Can Pretrained Language Models Derive Correct Semantics from Corrupt Subwords under Noise?

Xinzhe Li, **Ming Liu**, **Shang Gao**

School of IT, Deakin University, Australia
{lixinzhe, m.liu,shang.gao}@deakin.edu.au

Abstract

For Pretrained Language Models (PLMs), their susceptibility to noise has recently been linked to subword segmentation. However, it is unclear which aspects of segmentation affect their understanding. This study assesses the robustness of PLMs against various disrupted segmentation caused by noise. An evaluation framework for subword segmentation, named Contrastive Lexical Semantic (CoLeS) probe, is proposed. It provides a systematic categorization of segmentation corruption under noise and evaluation protocols by generating contrastive datasets with canonical-noisy word pairs. Experimental results indicate that PLMs are unable to accurately compute word meanings if the noise introduces completely different subwords, small subword fragments, or a large number of additional subwords, particularly when they are inserted within other subwords.

1 Introduction

The capability to understand the meaning of noisy words through character arrangements is a crucial aspect of human cognitive abilities (Rawlinson, 2007). This capability is highly sought after in practical applications such as machine translation and sentiment analysis (Belinkov and Bisk, 2018). However, despite their success in in-distribution test data with standardized word forms, Pretrained Language Models (PLMs), which serve as the backbone models, tend to perform poorly on rare or noisy words (Kumar et al., 2020; Baron, 2015). These noisy words may be caused by accidental typos (Belinkov and Bisk, 2018) or spelling variants on social media (Ritter et al., 2010).

Prior studies show that most subword-based PLMs perform poorly under noise largely due to subword segmentation (Zhuang and Zuccon, 2022), while character-based PLMs show more robustness (El Boukkouri et al., 2020). Examining the impact of subword segmentation factors on PLMs is also crucial for defending against the adversarial attacks that leverage the sensitivity of subword segmentation to noise (Liu et al., 2022). However, rare work has investigated how the subword segmentation from noisy words affects the word meaning.

To help address this question, we design and develop a contrastive framework (CoLes) to assess the robustness of PLMs in the face of various forms of segmentation corruption. As subword segmentation can be influenced by noise in various ways, such as adding extra subwords or losing original subwords, we systematically categorize the ways into four main categories and two additional subcategories based on three subword sets, as exemplified in Table 1. Two types of noise models are proposed to effectively generate all the types of corruption except missing corruption, and a contrastive dataset consisting of noisy and standard word pairs is created. This framework enables us to evaluate the significance of preserved subwords and the impact of subwords added by noise.

The experimental results provide the following insights: 1) complete corruption: the PLMs struggle to infer meaning accurately if no subwords from the original segmentation are retained. The worst performance is observed when the meaning of original words is stored in the embedding; 2) partial corruption: preserving larger subword chunks can aid the understanding of PLMs, whereas retaining smaller subword pieces tend to be ineffective; and 3) additive corruption: even with all original subwords, however, the addition of subwords can harm the meaning of words, particularly when they are placed within other subwords. The more additive subwords, the greater the deviation in word semantics. All the results are consistent on the three PLMs with different vocabularies and segmentation algorithms.

2 Contrastive Lexical-Semantic Probe

The CoLeS probe framework has segmentation corruption and noise models that produce noisy

*Proceedings of the The 12th Joint Conference on Lexical and Computational Semantics (*SEM 2023)*, pages 165–173

Corruption Types	Examples	Segmentation Sets		
		Missing	Overlap	Additive
Complete (intact)	tasty → taaasty	tasty		ta, aa, sty
Complete	stun → stunn	s, tun		stu, nn
Partial	effectiveness → efeectiveness	effect	iveness	efe, ect
Additive (infix)	insubstantial → insuubstantial		ins, ub, stan, tial	u
Additive (affix)	hilarious → hilariousss		hil, ario, us	s, s
Missing	insubstantial → insstantial	ub	ins, stan, tial	

Table 1: Examples of different types of segmentation corruption. Complete/partial: completely/partially disrupting the original segmentation; additive: creating unnecessary subwords; missing: ignoring a token. A distinct form of complete corruption, referred to as "intact corruption", arises when a clean word is tokenized into a single subword that does not appear in the segmentation of its noisy counterpart. In the given example of intact corruption, the term "tasty" serves as an intact token.

words leading to different types of segmentation corruption. These noisy words, along with their corresponding canonical forms, are organized in a contrastive lexical dataset $\mathbb{D}_{\text{contrastive}}$ [1]. An evaluation protocol is designed to examine the effect of various corruption types.

