Contextual Text Denoising with Masked Language Models

Yifu Sun [*]
Tencent
yifusun2016@outlook.com

Haoming Jiang
Georgia Tech
jianghm@gatech.edu

Abstract

Recently, with the help of deep learning models, significant advances have been made in different Natural Language Processing (NLP) tasks. Unfortunately, state-of-the-art models are vulnerable to noisy texts. We propose a new contextual text denoising algorithm based on the ready-to-use masked language model. The proposed algorithm does not require retraining of the model and can be integrated into any NLP system without additional training on paired cleaning training data. We evaluate our method under synthetic noise and natural noise and show that the proposed algorithm can use context information to correct noise text and improve the performance of noisy inputs in several downstream tasks.

1 Introduction

Based on our prior knowledge and contextual information in sentences, humans can understand noisy texts like misspelled words without difficulty. However, NLP systems break down for noisy text. For example, Belinkov and Bisk (2017) showed that modern neural machine translation (NMT) system could not even translate texts with moderate noise. An illustrative example of English-to-Chinese translation using Google Translate [1] is presented in Table 1.

Text correction systems are widely used in real-world scenarios to address noisy text inputs problem. Simple rule-based and frequency-based spell-checker are limited to complex language systems. More recently, modern neural Grammatical Error Correction (GEC) systems are developed with the help of deep learning (Zhao et al., 2019; Chollampatt and Ng, 2018). These GEC systems heavily rely on annotated GEC corpora, such as CoNLL-2014 (Ng et al., 2014). The parallel GEC corpora, however, are expansive, limited, and even unavailable for many languages. Another line of researches focuses on training a robust model that inherently deals with noise. For example, Belinkov and Bisk (2017) train robust character-level NMT models using noisy training datasets, including both synthetic and natural noise. On the other hand, Malykh et al. (2018) consider robust word vectors. These methods require retraining the model based on new word vectors or noise data. Retraining is expensive and will affect the performance of clean text. For example, in Belinkov and Bisk (2017), the robustness scarifies the performance of the clean text by about 7 BLEU score on the EN-FR translation task.

In this paper, we propose a novel text denoising algorithm based on the ready-to-use masked language model (MLM, Devlin et al. (2018)). Notice that we are using English Bert. For other languages, We need to use MLM model pre-trained on that specific language. The design follows the human cognitive process that humans can utilize the context, the spell of the wrong word (Mayall et al., 1997), and even the location of the letters on the keyboard to correct noisy text. The MLM essentially mimics the process that the model predicts the masked words based on their context. There are several benefits of the proposed method:

- Our method can make accurate corrections based on the context and semantic meaning of the whole sentence as Table 1 shows.
- The pre-trained masked language model is ready-to-use (Devlin et al., 2018; Liu et al., 2019). No extra training or data is required.
- Our method makes use of Word Piece embeddings (Wu et al., 2016) to alleviate the out-of-vocabulary problem.

2 Method

Our denoising algorithm cleans the words in the sentence in sequential order. Given a word, the

[*] Work done at Georgia Tech.

[1] https://translate.google.com; Access Date: 08/09/2019

Method	Input Text	Google Translate
Clean Input	there is a fat duck swimming in the lake	湖里 有一只胖鸭子在游泳
Noisy Input	there is a fat dack swimming in the leake	在 leake 里游泳时有一个 胖子
Spell-Checker	there is a fat sack swimming in the leak	在 泄露处 有一个肥胖 袋在游泳
Grammaly[2]	there is a fat dack swimming in the lake	湖里 游泳很胖
Ours	there is a fat duck swimming in the lake	湖里 有一只胖鸭子在游泳

Table 1: Illustrative example of spell-checker and contextual denoising.

algorithm first generates a candidate list using the MLM and then further filter the list to select a candidate from the list. In this section, we first briefly introduce the masked language model, and then describe the proposed denoising algorithm.

