Continual Pre-training on Character-level Noisy Texts Makes Decoder-based Language Models Robust Few-shot Learners

Takeshi Kojima, Yutaka Matsuo, Yusuke Iwasawa
The University of Tokyo, Japan
t.kojima@weblab.t.u-tokyo.ac.jp

Abstract

Recent decoder-based pre-trained language models (PLMs) generally use subword tokenizers. However, adding character-level perturbations drastically changes the delimitation of texts by the tokenizers, leading to the vulnerability of PLMs. This study proposes a method of continual pre-training to convert decoder-based PLMs with subword tokenizers into perturbation-robust few-shot in-context learners. Our method continually trains decoder-based PLMs to predict the next tokens conditioning on artificially created character-level noisy texts. Since decoder-based language models are auto-regressive, we skip noised words from the target optimization. In addition, to maintain the same word prediction performance under noisy text as clean text, our method employs word distribution matching between the original PLMs and training models. We conducted experiments on various subword-based PLMs, including GPT2, Pythia, Mistral, Gemma2, and Llama3, ranging from 1B to 8B parameters. The results demonstrate that our method consistently improves the performance of few-shot in-context learning on downstream tasks which contain actual typos or misspellings as well as artificial noise.[1]

1 Introduction

Recent decoder-based pre-trained language models (PLMs) have demonstrated remarkable generalization performance in natural language processing (NLP) (Achiam et al., 2023; Dubey et al., 2024; Gemma Team et al., 2024). These general-purpose PLMs exhibit high performance on versatile tasks with zero-shot or few-shot in-context learning settings after pre-training (Brown et al., 2020).

Subword-based tokenization is a mainstream approach for most of the recent PLMs to tokenize texts for the model inputs, such as Byte Pair Encoding (BPE) (Sennrich, 2015) or SentencePiece (Kudo, 2018). Subword-based tokenization is a compromise approach between word-level and character-level tokenization to achieve efficient data compression and fast inference while maintaining the performance of the models.

However, it is found that adding character-level perturbations drastically changes the delimitation of texts by subword-based tokenizers, leading to the vulnerability of PLMs. For example, Cao et al. (2023) have demonstrated that when we scramble characters except for the first and last ones within each word in a text, the reconstruction performance, as well as downstream task performance of most large language models (LLMs), significantly drops although humans can recognize it (a.k.a typoglycemia).

Some prior work has improved the robustness of decoder-based PLMs by pre-training the models that do not utilize subword tokenizer from scratch (Tay et al., 2021; Xue et al., 2022). These methods generally delimitate words into character-level or byte-level pieces before the input to neural network architecture. In addition, its structure is specially designed for efficient calculations specific to the tokenization method. Therefore, this method requires a considerable computational burden because we must pre-train the model from scratch.

This study proposes a method to convert off-the-shelf decoder-based PLMs with subword tokenizers into perturbation-robust few-shot in-context learners by continually pre-training the models. Inspired by the previous work (Aepli and Sennrich, 2022) that makes **encoder-based** PLMs more robust by continually pre-training on character-level noisy texts, we propose a method to turn **decoder-based** PLMs into perturbation-robust few-shot in-context learners

[1]Our code is available at https://github.com/kojima-takeshi188/charnoise.

Transactions of the Association for Computational Linguistics, vol. 13, pp. 831–847, 2025. https://doi.org/10.1162/tacl.a.21
Action Editor: Yuval Pinter. Submission batch: 10/2024; Revision batch: 2/2025; Published 8/2025.

Method	Architecture	Tokenizer	Training Data	Need pre-training from scratch?	Need fine-tuning for each specific task?
Clark et al. (2022)	Encoder	Character (Byte)	Clean	✗ Yes	✗ Yes
Aepli and Sennrich (2022); Srivastava and Chiang (2023)	**Encoder**	**Subword**	**Artificial Noise**	✓ No (Use PLMs or **Continually pre-train PLMs)**	✗ Yes
Tay et al. (2021); Xue et al. (2022)	Decoder (*1)	Character (Byte)	Clean	✗ Yes	✗ Yes
Huang et al. (2023)	Decoder (*1)	Subword	Artificial Noise	✓ No (Use PLMs)	✗ Yes (*2)
Ours	**Decoder**	**Subword**	**Artificial Noise**	✓ No (**Continually pre-train PLMs)**	✓ No (few-shot ICL)

Table 1: Method comparison of robustifying PLMs against character-level perturbations. Methods that did not assume PLMs (i.e., models that are not pre-trained) are excluded from this table. See the details in Section 2. (*1) Encoder-Decoder Models, e.g., T5-based model. (*2) This method is a supervised learning approach only for tasks of character-level manipulation of words, thus limiting the generalizability to many NLP tasks.

by continually pre-training the models on character-level noisy texts. Specifically, our method trains decoder-based PLMs to predict the next token conditioning on artificially curated character-level noisy texts. Since decoder-based language models are auto-regressive, we skip noised words from the target optimization. In addition, to avoid degrading the performance of the original model, our method introduces the minimization of KL distance for the predicted word distributions.

We conducted experiments on six decoder-based PLMs with subword tokenizers ranging from 1B to 8B parameters. The experimental results demonstrate that our method consistently improves the performance on artificially noised downstream tasks, including classification, summarization, and reconstruction with few-shot in-context learning settings. Interestingly, our method generalizes not only to in-distribution noises but also to out-of-distribution noise types. Our method has also been shown to improve the performance of real-world noisy tasks such as typo correction within texts and classification of texts containing real-world spelling errors, while maintaining the performance of general downstream tasks which does not contain noises.

2 Related Work

Subword Tokenizers Subword tokenizers split texts into words and word pieces, usually longer chunks than character-level tokenization. The representative subword tokenization methods include BPE (Sennrich, 2015) and unigram language models (also known as SentencePiece) (Kudo, 2018). Due to the computational efficiency via text data compression with limited vocabulary size, subword tokenization has become a de facto standard method for recent PLMs (Dubey et al., 2024; Gemma Team et al., 2024).

Vulnerability to Character-level Noise Some studies report that even high-performance language models suffer from input perturbations, including word-level ones (Sinha et al., 2021; Abdou et al., 2022) and character-level ones (Moradi and Samwald, 2021; Cao et al., 2023; Shin and Kaneko, 2024). Character-level perturbations are seemingly trivial noise for human readers. However, they make a big difference from the perspective of subword tokenizers. For example, when we put noise in ‘‘apple’’ to make it ‘‘applle’’, the tokenization result changes from [‘apple’] (one token) to [‘app’, ‘l’, ‘le’] (three tokens). This drastic input change is expected to cause subword-based language model performance degradation.

Methods of Robustifying PLMs to Character-level Noise We mainly summarize prior work on improving robustness to character-level perturbation for PLMs. Table 1 summarizes method comparison results. Some studies (Tay et al., 2021; Libovický et al., 2022) recognize text as a sequence of bytes expressed in UTF-8 and use the value of each byte (0–255) as token IDs. Then, they propose character-aware encoder-decoder models based on T5 (Roberts et al., 2019) by changing network architecture and training the models from scratch. Clark et al. (2022) have also proposed tokenization-free character-level encoder models based on BERT (Devlin, 2018). However, these methods require a substantial computational burden during the pre-training phase.