2.1 Segmentation Corruption under Noise

A PLM consists of a tokenizer $\text{Seg}(\cdot)$, which segments a given word w into a sequence of subwords, i.e., $\text{Seg}(w) = (\tilde{w}_1, ..., \tilde{w}_K)$, and a PLM encoder $\text{Enc}(\cdot)$, which takes $\text{Seg}(w)$ and outputs a word representation. Formally, the segmentation of a canonical word $\text{Seg}(w)$ can be represented as a set $\mathbb{S}$, while the segmentation of a noisy word $\text{Seg}(\tilde{w})$ can be represented as set $\tilde{\mathbb{S}} = \{\tilde{w}_1, ..., \tilde{w}_K\}$. We can then utilize set operations to define the overlap set (consisting of retained subwords), the missing set, and the additive set (comprising additional tokens that are not present in $\mathbb{S}$) as $\mathbb{O} = \mathbb{S} \cap \tilde{\mathbb{S}}$, $\mathbb{M} = \mathbb{S} - \mathbb{O}$ and $\mathbb{A} = \tilde{\mathbb{S}} - \mathbb{O}$, respectively.

The set data structure cannot count duplicated tokens, which frequently occur in additive corruption scenarios, such as the additive (affix) corruption example presented in Table 1. Hence, we utilize a multiset implementation of $\mathbb{S}$ and $\tilde{\mathbb{S}}$ since such a data structure also stores the frequencies of elements, helping us assess the impact of duplicated tokens. Since the multiset implementation only includes unique elements without considering their order of appearance, we further differentiate the two types of additive corruption by iteratively comparing elements from two queue implementations of $\text{Seg}(w)$ and $\text{Seg}(\tilde{w})$.

In this study, we distinguish a unique category of corruption referred to as "intact corruption" from complete corruption, as the canonical words in this category (with whole-word vectors) remain unchanged. In total, there are six different types of corruption, as outlined in Table 1.

Identification of corruption types. During the evaluation, we need to filter each word pair according to its corruption type. First, we segment each word pair in $\mathbb{D}_{\text{contrastive}}$ by a model-specific tokenizer $\text{Seg}(\cdot)$ into subwords $(\mathbb{S}, \tilde{\mathbb{S}})$. We then identify the corruption type according to the following conditions: 1) Complete corruption: $\mathbb{S}$ and $\tilde{\mathbb{S}}$ are disjoint, i.e., $\mathbb{O} = \emptyset$. If the length of the missing set $\mathbb{M}$ is 1, this noise leads to intact corruption; 2) Partial corruption: the corruption only occurs to one of the subwords (i.e., the one in $\mathbb{M}$), and the other subwords (i.e., those in $\mathbb{O}$) are not affected. The prerequisite is that there exist more than one subwords in the original segmentation set $\mathbb{S}$. We can find such word pairs satisfy $\mathbb{M}, \mathbb{O}, \mathbb{A} \neq \emptyset$; 3) The conditions for additive corruption and missing corruption are $\mathbb{S} \in \tilde{\mathbb{S}}$ (or $\mathbb{M} = \emptyset$) and $\tilde{\mathbb{S}} \in \mathbb{S}$ (or $\mathbb{A} = \emptyset$), respectively. [2]

2.2 Creation of Contrastive Dataset

Most prior noisy datasets added noise to sentences, not individual words (Belinkov and Bisk, 2018; Kumar et al., 2020; Warstadt et al., 2019; Hagiwara and Mita, 2020). Besides, as contrastive datasets containing both the original and noisy form of a word are not readily available, we create our own lexical dataset which includes both forms. Examples of the generated dataset can be found in Table 2.

[1] Sentiment lexicon used is from https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#lexicon.

[2] See https://github.com/xinzhel/word_corruption/blob/main/word_corruption.py for concrete implementation.

