BLISS: Robust Sequence-to-Sequence Learning via Self-Supervised Input Representation

Anonymous ACL submission

Abstract

Data augmentations (DA) are the cores to achieving robust sequence-to-sequence learning on various NLP tasks. However, most of the DA approaches force the decoder to make predictions conditioned on the perturbed input representation, which we argue may make the sequence-to-sequence learning sub-optimal. In response to this problem, we propose a framework-level robust sequence-to-sequence learning approach, namely BLISS, via self-supervised input representation, which has the great potential to complement the data-level augmentation approaches. The core idea is to supervise the sequence-to-sequence framework with both the supervised ("input→output") and self-supervised ("perturbed input→input") information. Experimental results show that our BLISS outperforms the vanilla Transformer and five contrastive baselines on several NLP benchmarks, including machine translation, grammatical error correction and text summarization. Extensive analyses reveal that BLISS learns robust representations and rich linguistic knowledge, confirming our claim. Source code will be released upon publication.

1 Introduction

Sequence-to-sequence learning (Sutskever et al., 2014) has advanced the state-of-the-art in various natural language procecssing (NLP) tasks, such as machine translation (Bahdanau et al., 2015a; Wu et al., 2016; Vaswani et al., 2017), grammatical error correction (Kiyono et al., 2019; Kaneko et al., 2020) and text summarization (Wang et al., 2019; Zhang et al., 2020a). Sequence-to-sequence learning models are generally implemented with an encoder-decoder framework, in which the encoder receives the input sentence and predictions of the decoder are correspondingly supervised by matching the cross-entropy of ground-truth. That is, the existing sequence-to-sequence learning frameworks are supervised by the *direct correlation between the input and the output*.

To achieve robust sequence-to-sequence learning, many data augmentation methods (Kobayashi, 2018; Wu et al., 2019; Gao et al., 2019; Cheng et al., 2020; Chen et al., 2021; Morris et al., 2020) are proposed to enrich the training dataset by automatically or manually creating the perturbed input. For example, Wei and Zou (2019) show that simple data augmentation strategies, e.g. insert, swap and deletion, works well for the low-resource settings. Kobayashi (2018); Wu et al. (2019); Gao et al. (2019) employ the language models to generate the substitutions for the subset of the input sentence. Cheng et al. (2020); Chen et al. (2021); Morris et al. (2020) adopt the adversarial techniques to generate the adversarial samples to enhance the model generalization. Although those data-level approaches are straightforward and easy to use, all the above methods *force the decoder to make lexical choices conditioned on the perturbed input representation*, which we argue are sub-optimal for sequence-to-sequence learning.

In response to this problem, we propose a framework-level robust approach to making the most of the perturbed input in sequence-to-sequence learning via self-supervised input representation. The core idea is supervising the sequence-to-sequence framework with both the *correlation* between input and output, and *self-supervisions* between perturbed input and original input. In particular, we employ two extremely simple and effective data augmentation techniques, i.e. shuffle and replacement, as the input perturbing function. Then, we propose a smoothness controller to harness the perturbing degree. Based on the perturbed input, we correspondingly design a self-supervision mechanism upon the top of the encoder, where we choose the token prediction and position prediction as two self-supervised objectives to restore the perturbed subset. By doing so, we can achieve robust sequence-to-sequence learning by fully exploiting the super-

vised ("input→output") and self-supervised ("perturbed input→input") information.

We validated our approach on several sequence-to-sequence NLP tasks in §4.4, including machine translation (Bahdanau et al., 2015b; Vaswani et al., 2017), grammatical error correction (Wang et al., 2019; Zhang et al., 2020b) and text summarization (Kiyono et al., 2019; Kaneko et al., 2020), across five datasets. We show that our BLISS consistently outperforms the strong baseline Transformer and five competitive data augmentation approaches. Experiments on translation show that our proposed BLISS yields consistent improvements, ranging from 0.5 upto 2.1 BLEU points. As for correction and summarization tasks, we achieve +2.0 $F_{0.5}$ and +0.4 Rouge-L improvements against strong Transformer, demonstrating the effectiveness and universality of our approach. We conduct comprehensive analyses in §4.5 to understand when and why our BLISS works. We show that our framework-level self-supervised BLISS can be combined with existing augmentation approach, e.g. SwitchOut (Wang et al., 2018), to achieve further improvement. Also, our BLISS is robust to inference noises and hyper-parameters compared to baselines. Importantly, through probing task (Conneau and Kiela, 2018), we found that our model could preserve significantly rich linguistic knowledge against vanilla Transformer. Our main contributions can be summarized as:

- We introduce a robust sequence-to-sequence learning framework via self-supervised input representation, which has the potential to complement existing data augmentation approaches.
- Our approach provide a unified framework to make the most of existing supervised signals, i.e. correlation between input and output, and self-supervised signals, i.e. self-supervisions between perturbed input and original input.
- We empirically validate the the effectiveness and universality on extensive experiments across tasks and datasets.

2 Related Work

Our work is inspired by two lines of research: i) designing self-supervisions and ii) data augmentation.