Other studies have proposed more lightweight approaches: fine-tuning off-the-shelf PLMs for specific tasks. Aepli and Sennrich (2022) and Srivastava and Chiang (2023) updated parameters of encoder models, i.e., BERT by continual pre-training and/or finetuning on artificially noised texts to make the models robust against input perturbation and cross-lingual tasks. However, research on continual pre-training of decoder models with noisy texts has not been conducted so far.

Huang et al. (2023) have proposed injecting character-level information into assigned dimensions in hidden states at the fine-tuning phase by interchange intervention training (Geiger et al., 2022). However, this method is a supervised learning approach only for tasks of character-level manipulation of words, e.g., character reversal or unscrambling. Therefore, we cannot use this method for learning most NLP tasks such as text classification or continuation of writing a sentence. In addition, we cannot apply this method to continual pre-training for decoder-based language models because it generally adopts a self-supervised learning approach, i.e., next token prediction (Achiam et al., 2023; Dubey et al., 2024).

Other Robust Methods before PLMs Even before the advent of PLMs, many studies conducted research on the character-level robustness of language models. Some studies (Kim et al., 2016; Wieting et al., 2016; Akbik et al., 2019) input character-level information into language models, validating the effectiveness and robustness on various tasks.

Other studies improved robustness against the character-level perturbations by using subword tokenizers. Stochastic tokenization (Hiraoka et al., 2019) tokenized a training text stochastically by using a language model specialized for subword tokenization to verify the performance improvement on sentiment classification tasks. Similarly, BPE dropout (Provilkov et al., 2020) tokenized a training text with a BPE tokenizer by sampling segmentation of text stochastically to make the model robust against infrequent tokens, demonstrating the performance gain in machine translation tasks. However, the effectiveness of these methods has not been validated in the learning paradigm of PLMs, i.e., unsupervised pre-training followed by fine-tuning or zero-shot/few-shot in-context learning in downstream tasks.

3 Our Method

Inspired by the previous work (Aepli and Sennrich, 2022) which makes **encoder-based** PLMs more robust by continually pre-training on character-level noisy texts, we propose a method to turn **decoder-based** PLMs into perturbation-robust few-shot in-context learners by continually pre-training the models on character-level noisy texts. However, we cannot directly apply the method of encoder-based PLMs to decoder-based ones because of the difference in learning mechanisms: encoder-based PLMs are trained by masked token prediction, while decoder-based PLMs are trained by autoregression or next token prediction. Therefore, we propose a new learning approach for decoder-based PLMs. See Figure 1 to grasp the differences in learning methods between encoder and decoder PLMs for noisy text. In this section, first, we define character-level artificial noise used for our continual pre-training. Second, we define an objective function specific to decoder-based PLMs for continual pre-training on the defined noisy text.

3.1 Artificially Noised Corpus

We put character-level noises into the pre-training corpus. Table 2 lists synthetic noise types for making noisy texts. Specifically, we define four noise types {Insert, Delete, Replace, Swap} as in-distribution (ID) noises because they are used for both training and evaluation. These four noise types are based on prior work (Srivastava and Chiang, 2023). We also define the three additional noise types {Split, Uppercase, Mask} as

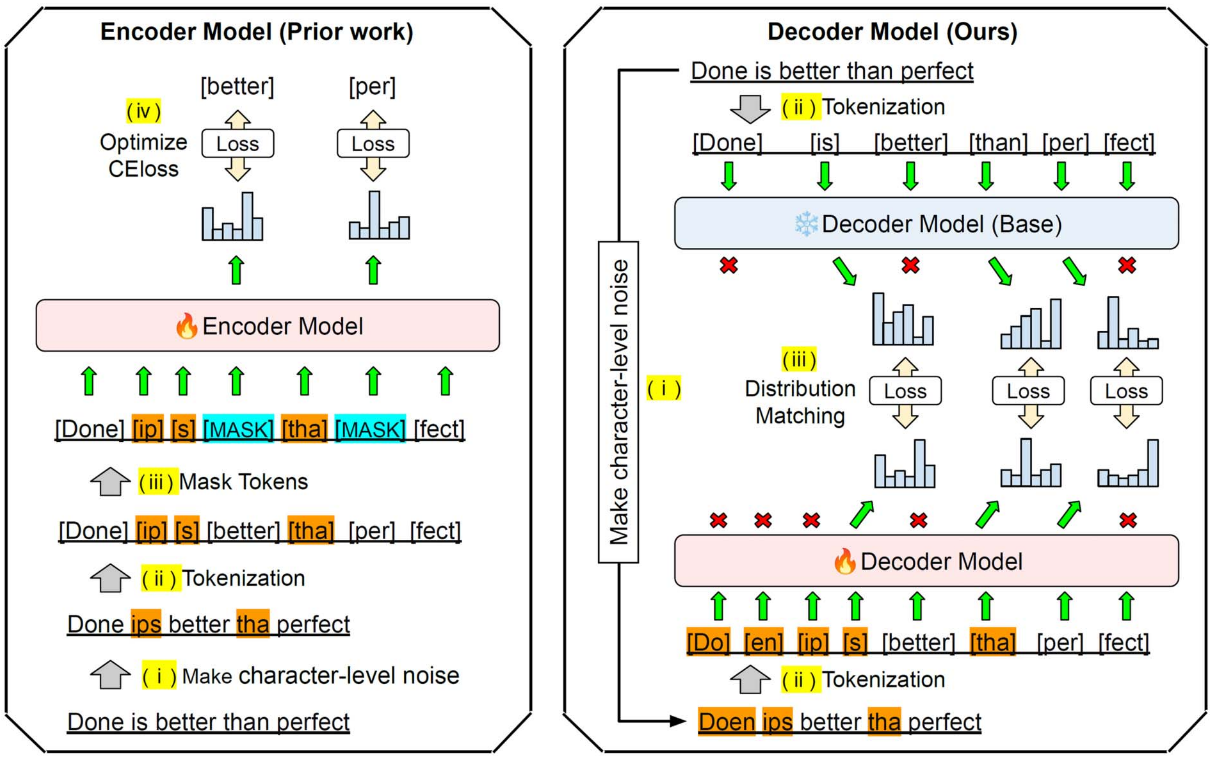
Encoder Model (Prior work)
[better]
[per]
(iv)
Optimize CEloss
Loss
Loss
Encoder Model
[Done] [ip] [s] [MASK] [tha] [MASK] [fect]
(iii) Mask Tokens
[Done] [ip] [s] [better] [tha] [per] [fect]
(ii) Tokenization
Done ips better tha perfect
(i) Make character-level noise
Done is better than perfect
Decoder Model (Ours)
Done is better than perfect
(ii) Tokenization
[Done] [is] [better] [than] [per] [fect]
Decoder Model (Base)
Make character-level noise
(i)
(iii)
Distribution Matching
Loss
Loss
Loss
Decoder Model
[Do] [en] [ip] [s] [better] [tha] [per] [fect]
(ii) Tokenization
Doen ips better tha perfect

Figure 1: Overview of continual pre-training methods on noisy texts. Left: A method for encoder-based PLMs (Aepli and Sennrich, 2022). Right: A method for decoder-based PLMs (ours).