2.1 Masked Language Model

Masked language model (MLM) masks some words from a sentence and then predicts the masked words based on the contextual information. Specifically, given a sentence $\boldsymbol{x} = \{x_i\}_{i=1}^L$ with L words, a MLM models

$$p(x_j|x_1, ..., x_{j-1}, [MASK], x_{j+1}, ..., x_L),$$

where $[MASK]$ is a masking token over the j-th word. Actually, MLM can recover multiple masks together, here we only present the case with one mask for notation simplicity. In this way, unlike traditional language model that is in left-to-right order (i.e., $p(x_j|x_1, ..., x_{j-1})$), MLM is able to use both the left and right context. As a result, a more accurate prediction can be made by MLM. In the following, we use the pre-trained masked language model, BERT (Devlin et al., 2018). So no training process is involved in developing our algorithm.

2.2 Denoising Algorithm

The algorithm cleans every word in the sentence with left-to-right order except for the punctuation and numbers by masking them in order. For each word, MLM first provide a candidate list using a transformed sentence. Then the cleaned word is selected from the list. The whole process is summarized in Algorithm 1.

Text Masking The first step is to convert the sentence $\boldsymbol{x}$ into a masked form $\boldsymbol{x}'$. With the use of Word Piece tokens, each word can be represented by several different tokens. Suppose the j-th word (that needs to be cleaned) is represented by the j_s-th token to the j_e-th token, we need to mask them out together. For the same reason, the number of tokens of the expected cleaned word is unknown. So we use different number of masks to create the masked sentence $\{\boldsymbol{x}'_n\}_{n=1}^N$, where $\boldsymbol{x}'_n$ denotes the masked sentence with n-gram mask. Specifically, given $\boldsymbol{x} = x_1, ..., x_{j_s}, ..., x_{j_e}, ..., x_L$, the masked form is $\boldsymbol{x}'_n = x_1, ..., [MASK] \times n, ..., x_L$. We mask each word in the noisy sentence by order. The number of masks N can not be too small or too large. The candidate list will fail to capture the right answer with a small N. However, the optimal answer would fit the noisy text perfectly with a large enough N. Empirically, we find out $N = 4$ is sufficiently large to obtain decent performance without too much overfitting.

Text Augmentation Since the wrong word is also informative, so we augment each masked text $\boldsymbol{x}'_n$ by concatenating the original text $\boldsymbol{x}$. Specifically, the augmented text is $\widetilde{\boldsymbol{x}}_n = \boldsymbol{x}'_n[SEP]\boldsymbol{x}$, where $[SEP]$ is a separation token.[3]

Compared with directly leaving the noisy word in the original sentence, the masking and augmentation strategy are more flexible. It is benefited from that the number of tokens of the expected word does not necessarily equal to the noisy word. Besides, the model pays less attention to the noisy words, which may induce bias to the prediction of the clean word.

Candidate Selection The algorithm then constructs a candidate list using the MLM, which is semantically suitable for the masked position in the sentence. We first construct candidate list V_c^n for each $\widetilde{\boldsymbol{x}}_n$, and then combine them to obtained the final candidate list $V_c = V_c^1 \cup \cdots \cup V_c^N$. Note that we need to handle multiple masks when $n > 1$. So we first find k most possible word pieces for each mask and then enumerate all possible combinations to construct the final candidate list. Specifically,

$$\begin{aligned} V_c^n = &\text{Top-}k\{p([MASK]_1 = w|\widetilde{\boldsymbol{x}}_n)\}_{w \in V} \\ &\times \cdots \times \text{Top-}k\{p([MASK]_n = w|\widetilde{\boldsymbol{x}}_n)\}_{w \in V}, \end{aligned}$$

where V is the whole vocabulary and $\times$ means the Cartesian product.

[2] https://app.grammarly.com; Access Date: 08/09/2019

[3] In BERT convention, the input also needs to be embraced with a $[CLS]$ and a $[SEP]$ token.

There may be multiple words that make sense for the replacement. In this case, the spelling of the wrong word is useful for finding the most likely correct word. We use the edit distance to select the most likely correct word further.

$$w_c = \arg\min_{w \in V_c} E(w, x_j),$$

where $E(w, x_j)$ represent the edit distance between w and the noisy word x_j.

