Recovery Should Never Deviate from Ground Truth: Mitigating Exposure Bias in Neural Machine Translation

Jianfei He[1], **Shichao Sun**[2], **Xiaohua Jia**[1], **Wenjie Li**[2]
[1] City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong
[2] The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong
jianfeihe-2c@my.cityu.edu.hk, bruce.sun@connect.polyu.hk
csjia@cityu.edu.hk, wenjie.li@polyu.edu.hk

Abstract

In Neural Machine Translation, models are often trained with teacher forcing and suffer from exposure bias due to the discrepancy between training and inference. Current token-level solutions, such as scheduled sampling, aim to maximize the model's capability to recover from errors. Their loss functions have a side effect: a sequence with errors may have a larger probability than the ground truth. The consequence is that the generated sequences may deviate from the ground truth. This side effect is verified in our experiments. To address this issue, we propose using token-level contrastive learning to coordinate three training objectives: the usual MLE objective, an objective for recovery from errors, and a new objective to explicitly constrain the recovery in a scope that does not impact the ground truth. Our empirical analysis shows that this method effectively achieves these objectives in training and reduces the frequency with which the third objective is violated. Experiments on three language pairs (German-English, Russian-English, and English-Russian) show that our method outperforms the vanilla Transformer and other methods addressing the exposure bias.

1 Introduction

Like many other text generation tasks, models for Neural Machine Translation (NMT) (Bahdanau et al., 2014) are usually trained with *teacher forcing*. During training, ground truth tokens are used as target prefixes to the decoder, and the model learns to predict the next token conditioned on the ground truth. There is a discrepancy between this training method and inference. In inference, the ground truth tokens are not available. The target prefixes to the decoder are tokens previously generated by the model, which may include some errors. This discrepancy is referred to as *exposure bias* (Bengio et al., 2015; Ranzato et al., 2016). The main concern about exposure bias is *error accumulation*. If one error happens at one step, it is incorporated into the future steps and leads to more errors. Although there are still some doubts about whether exposure bias is a big issue for text generation (He et al., 2021), more research shows that this issue matters for NMT (Wu et al., 2018; Wang and Sennrich, 2020; Korakakis and Vlachos, 2022).

There are two approaches to mitigate the exposure bias, working at the token and sequence levels, respectively.

The token-level solutions, for example, *scheduled sampling* (Bengio et al., 2015; Mihaylova and Martins, 2019; Liu et al., 2021), usually use the tokens sampled from the model to replace the ground truth in training. The objective is to simulate the possible errors in inference and recover from these errors to reduce the error accumulation.

The sequence-level solutions directly maximize the total quality of the generated sequences with a sequence-level loss function (Ranzato et al., 2016; Shen et al., 2016; Edunov et al., 2018). There is still debate whether these solutions are stable and effective (Choshen et al., 2019; Kiegeland and Kreutzer, 2021).

This paper focuses on mitigating the exposure bias with token-level objectives.

The loss functions used in most token-level solutions have a side effect. They aim to increase the model's capability to recover from errors by maximizing the probability of the next token conditioned on some error tokens. Consequently, a sequence with errors may have a larger probability than the ground truth, and the generated sequences may deviate from the ground truth. This side effect is verified in our experiments. We discover a missing objective behind this side effect that can explicitly constrain the recovery in a scope that does not impact the ground truth. We propose to use *token-level contrastive learning* and coordinate three training objectives: the usual Maximum Likelihood Estimation (MLE) objective, an objective for recovery from errors, and a new objective constraining the recovery. Our empirical analysis shows that this method effectively meets three objectives in training. Particularly our method reduces the frequency that the third objective is violated. We conduct experiments on German-English (De–En), Russian-English (Ru–En), and English-Russian (En–Ru). Results show that our method outperforms the vanilla Transformer and other methods addressing the exposure bias.

2 Related Work

2.1 Exposure Bias and Methods to Migitate It

The existence of exposure bias is well recognized (Bengio et al., 2015; Ranzato et al., 2016), but its impact is still under debate. He et al. (2021) find that the distortion from exposure bias is limited in open-ended generation tasks. They hypothesize that the self-recovery ability of the language model is countering that distortion. In NMT, Wu et al. (2018) and Korakakis and Vlachos (2022) prove the *error accumulation* from exposure bias using *prefix switching*. They use different types of prefixes on the target side and measure the difference in the quality of the predictions. Typical prefixes include ground truth, predictions from the system, and random tokens. Wang and Sennrich (2020) provide indirect evidence for exposure bias in NMT. They train models with Minimum Risk Training (MRT), which has a sequence-level objective and inherently avoids exposure bias. The better performance of MRT than MLE justifies that exposure bias is harmful. Besides NMT, Chiang and Chen (2021) and Arora et al. (2022) quantify exposure bias in open-ended text generation tasks such as text completion.

Two categories of approaches have been proposed to mitigate exposure bias.

