV)*.

Hyperparameters	**Values**
Batch size	128
Total training steps	500,000
Adam ϵ	1e-6
Adam β_1	0.9
Adam β_2	0.999
Sequence length	512
Learning rate	1e-4 for Last Char 2e-4 for other models
Learning rate schedule	linear warmup
Warmup steps	10,000
Weight decay	0.01
Attention dropout	0.1
Dropout	0.1

Table 4: Hyperparameters for pretraining.

Appendices

A Masked n Characters Prediction

The task of predicting first or last n characters of a masked token (n Chars) is trained with the token-level cross-entropy loss averaged over the masked ones only, following First Char (Yamaguchi et al., 2021). We mask 15% of input tokens the same as in BERT.

Our method generates a label dictionary for n Chars similar to that of First Char. We used a pretrained tokenizer of RoBERTa (`roberta-base`) provided by the Transformers library and generated a label of each token in the vocabulary by picking out a specified number of characters. We did not count a special blank character of Ġ when generating labels. The average number of characters per token in the `roberta-base` vocabulary is 5.72 ± 1.78.

B Detailed Experimental Setup

We trained our models with four NVIDIA Tesla V100 (32GB) for pretraining and one for fine-tuning. Note that we used eight V100 GPUs for MLM to match the total batch size of 128 used for other models.

B.1 Pretraining

We pretrained all models for 500k steps following Alajrami and Aletras (2022) and optimized the models with AdamW (Loshchilov and Hutter, 2019). Table 4 shows the hyperparameter settings used in pretraining.

B.2 Fine-tuning

Table 5 lists the hyperparameters for fine-tuning models on GLUE, SQuAD, and UD benchmarks.

Hyperparameters	GLUE	SQuAD	UD
Batch size	32	24	32
Maximum number of epochs	20	10	10
Adam ϵ	1e-8	1e-8	1e-8
Adam β_1	0.9	0.9	0.9
Adam β_2	0.999	0.999	0.999
Sequence length	128	384	512
Learning rate	3e-5	3e-5	5e-5 for BERT, 1e-3 for BAP
Learning rate schedule	linear warmup	linear warmup	linear warmup
Warmup steps	First 6% of steps	First 6% of steps	First 10% of steps
Weight decay	0.01	0.01	0.01
Attention dropout	0.1	0.1	0.1
Dropout	0.1	0.1	0.1
Early stopping criterion	No improvements over 5% of steps	No improvements over 2.5% of steps	None

Table 5: Hyperparameters for fine-tuning.

For GLUE and SQuAD, we used early stopping. For UD, we compute an average for each token over the top four layers of the BERT hidden representations and use it as an input to BAP. The dimensionalities of arc and relation features given to each biaffine module are 500 and 100, respectively.

C GLUE

C.1 Evaluation Metrics

Following previous work (Aroca-Ouellette and Rudzicz, 2020), we report matched accuracy for MNLI (Williams et al., 2018), Matthews correlation for CoLA (Warstadt et al., 2019), Spearman correlation for STS-B (Cer et al., 2017), accuracy for MRPC (Dolan and Brockett, 2005), F1 scores for QQP[7] and SQuAD, and accuracy for all other tasks, including SST-2 (Socher et al., 2013), QNLI (Wang et al., 2019) and RTE (Dagan et al., 2005; Bar Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009).

C.2 Results

Table 6 shows the detailed GLUE results with standard deviations for each mean value. We can see that standard deviations on low-resource corpora with fewer than 10k samples tend to be larger than that of high-resource corpora.

[7] https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

D Relative Performance Difference from MLM

Table 7 shows the results on GLUE, SQuAD, and UD benchmarks along with the relative performance differences from MLM in percent.

E License

SQuAD and UD (EN-EWT) are distributed under the CC BY-SA 4.0 license. GLUE has different licenses but is freely available for typical machine learning uses. We used all of the corpora for benchmarking purposes only and did not modify their contents.