Algorithm 1: Denoising with MLM

```
Input: Noisy sentence $\boldsymbol{x} = \{x_i\}_{i=1}^{L}$
Output: Denoised sentence $\boldsymbol{x} = \{x_i\}_{i=1}^{L}$
for $i = 1, 2, ..., L$ do
    $\{\boldsymbol{x}'_n\}_{n=1}^{N} = \text{Masking}(\boldsymbol{x})$ ;
    $\{\widetilde{\boldsymbol{x}}_n\}_{n=1}^{N} = \{\text{Augment}(\boldsymbol{x}'_n, \boldsymbol{x})\}_{n=1}^{N}$ ;
    for $n = 1, 2, ..., N$ do
        $V_c^n = \text{Candidate}(\widetilde{\boldsymbol{x}}_n)$ ;
    end
    $V_c = V_c^1 \cup \cdots \cup V_c^N$ ;
    $w_c = \arg\min_{w \in V_c} E(w, x_j)$ ;
    $x_i = w_c$;
end
```

3 Experiment

We test the performance of the proposed text denoising method on three downstream tasks: neural machine translation, natural language inference, and paraphrase detection. All experiments are conducted with NVIDIA Tesla V100 GPUs. We use the pretrained pytorch Bert-large (with whole word masking) as the masked language model [4]. For the denoising algorithm, we use at most $N = 4$ masks for each word, and the detailed configuration of the size of the candidate list is shown in Table 2. We use a large candidate list for one word piece which covers the most cases. For multiple masks, a smaller list would be good enough.

For all tasks, we train the task-specific model on the original clean training set. Then we compare the model performance on the different test sets, including original test data, noise test data, and cleaned noise test data. We use a commercial-level spell-checker api [5] as our baseline method.

In this section, we first introduce how the noise is generated, and then present experimental results of three NLP tasks.

[4]https://github.com/huggingface/pytorch-pretrained-BERT

[5]https://rapidapi.com/montanaflynn/api/spellcheck; Access Date: 08/09/2019

No. of $[MASK]$ (n)	Top k	Size
1	3000	3000
2	5	25
3	3	27
4	2	16
Total:		3068

Table 2: Size of the candidate list

3.1 Noise

To control the noise level, we randomly pick words from the testing data to be perturbed with a certain probability. For each selected word, we consider two perturbation setting: artificial noise and natural noise. Under *artificial noise* setting, we separately apply four kinds of noise: Swap, Delete, Replace, Insert with certain probability. Specifically,

- Swap: We swap two letters per word.
- Delete: We randomly delete a letter in the middle of the word.
- Replace: We randomly replace a letter in a word with another.
- Insert: We randomly insert a letter in the middle of the word.

Following the setting in (Belinkov and Bisk, 2017), the first and the last character remains unchanged.

For the *artificial noise*, we follow the experiment of Belinkov and Bisk (2017) that harvest naturally occurring errors (typos, misspellings, etc.) from the edit histories of available corpora. It generates a lookup table of all possible errors for each word. We replace the selected words with the corresponding noise in the lookup table according to their settings.

3.2 Neural Machine Translation

We conduct the English-to-German translation experiments on the TED talks corpus from IWSLT 2014 dataset [6]. The data contains about $160,000$ sentence pairs for training, $6,750$ pairs for testing.

We first evaluate the performance using a 12-layer transformer implemented by fairseq (Ott et al., 2019). For all implementation details, we follow the training recipe given by fairseq [7]. We also evaluate the performance of Google Translate.

[6]https://wit3.fbk.eu/archive/2014-01/texts/en/de/en-de.tgz

[7]https://github.com/pytorch/fairseq/tree/master/examples/translation

For the artificial noise setting, we perturb 20% words and apply each noise with probability 25%. For that natural noise setting, we also perturb 20% words. All experiment results is summarized in Table 3, where we use BLEU score (Papineni et al., 2002) to evaluate the translation result.