The token-level approach usually uses the tokens sampled from the model to replace the ground truth in training. Bengio et al. (2015) propose *Scheduled Sampling (SS)*, which dynamically takes samples from the model's predictions and replaces the ground truth used for the decoder. Zhang et al. (2019) further extend the sample space with beam search and choose the candidate translation with a sentence-level metric such as BLEU. Mihaylova and Martins (2019) implement SS to Transformer (Vaswani et al., 2017) using *two-pass decoding*. The first pass gets the predictions from the model, which are used as input to the second decoder according to the scheduler. Liu et al. (2021) propose *Confidence-Aware Scheduled Sampling (CASS)* which also uses the two-pass decoding. They improve the performance by choosing the inputs to the second decoder based on the log probability of the ground truth token. Model predictions are only used when the model is confident and has a high probability (above 0.9 in their paper). Goodman et al. (2020) propose *TeaForN* to mitigate exposure bias. They use a stack of decoders to allow the model to update based on N prediction steps. Each decoder's output is used to calculate the loss component at this decoder and is also used as the input of the next decoder. The overall loss is the weighted sum of losses from all decoders.

There are some doubts about SS. Huszár (2015) proves that SS has an improper training objective. Experiments in Mihaylova and Martins (2019) show that SS performs worse than teacher forcing for De–En. Korakakis and Vlachos (2022) use the ground truth tokens as prefixes for the decoding on a model trained with SS and find that its performance is bad compared to the MLE model. They conclude that *finetuning* with SS results in *catastrophic forgetting* (French, 1999). To avoid forgetting, they use Elastic Weight Consolidation (EWC) to regularize conditioning with model-generated prefixes. This method is similar to TFN for using a weight for prediction. But EWC works at the training parameters level, not at the loss level like TFN.

The sequence-level approach uses a sequence-level loss function and directly maximizes the total quality of the generated sequences. Ranzato et al. (2016) propose MIXER, based on a reinforcement-learning algorithm REINFORCE. MRT (Shen et al., 2016; Wang and Sennrich, 2020) aims to minimize the risk by preference to the candidate with the largest similarity to other candidates. Edunov et al. (2018) provide a summary of classic sequence-level loss functions. There is some debate on the effectiveness of these methods. Choshen et al. (2019) identify multiple weaknesses of MIXER and MRT and suspect that they do not optimize the expected reward, while Kiegeland and Kreutzer (2021) provide empirical counter-evidence to these claims.

The sequence-level approach is usually hard to converge from randomly initialized parameters and requires a baseline model trained at the token level as a starting point. In this sense, a token-level solution can be complementary to the sequence-level approach.

2.2 Using Contrastive Learning (CL) in NLP

Sun and Li (2021) apply CL to mitigate exposure bias for text summarization. They use the gold references and low-quality predictions as the positive and negative samples, respectively. The average log probability of sequences is used for the loss. Liu et al. (2022) use CL to calibrate the model. The objective is that higher-quality candidates tend to have higher log probability and are more likely to be chosen from the n-best list at the decision phase. All these methods use CL in sequence-level objectives, while our method works at the token level.

Yang et al. (2019) and Pan et al. (2021) apply CL to NMT, but they address specific issues, namely word omission errors and interim presentation for many-to-many multilingual NMT, respectively. Su et al. (2022) use CL to calibrate the model's representation space for tokens, mitigating the issue of anisotropic distribution of token representations.

3 Approach

3.1 Discover the Missing Objective

We analyze the objectives used by the current token-level methods and discover a missing objective.

We use X and y_i to denote the source sentence and the ground truth token for step i. $\hat{y}_i$ is a target token different from y_i at step i.

At step i, the MLE training with teacher forcing maximizes $p(y_i|X, y_1, ..., y_{i-1})$. If the model is effectively trained, it implies that, for any $\hat{y}_i$,

$$p(y_i|X, y_{<i}) \ > \ p(\hat{y}_i|X, y_{<i}). \qquad (1)$$

The popular token-level methods addressing exposure bias, such as Scheduled Sampling, usually aim to enhance recovery capability from errors by *maximizing* $p(y_i|X, y_{<i-1}, \hat{y}_{i-1})$, which implies that, for the sampled $\hat{y}_{i-1}$ and any $\hat{y}_i$,

$$p(y_i|X, y_{<i-1}, \hat{y}_{i-1}) \ > \ p(\hat{y}_i|X, y_{<i-1}, \hat{y}_{i-1}). \qquad (2)$$

Note: when $\hat{y}_{i-1}$ is the first error, $y_{<i-1}$ are all ground truth tokens. Otherwise, $y_{<i-1}$ may include sample tokens.

However, *maximizing* $p(y_i|X, y_{<i-1}, \hat{y}_{i-1})$ has a side effect. Although it is good for recovery, it may impact the ground truth. If $p(y_i|X, y_{<i-1}, \hat{y}_{i-1})$ exceeds $p(y_i|X, y_{<i})$, the sequence $(y_{<i-1}, \hat{y}_{i-1}, y_i)$ may have a larger probability than the ground truth $(y_{<i-1}, y_{i-1}, y_i)$. This side effect is observed in our experiments (Subsection 5.2).