	Noise Type	Example	Tokenization result (by GPT-2)
ID	Insert	View → Viiew	[‘View’] → [‘Vi’, ‘iew’]
	Delete	Musicians → Musicins	[‘Mus’, ‘icians’] → [‘Music’, ‘ins’]
	Replace	usually → usuylly	[‘usually’] → [‘us’, ‘uy’, ‘lly’]
	Swap	recorded → cerorded	[‘recorded’] → [‘cer’, ‘ord’, ‘ed’]
OOD	Split	know → kn-ow	[‘know’] → [‘kn’, ‘-’, ‘ow’]
	Uppercase	unbiased → unbiAsed	[‘un’, ‘biased’] → [‘un’, ‘bi’, ‘As’, ‘ed’]
	Mask	have → hav_	[‘have’] → [‘h’, ‘av’, ‘_’]

Table 2: List of synthetic noise type. ID: In-distribution noises which are used for both training and test. OOD: Out-of-distribution noises which are used only for test.

out-of-distribution (OOD) noises because they are used only for evaluation.

- **Insert** one randomly chosen alphabet into a random position in a word.
- **Delete** one character at a randomly chosen position from a word.
- **Replace** one character at a randomly chosen position in a word with a randomly sampled alphabet letter.
- **Swap** two characters at randomly chosen positions in a word.
- **Split** a word into two pieces by inserting ‘‘-’’ (hyphen) within a word.
- **Uppercase** one character at a randomly chosen position in a word.
- **Mask** one character at a randomly chosen position in a word.

Table 2 describes an example for each noise type and its tokenization results by GPT-2 tokenizer.

We define noise severity s as the probability of applying noise to each word in a document. Suppose that each mini-batch consists of a total of B documents $\{x_1, \ldots, x_i, \ldots, x_B\}$. For i-th

Original Text:
Also time limits can be adapted. Timing will be continued if possible. Due to security reasons the OC has the right to cancel, stop or end the competition if necessary.
Syntheiszed Noisy Text:
Also tiem limits car be adpted . Timgni wfill be continued if pssible . Duqe to security reasonso yhe OC zas the righht to cancel, stopg or ed the competition if necessarcy .

Table 3: Example of artificial noise to continually pre-train models. The original text is extracted from FineWeb.

document $x_i = \{x_{i,1}, .., x_{i,j}, .., x_{i,T_i}\}$ which consists of T_i words, we set the noise severity s_i by randomly sampling a value from the following distribution.

$$s_i = \begin{cases} 0 & (u_1 \leq 0.5) \\ u_2 & (u_1 > 0.5), \end{cases} \tag{1}$$

where u_1 and u_2 are independently sampled values from a uniform distribution. The intuition behind the above conditional formula is that in order to retain the performance on clean text corpus, we mix both clean text corpus and noisy text corpus. We set the mixing ratio as 1:1 for a hyperparameter setting. A synthesized noisy text $x'_i = \{x'_{i,1}, .., x'_{i,j}, .., x'_{i,T_i}\}$ is created by applying the following conditional formula for each word.

$$x'_{i,j} = \begin{cases} noise(x_{i,j}) & (u_{i,j} \leq s_i) \\ x_{i,j} & (u_{i,j} > s_i), \end{cases} \tag{2}$$

where $u_{i,j}$ denotes randomly selected value from uniform distribution, and $noise(\cdot)$ denotes character-level noising function randomly chosen among {Insert, Delete, Replace, Swap}. In this experiment, we apply noise to only ASCII words to avoid unexpected errors during tokenization. Table 3 describes an example of synthesized noisy text following this algorithm.

3.2 Objective Function

We continually pre-train decoder-based PLMs with the aforementioned synthesized noisy corpus by following the concept of next-token prediction with several modifications specific to decoder-based models. This section defines the objective function for the training.

As a pre-processing, we convert clean text x_i and corresponding synthesized noisy text x'_i into a sequence of tokens by using subword tokenizers: $\tilde{z}_i = \{\tilde{z}_{i,1}, .., \tilde{z}_{i,j}, .., \tilde{z}_{i,\tilde{T}_i}\}$ as a sequence of clean tokens and $z'_i = \{z'_{i,1}, .., z'_{i,j}, .., z'_{i,T'_i}\}$ as a sequence of noisy tokens. Note that sequence length $\tilde{T}_i$ and T'_i are different after tokenization.

Let P_{θ^*} and P_θ be language models with original parameter θ^* and training parameter θ. At the beginning of the continual pre-training, $\theta = \theta^*$. For simplicity, we denote a probability of outputting a vocabulary v at position j conditioning on previous tokens in the noisy text as below.

$$P'_\theta(v_{i,j}) = P_\theta(v_{i,j} | z'_{i,1}, \ldots, z'_{i,j-1}). \tag{3}$$

Since decoder-based language models are autoregressive, our method skips noisy words from the target optimization to prevent models from learning to output noisy texts. Recall that there are still some clean words in noisy texts depending on the value of s_i (See Equation 2). We define a subset of token indices belonging to clean words in a noisy text as J_i. Each index $j \in J_i$ in noisy text has a corresponding index in clean text. We define the index mapping function as $k(j)$. Here, we denote a probability of outputting a vocabulary v at position $k(j)$ conditioning on previous tokens in a clean text as below.

$$P_{\theta^*}(v_{i,k(j)}) = P_{\theta^*}(v_{i,k(j)} | \tilde{z}_{i,1}, \ldots, \tilde{z}_{i,k(j)-1}). \tag{4}$$

To avoid the model from predicting noise, our method employs word distribution matching only for clean tokens between the original PLMs and training models. Specifically, Our objective function $\mathcal{L}$ minimizes soft label cross-entropy between $P_{\theta^*}(v_{i,k(j)})$ and $P'_\theta(v_{i,j})$ as below.

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{B} \mathcal{L}_i, \quad N = \sum_{i=1}^{B} |J_i|, \tag{5}$$

$$\mathcal{L}_i = -\sum_{j \in J_i} \sum_{v \in V} P_{\theta^*}(v_{i,k(j)}) \log P'_\theta(v_{i,j}). \tag{6}$$

For each mini-batch, we minimize the above objective function with regard to trainable parameters θ using gradient descent approaches. The overview of our method is described at Figure 1.

4 Experiment

4.1 Model and Dataset

To validate the effectiveness of our method, we conduct experiments with a wide variety of PLMs with subword tokenizers. Specifically, we use five models including GPT2-XL (1.5B) (Radford et al., 2019), Pythia-2.8B (Biderman et al., 2023), Mistral-7B(-v0.1) (Jiang et al., 2023), Gemma2-2B (Gemma Team et al., 2024), and Llama3-8B (Dubey et al., 2024). All the model parameters are downloaded from HuggingFace (Wolf et al., 2019). Although each model used different corpora at the pre-training stage, they all share the same point: They included website crawl data on the Internet. In light of this common point in this experiment, we use FineWeb corpus (Penedo et al., 2024) for continual pre-training for all the models. FineWeb comprises over 15T tokens of cleaned and deduplicated English web data from CommonCrawl. Our experiment uses random samples from the ‘‘sample-10BT’’ subset from HuggingFace.