Canonical Words	Keyboard	Swap	Letter-reduplication
bad	NA	NA	badddddddd, baaaadddd, bbbbaaaaddddd
crazy	craxy	carzy	crazzyyyyyyyyy, crazzzzzy
amazing	amazijg	amzaing	amazing, amazinnng, amazinggg, amaaazzziiingggg

Table 2: Examples of contrastive datasets with canonical-noisy word pairs. Three types of noise models are applied: Swap-typos, Keyboard typos and letter reduplication. NA: we discard generated noisy words since typos on these words generate noisy words that are even unrecognizable to humans.

Noise models. Two sources of noise models are used to generate the lexical dataset. Findings given in Appendix E indicate that both types of noise models have comparable effects on model performance.

1) **Naturally and frequently occurring typos.** Users often type neighboring keys due to mobile penetration across the globe and fat finger problem (Kumar et al., 2020), while typing quickly may result in swapping two letters Belinkov and Bisk (2018). We refer to them as *Keyboard* and *Swap* typos, respectively. Our implementation of these typos is based on Wang et al. (2021). Specifically, for *Keyboard*, we only use letters in the English alphabet within one keyboard distance as the substitute symbols. Further, we avoid unrecognizable word forms (e.g., "bad→bqd" or "top → tpp") by selecting words with more than four characters.

According to the psycholinguistic study (Davis, 2003), to make noisy words recognizable for humans, we only apply noise to the middle characters and keep characters at the beginning and the end. Besides such a constraint, *Swap* typo also requires at least two distinct characters in the middle for swapping. However, words like "aggressive" can still be transformed into the same word by swapping "ss", so we transform them until we get a distinct word. Finally, we set a one-edit constraint for typos.

2) **Non-standard orthography.** We gather words with letter reduplication from 1.6 million tweets (Go et al., 2009). To create the canonical and noisy word pairs, we match specific noisy word forms (e.g. words with repeated letters for emphasis) to their corresponding canonical forms (a sequence of definite characters). We use simple regular expression patterns to search for words with repeated letters [3]. Examples in Table 3 show how effective these types of noise are in triggering different types of segmentation corruption.

[3]For example, pattern "\bb+a+d+" for "bad" matches "badddddddd".

Data-generating process. We create a contrastive dataset, $\mathbb{D}_{\text{contrastive}}$, by applying the noise models to the lexical dataset $\mathbb{D}_{\text{canonical}}$, which contains words in their canonical form. The noise models are applied to each word in $\mathbb{D}_{\text{canonical}}$ to create two misspelled words. Additionally, a random number of noisy words is extracted from the collection of 1.6 million tweets. As for the lexical dataset, $\mathbb{D}_{\text{canonical}}$, we choose adjectives from a sentiment lexicon that, by definition, provides positive or negative sentiment labels for use with downstream classifiers.

Evaluation. To assess the extent to which the meanings of noisy words diverge from the standard word forms, we calculate the cosine similarity between $\text{Enc}(\mathbb{S})$ and $\text{Enc}(\tilde{\mathbb{S}})$. For words that consist of multiple subwords, we aggregate their vectors into a single representation by averaging the token embeddings obtained from the PLMs. It is important to note that the output embedding spaces of PLMs exhibit varying levels of anisotropy (Ethayarajh, 2019; Yan et al., 2021; Gao et al., 2019). Thus, the similarity scores cannot be directly compared across different models. It is necessary to set a baseline by computing the similarity between $\text{Enc}(\mathbb{S})$ and a random embedding (we use the embedding of token "the", i.e., $\text{Enc}(\text{the})$).

Additionally, we fine-tune downstream classifiers denoted as $y = \text{Cls}(x)$, where y represents an arbitrary semantic dimension and x corresponds to the encoded representation obtained from the PLMs $\text{Enc}(\text{Seg}(\cdot))$. We focus on sentiment classification as individuals frequently use sentiment words creatively on social media to express their emotions. To conduct our experiments, the sentiment of each word and its noisy variations is derived from the sentiment lexicon.

To gauge the semantic deviation caused by noise, we measure the accuracy of the noisy counterparts of words that are accurately classified in their original form.

Tokenizers	Intact	Complete	Partial
BERT	0.36	0.14	0.49
RoBERTa	0.46	0.12	0.42
ALBERT	0.38	0.13	0.49

(a) Typos.