This side effect implies that the model's prediction may deviate from the ground truth and generate a sequence with an error. This is particularly probable when beam search is used for decoding, where several $\hat{y}_{i-1}$ tokens have a chance to remain in the hypothese set and enter the next step during decoding.

The objective in Inequality (3) is *missing* in current training objectives:

$$p(y_i|X, y_{<i}) > p(y_i|X, y_{<i-1}, \hat{y}_{i-1}). \qquad (3)$$

With this objective, the recovery is explicitly constrained in a scope not to impact the ground truth. We propose to include it in training.

These three inequalities represent three objectives that we want to achieve. We denote them as Obj_{MLE}, Obj_{Rec} and Obj_{CRec} for Inequality (1), (2), and (3), respectively. *CRec* stands for *Constraining the Recovery*.

3.2 Token-Level Contrastive Learning

The key component in the loss function of contrastive learning is a *max* function:

$$\max\{0, \rho + S_{negative} - S_{positive}\}, \qquad (4)$$

where $S_{negative}$ and $S_{positive}$ are scores for negative and positive samples, ρ is a hyperparameter for the margin. This function implies that when the score of the negative sample plus a margin is larger than the score of the positive sample, it outputs a positive loss. Otherwise, the loss is zero. The objective is that the score of the negative sample is constrained to be at least one margin lower than the score of the positive sample.

We apply contrastive learning at the token level. The left terms in Inequality (2) and (3) are used as the scores of positive samples, while their right terms are the scores of negative samples.

3.3 Coordinate Three Objectives in One Loss Function

Three objectives in Subsection 3.1 are combined in our loss function using *multi-task learning*.

We follow the two-pass decoding in Mihaylova and Martins (2019) and Liu et al. (2021), as illustrated in Figure 1.

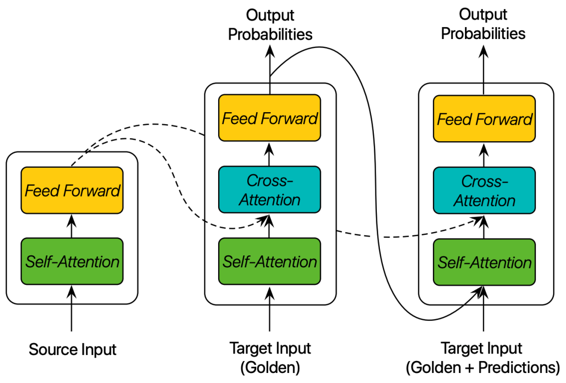


Figure 1: Scheduled sampling for the transformer with two-pass decoding (Mihaylova and Martins, 2019; Liu et al., 2021)

The first decoder is trained with teacher forcing, and its output is used for the Obj_{MLE} (Inequality 1). The Negative Log-Likelihood (NLL) with Label Smoothing (Edunov et al., 2018) is used:

$$\begin{aligned}\mathcal{L}_{MLE} = -\sum_{i=1}^{n} log\ p(y_i|X, y_{<i}) \\ -D_{KL}(f \parallel p(y_i|X, y_{<i})),\end{aligned} \tag{5}$$

where f is uniform prior distribution over all tokens in the vocabulary with the size of V, $f = \frac{1}{V}$.

We use the same strategy and hyperparameters in *Confidence-Aware Scheduled Sampling* (Liu et al., 2021) to decide the inputs to the second decoder. Predicted tokens and random tokens are used as target inputs for high-confidence positions, and the ground-truth tokens are used for low-confident positions. The decision rule can be expressed in Equation (6) below.

$$y_{i-1} = \begin{cases} y_{i-1} & if\ p(y_i|X, y_{<i}) \leq 0.9 \\ \hat{y}_{i-1} & if\ 0.9 < p(y_i|X, y_{<i}) \leq 0.95 \\ y_{rand} & if\ p(y_i|X, y_{<i}) > 0.95 \end{cases} \tag{6}$$

When the probability of the ground truth token at step i in the first decoder is no greater than 0.9, the ground truth token y_{i-1} is chosen as input for the second decoder to reinforce the teacher forcing. When the probability is between 0.9 and 0.95, the token with the maximum probability at step $i-1$ is used to simulate the model prediction in inference. When the probability is larger than 0.95, a token randomly sampled from the target sentence is used.

The output from the second decoder is used with contrastive learning for the Obj_{Rec} and Obj_{CRec}.

To meet the Obj_{Rec} from Inequality (2), we use the function below to formulate the *recovery loss*:

$$\begin{aligned}\mathcal{L}_{Rec} = \max\{0, \rho + log\ p(\hat{y}_i|X, y_{<i-1}, \hat{y}_{i-1}) \\ -log\ p(y_i|X, y_{<i-1}, \hat{y}_{i-1})\}.\end{aligned} \tag{7}$$

We use the function below to formulate the loss for the Obj_{CRec} (Inequality 3) to *constrain recovery*:

$$\begin{aligned}\mathcal{L}_{CRec} = \max\{0, \rho + log\ p(y_i|X, y_{<i-1}, \hat{y}_{i-1}) \\ -log\ p(y_i|X, y_{<i})\}.\end{aligned} \tag{8}$$

The overall loss function is a weighted sum of three components:

$$\mathcal{L} = \frac{\mathcal{L}_{MLE} + \alpha\mathcal{L}_{Rec} + \alpha\mathcal{L}_{CRec}}{1 + 2\alpha}, \tag{9}$$

where α is a hyperparameter as the weight.