Model	MNLI 393k	QQP 364k	QNLI 105k	SST 67k	CoLA 8.6k	STS 5.7k	MRPC 3.7k	RTE 2.5k
MLM	82.3 (0.3)	86.9 (0.2)	89.2 (0.1)	91.8 (0.4)	58.0 (2.0)	87.0 (0.4)	86.7 (0.4)	64.8 (1.0)
First 9 Chars	81.6 (0.2)	86.4 (0.3)	89.2 (0.4)	91.9 (0.7)	53.0 (2.5)	85.6 (0.5)	85.2 (1.1)	58.2 (4.5)
First 5 Chars	82.0 (0.2)	86.6 (0.1)	89.3 (0.3)	91.1 (0.3)	51.8 (2.4)	85.6 (0.3)	85.5 (0.2)	59.2 (2.2)
First 4 Chars	82.0 (0.2)	86.6 (0.2)	89.6 (0.3)	91.3 (0.4)	54.2 (2.0)	85.5 (0.4)	85.7 (0.7)	57.3 (3.1)
First 3 Chars	81.9 (0.3)	86.8 (0.2)	88.7 (0.7)	90.7 (0.5)	52.0 (1.2)	85.9 (0.4)	85.6 (0.5)	58.9 (2.2)
First 2 Chars	81.1 (0.4)	86.5 (0.1)	88.6 (0.4)	90.8 (0.4)	51.1 (1.0)	85.1 (0.3)	83.7 (1.0)	60.6 (4.8)
First 1 Char	80.5 (0.5)	86.3 (0.1)	88.5 (0.2)	90.4 (0.2)	48.6 (2.0)	84.7 (1.0)	83.3 (2.0)	60.0 (1.4)
First Char	80.7 (0.3)	86.3 (0.2)	88.2 (0.3)	90.6 (0.3)	50.0 (1.7)	85.1 (0.2)	85.4 (0.7)	59.5 (1.1)
Last 9 Chars	82.1 (0.1)	86.7 (0.3)	89.3 (0.3)	91.4 (0.7)	55.0 (1.3)	85.6 (0.3)	85.1 (0.9)	57.6 (2.2)
Last 5 Chars	81.8 (0.1)	86.4 (0.1)	89.1 (0.3)	91.3 (0.6)	54.8 (1.8)	85.8 (0.3)	85.4 (1.0)	58.7 (4.8)
Last 4 Chars	81.6 (0.2)	86.6 (0.1)	89.4 (0.2)	90.2 (0.5)	56.0 (0.7)	85.6 (0.7)	86.2 (0.5)	56.9 (2.5)
Last 3 Chars	81.3 (0.2)	86.4 (0.4)	88.9 (0.3)	91.0 (0.7)	53.0 (2.0)	84.9 (0.2)	84.9 (1.0)	56.1 (3.7)
Last 2 Chars	81.0 (0.4)	86.3 (0.3)	88.0 (0.3)	90.7 (0.5)	50.7 (2.6)	84.5 (0.6)	85.7 (0.9)	58.6 (2.1)
Last 1 Char	80.2 (0.2)	86.3 (0.2)	87.9 (0.4)	90.4 (0.4)	54.8 (1.8)	84.6 (0.4)	84.8 (1.6)	61.2 (0.8)
Last Char	79.8 (0.3)	86.0 (0.1)	87.5 (0.1)	90.2 (0.3)	48.8 (2.6)	85.2 (0.3)	85.2 (0.6)	55.7 (3.4)

Table 6: Results on GLUE dev sets with standard deviations over five runs in parentheses. Values under dataset names are the number of their corresponding training samples.