Intact	Complete	Partial	Additive	
			affix	infix
0.70	0.02	0.06	0.22	0
0.61	0	0.06	0.30	0.01
0.61	0.02	0.06	0.29	0.02

(b) Letter Reduplication.

Table 3: Frequency of each segmentation corruption.

3 Experimental Results

Experiments are performed on three widely used PLMs: **BERT$_{\text{BASE}}$**, **RoBERTa$_{\text{BASE}}$** and **ALBERT$_{\text{BASE}}$** (See Appendix A for details). **BERT** (Devlin et al., 2019) accepts inputs from a Wordpiece tokenizer (Schuster and Nakajima, 2012), while **RoBERTa** (Liu et al., 2019), another popular frequent-based segmentation scheme, uses BPE (Sennrich et al., 2016). For comparison, we include **ALBERT** (Lan et al., 2020) with a probabilistic tokenizer called Sentencepiece (Kudo and Richardson, 2018).

Subwords retention is important for maintaining the correct semantics. Table 4 shows the severity of semantic deviation for each type of corruption. Generally, the more subwords the segmentation retains, the better the semantics are maintained (additive corruption $>$ partial corruption $>$ complete and intact corruption). Under additive corruption, the PLMs can always maintain more semantics from noisy words than random words (the baseline), while only RoBERTa has similarity score higher than the baseline under partial corruption. All the PLMs cannot infer word meaning from complete corruption.

What subwords, if retained, would enhance the comprehension of PLMs? We find that partial corruption can preserve word meaning if it retains a significant portion of the words, such as "upset" for "upsetting" or "phenomena" for "phenomenal" (See Appendix B). This is backed up by the finding that PLMs have the capability of learning morphological information, where stems contain more semantic meaning in a word compared to smaller components such as inflectional morphemes (Hofmann et al., 2021).

Are words more impacted by noise under complete corruption if their meaning is stored in the embeddings? According to Hofmann et al. (2021), if a word is represented as a single vector, PLMs can access its meaning directly from the embedding (referred to as the "storage route") instead of deducing it from the combination of subwords (known as the "computation route"). We presume that PLMs struggle to maintain the original meaning of these words when exposed to noise. We classify this type of corruption as "intact corruption", which is a particular variation to complete corruption. To validate our assumption, we evaluate the performance of PLMs on words under intact corruption. Results show that words with intact corruption consistently perform worse than those with complete corruption, despite both having completely distinct subwords. Although intact corruption consistently yields the lowest similarity score, the PLMs may still be able to better infer some semantic dimensions, such as sentiment, under intact corruption compared to complete corruption. (Appendix C).

Presence of additive subwords can damage the meaning of words, particularly when they are inserted in the middle of other subwords. In some cases, words under additive corruption (keeping all subwords) can perform worse than those under partial corruption (keeping only some subwords), as seen in the letter reduplication experiment (Appendix C). The finding suggests that the retention of subwords is not the only factor impacting the performance of PLMs. To uncover other factors affecting the word meaning, we analyzed 10 worst and best instances for each corruption type based on similarity scores (Appendix B). All the poorly performing cases have incorrect predictions, further highlighting the damaging impact on semantic meaning. The results show that the number of additive tokens (i.e., the cardinality of $\mathbb{A}$) is a distinct feature between good and bad instances. All the good cases have only 1 additive token, while the bad cases have at least 2 additive tokens (3.8 for partial corruption and 8.7 for additive corruption on average).

Thus, our hypothesis is that as the number of additive subwords increases, PLMs will have dif-

Models	Intact	Complete	Partial	Additive	Baseline
BERT	0.29	0.41	0.58	0.69	0.69
RoBERTa	0.54	0.66	0.76	0.85	0.72
ALBERT	0.41	0.47	0.62	0.74	0.68

(a) Similarity.

Intact	Complete	Partial	Additive
0.56	0.65	0.8	0.91
0.66	0.60	0.75	0.95
0.61	0.63	0.76	0.93

(b) Accuracy.

Table 4: Performance of PLMs under various types of corruption. Similarity scores of pretrained representations and accuracy of downstream classifiers are evaluated. The best result per row is highlighted in gray, and the second-best is in light gray. As a baseline, we compare the similarity scores between canonical and random words ("the" used). The unaffected accuracy is 1 since the canonical forms selected for evaluation are always correctly predicted.