4 Experiments

4.1 Datasets

Our experiments use the corpora from WMT[1]. Wang and Sennrich (2020) claim that the methods reducing exposure bias with sequence-level objectives such as MRT can particularly enhance the model's resilience to domain shift. To evaluate this claim, we conduct Out-Of-Domain (OOD) tests on De–En and Ru–En.

For De–En, we use Europarl v7, News-commentary-v12, and Common Crawl for training, Newstest2014 for validation, and Newstest2021 and EMEA[2] for in-domain and OOD testing respectively.

For Ru–En and En–Ru, we use ParaCrawl v9, News-commentary-v10, and Common Crawl for training, Newstest2014 for validation, and Newstest2021 for in-domain testing. The OOD tests for

[1]http://www.statmt.org
[2]https://opus.nlpl.eu/EMEA.php

Ru–En use the test set for the Biomedical Translation Task in WMT22[3].

These original datasets are filtered to remove low-quality data. 350 million sentences are randomly selected with the conditions below:

- The length of source and target sentences are within the range of 5 to 300.
- The disparity between the source and target sentence length does not exceed five times.

The number of sentence pairs in the final training sets for each language pair is: De–En 2.6 million, Ru–En 2.9 million, En–Ru 2.9 million.

4.2 Models

We compare our method to the vanilla Transformer model and reimplement five methods aiming at mitigating exposure bias for comparison.

- *TX* is the vanilla Transformer.
- *SS* (Mihaylova and Martins, 2019) is a typical Scheduled Sampling method based on 2-pass decoding with Transformer. We use Inverse Sigmod Decay for scheduling since it performs better than other scheduling algorithms according to Liu et al. (2021).
- *CASS* (Liu et al., 2021) is Confidence-Aware Scheduled Sampling using the best configuration in their paper, which outperforms TFN, MIXER, and MRT in their experiments.
- *TFN* (Goodman et al., 2020) uses 2 stacking decoders and combine their loss functions. According to their paper's recommendation, we use 0.4 as the second decoder's weight and shared parameter for both decoders.
- *MIXER* (Ranzato et al., 2016): Our implementation follows Kiegeland and Kreutzer (2021).
- *MRT* (Shen et al., 2016): We use 4 candidates and do not include the gold reference, same as Wang and Sennrich (2020).

Our method is denoted as *TCL* (*Token-level Contrastive Learning*). The margin ρ for the contrastive learning is set to 0.01. This means that the probability of a negative sample is allowed to reach 99% of the probability of a positive sample maximally. We conducted preliminary experiments on the weight α in the loss function. The models with $\alpha = 0.5$ got bad performance. Our results in this paper are from experiments using models trained with $\alpha = 0.1$.

Our implementation is based on the Fairseq toolkit (Ott et al., 2019) with a typical configuration [4] similar to the original Transformer (Vaswani et al., 2017). We use the BPE (Sennrich et al., 2016) mode in SentencePiece[5] for subwords with 32,000 updates and use a shared vocabulary for source and target. We use beam search for decoding. The beam size is 4.

Our experiments are based on Transformer Base (about 60 million parameters). Both the dropout rate and Lable Smoothing are set to 0.1 for all models.

The models for vanilla Transformer *TX* are trained for a minimum of 20 epochs, stopping if the validation loss does not decrease for 20 consecutive epochs. The other baseline methods and TCL use these vanilla Transformer models as pre-trained models for finetuning. During finetuning, we adopt the same early stop policy as Choshen et al. (2019), where the process is terminated if the validation loss does not decrease for ten consecutive epochs.

The token-level methods (*CASS*, *CASS*, *TFN*, and *TCL*) have similar speeds in training. It takes about 30 minutes to finish one epoch with 10 GPUs. The sequence-level methods (*Mixer* and *MRT*) are much slower since they use online samples during training. It takes *MIXER* and *MRT* about 10 hours and 14 hours to finish one epoch with 10 GPUs, respectively. This result is consistent with the experiments in Edunov et al. (2018). They find that online sampling methods can be 26 times slower than the corresponding offline methods.

We do significance tests for token-level methods. We train models with five different seeds (1–5) and report the mean and standard error over these independent runs. We use the default seed (1) in Fairseq for other experiments. We do not have significance tests for the sequence-level methods (*MIXER* and *MRT*) since they are too slow.

All GPUs that we use are Nvidia GF1080Ti.