4.2 Training and Inference Settings

To continually pre-train models, we employ LoRA tuning (Hu et al., 2021) instead of full-parameter tuning for the following reasons. When calculating distribution matching loss between the base model and training model in Equation 6, we must run forward pass twice while retaining parameters for each of the two models, which causes memory increase for the training. LoRA tuning can drastically reduce training cost and memory usage for our method because LoRA tuning not only reduces trainable parameters but also retains original parameters unchanged and it can use them by just optionally disabling additional LoRA weights. In addition, we assume that LoRA tuning tends to prevent catastrophic forgetting (degradation) because LoRA is hypothesized to constrain the trained model from diverging significantly from the base model by training fewer parameters (Biderman et al., 2024). We set LoRA weight to all the linear layers, including MLP blocks, Attention blocks, Projection, and Embedding layers. We set r = 16, alpha = 8, and dropout = 0.0 for LoRA.

We train each model for 1000 global steps. For training, we set the context length to 1024 for GPT2-XL, 2048 for Pythia-2.8B (the same as the default settings), and 8192 for the other models. In order to mimic the pre-training stage and efficiently use computational resources, we concatenate each document with a BOS token (or EOS token if it is not defined) and expand the input sequence up to the context length limitation. We set the mini-batch size to 64 for GPT2-XL, 32 for Pythia-2.8B, and 8 for the other models to equalize the total number of tokens (= context length * mini-batch size * global steps) used for continual pre-training across models. The total number of tokens is approximately 64M tokens.

We use AdamW with weight decay = 0.01 as optimizer. Learning rate is set as 1e-4 constant (without scheduler setup). To avoid out-of-memory error and achieve faster training and inference, parameter dtype is set as BFloat16, and flash attention v2 (Dao, 2023) is enabled for all the models. All the experiments are run on parallel H200 (141GB) GPU servers, and each experiment is conducted on a single H200 GPU server.

5 Result

To validate that our method of continual pre-training effectively converts subword-based PLMs into perturbation-robust few-shot in-context learners, we conducted the following evaluations after continual pre-training of PLMs: synthesized noisy tasks (classification, reconstruction, and summarization), real-world noisy tasks (typo correction, noisy text classification), and baseline study.

5.1 Synthesized Noisy Tasks

5.1.1 Text Classification

To evaluate if models can predict correct labels given character-level noisy text, we applied noise to SST-2 (Socher et al., 2013) and AG News (Zhang et al., 2015) texts for text classification benchmarks. Both are standard benchmarks with moderate difficulties for the character-level perturbation experiments. SST-2 is a binary classification task that requires classifying review texts as positive or negative. AG News is a four-valued classification task that requires classifying news text into world, sports, technology, and business.

We make noise into the text with a severity level of 0.5. We evaluate the performance under each individual noise type, including four types of ID noise (Insert, Delete, Replace, Swap) and three types of OOD noise (Swap, Split, Uppercase). Additionally, we evaluated ‘‘multi’’ noise, in which we made multiple varieties of ID or OOD noises by randomly changing the noise type for each word. Because of the large number of experiments, we

Task	Model	Method	w/o noise	noise (in-distribution)					noise (out-of-distribution)			
				insert	swap	del.	rep.	multi	mask	u.c.	split	multi
SST-2	GPT2	Base	62.4	53.2	51.1	51.6	52.2	50.7	51.7	53.7	52.8	51.5
	(XL)	Ours	**+2.1**	**+3.2**	**+4.1**	**+2.4**	**+3.7**	**+4.4**	**+0.8**	**+3.8**	**+2.2**	**+1.6**
	Gemma2	Base	93.0	88.9	85.3	88.6	87.8	88.9	90.8	91.6	91.6	90.5
	(2B)	Ours	±0.0	**+1.7**	**+2.6**	**+1.8**	**+1.1**	**+1.9**	**+0.6**	**+1.4**	**+0.3**	±0.0
	Pythia	Base	88.1	81.3	74.3	76.7	77.2	77.1	77.3	79.7	81.4	79.6
	(2.8B)	Ours	**+1.9**	**+3.9**	**+5.4**	**+3.9**	**+4.6**	**+4.6**	**+2.3**	**+1.8**	**+2.6**	**+1.9**
	Mistral	Base	95.2	92.7	88.2	92.7	90.6	90.8	90.0	93.6	94.4	92.3
	(7B)	Ours	−0.3	**+1.9**	**+4.2**	**+0.2**	**+2.6**	**+2.8**	**+1.6**	**+1.0**	**+0.3**	**+1.4**
	Llama3	Base	94.6	92.1	88.9	91.2	89.6	90.9	91.7	93.8	93.9	92.0
	(8B)	Ours	±0.0	**+1.5**	**+2.4**	**+1.6**	**+1.8**	**+2.1**	**+0.3**	**+0.8**	**+0.7**	**+0.7**
AG NEWS	GPT2	Base	66.4	48.4	42.7	50.6	42.3	47.5	28.3	43.2	51.5	39.4
	(XL)	Ours	**+0.8**	**+7.5**	**+8.1**	**+3.6**	**+10.7**	**+5.8**	**+2.7**	**+3.7**	±0.0	**+2.5**
	Gemma2	Base	71.1	65.8	63.5	64.2	65.2	64.4	60.1	65.3	59.7	59.6
	(2B)	Ours	−0.5	**+3.4**	**+3.4**	**+1.8**	**+1.6**	**+2.5**	±0.0	**+1.9**	**+0.1**	**+0.5**
	Pythia	Base	72.2	73.6	70.1	73.3	72.1	73.4	70.3	73.6	76.4	74.4
	(2.8B)	Ours	**+0.8**	**+2.1**	**+2.9**	**+2.6**	**+3.1**	**+2.5**	−0.6	**+1.8**	−1.0	**+1.0**
	Mistral	Base	86.2	82.7	82.6	84.6	83.0	84.3	83.3	83.8	84.3	83.7
	(7B)	Ours	**+0.5**	**+3.5**	**+2.7**	**+2.6**	**+3.1**	**+2.3**	**+2.5**	**+1.7**	**+0.6**	**+1.4**
	Llama3	Base	83.3	78.9	79.3	78.1	79.6	78.5	80.0	80.8	80.6	78.2
	(8B)	Ours	**+0.5**	**+3.1**	**+2.9**	**+4.5**	**+3.3**	**+3.6**	**+1.9**	**+1.8**	**+2.0**	**+3.1**

Table 4: Performance of text classification. Metric is accuracy. del.: delete, rep.: replace, u.c.: uppercase.

limited the total number of evaluation samples to 1000 from the test set for AG News with a strict balance of the number of labels. We used all the validation samples (872) for the evaluation of SST-2. We set the number of in-context exemplars as four for both benchmarks across all the models. Regarding the in-context template, SST-2 used ''Choose a sentiment from ''positive'' or ''negative.'' review: {text} label: {label} . . . (repeat)''. AG News used ''Choose a topic from ''world'', ''sports'', ''business'', or ''technology''. input: {text} type: {label} . . . (repeat)''. We limit each text length to 100 tokens.