Designing Self-Supervisions Self-supervision signals have been widely investigated in language model pretraining and unsupervised learning. Devlin et al. (2019) propose the mask language model, where they substitute a subset of tokens in the input sentence by a special symbol [MASK], and then predicts the missing tokens by the residual ones. MASS (Song et al., 2019) presents a sequence-to-sequence pre-training framework, which takes non-mask tokens as the encoder input and leverages masked tokens as the decoder input as well as the to-be-predicted target. STRUCTBERT (Wang et al., 2020) extends BERT by leveraging the structural information: word-level ordering and sentence level ordering. SpanBERT (Joshi et al., 2020) masks random contiguous spans rather than individual tokens and additionally introduces span-boundary objective. Different from these works that apply self-supervisions to the cost pre-train stage and fine-tune them on the down-stream tasks, we design the self-supervision objectives for input sentence to complement the existing MLE generation objectives to achieve further improvement.

Similar to our work, there exists several works that combine self-supervisions with from-scratch sequence-to-sequence model training. Guo et al. (2020b) introduce mask task to non-autoregressive translation model to fully exploit the undertrained encoder. Siddhant et al. (2020) propose to make use of monolingual data by self-supervisions in multilingual translation. Cheng et al. (2021) combine self-supervised and supervised learning to optimize the machine translation models especially for the rich-resource settings. Different from these works, we propose a plug-and-play self-supervised input representation approach for general sequence-to-sequence tasks, which could be used to complement any data augmentation approaches and consistently enhance the model performance.

Data Augmentation Artetxe et al. (2018); Lample et al. (2018) randomly shuffle the words within a fixed window size to construct the perturbed sentence. Iyyer et al. (2015) drop some words randomly in the source sentence for learning an auto-encoder to help train the unsupervised NMT model. Xie et al. (2017) replace the word with a placeholder token or a word sampled from the frequency distribution of vocabulary. Wang et al. (2018) introduce SwitchOut, which replace words in the source/target sentences with other words form the source/target vocabulary. Guo et al. (2020a) introduce

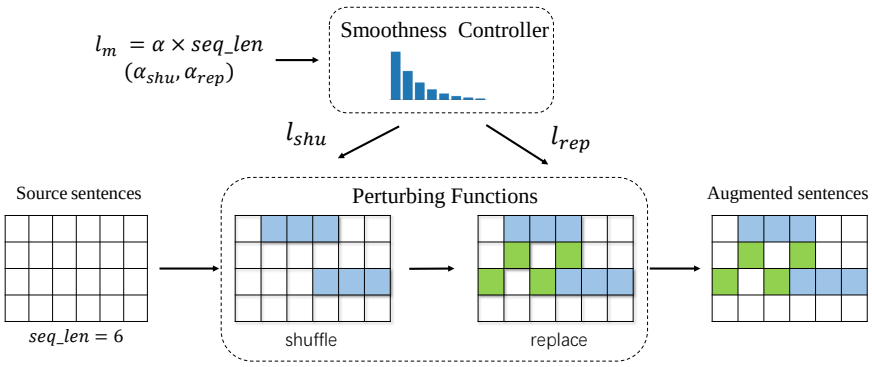

Figure 1: Illustration of the proposed smooth augmented data generator in §3.2, which consists of two components, i.e. perturbing functions and smoothness controller, represented by dashed rounded rectangles, respectively. The blue block represent tokens been shuffled while the green block represent tokens been replaced.

SeqMix to mix up pairs of source sentences or decoder input sentences. Wei and Zou (2019) experiments with easy data augmentation methods like randomly insert, swap and delete, but they found these simple methods take little effect with full datasets. Our work significantly differs from these work. We do not predict the target lexicons conditioned on these perturbed input directly. Rather, we propose to recover the noised input with encoder, thus the conditional representation for decoder preserve much linguistic knowledge (See §4.5).

3 Self-Supervised Input Representation

In this section, we first review the sequence-to-sequence learning in §3.1. Then we introduce the smoothed data augmentation technique, namely SMOOTH AUGMENTED DATA GENERATOR in §3.2. Finally §3.3 elaborates our proposed SELF-SUPERVISED INPUT REPRESENTATION approach.

3.1 Preliminaries

Sequence-to-Sequence Learning Given the target sequence $\mathbf{y} = \{y_1, y_2, ..., y_t\}$ conditioned on a source sequence $\mathbf{x} = \{x_1, x_2 ..., x_s\}$, the objective of Seq2Seq model can be formulated as $\hat{\mathbf{y}} = argmax\ logP(\mathbf{y}|\mathbf{x})$. Up to now, Transformer advanced the state of art results compared to other architectures. Thus, we employ Transformer as the strong baseline and test bed. Transformer consists of an encoder equipped with several identical layers to map the source sequence $\mathbf{x}$ into intermediate representation $\mathbf{h}$ and a decoder equipped with several identical layers take $\mathbf{h}$ as input and generates $\mathbf{y}$ target sequences autoregressively:

$$\mathbf{h} = enc(\mathbf{x} + \mathbf{pos\_emb}(\mathbf{x}))$$
$$\hat{\mathbf{y}}_{\leq \mathbf{t}} = dec(\hat{\mathbf{y}}_{<\mathbf{t}} + \mathbf{pos\_emb}(\hat{\mathbf{y}}_{<\mathbf{t}}), \mathbf{h})$$

Encoder and decoder composed of position-wise feed-forward network, multi-head dot-product attention network and so on, you can refer to Vaswani et al. (2017) for more details. Noticeable, tokens as well as its position embedding calculated by $pos\_emb(\cdot)$ are token as the input of encoder or decoder, which prove the existence and necessity of position information, we design position auxiliary task to restore position information in encoder representation. Decoder generate target sentence autoregressively until meeting special symbol $< eos >$. Finally, the output of the decoder $\hat{\mathbf{y}}$ is projected into the probability $P(\mathbf{y}|\mathbf{x})$, and the optimization objective can be formulated as:

$$\text{argmax}_\theta \text{log} P(\mathbf{y}|\mathbf{x}; \theta_{enc}, \theta_{dec}) \quad (1)$$

where θ_{enc} and θ_{dec} denote the parameters of the encoder and decoder respectively.