[3]`https://www.statmt.org/wmt22/biomedical-translation-task.html`

[4]`https://github.com/facebookresearch/fairseq/tree/main/examples/scaling_nmt`

[5]`https://github.com/google/sentencepiece`

	De–En			Ru–En			En–Ru		
Metrics	BLEU	Meteor	Comet	BLEU	Meteor	Comet	BLEU	Meteor	Comet
TX	27.57	49.72	75.01	30.15	49.43	74.93	15.87	29.13	63.97
SS	27.78±.08	49.76±.12	75.16±.01	30.44±.11	49.64±.13	75.16±.07	16.78±.11	30.54±.24	65.95±.34
CASS	27.86±.18	49.74±.07	75.26±.06	30.59±.16	**49.85±.10**	75.39±.02	17.10±.28	31.08±.05	66.36±.49
TFN	27.62±.23	49.63±.19	75.16±.09	30.44±.10	49.74±.07	75.33±.09	17.04±.18	30.87±.30	66.62±.09
MIXER	27.84	49.74	**75.33**	30.03	49.67	75.36	**17.65**	**31.64**	66.77
MRT	27.41	49.52	75.29	30.39	49.69	75.07	17.15	31.29	66.04
TCL	**28.10±.16**	**49.94±.13**	**75.33±.07**	**30.59±.17**	**49.81±.13**	**75.50±.18**	**17.35±.16**	**31.56±.24**	**66.83±.13**
∆ (-TX)	**0.53**	**0.22**	**0.32**	**0.44**	**0.38**	**0.57**	**1.48**	**2.43**	**2.86**

Table 1: Performance of different methods for the in-domain tests (Newstest2021). We report mean and standard error over five independent training runs with seeds 1–5 for the token-level methods. The scores of TCL and those better than TCL are highlighted in **Bold**. ∆ is the gain of TCL compared to TX.

	De–En			Ru–En		
Metrics	BLEU	Meteor	Comet	BLEU	Meteor	Comet
TX	25.75	41.62	67.93	34.94	52.01	74.91
SS	26.17±.14	42.09±.07	68.13±.08	35.66±.06	52.51±.17	75.20±.10
CASS	26.32±.12	42.03±.07	68.23±.09	35.54±.15	52.39±.25	**75.28±.12**
TFN	26.41±.08	42.04±.06	68.32±.07	**35.85±.08**	52.57±.13	75.23±.09
MIXER	26.62	42.20	**68.50**	35.66	52.22	75.18
MRT	26.36	42.05	68.15	35.39	52.55	75.22
TCL	**26.62±.20**	**42.17±.19**	**68.34±.07**	**35.82±.11**	**52.61±.07**	**75.24±.07**
∆ (-TX)	**0.87**	**0.55**	**0.41**	**0.88**	**0.60**	**0.33**

Table 2: Performance of different methods for out-of-domain (OOD) tests. Denotations are the same as Table 1.

4.3 Evaluation and Results

We use BLEU, Meteor, and Comet for evaluation. For BLEU, We use SacreBLEU [6] (Post, 2018) [7]. For Meteor[8], we use version 1.5. For Comet[9], we use the *wmt22-comet-da* model, which scales the scores between 0 and 1. Scores for all metrics are multiplied by 100.

Table 1 and Table 2 illustrate the performance of methods for in-domain test sets (Newstest2021) and out-of-domain test sets, respectively.

TCL outperforms the vanilla Transformer in all tests. TCL gets the best performance among token-level methods in tests except for three cases, highlighted in the tables in **Bold**. The differences in scores between TCL and these three exceptions are very small (less than 0.1).

TCL is ten times more efficient in training compared to those two sequence-level methods. TCL still outperforms those methods in the majority of the tests.

TCL gets larger gains in the OOD tests than in the in-domain tests. This is consistent with the conclusion in Wang and Sennrich (2020). They claim that exposure bias is more influential in domain shift, although their experiment uses the method *MRT*.

Our analysis in Section 5.2 demonstrates that TCL achieves both *recovery* and *constraining recovery* and mitigates the exposure bias. The analysis in Section 5.3 shows the effectiveness of this method by tracking the values of three components in the loss function in training.

5 Analysis

Besides the overall performance, we investigate how these three objectives are met and whether the loss function effectively coordinates these objectives.

We start by using the *prefix switching* method. Then, we directly measure how often the three objectives in Subsection 3.1 are *NOT* met during decoding for each method. Finally, we verify the effectiveness of the loss function in Subsection 3.3 by monitoring how the values of these components

[6] `https://github.com/mjpost/sacreBLEU`
[7] case.mixed+numrefs.1+smooth.exp+tok.13a+version.2.3.1
[8] `http://www.cs.cmu.edu/~alavie/METEOR/`
[9] `https://github.com/Unbabel/COMET`

	De–En		**Ru–En**		**En–Ru**	
	Prefix	**Normal**	**Prefix**	**Normal**	**Prefix**	**Normal**
TX	41.37	27.57	42.87	30.15	30.25	15.87
SS	41.20	27.75	43.28	30.20	30.91	16.86
CASS	41.16	27.70	43.42	30.35	**31.01**	17.19
TFN	**41.77**	27.25	**43.47**	30.30	30.83	17.29
MIXER	41.40	27.84	43.43	30.03	30.54	**17.65**
MRT	40.96	27.41	43.42	**30.39**	30.83	17.15
TCL	41.65	**28.48**	43.44	**30.39**	30.79	17.33

Table 3: The inconsistency between *prefix switching* test (denoted as *Prefix*) and normal tests. Best BLEU scores are highlighted in **Bold**.

in our loss function change during training TCL and its variants.