Table 4 describes the result of the experiment. The result demonstrates that our method (Ours) does not drop the text classification performance from the original PLMs (Base: Baseline). In addition, our method consistently outperforms the baseline in various types of ID noise settings across models. Interestingly, our method also tends to outperform baselines in OOD noise settings. This tendency implies that our method generalizes to unseen noise types.

We conducted extensive experiments in which the noise severity was changed from 0.0 to 1.0 in 0.25 increments. For simplicity, this experiment only focuses on ID multi-noise and OOD multi-noise. Figure 2 shows the results of SST-2 on Pythia-2.8B and ones of AG News on Mistral-7B. The result clearly shows that when the noise severity is higher, the baselines' performance significantly drops, while our method keeps the performance comparatively good. Eventually, the discrepancy between our method and baseline has become more significant. This result indicates that our method exhibits a more outstanding robustness superior to the original PLMs.

In addition, to confirm the trend of performance change throughout model training, we measured the accuracy of multi-ID noise, multi-OOD noise, and without noise on AG NEWS every 250 steps until 1000 steps. The results in Figure 3 demonstrate that while the performance without noise stays almost the same, the performance of ID noise and OOD noise increases drastically at the beginning stage and keeps rising moderately with some fluctuations until the end of the training.

5.1.2 Text Summarization

To analyze the performance of models in more complex and challenging tasks, we conducted experiments of summarization in character-level noisy input settings. Specifically, we use XSUM (Narayan et al., 2018) benchmark and apply noise to source documents. We made ID-multi noise into source documents with a severity level = 0.5. because of the large number of experiments, we

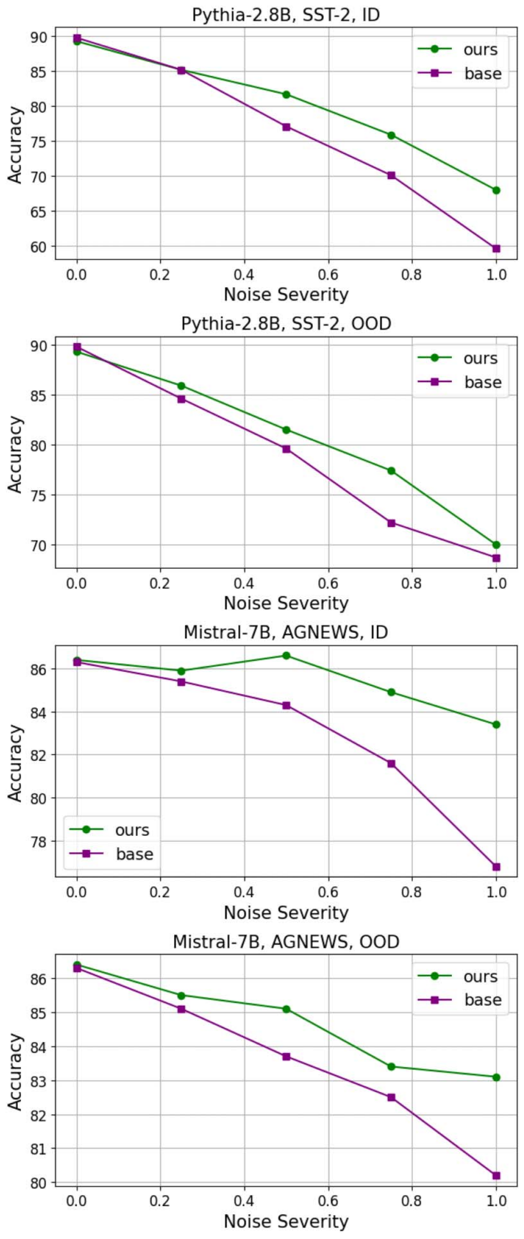

Figure 2: Noise severity and prediction performance.

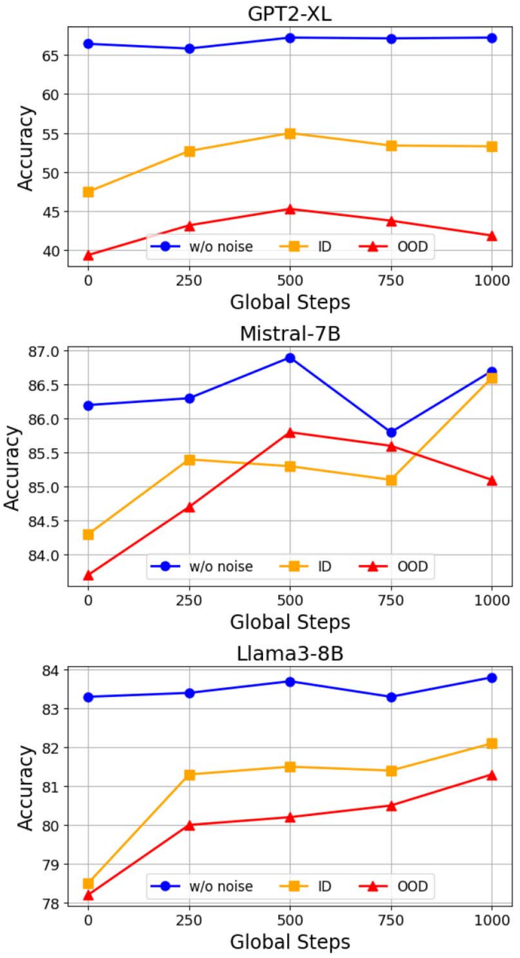

Figure 3: Effect of training data size on performance.

limited the total number of evaluation samples to 200 from the test set. We set the number of in-context exemplars as one across all the models. We used the following in-context template: ‘Summarize the following documents. document: {document} summary: {summary} . . . (repeat)’’. We limit source document length to 400 tokens and limit predicted summary text length to 100.

Results on Table 5 demonstrate that our method consistently outperforms the baseline in a noisy source document setting for Gemma2-2B, Pythia-2.8B, Mistral-7B, and Llama3-8B. The only exception was GPT2-XL. It is assumed that the text generation ability of GPT2-XL was not originally high due to the small parameter size, and continual pre-training on noisy text collapsed the ability.

5.1.3 Text Reconstruction

In order to analyze the performance of models by not only classification tasks but also text generation tasks, we conducted experiments on reconstructing original clean text from artificially character-level noised ones. Specifically, we measure the performance on text reconstruction from typoglycemia (Cao et al., 2023), i.e., scramble characters except for the first and last ones within each word in a text. This noise type is regarded as OOD. We randomly chose 1000 clean English texts from flores200 (Costa-jussà et al., 2022) (devtest subset) and applied typoglycemia noise with severity level $= 0.5$. We set the number of in-context exemplars as four for few-shot ICL across all the models. We used the following in-context template: ‘Fix typos in the following

Model	Method	without noise			with noise		
		ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-1	ROUGE-2	ROUGE-L
GPT2-XL	Base	18.3	3.8	14.5	17.6	2.9	13.1
	Ours	**+0.2**	**+0.2**	**+0.3**	−1.7	−0.4	−0.9
Gemma2-2B	Base	27.8	8.8	21.6	28.0	9.0	22.3
	Ours	−0.8	−0.7	−1.0	**+0.6**	**+0.6**	**+0.5**
Pythia-2.8B	Base	24.7	6.0	19.3	23.0	5.2	17.5
	Ours	−0.5	−0.2	−0.7	**+1.6**	**+0.4**	**+0.8**
Mistral-7B	Base	35.8	14.5	28.5	30.8	11.0	24.5
	Ours	**+0.1**	−0.1	−0.3	**+4.3**	**+2.9**	**+3.7**
Llama3-8B	Base	34.4	13.2	26.8	29.6	10.4	23.5
	Ours	−0.2	**+0.1**	−0.3	**+1.8**	**+0.8**	**+0.6**

Table 5: Perfomance on summarization task from noisy source documents. We use XSUM dataset and applied ID-multi noise with serverity level = 0.5 to source documents.