3.2 Smooth Augmented Data Generator

As Seen in Figure 1, our Smooth Augmented data Generator compose of two parts, perturbing functions and smoothness controller.

Perturbing Functions As shown in Figure 1, we feed the source sentences into two perturbing functions, shuffle function and replace function sequentially. For each function, we randomly select γ percentage of source sentences for generating augmented data. Specifically, we randomly shuffle tokens within a l_{shu} sized window in the shuffle function, and randomly replace l_{rep} tokens in the

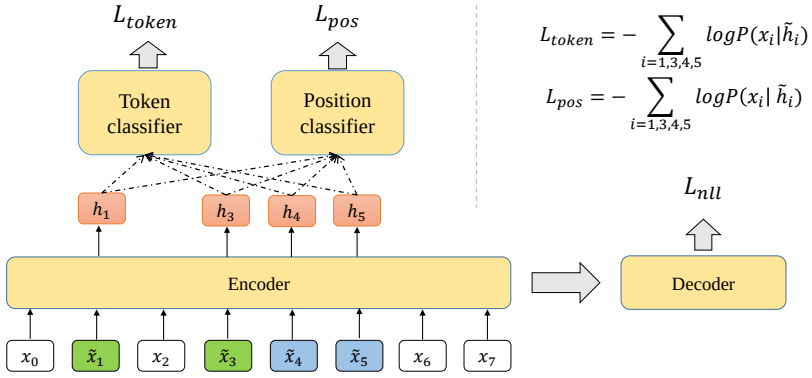

Figure 2: The illustration of our proposed self-supervised input representation (§3.3) in sequence-to-sequence learning framework. We add two classifier to predict the token and position of perturbed tokens synthesized by the smooth augmented data generator in §3.2. The meaning of blue rectangle and green rectangle is the same as in Figure 1. The red rectangles represent disturbed tokens' intermediate representation produced by the top layer of encoder.

source sentences with other words form the source vocabulary in replace function.

Smoothness Controller We set $\alpha_{shu}, \alpha_{rep}$ to control the maximum number of shuffled and replaced tokens respectively. Without smoothness controller, models can only see augmented data with $\alpha_{shu}L$ shuffle tokens of $\alpha_{rep}L$ replaced tokens, where L is the length of sequence. To balance the diversity and similarity of augmented data, we design a smoothness controller to get a smooth distribution of augmented data with different perturbing tokens. Specifically, we hypothesis sampling the number of perturbed tokens l from geometric distribution $l|_{l<=\alpha L} \sim Geometric(p)$, where p is between 0 and 1[1]. Then, since l is limited by a upper bound, we normalize the distribution of sampling l. Finally we sample l according to the probability distribution expressed as in Equation 2. For shuffle function and replacement function, we repeat the above procedures individually and obtain l_{shu} and l_{rep} for perturbing operations.

$$P(l) = \frac{p(1-p)^{l-1}}{\sum_{i=1}^{L_m} p(1-p)^{i-1}} \delta_{1-L_m}(l) \quad (2)$$

$\delta_{1-L_m}(l)$ equal 1 when l is equal or greater than 0 and equal or smaller than L_m otherwise 0.

3.3 Self-Supervised Input Representation

Inspired by mask language model (Devlin et al., 2019), which mask tokens in source sentences and predict masked tokens on the output, we take similar procedure but two differences distinguish us between them. First, our method is used in the downstream tasks with labeled datasets while mask language model take effects in pre-training tasks with unlabeled datasets, so our method works in parallel with mask language models and is complementary with them. Second, prior studies only take token as ground truth label to supervised output. To our knowledge, we are the first to take positions as supervised labels.

Specifically, we design two softmax classifier to predict token and position by $\tilde{\mathbf{h}}$ respectively. Token classifier are responsible to predict the origin tokens of $\tilde{\mathbf{x}}$ while the position classifier predict the position of perturbed tokens. And corresponding loss function L_{token}, L_{pos} is expressed as Equation 3 and Equation 4. where x_i, p_i denote the origin tokens and absolute position, $W_{token} \in \mathbf{R}^{e\times v}$ and $W_{pos} \in \mathbf{R}^{e\times p_m}$ represents the parameters of softmax classifier, and e, v, p_m denote embedding dimension, vocabulary size and maximum position index. Following the preliminary trials, we set $p_m = 400$.

$$\mathcal{L}_{token} = \sum_i logP(x_i|\tilde{h}_i, W_{token}, \theta_{enc}) \quad (3)$$

$$\mathcal{L}_{pos} = \sum_i logP(p_i|\tilde{h}_i, W_{pos}, \theta_{enc}) \quad (4)$$

$$\mathcal{L}_{nll}(\tilde{\mathbf{x}}, \mathbf{y}) = \log P(\mathbf{y}|\tilde{\mathbf{x}}; \theta_{enc}, \theta_{dec}) \quad (5)$$

By integrating the above two loss functions with the traditional negative log-likelihood loss function as Equation 5, the complete objective function of

[1] after preliminary studies, we set p=0.2 for all tasks

our model is expressed as Equation 6:

$$\underset{\theta}{\text{argmax}}\ \mathcal{L}(x,y) = \mathcal{L}_{nll} + \lambda_{token}\mathcal{L}_{token} + \lambda_{pos}\mathcal{L}_{pos} \tag{6}$$

where $\theta = \{\theta_{enc}, \theta_{dec}, W_{token}, W_{pos}\}$, λ_{token} and λ_{pos} are hyper-parameters that balance the weights of different self-supervision objectives.