Models	Infix	Suffix
BERT	0.59	0.70
RoBERTa	0.85	0.95
ALBERT	0.66	0.74

(a) Similarity.

Infix	Suffix
0.74	0.91
0.95	1
0.82	0.94

(b) Accuracy.

Table 5: Comparison of two types of additive corruption.

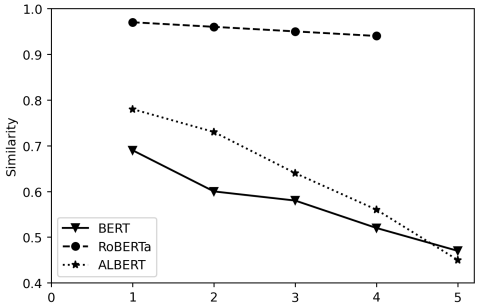

Figure 1: Correlation between the number of additive subwords and the cosine similarity of noisy words with their canonical forms. The range of quantity of additive subwords is subject to change depending on the tokenizer used.

ficulty determining the correct meaning of words. We test the hypothesis by examining the performance of PLMs on both additive and intact corruption, where the missing and overlap sets remain constant. For additive corruption, we limit our experiments to only one unique additive subword and vary its frequency. We find 23 words with at least 3 noisy versions, each creating an additive set with the same element but different multiplicities. Take "amazing" as an example: one of its noisy instances ("amazinggggggg") has the multiplicity of 3 according to its additive set $\mathbb{A} = \{$"gg", "gg", "gg"$\}$ while "gg" only appears twice in another instance ("amazinggggg"). We sort every collection of noisy words in either of two ways, depending on the similarity scores or the multiplicities of additive tokens. In 17 out of 23 collections, these two sorting criteria produce identical results. This discovery also holds true for intact corruption, where the subwords within an additive set are typically diverse. Figure 1 illustrates a strong negative correlation between the number of additive tokens and the average similarity of noisy words for all the three models under intact corruption, where the sizes of missing sets and overlap sets are fixed to 1 and 0.

Besides, as shown in Table 5, additive subwords placed within subwords cause more harm than those that act as suffixes.

4 Conclusion

We proposed the CoLeS framework which can evaluate how corrupt segmentation under noise affects PLMs' understanding. The experimental results show that three challenges can impair the PLMs' understanding of noisy words: insertion of additive subwords (especially within existing subwords), loss of original subwords, and incapacity of computing the word meanings through the aggregation of smaller subword units.

Reproducibility. Data and source code for noisy data generation, corruption types identification and PLMs' performance evaluation are released on Github [4].

Limitations

The omission of missing corruption from the evaluation process is justified due to its infrequent occurrence in real-world scenarios (refer to Appendix D for elaboration). Nevertheless, further investigation into rare instances of missing corruption may be warranted for research purposes. Our evaluation of language models was limited to auto-encoders based on the BERT architecture. Future studies are anticipated to expand the scope of PLMs under

[4] https://github.com/xinzhel/word_corruption

consideration [5].

References

Naomi S Baron. 2015. *Words onscreen: The fate of reading in a digital world*. Oxford University Press, USA.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *ICLR*.

Matt Davis. 2003. Psycholinguistic evidence on scrambled letters in reading.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Jun'ichi Tsujii. 2020. CharacterBERT: Reconciling ELMo and BERT for word-level open-vocabulary representations from characters. In *CoLing*, pages 6903–6915.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *EMNLP-IJCNLP*, pages 55–65, Hong Kong, China.

Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tieyan Liu. 2019. Representation degeneration problem in training natural language generation models. In *ICLR*.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*.

Masato Hagiwara and Masato Mita. 2020. GitHub typo corpus: A large-scale multilingual dataset of misspellings and grammatical errors. In *Language Resources and Evaluation Conference (LREC)*.

Maria Heath. 2018. Orthography in social media: Pragmatic and prosodic interpretations of caps lock. *Proceedings of the Linguistic Society of America*.

Valentin Hofmann, Janet Pierrehumbert, and Hinrich Schütze. 2021. Superbizarre is not superb: Derivational morphology improves BERT's interpretation of complex words. In *ACL-IJCNLP*.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *EMNLP: System Demonstrations*, pages 66–71.