5.1 Using Prefix Switching to Quantify Exposure Bias Is Not Reliable

Prefix Switching is often used to quantify exposure bias (Wu et al., 2018; Korakakis and Vlachos, 2022). We use various lengths of ground truth tokens as prefixes and measure the average quality of the part of the sequence from the model's prediction. The length of the prefix varies from 1 to N-1, where N is the length of the reference. After decoding, we measure the average *sentence-BLEU* scores of the prediction part of sequences. If the length of a prediction part is shorter than 4, it is not considered for the average.

Table 3 shows the results for three language pairs on the in-domain test sets using the Prefix Switching and the normal tests. In the normal tests, there are no ground-truth prefixes during decoding.

The results of these two tests are inconsistent. For example, TFN gets the best BLEU score in De–En in *Prefix Switching* testing. But it gets a score lower than the vanilla Transformer in the normal test. It reflects that using prefix switching to quantify the exposure bias may not be reliable. This issue requires further investigation.

5.2 Analysis if Three Objectives Are Met or Not

We directly detect how many times these three objectives (Obj_{MLE}, Obj_{Rec} and Obj_{CRec}) in Subsection 3.1 are met or not in decoding.

Similar to prefix switching, we use various lengths of ground truth as prefixes to the decoder. In this experiment, we only need to monitor one or two steps of decoding, not requiring the decoder to finish a prediction with an *End-of-Sentence (EOS)*.

Assume that *N* is the length of the gold reference in subwords and *k* is the length of the prefix which enumerates between 0 and N.

We need a set of *non ground-truth* tokens, $\hat{y}_{k-1}$, to test Obj_{Rec} and Obj_{CRec}. $\hat{y}_{k-1}$ is corresponding to $\hat{y}_{i-1}$ in Inequality (2) and (3). It is intractable to enumerate all tokens in the vocabulary. We choose the *m* tokens with the top probabilities from outputs at the step *k-2* and test with each of them by appending it to the ground truth prefixes $y_{<k-1}$ for decoding at step *k-1*. The ground truth token is taken out if it is in this top-m token set. We use $m \in \{1, 5, 10\}$ since the size of the beam search is usually not greater than 10 in practice.

Once $\hat{y}_{k-1}$ is selected, the decoder uses $(X, y_{<k-1}, y_{k-1})$ and $(X, y_{<k-1}, \hat{y}_{k-1})$ as inputs *respectively* and get both $p(y_k|X, y_{<k})$ and $p(y_k|X, y_{<k-1}, \hat{y}_{k-1})$. If $p(y_k|X, y_{<k})$ is the maximum in its decoding step, we can tell that Obj_{MLE} is met. If $p(y_k|X, y_{<k-1}, \hat{y}_{k-1})$ is the maximum in its step, Obj_{Rec} is met.

We use the joint probability of *bi-gram* to test Obj_{CRec}, the third (missing) objective, since the total probability of the sequence is used in decoding. Inequality (10) below is the criterion:

$$\begin{aligned} p(y_k, y_{k-1}|X, y_{<k-1}) &= \\ p(y_k|X, y_{<k}) * p(y_{k-1}|X, y_{<k-1}) & \\ &> \\ p(y_k, \hat{y}_{k-1}|X, y_{<k-1}) &= \\ p(y_k|X, y_{<k-1}, \hat{y}_{k-1}) * p(\hat{y}_{k-1}|X, y_{<k-1}) & \end{aligned} \tag{10}$$

Table 4, 5, and 6 illustrates results for different methods for De–En, Ru–En and En–Ru respectively. The event of *Not Met* is counted for each step for each objective. When the number of *non ground-truth* tokens (m) is larger than 1, such an event may happen more than once at one step. The

	Obj_{MLE}	Obj_{Rec}			Obj_{CRec}		
		Top1	**Top5**	**Top10**	**Top1**	**Top5**	**Top10**
TX	0.316	0.277	3.087	6.852	0.125	0.362	0.534
SS	0.314	**0.275**	3.091	6.855	0.127	0.362	0.530
CASS	0.314	**0.275**	**3.078**	**6.806**	0.127	0.366	0.538
TFN	**0.313**	**0.275**	3.117	6.933	0.138	0.388	0.566
TCL	0.314	**0.275**	3.088	6.868	**0.123**	**0.354**	**0.521**
MIXER	0.317	0.279	3.101	6.887	0.125	0.363	0.532
MRT	0.314	0.274	3.080	6.842	0.127	**0.353**	**0.513**

Table 4: Failure rates of three objectives for De–En. **Smaller is better**. The smallest ones are highlighted in **Bold**. The values in this table are how often the objective is *NOT* met, divided by the total number of tests (24760 in this case). *Top-m* denotes that the number of *non ground-truth* tokens ($\hat{y}_{k-1}$) used in test is *m*. CASS has a larger failure rate for the third objective Obj_{CRec} than the vanilla Transformer. This result reflects that CASS has enhanced recovery *too much* that it deviates from the ground truth. TCL is the only token-level method with lower failure rates for all objectives than the vanilla Transformer. The two sequence-level methods are not supposed to have the deviation issue, but they are tested here for reference.