Model	Method	Accuracy(↑)	Distance(↓)
GPT2-XL	Base	0.6	38.43
	Ours	**+0.2**	+25.72
Gemma2-2B	Base	27.7	9.59
	Ours	**+3.2**	**−0.53**
Pythia-2.8B	Base	1.8	40.70
	Ours	**+2.0**	**−6.10**
Mistral-7B	Base	34.6	8.19
	Ours	**+5.9**	**−1.91**
Llama3-8B	Base	45.7	4.78
	Ours	**+2.8**	**−0.40**

Table 6: Performance on text reconstruction from typoglycemia, i.e., we scramble characters except for the first and last ones within each word in a text.

texts. Source Text: {Source Text} Target Text: {Target Text} . . . (repeat)''. We limit source document length to 50 tokens. We measure the recovery rate by the accuracy of exact matching and the edit distance between the reconstructed text and the original text.

Results in Table 6 demonstrate that our method consistently outperforms the baseline across all the models.

5.2 Real-World Tasks

5.2.1 Spelling Error Correction

To validate the effectiveness of our method on real-world noisy tasks, we conducted experiments of measuring typo and misspelling correction performance on GitHub Typo Corpus and (Hagiwara and Mita, 2020) and BEA-60K (Jayanthi et al., 2020). GitHub Typo Corpus is a large-scale dataset of texts containing misspellings, grammatical errors, and their corrections harvested from GitHub. BEA-60K is a collection of text datasets containing spelling mistakes (6.8% of all tokens) and the corresponding corrections.

We filter out samples whose source document length exceeds 50 tokens. In addition, we limited the total number of evaluation samples to 1000. We set the number of in-context exemplars as four across all the models for few-shot ICL. We used the same in-context template as the typoglycemia task (See Section 5.1.3). To enhance the quality of GitHub Typo Corpus, we used samples whose ``prob_typo'' is larger than 0.95. To enhance the readability of samples in BEA-60K, we conducted proper data pre-processing for the samples before evaluation, e.g., erasing blank spaces before periods, punctuation, and questions. We measure the recovery rate with the accuracy of exact matching and edit distance between reconstructed and original text. Results in Table 7 demonstrate that our method consistently outperforms the baseline across all the models. Especially our method has gained a 18.8-point improvement in accuracy on the GitHub Typo Corpus for Pythia-2.8B.

5.2.2 Typo-Contained Text Classification

We measure the performance of classification tasks for texts containing spelling errors. We inserted a collection of typing errors committed by humans in the real world into the following four benchmarks: SST-2 (2-way classification), AG News (4-way), SST-5 (5-way) (Socher et al., 2013), and Amazon Review in English from

Model	Method	Github Typo		BEA-60K	
		Acc.	Dist.	Acc.	Dist.
GPT2	Base	5.2	4.57	19.0	2.30
(XL)	Ours	**+10.3**	**−0.31**	**+15.3**	+0.59
Gemma2	Base	69.5	0.81	63.7	1.11
(2B)	Ours	**+11.8**	**−0.10**	**+6.9**	**−0.19**
Pythia	Base	41.7	1.04	56.6	1.50
(2.8B)	Ours	**+18.8**	**−0.08**	**+5.5**	**−0.09**
Mistral	Base	80.2	0.38	67.3	1.15
(7B)	Ours	**+3.1**	**−0.09**	**+3.8**	**−0.21**
Llama3	Base	82.9	0.31	71.1	0.96
(8B)	Ours	**+1.1**	**−0.02**	**+1.8**	**−0.14**

Table 7: Typo and misspelling correction performance. Evaluation metrics are accuracy and Levenshtein distance (edit distance).

Model	Method	SST-2	AG	SST-5	Amazon
GPT2	Base	53.2	48.3	19.4	29.0
(XL)	Ours	**+3.6**	**+4.1**	−0.8	−0.6
Gemma2	Base	83.4	63.8	39.4	49.3
(2B)	Ours	**+1.0**	**+1.4**	**+1.1**	−0.3
Pythia	Base	69.8	69.2	30.2	35.7
(2.8B)	Ours	**+1.9**	**+4.3**	**+4.0**	**+2.7**
Mistral	Base	84.3	82.6	43.1	52.6
(7B)	Ours	**+3.3**	**+1.4**	**+1.4**	**+0.1**
Llama3	Base	85.1	77.6	40.4	50.8
(8B)	Ours	**+3.2**	**+3.1**	**+1.7**	**+2.9**

Table 8: Performance of text classification containing real-world spelling errors. Evalation metric: accuracy.

Multilingual Amazon Reviews Corpus (5-way) (Keung et al., 2020). Specifically, following Bast et al. (2021), we use a collection of spelling errors from Peter Norvig[2] as a dictionary which consists of 7841 words and their corresponding typos. We replace each word with a misspelled one by a higher probability of 50% in order to clearly show the impact of typos on performance. We limited the total number of evaluation samples by randomly sampling 1000 from test sets for each benchmark. The result in Table 8 shows that our proposed method achieves consistent improvement over base PLMs across benchmarks and models, indicating that our method makes PLMs more robust against real-world typos for downstream tasks.

[2]`https://norvig.com/ngrams/spell-errors.txt`.

5.2.3 Other NLP Tasks

To confirm that our method does not degrease NLP performance, we conduct further evaluations on other NLP tasks: a perplexity (PPL) check on the representative pre-training corpus, standard downstream English tasks, and cross-lingual task performance. Specifically, we check the perplexity on hold-out 1000 documents for FineWeb and RedPajama-V2 (Computer, 2023). Regarding downstream English tasks, we use ARC-Easy, ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), OpenbookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), and Winogrande (Sakaguchi et al., 2020) for the evaluation. In addition, we conducted evaluations on cross-lingual tasks, including PAWS-X (Yang et al., 2019) and XNLI (Conneau et al., 2018). The target languages are English, Spanish, French, and German. We use Language Model Evaluation Harness (Gao et al., 2024) to evaluate English downstream and cross-lingual tasks. We limited the total number of evaluation samples to 1000 and set the number of few-shot in-context exemplars as four across all the tasks and models. Evaluation metric is accuracy for all the downstream tasks. The results of the experiments are described in Table 9 and Table 10. It was found that perplexity is almost the same before and after continual pre-training. Regarding English and cross-lingual downstream tasks, there is no drastic performance change or strong tendencies across models and tasks, indicating that our method retains general-purpose NLP skills.