Ankit Kumar, Piyush Makhija, and Anuj Gupta. 2020. Noisy text data: Achilles' heel of BERT. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*.

[5] Instructions within our codebase facilitate the evaluation of various types of pre-trained language models accessible via Huggingface`https://huggingface.co/models`.

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 *ICLR*.

Aiwei Liu, Honghai Yu, Xuming Hu, Shu'ang Li, Li Lin, Fukun Ma, Yawen Yang, and Lijie Wen. 2022. Character-level white-box adversarial attacks against transformers via attachable subwords substitution. In *EMNLP*.

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

Graham Rawlinson. 2007. The significance of letter position in word recognition. *IEEE Aerospace and Electronic Systems Magazine*.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of Twitter conversations. In *NAACL-HLT*, pages 172–180.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

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

Xiao Wang, Qin Liu, Tao Gui, Qi Zhang, et al. 2021. Textflint: Unified multilingual robustness evaluation toolkit for natural language processing. In *ACL-IJCNLP: System Demonstrations*, Online.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *TACL*.

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 *EMNLP: System Demonstrations*.

Yuanmeng Yan, Rumei Li, Sirui Wang, Fuzheng Zhang, Wei Wu, and Weiran Xu. 2021. ConSERT: A contrastive framework for self-supervised sentence representation transfer. In *ACL-IJCNLP*.

Shengyao Zhuang and Guido Zuccon. 2022. Characterbert and self-teaching for improving the robustness of dense retrievers on queries with typos. SIGIR.

A Fine-tuning Pretrained Language Models

All the PLMs use BERT-based architecture, i.e., the encoding part of the transformer (Vaswani et al., 2017). **BERT$_{\text{BASE}}$** (110M parameters) and **RoBERTa$_{\text{BASE}}$** (125M parameters) are pretrained on BookCorpus and Wikipedia as masked language models. Only the pretraining of **ALBERT$_{\text{BASE}}$** (11M parameters) includes extra news and web data (Wolf et al., 2020). They are then fine-tuned for sentiment classification on the SST-2 dataset. All the models are publicly available on the Huggingface Hub website `https://huggingface.co/textattack`. Some configurations are shown as below. The BERT and RoBERTa models are fine-tuned using a learning rate of $2e^{-5}$ with no scheduling employed. The batch size is set to 32, and the training process spans 3 epochs, maintaining a gradient norm of 1. ALBERT is fined-tuned with a learning rate of $3e^{-5}$, a batch size of 32, and a total of 5 training epochs.

B Good and Bad Cases

Figure 2 shows the good and bad cases of partial and additive corruption under letter reduplication.

Partial

Bad

clean_word	noisy_word	overlap_set	missing_set	additive_set
enrich	enricch	{'en': 1}	{'rich': 1}	{'ric': 1, 'ch': 1}
nifty	niffty	{'ni': 1}	{'ft': 1, 'y': 1}	{'ff': 1, 'ty': 1}
genial	geniaaaaaaaal	{'gen': 1}	{'ial': 1}	{'ia': 1, 'aa': 3, 'al': 1}
redeem	redeeem	{'red': 1}	{'eem': 1}	{'ee': 1, 'em': 1}
relish	rellish	{'re': 1}	{'lish': 1}	{'llis': 1, 'h': 1}
loot	looooooooooottttttt	{'lo': 1}	{'ot': 1}	{'oo': 5, 'ott': 1, 'tt': 2, 't': 1}
loot	looooooooooot	{'lo': 1}	{'ot': 1}	{'oo': 5, 'oot': 1}
loot	looooooooot	{'lo': 1}	{'ot': 1}	{'oo': 4, 'oot': 1}
chic	chicccccc	{'chi': 1}	{'c': 1}	{'cc': 3}
stun	sttun	{'n': 1}	{'stu': 1}	{'st': 1, 'tu': 1}