	Obj_{MLE}	Obj_{Rec}			Obj_{CRec}		
		Top1	**Top5**	**Top10**	**Top1**	**Top5**	**Top10**
TX	0.288	0.254	3.078	6.891	0.104	0.281	0.394
SS	0.285	0.251	3.083	6.891	0.104	0.276	0.388
CASS	0.286	0.250	3.068	**6.838**	0.106	0.285	0.400
TFN	**0.284**	**0.249**	3.103	6.955	0.115	0.303	0.418
TCL	0.285	0.251	**3.067**	6.865	**0.103**	**0.275**	**0.387**
MIXER	0.285	0.252	3.078	6.892	0.104	0.275	0.388
MRT	0.285	0.251	3.079	6.874	0.102	0.274	0.380

Table 5: Failure rates of three objectives for Ru–En. **Smaller is better**. The denotations are the same as Table 4. The total number of tests is 27828 in this case.

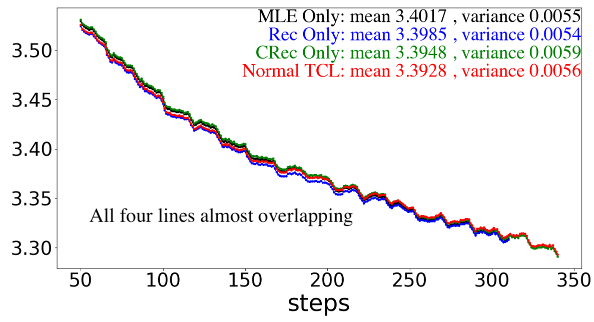
	Obj_{MLE}	Obj_{Rec}			Obj_{CRec}		
		Top1	**Top5**	**Top10**	**Top1**	**Top5**	**Top10**
TX	0.379	0.356	3.464	7.559	0.149	0.492	0.776
SS	0.375	0.351	3.462	7.560	0.145	0.487	0.774
CASS	**0.373**	**0.348**	**3.431**	**7.492**	0.151	0.504	0.794
TFN	**0.373**	0.353	3.464	7.558	0.159	0.519	0.813
TCL	0.376	0.352	3.451	7.536	**0.144**	**0.486**	**0.770**
MIXER	0.375	0.352	3.467	7.577	0.140	0.467	0.740
MRT	0.373	0.349	3.448	7.540	0.146	0.479	0.757

Table 6: Failure rates of three objectives for En–Ru. **Smaller is better**. The denotations are the same as Table 4. The total number of tests is 42442 in this case.

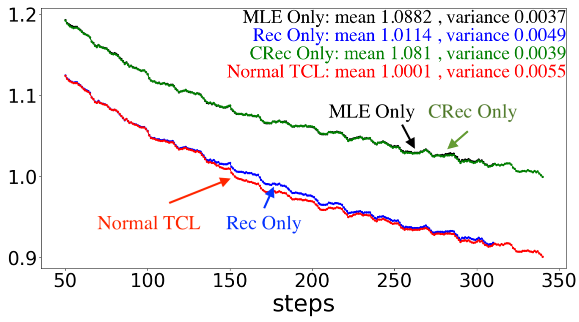
total number of events is then divided by the number of steps (for example, 24760 in De–En). The results are the *average failure rate* per token.

These tables show that CASS has the lowest failure rates for the second objective Obj_{Rec} in both De–En and En–Ru. CASS also gets relatively low failure rates for this objective in Ru–En. These results demonstrate that CASS successfully enhances the recovery capability. However, CASS has larger failure rates for the third objective Obj_{CRec} than the vanilla Transformer in all three language pairs. This result reveals that CASS has enhanced recovery *too much* that it deviates from the ground truth, which is the side effect described in Subsection 3.1.

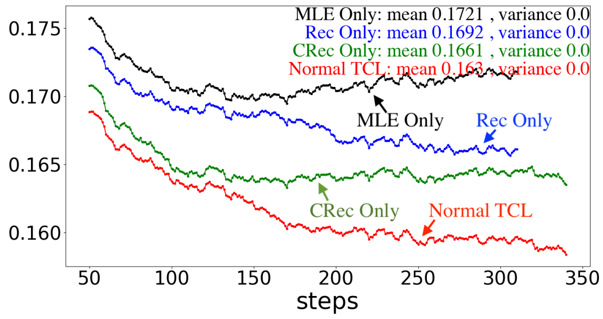
Our method TCL gets the lowest failure rate for the third objective Obj_{CRec} among the token-level methods in all tests. Furthermore, TCL is the only token-level method with lower failure rates for all objectives than the vanilla Transformer in Ru–En and En–Ru. It achieves a *pareto optimality* in the sense of improvement on both objectives: *recovery*