5.3 Baseline Comparison

To confirm the advantage of our proposed method, we compare its performance with the following baseline approaches.

- **Next Token Prediction (NTP) on Clean Text:** Following the conventional self-supervised pre-training of decoder-based language models, this method continually pre-trains base models by optimizing joint probabilities over tokens in clean texts as the product of conditional probabilities (Radford et al., 2019; Brown et al., 2020). It employs cross-entropy loss for probability optimization. For simplicity, we define this optimization approach as next token prediction (NTP). Based on the mathematical

Model	Method	PPL(↓)		Downstream Tasks					
		FW	RP	ARC-C	ARC-E	HS	OBQA	PIQA	WG
GPT2-XL	Base	25.60	26.97	26.1	60.5	40.7	22.8	71.6	55.8
	Ours	**−0.01**	+0.86	−0.5	−0.1	−0.1	−0.6	±0.0	**+0.4**
Gemma2-2B	Base	16.94	16.04	50.4	81.5	49.9	33.8	79.7	70.1
	Ours	**−0.01**	+0.08	**+0.2**	**+0.1**	±0.0	−0.4	−0.7	**+1.0**
Pythia-2.8B	Base	18.51	10.68	30.6	66.1	43.4	25.6	74.3	59.0
	Ours	+0.02	+0.15	**+0.9**	−0.1	**+0.2**	**+0.2**	**+0.1**	**+1.1**
Mistral-7B	Base	9.49	6.84	57.6	82.9	53.5	37.2	80.4	76.1
	Ours	+0.08	+0.21	−0.8	**+0.3**	−0.2	−0.4	−0.2	±0.0
Llama3-8B	Base	12.60	8.16	54.4	82.6	52.7	38.0	79.6	77.1
	Ours	+0.12	+0.08	−0.4	±0.0	**+0.2**	−0.6	−0.2	−0.3

Table 9: Performance of downstream English tasks and perplexity on major pre-training corpus.

Model	Method	PAWS-X				XNLI			
		de	en	es	fr	de	en	es	fr
GPT2-XL	Base	51.9	56.7	56.6	54.0	34.8	46.6	32.3	34.1
	Ours	**+0.2**	−0.4	**+1.1**	**+0.8**	**+0.1**	**+0.5**	**+1.2**	−1.0
Gemma2-2B	Base	63.8	67.9	62.2	60.2	43.9	48.2	44.7	46.7
	Ours	−0.2	−1.3	−2.1	−0.7	**+1.4**	**+0.3**	−0.8	**+0.2**
Pythia-2.8B	Base	58.2	60.8	55.3	54.7	42.8	47.4	39.7	43.0
	Ours	−1.2	−0.5	**+0.9**	**+1.0**	**+1.6**	−0.1	−0.3	−1.0
Mistral-7B	Base	71.6	69.9	69.2	68.6	45.1	50.2	44.8	45.5
	Ours	−0.8	**+0.4**	−1.9	−0.1	±0.0	−0.5	−0.4	**+0.5**
Llama3-8B	Base	67.3	67.2	64.5	63.7	44.6	46.6	45.4	46.7
	Ours	**+1.8**	−1.0	−1.7	−0.6	**+0.1**	**+1.1**	**+1.0**	**+1.0**

Table 10: Cross-lingual task performance.

symbols defined in Section 3, the objective function is defined as:

$$\mathcal{L} = -\frac{1}{B\tilde{T}} \sum_{i=1}^{B} \sum_{j=1}^{\tilde{T}} \log P_\theta(\tilde{z}_{i,j} | \tilde{z}_{i,<j}), \quad (7)$$

where $\tilde{z}_{i,<j}$ denotes $\{\tilde{z}_{i,1}, \ldots, \tilde{z}_{i,j-1}\}$.

- **NTP on Clean Text with Dropout BPE:** Instead of adding the artificial noises to original texts, this method applies BPE dropout to clean text (Provilkov et al., 2020). BPE dropout is a simple and effective subword regularization method by stochastically segmenting each word to produce multiple segmentations. This method enhances the language model to better learn word compositionality and to be more robust against segmentation errors. Prior work Provilkov et al. (2020) has shown that models trained with this method achieve better and more robust performance, although it has not been empirically proven to be effective for continual training of decoder-based PLMs.
- **NTP on Noisy Text:** This method continually pre-trains models with NTP on noisy text data instead of clean data. The objective function of this method is defined as follows.

$$\mathcal{L} = -\frac{1}{BT'} \sum_{i=1}^{B} \sum_{j=1}^{T'} \log P_\theta(z'_{i,j} | z'_{i,<j}). \quad (8)$$

- **NTP on Noisy Text with Sparse Token Optimization:** To ablate the importance of distribution matching in our method, this method trains models with next token prediction objective functions plus sparse token optimization, i.e., skip noisy tokens from

Model	Method	Typo	ID	OOD	w/o noise
GPT2-XL	Base	48.3	47.5	39.4	66.4
	NTP on clean text	38.3	36.4	32.5	54.1
	NTP on clean text + BPE Dropout	35.7	35.3	30.2	52.9
	NTP on noisy text	36.7	37.1	29.6	52.5
	NTP on noisy text + sparse token optimization	41.0	39.0	33.3	55.6
	Ours	**52.4**	**53.3**	**41.9**	**67.2**
Llama3-8B	Base	77.6	78.5	78.2	83.3
	NTP on clean text	71.8	73.9	65.3	68.9
	NTP on clean text + BPE Dropout	7.2	4.0	2.0	1.1
	NTP on noisy text	8.6	0.8	2.9	4.3
	NTP on noisy text + sparse token optimization	48.5	41.1	49.4	63.3
	Ours	**80.7**	**82.1**	**81.3**	**83.8**

Table 11: Baseline Study. Evaluation metric is accuracy on AG News classification.

the optimization target. This is realized just by replacing distribution matching loss in Equation 6 with cross-entropy loss:

$$\mathcal{L}_i = -\sum_{j\in J} \log P_\theta(z'_{i,j}|z'_{i,<j}). \tag{9}$$

We continually pre-trained PLMs with each of the above baselines in the same settings as our method as described in Section 4. After the training, we compared the classification performance for AG News with ID-multi noise, OOD-multi noise, and actual spelling errors or typos. The evaluation follows the same settings as Section 5.1.1 for OOD and ID noise as well as Section 5.2.2 for typos. The results are summarized in Table 11. It demonstrates that our method consistently improved performance, whereas the above-mentioned baseline methods degrade performance across models. Regarding the performance drop of continual pre-training (NTP) on clean text, we assume that training instability is caused by the small batch size during the training (see Section 4) although the setting is the same as our method. The result indicates that our method achieves further training stability compared to the conventional training objective function or NTP. BPE dropout caused more training instability when applied to the pre-training of decoder-based PLMs in our experiment setting. The possible reason is that it caused training instability by making the models predict a wide variety of delimitations for each of the words in an auto-regressive manner. Continual pre-training (NTP) on noisy text caused more performance drop than clean text. While sparse token optimization alleviates performance degradation to some extent, there is still a gap from our method. In contrast, our method, or distribution matching on noisy text with sparse token optimization in other words, achieves better and more robust performance than the baseline methods.