Additive

clean_word	noisy_word	overlap_set	additive_set
good	ggoodd	{'good': 1}	{'g': 1, 'd': 1}
decent	decentttttt	{'decent': 1}	{'tt': 2}
	warmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm		{'mm': 19,
warm	mmmmmmmmm	{'warm': 1}	'm': 1}
hot	hottttttttttttttttttttttttttttttttttt	{'hot': 1}	{'tt': 18}
super	superrrrrr	{'super': 1}	{'rr': 2, 'r': 1}
super	superrrrrrrr	{'super': 1}	{'rr': 3, 'r': 1}
super	superrrrrrrrrr	{'super': 1}	{'rr': 5}
hot	hottttttttt	{'hot': 1}	{'tt': 4, 't': 1}
			{'ee': 16, 'e':
free	freeeeeeeeeeeeeeeeeeeeeeeeeeeeeee	{'free': 1}	1}
free	freeeeeeeeeeeeeeeeeeeeeeeeeeeeee	{'free': 1}	{'ee': 16}

Good

clean_word	noisy_word	overlap_set	missing_set	additive_set
uneventful	uneventfull	{'uneven': 1, 'tf': 1}	{'ul': 1}	{'ull': 1}
snobish	snobbish	{'s': 1, 'nob': 1}	{'ish': 1}	{'bis': 1, 'h': 1}
disown	dissown	{'di': 1, 'wn': 1}	{'so': 1}	{'sso': 1}
phenomenal	phenomenall	{'phenomena': 1}	{'l': 1}	{'ll': 1}
stupendous	stupendouss	{'stu': 1, 'pen': 1, 'dou': 1}	{'s': 1}	{'ss': 1}
sucky	suckky	{'suck': 1}	{'y': 1}	{'ky': 1}
crappy	crapppy	{'crap': 1}	{'py': 1}	{'ppy': 1}
upseting	upsetting	{'upset': 1}	{'ing': 1}	{'ting': 1}
dismal	dismall	{'di': 1, 'sma': 1}	{'l': 1}	{'ll': 1}
nicer	niceer	{'nice': 1}	{'r': 1}	{'er': 1}

clean_word	noisy_word	overlap_set	additive_set
abysmal	abysmall	{'ab': 1, 'ys': 1, 'mal': 1}	{'l': 1}
ungrateful	ungratefull	{'un': 1, 'grate': 1, 'ful': 1}	{'l': 1}
vengeful	vengefull	{'ve': 1, 'nge': 1, 'ful': 1}	{'l': 1}
sceptical	scepticall	{'sc': 1, 'ept': 1, 'ical': 1}	{'l': 1}
gloomy	gloomyy	{'gloom': 1, 'y': 1}	{'y': 1}
awkwardness	awkwardnesss	{'awkward': 1, 'ness': 1}	{'s': 1}
shameful	shamefull	{'shame': 1, 'ful': 1}	{'l': 1}
awesomeness	awesomenesss	{'awesome': 1, 'ness': 1}	{'s': 1}
caustic	causticc	{'ca': 1, 'ust': 1, 'ic': 1}	{'c': 1}
craziness	crazinesss	{'cr': 1, 'azi': 1, 'ness': 1}	{'s': 1}

Figure 2: Good and bad cases of partial and additive corruption under letter reduplication.

C PLM Robustness to Segmentation Corruption under Different Types of Noise

Table 6 displays the robustness of PLMs to segmentation corruption under various forms of noise. The results are largely consistent with those seen in Table 4. However, we notice that for letter reduplication, PLMs may perform worse with additive corruption than with partial corruption. Additionally, the accuracy of intact corruption can be better than that of complete corruption, despite they consistently having the lowest similarity score.

D Noisy words for Missing Corruption

As per the findings of Heath et al. (Heath, 2018), English word recognition by humans is predominantly influenced by consonants. Consequently, our investigation aims to identify abbreviations that disregard vowels and certain consonants when examining tweets. To be precise, an abbreviation is considered acceptable if its first letter and arbitrary consonants appear in a sequence that adheres to canonical words. For instance, the pattern of regular expression for term "sorry" can be "\bsr?r?y?" in such cases. However, we find that even humans have difficulties in recognizing all these abbreviations. While the inclusion of all consonants may enhance human recognition, we contend that assessing this form of corruption is

Models	Intact	Complete	Partial	Additive
BERT	0.24	0.34	0.62	0.69
RoBERTa	0.54	0.73	0.8	0.85
ALBERT	0.42	0.53	0.77	0.74

(a) Similarity.

Intact	Complete	Partial	Additive
0.54	0.47	0.92	0.91
0.54	0.73	0.87	0.95
0.63	0.79	0.91	0.93

(b) Accuracy.