In conclusion, we add smooth augmented data generator for source sentences, and restore it's original token and position information on the encoder output. The basic intuition behind is that although the augmented sequence information is distorted locally, but the contextual information is preserved, a robust encoder should has the ability to restore correct information from distorted sentences. Besides the token of sequence, the position of tokens in the sequence play an importance role of the sequence information. So, we design the encoder to predict the position of swapped tokens to help encoder understanding the position information of sequence.

4 Experiments

4.1 Tasks and Datasets

To validate the effectiveness of BLISS, we conducted experiments on three representative tasks, which vary from the distance between input and output domains and the scale of training data:

Machine Translation takes a sentence in one language as input, and outputs a semantically-equivalent sentence in another language. We evaluate our method on three widely-used benchmarks: IWSLT14 German→English (IWSLT14 De-En[2], Nguyen et al. (2020)), WMT16 English→Romanian (WMT16 En-Ro[3], Gu et al. (2018)), and WMT14 English-German (WMT14 En-De[4], Vaswani et al., 2017). We strictly follow the dataset configurations of previous works for a fair comparison. For the IWSLT14 De-En task, we train the model on its training set with 160K training samples,and evaluate on its test set. For the WMT14 En-De task, we train the model on the training set with 4.5M training samples, where newstest2013 and newstest2014 are used as the validation and test set respectively. As for the WMT16 En-Ro task which has 610K training pairs, we utilize newsdev2016 and newstest2016 as the validation and test set. For each dataset, we tokenize the sentences by Moses (Koehn et al., 2007) and segment each word into subwords using Byte-Pair Encoding (BPE, Sennrich et al., 2016), resulting in a 32K vocabulary shared by source and target languages. All the translation tasks are evaluated with tokenized BLEU (Papineni et al., 2002) score.

Grammatical Error Correction takes a sentence with grammatical errors as input and generates a corrected sentence. We evaluate our method on CONLL14 dataset[5], which has 1.4M training samples. We closely follow Chollampatt and Ng 2018 to preprocess the data. The MaxMatch (M^2) scores (Dahlmeier and Ng, 2012) were used for evaluation with Precision, Recall, and $F_{0.5}$ values.

Text Summarization takes a long-text document as input, and generates a short and adequate summary in the same language. We evaluate our method on the the most representative summarization benchmark CNN/Daily Mail corpus[6], which contains 300K training samples. We follow (Ott et al., 2019) to preprocess the data. During testing, the minimum length was set to 55 and the maximum length was set to 140, which were tuned on the development data. We also follow Paulus et al. 2018 to disallow repeating the same trigram. We evaluate the summarization performance with the standard ROUGE metric (Lin, 2004), i.e. Rouge-1, Rouge-2, and Rouge-L.

The machine translation task has distant input/output domains (i.e. in different languages), while the other tasks has similar input/output domains (i.e. in the same language). Details of the datasets are listed in Appendix A.1.

4.2 Implementation

Our model is based on the Transformer (Vaswani et al., 2017) architecture due to its state-of-the-art performance and all the models are implemented by the open-source toolkit fairseq[7] (Ott et al., 2019). For better reproduction, we employ the base Transformer (d_{model} = d_{hidden} = 512, n_{layer} = 6, n_{head} = 8) for all tasks in this paper. All models were trained on `NVIDIA DGX A100` cluster. The hyper-parameters of training of different tasks

[2] https://wit3.fbk.eu/

[3] https://www.statmt.org/wmt16/translation-task

[4] new://www.statmt.org/wmt14/translation-task

[5] https://www.comp.nus.edu.sg/~nlp/conll14st.html

[6] https://huggingface.co/datasets/cnn_dailymail

[7] https://github.com/pytorch/fairseq

	Translation			Correction			Summarization		
	De-En	**En-Ro**	**En-De**	**Prec.**	**Recall**	**$F_{0.5}$**	**RG-1**	**RG-2**	**RG-L**
VANILLA	35.1	34.7	27.3	58.7	33.8	51.2	40.1	17.6	36.8
DROPOUT	36.3	36.4	27.2	**60.8**	34.2	52.6	40.4	17.7	37.1
BLANK	36.5	**36.7**	27.6	59.3	32.7	51.0	40.0	17.5	36.8
SHUFFLE	35.8	35.4	27.0	52.5	33.9	47.3	40.1	17.3	36.8
SEQMIX	36.4	36.3	27.4	58.3	33.5	50.8	40.2	17.6	36.9
SWITCHOUT	36.5	36.4	27.5	60.3	34.1	52.3	40.6	17.9	37.1
BLISS (ours)	**36.6**†	**36.7**†	**27.9**†‡	60.2†	**36.3**†‡	**53.2**†‡	**40.6**†	**17.9**†	**37.2**†

Table 1: Experimental results of the proposed BLISS method on the Seq2Seq tasks. Results marked with † are statistically significant compared to vanilla transformer, with ‡ are statistically significant compared to best baseline.

and datasets are listed in Appendix A.1. The hyper-parameters of our methods, including γ, α_{shu}, α_{rep}, λ_{token}, λ_{pos}, p are also listed in Appendix A.2.