6 Discussion

Our experiments have demonstrated that decoder-based PLMs with subword tokenizers can acquire higher robustness against artificial character-level noise and real-world spelling errors (typos) by our proposed method. However, as shown in Table 4, the degree of improvement in robustness tends to decrease as the parameter size of models increases. This phenomenon is attributed to the fact that as the model becomes larger, the performance degradation itself tends to diminish even when the noise is added to input texts. The possible reason for this phenomenon is that a large pre-training corpus created by scraping through the Internet websites contains noise such as typos with a certain probability, and as the model grows, the relationship between typo words and their correct words is learned in a self-supervised way. However, even if it is true, our method can further robustify PLMs against more general noise including typos regardless of model size as shown in our experiments. In addition, our method tends to make PLMs more robust against noise as the number of training steps increases (see Figure 3). There is a possibility that more training steps make larger models acquire more robustness.

7 Conclusion

This study has proposed a method of continual pre-training to convert decoder-based PLMs with subword tokenizers into perturbation-robust few-shot in-context learners. The results demonstrate that our method consistently improves the performance of artificial and real-world noisy tasks while maintaining the performance of downstream NLP tasks without noise. We hope this study will inspire further improvements in the robustness of LLMs, such as prompt attack defense or perturbation-robust search systems.

Acknowledgments

We would like to thank our TACL action editor, Yuval Pinter, and all the anonymous reviewers for their helpful feedback, which has greatly improved our paper.

References

Mostafa Abdou, Vinit Ravishankar, Artur Kulmizev, and Anders Søgaard. 2022. Word order does matter and shuffled language models know it. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6907–6919, Dublin, Ireland. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-long.476

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, and Shawn Jain. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Noëmi Aepli and Rico Sennrich. 2022. Improving zero-shot cross-lingual transfer between closely related languages by injecting character-level noise. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 4074–4083, Dublin, Ireland. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.findings-acl.321

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, Minneapolis, Minnesota. Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-4010

Hannah Bast, Matthias Hertel, and Mostafa M. Mohamed. 2021. Tokenization repair in the presence of spelling errors. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 279–289, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.conll-1.22

Dan Biderman, Jose Gonzalez Ortiz, Jacob Portes, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P. Cunningham. 2024. Lora learns less and forgets less. *arXiv preprint arXiv:2405.09673*.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Yonatan Bisk, Rowan Zellers, Ronan Le bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7432–7439. `https://doi.org/10.1609/aaai.v34i05.6239`

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Qi Cao, Takeshi Kojima, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Unnatural error correction: Gpt-4 can almost perfectly handle unnatural scrambled text. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8898–8913. `https://doi.org/10.18653/v1/2023.emnlp-main.550`

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91. `https://doi.org/10.1162/tacl_a_00448`

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? Try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Together Computer. 2023. Redpajama: an open dataset for training large language models.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium. Association for Computational Linguistics. `https://doi.org/10.18653/v1/D18-1269`

Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe

Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, and Junteng Jia. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024. A framework for few-shot language model evaluation. `https://doi.org/10.5281/zenodo.12608602`

Atticus Geiger, Zhengxuan Wu, Hanson Lu, Josh Rozner, Elisa Kreiss, Thomas Icard, Noah Goodman, and Christopher Potts. 2022. Inducing causal structure for interpretable neural networks. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 7324–7338. PMLR.

Masato Hagiwara and Masato Mita. 2020. GitHub typo corpus: A large-scale multilingual dataset of misspellings and grammatical errors. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 6761–6768, Marseille, France. European Language Resources Association.

Tatsuya Hiraoka, Hiroyuki Shindo, and Yuji Matsumoto. 2019. Stochastic tokenization with a language model for neural text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1620–1629, Florence, Italy, Association for Computational Linguistics. `https://doi.org/10.18653/v1/P19-1158`

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Jing Huang, Zhengxuan Wu, Kyle Mahowald, and Christopher Potts. 2023. Inducing character-level structure in subword-based language models with type-level interchange intervention training. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12163–12180, Toronto, Canada. Association for Computational Linguistics. `https://doi.org/10.18653/v1/2023.findings-acl.770`

Sai Muralidhar Jayanthi, Danish Pruthi, and Graham Neubig. 2020. NeuSpell: A neural spelling correction toolkit. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 158–164, Online. Association for Computational Linguistics. `https://doi.org/10.18653/v1/2020.emnlp-demos.21`

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b (2023). *arXiv preprint arXiv:2310.06825*.

Phillip Keung, Yichao Lu, György Szarvas, and Noah A. Smith. 2020. The multilingual Amazon reviews corpus. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4563–4568, Online. Association for Computational Linguistics. `https://doi.org/10.18653/v1/2020.emnlp-main.369`

Yoon Kim, Yacine Jernite, David Sontag, and Alexander Rush. 2016. Character-aware neural language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30. https://doi.org/10.1609/aaai.v30i1.10362

T. Kudo. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*. https://doi.org/10.18653/v1/D18-2012

Jindřich Libovický, Helmut Schmid, and Alexander Fraser. 2022. Why don't people use character-level machine translation? In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2470–2485, Dublin, Ireland. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.findings-acl.194

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics. https://doi.org/10.18653/v1/D18-1260

Milad Moradi and Matthias Samwald. 2021. Evaluating the robustness of neural language models to input perturbations. *arXiv preprint arXiv:2108.12237*. https://doi.org/10.18653/v1/2021.emnlp-main.117

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745*. https://doi.org/10.18653/v1/D18-1206

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. The fineweb datasets: Decanting the web for the finest text data at scale.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.170

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740. https://doi.org/10.1609/aaai.v34i05.6399

Rico Sennrich. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*. https://doi.org/10.18653/v1/P16-1162

Andrew Shin and Kunitake Kaneko. 2024. Large language models lack understanding of character composition of words. *arXiv preprint arXiv:2405.11357*.

Koustuv Sinha, Prasanna Parthasarathi, Joelle Pineau, and Adina Williams. 2021. UnNatural language inference. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7329–7346, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.569

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. https://doi.org/10.18653/v1/D13-1170

Aarohi Srivastava and David Chiang. 2023. Fine-tuning BERT with character-level noise for zero-shot transfer to dialects and closely-related languages. In *Tenth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2023)*, pages 152–162, Dubrovnik, Croatia. Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.vardial-1.16

Yi Tay, Vinh Q. Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. 2021. Charformer: Fast character transformers via gradient-based subword tokenization. *arXiv preprint arXiv:2106.12672*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu,Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozińska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucińska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju-yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, and Lilly McNealus. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1504–1515. https://doi.org/10.18653/v1/D16-1157

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi 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 M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*. https://doi.org/10.18653/v1/2020.emnlp-demos.6

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306. https://doi.org/10.1162/tacl_a_00461

Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. 2019. PAWS-X: A cross-lingual adversarial dataset for paraphrase identification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3687–3692, Hong Kong, China. Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1382

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*. https://doi.org/10.18653/v1/P19-1472

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*.