Text Source	Google	Fairseq
Original	31.49	28.06
Artificial Noise	28.11	22.27
+ Spell-Checker	26.28	21.15
+ Ours	**28.96**	**25.80**
Natural Noise	25.22	17.29
+ Spell-Checker	20.90	15.04
+ Ours	**25.49**	**21.40**

Table 3: BLEU scores of EN-to-DE tranlsation

As can be seen, both fairseq model and Google Translate suffer from a significant performance drop on the noisy texts with both natural and synthetic noise. When using the spell-checker, the performance even drops more. Moreover, our purposed method can alleviate the performance drop.

3.3 Natural Language Inference

We test the algorithm on Natural Language Inference (NLI) task, which is one of the most challenge tasks related to the semantics of sentences. We establish our experiment based on the SNLI (the Stanford Natural Language Inference, Bowman et al. (2015)) corpus. Here we use accuracy as the evaluation metric for SNLI.

Here we use state-of-the-art 400 dimensional Hierarchical BiLSTM with Max Pooling (HBMP) (Talman et al., 2019). The implementation follows the publicly released code [8]. We use the same noise setting as the NMT experiments. All results are presented in Table 4. We observe performance improvement with our method. To see if the denoising algorithm would induce noises to the clean texts, we also apply the algorithm to the original sentence and check if performance will degrade. It can be seen that, unlike the traditional robust model approach, applying a denoising algorithm on a clean sample has little influence on performance.

As shown in the Table4, the accuracy is very close to the original one under the artificial noise. Natural noises contain punctuations and are more complicated than artificial ones. As a result, inference becomes much harder in this way.

[8]https://github.com/Helsinki-NLP/HBMP

Method	Original	Artificial Noise	Natural Noise
HBMP	84.0	75.0	74.0
+Spell-Checker	84.0*	63.0	68.0
+Ours	83.0*	**81.0**	**77.0**

Table 4: SNLI classification accuracy with artificial noise and natural noise. *: Applying denoising algorithm on original texts.

3.4 Paraphrase Detection

We conducted Paraphrase detection experiments on the Microsoft Research Paraphrase Corpus (MRPC, Dolan and Brockett (2005)) consisting of 5800 sentence pairs extracted from news sources on the web. It is manually labelled for presence/absence of semantic equivalence.

We evaluate the performance using the state-of-the-art model: fine-tuned RoBERTa (Liu et al., 2019). For all implemented details follows the publicly released code [9]. All experiment results is summarized in Table 5. We increase the size of the candidate list to $10000+25+27+16 = 10068$ because there are a lot of proper nouns, which are hard to predict.

Method	Original	Artificial Noise	Natural Noise
RoBERTa	84.3	81.9	75.2
+Spell-Checker	82.6	81.3	75.4
+Ours	83.6	**82.7**	**76.4**

Table 5: Classification F1 score on MRPC

4 Conclusion and Future Work

In this paper, we present a novel text denoising algorithm using ready-to-use masked language model. We show that the proposed method can recover the noisy text by the contextual information without any training or data. We further demonstrate the effectiveness of the proposed method on three downstream tasks, where the performance drop is alleviated by our method. A promising future research topic is how to design a better candidate selection rule rather than merely using the edit distance. We can also try to use GEC corpora, such as CoNLL-2014, to further fine-tune the denoising model in a supervised way to improve the performance.

[9]https://github.com/pytorch/fairseq/tree/master/examples/roberta

References

Yonatan Belinkov and Yonatan Bisk. 2017. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Shamil Chollampatt and Hwee Tou Ng. 2018. Neural quality estimation of grammatical error correction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2528–2539.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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

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*.

Valentin Malykh, Varvara Logacheva, and Taras Khakhulin. 2018. Robust word vectors: Context-informed embeddings for noisy texts. In *Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text*, pages 54–63.

Kate Mayall, Glyn W Humphreys, and Andrew Olson. 1997. Disruption to word or letter processing? the origins of case-mixing effects. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 23(5):1275.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The conll-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Aarne Talman, Anssi Yli-Jyrä, and Jörg Tiedemann. 2019. Sentence embeddings in nli with iterative refinement encoders. *Natural Language Engineering*, 25(4):467–482.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 2019. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. *arXiv preprint arXiv:1903.00138*.