4.3 Baselines

To validate the effectiveness of our methods, we compare our approach with following baselines:

- **Vanilla** (Vaswani et al., 2017): The original sequence-to-sequence training strategy without any data augmentation strategies.
- **Dropout** (Iyyer et al., 2015; Lample et al., 2018): Randomly dropping tokens with their best drop ratio 0.1.
- **Blank** (Xie et al., 2017): Randomly replacing word tokens with a placeholder, we leave their best setting ratio=0.1 as default.
- **Shuffle** (Artetxe et al., 2018; Lample et al., 2018): Randomly swapping words in nearby positions within a window size K=3.
- **SeqMix** (Guo et al., 2020a): Mixing sentence pairs on both the source and target side. We reimplement according to their public code[8].
- **SwithOut** (Wang et al., 2018): Replacing tokens with other tokens in vocabulary on the source side and target side. We reimplement according to the Appendix A.5 of their paper.

4.4 Main Results

Table 1 lists the performances of our models as well as strong baseline models on different tasks. Clearly, the proposed self-supervised input representation approach ("BLISS") significantly outperforms the vanilla Transformer in all cases, while there are still considerable differences among model variations. Specifically, on translation task, our BLISS equipped models achieve the best among all contrasted approaches, and encouragingly outperform the vanilla by averaged **+1.3** BLEU points. As for the grammatical error correction task, we achieve the **+2.0** $F_{0.5}$ scores improvement against the vanilla model, and notably, our robust self-supervised input representation approach recalls significantly more potential grammatical errors, i.e. **+2.5** percent. On the contrary, the existing data augmentation approaches, e.g. Shuffle, Blank and SeqMix, slightly undermine the GEC performance. We conjecture that such performance degradation for previous approaches is due to the lack of generalization across tasks, i.e. they are proposed for MT. As for summarization task, The results also show a promising trend against all baseline methods. All those findings demonstrate that our proposed robust self-supervised input representation approach ("BLISS") is effective and universal across language pairs and tasks.

4.5 Analysis

In this section, we provide some insights into when and why our BLISS works.

Effects of Each Component There are four carefully designed components: i) **perturbing functions** "aug" that performs shuffling and replacement to generate augmented data sequentially. ii) **Smoothness controller** "smooth" to generate augmented data with different noise degrees. iii) **Token auxiliary loss** "token" to supervise the lexical information of augmented input, which helps the

[8] https://github.com/dguo98/SeqMix/tree/main

	Translation			Correction			Summarization		
	De-En	**En-Ro**	**En-De**	**Prec.**	**Recall**	**$F_{0.5}$**	**RG-1**	**RG-2**	**RG-L**
Vanilla	35.1	34.7	27.3	58.7	33.8	51.2	40.1	17.6	36.8
BLISS	36.6	36.7	27.9	60.2	36.3	53.2	40.6	17.9	37.2
-aug-smooth	**36.4**	36.5	27.5	60.8	32.3	52.3	**40.4**	**17.6**	**36.9**
-smooth	36.5	36.5	27.6	**60.2**	34.9	52.6	**40.4**	17.7	37.0
-token	**36.4**	**36.2**	**27.4**	60.5	**32.0**	**51.3**	40.4	17.7	37.0
-pos	**36.4**	36.5	27.5	60.3	35.0	52.7	40.5	17.8	37.1

Table 2: Effects of removing each component. The metrics and datasets are same as that of Table 1. Bold represents the settings with the most performance degradation for each corresponding task.

	WMT14	**CoNLL14**	**CNN/DM**
SwitchOut	26.0	52.3	37.1
BLISS w/ S.	**26.3**	**52.9**	**37.1**

Table 3: Complementary to other work, i.e. SwitchOut (Wang et al., 2018). BLISS with SwitchOut-style augmentation function is denoted as "BLISS w/ S.". Translation, Correction and Summarization are evaluated with BLEU, $F_{0.5}$ and RG-L, respectively.

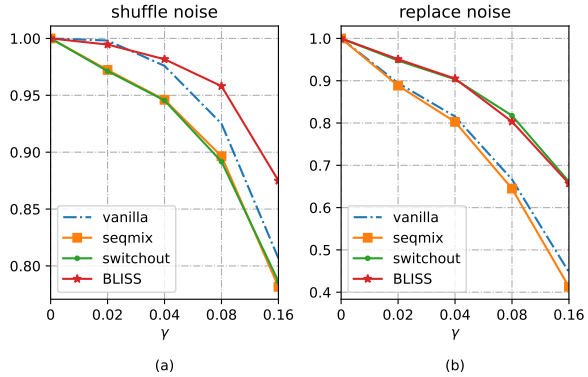

Figure 3: The performance drop when inference on noised testing data, where we test on WMT14 En-De and report the scaled BLEU scores. The noise types for the left and right figures are shuffling and replacing, respectively.

encoder capture robust token representation. iv) **Position auxiliary loss** "pos" to supervise the position information of augmented input. To verify the reasonableness of those components, we remove different components in Table 2, e.g. "-aug-smooth", "-smooth", "-token" and "-pos", as the ablation settings. **Takeaway:** *Our proposed BLISS performs worse when either component is removed, demonstrating the effectiveness of four proposed components.*