Letter Reduplication

Models	Intact	Complete	Partial	Additive
BERT	0.34	0.41	0.58	/
RoBERTa	0.55	0.65	0.75	/
ALBERT	0.4	0.47	0.61	/

(c) Similarity.

Intact	Complete	Partial	Additive
0.59	0.66	0.79	/
0.6	0.57	0.74	/
0.59	0.61	0.75	/

(d) Accuracy.

Typos

Table 6: Performance of PLMs under different types of corruption. *Similarity scores of pretrained representations* and *accuracy of downstream classifiers* are measured. The best result per row is highlighted in gray, the second-best is in light gray. There is no result for additive corruption under typos because intra-word noise (modifying characters except for the first and last characters) (i.e., typos) never results in additive corruption. Baseline similarity scores are calculated between canonical words and the word "the".

superfluous. This assertion stems from our demonstration in Table 7 that such aggressive search criteria are improbable to produce missing corruption.

Models	Intra
BERT	0.50%
RoBERTa	0.52%
ALBERT	0.43%

Table 7: Proportion of abbreviations causing missing corruption.

Provided below is a comprehensive inventory of the canonical words and their corresponding noisy counterparts responsible for inducing missing corruption. It is worth noting that these noisy words are completely imperceptible to human cognition.

- 24 word pairs under RoBERTa: enthral-enth, upgradable-upgr, abysmal-abys, chintzy-chzy, emphatic-emph, enslave-ensl, extraneous-extr, implacable-impl, implausible-impl, implicate-impl, imprudent-impr, inflame-infl, instable-inst, intransigent-intr, irksomeness-irks, obscenity-obsc, obtrusive-obtr, ungrateful-ungr, unscrupulous-unsc, unsteadily-unst, unsteadiness-unst, unsteady-unst, unsteady-unsty, untruthful-untr;
- 23 word pairs under BERT: enthral-enth, exemplar-expl, exemplar-empl, idyllic-idyl, stylish-styl, abysmal-abys, brutish-brsh, crummy-crmy, enslave-ensl, hysteric-hyst, impenitent-impt, incognizant-inct, inconstant-inct, inexplainable-inpl, infamy-inmy, inflame-infl, irksomeness-irks, obscenity-obsc, obtrusive-obtr, unscrupulous-unsc, unspeakable-unsp, untrue-untr, untruthful-untr;
- 2 word pairs under ALBERT: enthral-enth, exemplar-exmp.

E Performance of PLMs under Different Noise

We compare the effect of two noise models "Naturally and frequently occurring typos" and "Non-standard orthography" with both the lexicon dataset and two sentential datasets. For a fair comparison, we constrain

the length of *letter-reduplication* to 1. The accuracy of the noisy data and their standard deviation are reported in Table 8 and Table 9, respectively. It can be seen that the types of noise models in our experiments have no much distinction on model performance, except for the *Swap*.

Data	Noise Type	BERT	RoBERTa	ALBERT
Accuracy				
SST-2	Clean	0.93	0.85	0.92
	Keyboard	0.66	0.66	0.67
	Swap	0.71	0.72	0.72
	Letter-repetition	0.63	0.7	0.65
AG-News	Clean	0.95	0.8	0.92
	Keyboard	0.88	0.62	0.86
	Swap	0.89	0.62	0.86
	Letter-repetition	0.88	0.61	0.86
Similarity				
Setiment Lexicon	Keyboard	0.39	1	0.47
	Swap	0.45	1	0.5
	Letter-repetition	0.36	1	0.49
SST-2	Keyboard	0.5	0.52	0.61
	Swap	0.61	0.56	0.66
	Letter-repetition	0.46	0.55	0.58
AG-News	Keyboard	0.85	0.47	0.72
	Swap	0.87	0.5	0.75
	Letter-repetition	0.85	0.48	0.74

Table 8: Performance of PLMs under Different Noise

Data	BERT	RoBERTa	ALBERT
Similarity			
Lexicon	0.037	0	0.016
SST-2	0.061	0.016	0.035
AG-News	0.009	0.013	0.013
Accuracy			
SST-2	0.035	0.022	0.033
AG-News	0.004	0.004	0.004

Table 9: Standard deviations of PLMs' performance under different types of noise.