Model	MNLI 393k	QQP 364k	QNLI 105k	SST 67k	CoLA 8.6k	STS 5.7k	MRPC 3.7k	RTE 2.5k	GLUE Avg.	SQuAD 88k	UD 13k
MLM	82.3	86.9	89.2	91.8	58.0	87.0	86.7	64.8	80.8	88.1	88.8
First											
9 Chars	81.6 (-0.9)	86.4 (-0.6)	89.2 (0.0)	91.9 (+0.1)	53.0 (-8.7)	85.6 (-1.7)	85.2 (-1.6)	58.2 (-10.2)	78.9 (-2.4)	87.4 (-0.9)	88.5 (-0.3)
5 Chars	82.0 (-0.4)	86.6 (-0.3)	89.3 (+0.1)	91.1 (-0.7)	51.8 (-10.8)	85.6 (-1.6)	85.5 (-1.4)	59.2 (-8.7)	78.9 (-2.4)	87.9 (-0.3)	88.5 (-0.4)
4 Chars	82.0 (-0.4)	86.6 (-0.4)	89.6 (+0.4)	91.3 (-0.5)	54.2 (-6.7)	85.5 (-1.7)	85.7 (-1.1)	57.3 (-11.7)	79.0 (-2.3)	87.9 (-0.2)	88.8 (0.0)
3 Chars	81.9 (-0.5)	86.8 (-0.2)	88.7 (-0.5)	90.7 (-1.2)	52.0 (-10.5)	85.9 (-1.2)	85.6 (-1.2)	58.9 (-9.1)	78.8 (-2.5)	87.6 (-0.6)	88.1 (-0.8)
2 Chars	81.1 (-1.5)	86.5 (-0.5)	88.6 (-0.6)	90.8 (-1.1)	51.1 (-12.0)	85.1 (-2.2)	83.7 (-3.4)	60.6 (-6.5)	78.4 (-3.0)	86.8 (-1.5)	87.8 (-1.1)
1 Char	80.5 (-2.2)	86.3 (-0.7)	88.5 (-0.8)	90.4 (-1.5)	48.6 (-16.3)	84.7 (-2.7)	83.3 (-3.8)	60.0 (-7.5)	77.8 (-3.8)	86.1 (-2.3)	87.6 (-1.4)
First Char	80.7 (-2.0)	86.3 (-0.7)	88.2 (-1.1)	90.6 (-1.3)	50.0 (-13.8)	85.1 (-2.2)	85.4 (-1.5)	59.5 (-8.2)	78.2 (-3.2)	85.6 (-2.9)	87.8 (-1.2)
Last											
9 Chars	82.1 (-0.3)	86.7 (-0.2)	89.3 (+0.1)	91.4 (-0.4)	55.0 (-5.2)	85.6 (-1.7)	85.1 (-1.8)	57.6 (-11.1)	79.1 (-2.2)	88.4 (+0.3)	88.7 (-0.2)
5 Chars	81.8 (-0.6)	86.4 (-0.6)	89.1 (-0.1)	91.3 (-0.6)	54.8 (-5.6)	85.8 (-1.4)	85.4 (-1.4)	58.7 (-9.5)	79.2 (-2.1)	87.5 (-0.7)	88.5 (-0.4)
4 Chars	81.6 (-0.9)	86.6 (-0.3)	89.4 (0.3)	90.2 (-1.7)	56.0 (-3.5)	85.6 (-1.7)	86.2 (-0.5)	56.9 (-12.2)	79.1 (-2.2)	87.6 (-0.6)	88.4 (-0.4)
3 Chars	81.3 (-1.2)	86.4 (-0.6)	88.9 (-0.4)	91.0 (-0.9)	53.0 (-8.7)	84.9 (-2.4)	84.9 (-2.0)	56.1 (-13.5)	78.3 (-3.1)	87.0 (-1.3)	88.4 (-0.5)
2 Chars	81.0 (-1.6)	86.3 (-0.7)	88.0 (-1.3)	90.7 (-1.2)	50.7 (-12.7)	84.5 (-3.0)	85.7 (-1.1)	58.6 (-9.6)	78.2 (-3.3)	87.0 (-1.3)	88.0 (-1.0)
1 Char	80.2 (-2.5)	86.3 (-0.8)	87.9 (-1.4)	90.4 (-1.5)	54.8 (-5.5)	84.6 (-2.7)	84.8 (-2.1)	61.2 (-5.6)	78.8 (-2.5)	86.0 (-2.4)	87.8 (-1.1)
Last Char	79.8 (-3.0)	86.0 (-1.1)	87.5 (-1.9)	90.2 (-1.8)	48.8 (-15.9)	85.2 (-2.1)	85.2 (-1.7)	55.7 (-14.1)	77.3 (-4.4)	85.5 (-3.0)	88.1 (-0.8)

Table 7: Results on GLUE, SQuAD, and UD datasets along with their relative performance differences from MLM in percent.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Limitations

☑ A2. Did you discuss any potential risks of your work?
Ethics Statement

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

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

B ☑ Did you use or create scientific artifacts?

Section 3, Section 4, and Section 5.

☑ B1. Did you cite the creators of artifacts you used?
Section 1, Section 3, and Appendix C.

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Appendix E. The datasets we used are widely-used open-source benchmark datasets in NLP, and we just utilized them to evaluate downstream performances of models.

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
The datasets we used are widely-used open-source benchmark datasets in NLP, and we just utilized them to evaluate downstream performances of models, which is exactly the intended use.

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
The datasets we used are widely-used open-source benchmark datasets in NLP.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 3.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Table 1.

C ☑ Did you run computational experiments?

Section 4 and Section 5.

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 3 and Appendix.

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

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 3 and Appendix.

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 3, Section 4, and Section 5.

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 3 and Appendix.

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

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.