Complementary to Related Works Our proposed BLISS enables self-supervisions from the structure-level, thus has the great potential to complement existing strong data-level methods. Here we choose SwitchOut (Wang et al., 2018) due to its competitive performance in main experiments. We replace the vanilla simple augmentation function in BLISS, i.e. shuffle and replacement, with SwitchOut and conduct the experiments in Table 3. **Takeaway:** *Our proposed structure-level self-supervised approach BLISS achieves further improvement across different sequence-to-sequence tasks with advanced data augmentation functions, e.g. SwitchOut, showing its appealing expandability.*

BLISS is Robust to the Inference Noises Our self-supervised input representation is expected to tolerate the inputting noises to some extent. To validate our hypothesis, we inject two types of artificial noise, e.g. shuffle and replacement, into the test samples with different ratios ranging from {2%, 4%, 8% and 16%}. For shuffle noise, we select a span whose length is αl (l is the length of source sentence). Then we shuffle the order of words within the span. As for the replacement noise, we follow our replacement function but without smoothness controller, where we randomly replace αl tokens with other tokens in the vocabulary. Figure 3 lists the results when performing noisy inference on WMT14 En-De task. Additionally, we analysis how the auxiliary losses improve model robustness in Appendix **?? Takeaway:** *Compared with vanilla Transformer and existing contrastive variants, as noise increases, our model "BLISS" is significantly robust to both noise, demonstrating the robustness of our approach.*

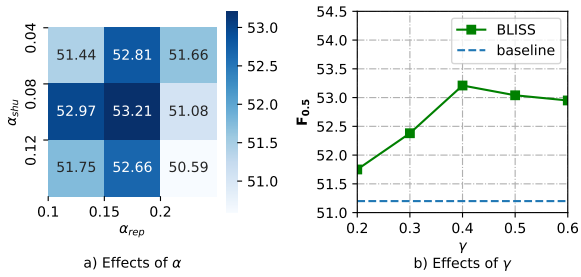

Figure 4: $F_{0.5}$ scores on CONLL14 dataset with different hyper parameters. Left: $\alpha_{shu}, \alpha_{rep}$. Right: γ.

Task		vanilla	BLISS
Surface	SeLen	93.1	**94.0**
	WC	**42.7**	41.9
Syntactic	TrDep	41.7	**44.0**
	ToCo	73.5	**75.3**
	BShif	69.3	**71.8**
Semantic	Tense	77.0	**77.5**
	SubN	77.3	**78.4**
	ObjN	75.0	**75.2**
	SoMo	50.4	**50.6**
	CoIn	62.2	**63.3**

Table 4: Performance on 10 probing tasks to evaluate the linguistic properties. Note that we train the model on WMT14 En-De.

BLISS is Robust to the Hyper-Parameters Data augmentation approaches are always sensitive to hyper-parameters. To dispel the doubt, we investigate whether our approach is robust to different hyper-parameters. We empirically study the effect of hyper parameters $\alpha_{shu}, \alpha_{rep}, \gamma$ on GEC task. We can observe from Figure 4 that with although the performance varies with hyper-parameters, the extreme values of the results are not significant, still outperforming the baseline approach. We give more analysis in Appendix A.5 to validate BLISS is not sensitive to hyper-parameters. **Takeaway:** *Our proposed BLISS is not sensitive to hyper-parameters, all hyper-parameters' variants outperform the baseline.*

BLISS Captures Better Linguistic Representation Intuitively, our proposed self-supervisions bring the capacity to correct artificial errors by restoring the token and position information, may help the encoder capture more linguistic knowledge. To verify this hypothesis, we quantitatively investigate it with probing tasks[9] (Conneau and Kiela, 2018) to study what linguistic properties are captured by the encoders. A probing task is a classification problem that focuses on simple linguistic properties of sentences. The 10 probing tasks are categorized into three groups: (1) "**Surface**" focuses on the simple surface properties learned from the sentence embedding; (2) "**Syntactic**" quantifies the syntactic reservation ability; and (3) "**Semantic**" assesses the deeper semantic representation ability. For each task, we trained the classifier on the train set, and validated the classifier on the validation set. We followed Hao et al., 2019 and Wang et al., 2019 to set the model configurations. We present the details for each probing tasks in Appendix A.3. To evaluate the representation ability of our BLISS, we compare the pretrained vanilla Transformer (Vaswani et al., 2017) and BLISS equipped machine translation model encoders, followed by a MLP classifier. Sepcifically, the mean of the top encoding layer, as sentence representation, will be passed to the classifier. Table 4 lists the results. **Takeaway:** *The proposed BLISS could preserve significant better surface, syntactic and semantic knowledge (Vanilla vs. BLISS = 65.1 vs. 66.2), confirming our hypothesis.*

5 Conclusion

In this paper, we investigate how to achieve robust sequence-to-sequence learning with self-supervised input representation. To achieve it, we propose to make the most of supervised signals and self-supervised signals with our proposed BLISS, where BLISS consists of a smooth augmented data generator and corresponding self-supervised objectives upon the top of the encoder. Experiments show that BLISS consistently outperforms the vanilla Transformer and other five data augmentation approaches in several datasets. Analyses show that BLISS indeed learns robust input representation and better linguistic information, confirming our hypothesis.

Future directions include validating our findings on more sequence-to-sequence tasks (e.g. dialogue and speech recognition) and model architectures (e.g. DynamicConv, Wu et al., 2018). Also, its worthy to explore our method to large scale sequence-to-sequence language model pretraining (e.g. BART, Lewis et al., 2020).

[9] https://github.com/facebookresearch/SentEval/tree/master/data/probing

References

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In *ICLR*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015a. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015b. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Guandan Chen, Kai Fan, Kaibo Zhang, Boxing Chen, and Zhongqiang Huang. 2021. Manifold adversarial augmentation for neural machine translation. In *Findings of ACL*.

Yong Cheng, Lu Jiang, Wolfgang Macherey, and Jacob Eisenstein. 2020. Advaug: Robust adversarial augmentation for neural machine translation. In *ACL*.

Yong Cheng, Wei Wang, Lu Jiang, and Wolfgang Macherey. 2021. Self-supervised and supervised joint training for resource-rich machine translation. In *ICML*.

Shamil Chollampatt and Hwee Tou Ng. 2018. A multilayer convolutional encoder-decoder neural network for grammatical error correction. In *AAAI*.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*. European Language Resources Association (ELRA).

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *NAACL*.

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

Fei Gao, Jinhua Zhu, Lijun Wu, Yingce Xia, Tao Qin, Xueqi Cheng, Wengang Zhou, and Tie-Yan Liu. 2019. Soft contextual data augmentation for neural machine translation. In *ACL*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *ICLR*.

Demi Guo, Yoon Kim, and Alexander M. Rush. 2020a. Sequence-level mixed sample data augmentation. In *EMNLP*.

Junliang Guo, Linli Xu, and Enhong Chen. 2020b. Jointly masked sequence-to-sequence model for non-autoregressive neural machine translation. In *ACL*.

Jie Hao, Xing Wang, Baosong Yang, Longyue Wang, Jinfeng Zhang, and Zhaopeng Tu. 2019. Modeling recurrence for transformer. In *NAACL-HLT*.

Mohit Iyyer, Varun Manjunatha, Jordan L. Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *ACL*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Trans. Assoc. Comput. Linguistics*, 8:64–77.

Masahiro Kaneko, Masato Mita, Shun Kiyono, Jun Suzuki, and Kentaro Inui. 2020. Encoder-decoder models can benefit from pre-trained masked language models in grammatical error correction. In *ACL*.

Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto, and Kentaro Inui. 2019. An empirical study of incorporating pseudo data into grammatical error correction. In *EMNLP-IJCNLP*.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *NAACL*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL*.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *ICLR*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

John X. Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP. In *EMNLP*.

Xuan-Phi Nguyen, Shafiq R. Joty, Wu Kui, and Ai Ti Aw. 2020. Data diversification: An elegant strategy for neural machine translation. In *NeurIPS*.

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 *NAACL-HLT*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *ICLR*.

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

Aditya Siddhant, Ankur Bapna, Yuan Cao, Orhan Firat, Mia Xu Chen, Sneha Reddy Kudugunta, Naveen Arivazhagan, and Yonghui Wu. 2020. Leveraging monolingual data with self-supervision for multilingual neural machine translation. In *ACL*.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: masked sequence to sequence pre-training for language generation. In *ICML*.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15:1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 5998–6008.

Liang Wang, Wei Zhao, Ruoyu Jia, Sujian Li, and Jingming Liu. 2019. Denoising based sequence-to-sequence pre-training for text generation. In *EMNLP*.

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

Xinyi Wang, Hieu Pham, Zihang Dai, and Graham Neubig. 2018. Switchout: an efficient data augmentation algorithm for neural machine translation. In *EMNLP*.

Jason W. Wei and Kai Zou. 2019. EDA: easy data augmentation techniques for boosting performance on text classification tasks. In *EMNLP-IJCNLP*.

Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. 2018. Pay less attention with lightweight and dynamic convolutions. In *ICLR*.

Xing Wu, Shangwen Lv, Liangjun Zang, Jizhong Han, and Songlin Hu. 2019. Conditional BERT contextual augmentation. In *ICCS*.

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. *In arXiv*.

Ziang Xie, Sida I. Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Dan Jurafsky, and Andrew Y. Ng. 2017. Data noising as smoothing in neural network language models. In *ICLR*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J Liu. 2020a. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *ICML*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020b. PEGASUS: pre-training with extracted gap-sentences for abstractive summarization. In *ICML*.

A Appendix

A.1 Detailed Description of Datasets and Training Setting

Table 6 gives more details of the benchmarks. It is noted that other unmentioned hyperparameters keep the same with the original paper of Transformer (Vaswani et al., 2017). All the models are implemented by the open-source toolkit fairseq (Ott et al., 2019).

A.2 Hyper-Parameters

We set $\lambda_{token} = \lambda_{pos}$=0.005 and $p = 0.2$ for all tasks, other hyper parameters vary in tasks as shown in Table 5.

	γ	α_{shu}	α_{rep}
WMT14 En-De	0.3	0.1	0.1
WMT16 En-Ro	0.4	0.1	0.1
IWSLT14 De-En	0.3	0.12	0.15
CNN/DM	0.4	0.08	0.15
CONLL	0.3	0.12	0.1

Table 5: hyper parameters of our methods in tasks.

A.3 Probing Tasks

We conducted 10 probing tasks to study what linguistic properties are captured by the encoder. 'SeLen' predicts the length of sentences in terms of number of words. 'WC' tests whether it is possible to recover information about the original words given its sentence embedding. 'TrDep' checks whether an encoder infers the hierarchical structure of sentences. In 'ToCo' task, sentences should be classified in terms of the sequence of top constituents immediately below the sentence node. 'BShif' tests whether two consecutive tokens within the sentence have been inverted. 'Tense' asks for the tense of the mainclause verb. 'SubN' focuses on the number of the main clause's subject. 'ObjN' tests for the number of the direct object of the main clause. In 'SoMo', some sentences are modified by replacing a random noun or verb with another one and the classifier should tell whether a sentence has been modified. 'CoIn' contains sentences made of two coordinate clauses. Half of sentences are inverted the order of the clauses and the task is to tell whether a sentence is intact or modified. We first extracted the sentence representations of input sentences by take average or encoder output. The classifier we used as the sentence as a Multi-Layer Perceptron(MLP) with a hidden dimention of 256. We optimized the model using the Adam optimizer with a leaning rate of 0.001 in 70 epochs for 'WC' and 'SoMo' task and 10 epochs for other tasks.

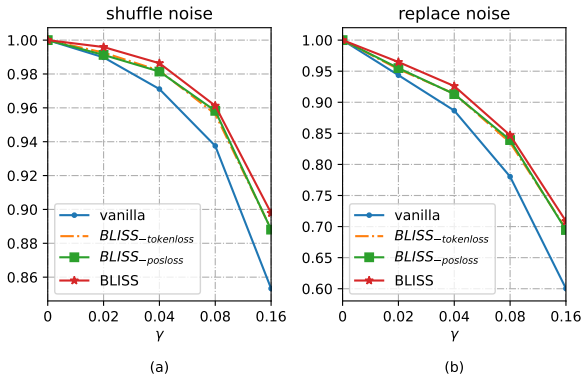

Figure 5: The performance drop when inference on noised testing data, where we test on IWSLT14 De-En and report the scaled BLUE scores. The noise types for the left and right figures are shuffling and replacing, respectively. Green and orange line represent BLISS models removing token loss and position loss individually.

A.4 How auxiliary loss improves model rebustness

We added experiments on the IWSLT DE-EN task and WMT En-Ro task as shown in Figure 5 and Figure 5, both in which we compare BLISS with variants removing token loss or position loss. From which we can draw two conclusions: 1) Removing token loss or position loss will damage the robustness of the BLISS model. 2) Token loss makes a more important difference in WMT 16 En-Ro task and makes a nearly equal difference in the IWSLT14 De-En task compared to position loss.

A.5 Hyper-Parameters Sensitivity Analysis

To validate that our model is not hyper-parameter sensitive, we do experiments with different values of hyper-parameters sampling from half of the optimal value to 1.5 times the optimal value and plot the boxplot graph below. As shown in Figure 7, the minimum values of each hyper-parameters are higher than baseline, proving the insensitivity of our hyper-parameters.

	Vocab	Sents			Training			Testing	
	Src/Tgt	Train	Dev	Test	Batch	Step	DP	Beam	LP
WMT14 En-De	32768	4.5M	3K	3K	64K	300K	0.2	5	0.6
WMT16 En-Ro	34976	0.6M	2K	2K	160K	15K	0.3	5	1.0
IWSLT14 De-En	10148	160215	7282	6750	32K	20K	0.3	5	1.0
CNN/DM	50264	0.3M	13K	11K	64K	70K	0.1	4	2.0
CONLL	33352	1.3M	5K	1K	64K	85K	0.2	6	0.6

Table 6: Statistics of the datasets and hyperparameters for the experiments. "Batch" denotes the number of source tokens and target tokens used in each training step. "DP" denotes the dropout value (Srivastava et al., 2014). "LP" denotes the length penalty (Wu et al. 2016). For GEC and text summarization tasks, we chose the checkpoint with best validation ppl for testing, for translation tasks, we choose the average of last five checkpoints for testing.

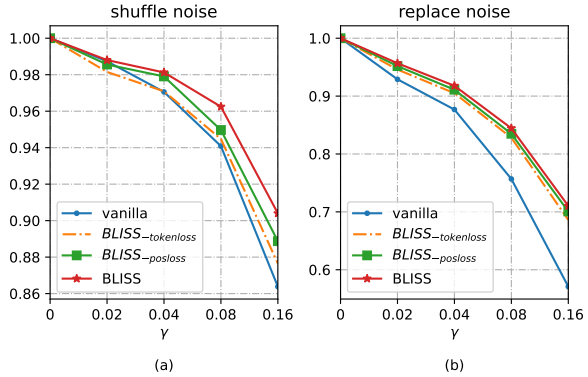

Figure 6: The performance drop when inference on noised testing data, where we test on WMT16 En-Ro and report the scaled BLUE scores. The noise types for the left and right figures are shuffling and replacing, respectively. Green and orange line represent BLISS models removing token loss and position loss individually.

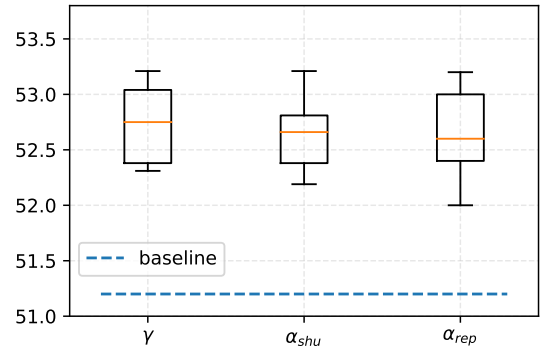

Figure 7: The box plot of three hyperparameters, $\gamma, \alpha_{shu}, \alpha_{hep}$. The blue dot line correspond to $F_{0.5}$ score of vanilla transformer. The upper bound, lower bound, middle line of each box behave the maximum, minimum and median value respectively.