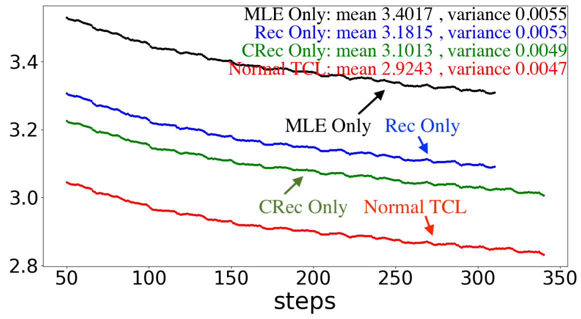
(a) The loss of MLE $\mathcal{L}_{MLE}$



(b) The loss of Recovery $\mathcal{L}_{Rec}$



(c) The loss of Constraining Recovery $\mathcal{L}_{CRec}$



(d) The total loss

Figure 2: Investigate the values of components in TCL's loss function for De–En in training. *Rec Only* denotes the model trained without applying the loss component $\mathcal{L}_{CRec}$. *CRec Only* denotes the model trained without applying the loss component $\mathcal{L}_{Rec}$. *MLE Only* denotes the model trained without applying both $\mathcal{L}_{Rec}$ and $\mathcal{L}_{CRec}$.

and *constraining recovery*. These results demonstrate that the exposure bias is mitigated by our method.

The sequence-level methods do not have the deviation issue discussed in this paper since they use sequence-level objectives in training. Their results are included in these failure rate tests for reference only. The results show that they perform well in this test, reflecting their effectiveness in mitigating exposure bias, although these methods are much slower than the token-level methods.

5.3 Effectiveness of Loss Components

There are three components in our loss function in Equation (9): $\mathcal{L}_{MLE}$, $\mathcal{L}_{Rec}$, and $\mathcal{L}_{CRec}$. We evaluate the effectiveness of these components by tracking their loss values in training TCL and its three variants by turning off one or two components. We use α_1 and α_2 to denote the weights for $\mathcal{L}_{Rec}$ and $\mathcal{L}_{CRec}$, respectively.

- *Normal TCL*: $\alpha_1 = \alpha_2 = 0.1$
- *Rec Only* (recovery): $\alpha_1 = 0.1, \alpha_2 = 0$
- *CRec Only* (constraining recovery): $\alpha_1 = 0, \alpha_2 = 0.1$
- *MLE Only*: $\alpha_1 = \alpha_2 = 0$

Figure 2 illustrates how each loss component's values vary in training for De–En. These values are reported every 100 updates during training and smoothed by taking the average with their ten right and left neighbors.

Figure 2a shows that the values of $\mathcal{L}_{MLE}$ for four models are almost the same. This component is not influenced by other two components.

Figure 2b shows the recovery loss $\mathcal{L}_{Rec}$. Even for the model *MLE Only* without $\mathcal{L}_{Rec}$ and $\mathcal{L}_{CRec}$, this loss decreases in training. This implies that models increase self-recovery capability during training even if no extra means are used to enhance it. This result supports the conclusion from He et al. (2021), although enhancing the recovery capability may not be enough to deny exposure bias's negative impact. The blue and red lines (*Rec Only* and *Normal TCL*) with the recovery component get smaller values than the other two models without this component. This illustrates that this component in the loss function effectively increases the capability of recovery.

Figure 2c shows the values of $\mathcal{L}_{CRec}$ (constraining recovery). Similar to the values of $\mathcal{L}_{Rec}$ in Figure 2b, even for the model *MLE Only* without $\mathcal{L}_{Rec}$

and $\mathcal{L}_{CRec}$, this loss decreases in training. The green and red lines (*CRec Only* and *Normal TCL*) with the component $\mathcal{L}_{CRec}$ get smaller values than the other two models without this component. This implies that using this component in the loss function effectively reduces the $\mathcal{L}_{CRec}$.

This loss surprisingly increases after a period of decreasing in training for *MLE Only* and *CRec Only*. This is the consequence of increasing the capability of self-recovery shown in Figure 2b with or without $\mathcal{L}_{Rec}$. The increasing of $p(y_i|X, y_{<i-1}, \hat{y}_{i-1})$ may result in the increase of values of $\mathcal{L}_{CRec}$ according to its definition in Equation (8). Current token-level methods that maximizes $p(y_i|X, y_{<i-1}, \hat{y}_{i-1})$ may make this contradiction more severe.

Figure 2d shows the total loss.

Table 7 shows the ablation tests using the BLEU scores for *Rec Only* (recovery) and *CRec Only* (constraining recovery) models compared to the vanilla Transformer and the normal TCL models. *Rec Only* gets worse performance than the vanilla Transformer. *CRec Only* have some gains. The normal TCL that combines these components gets extra improvement. Table 8 in Appendix A illustrates the results for En–Ru, and they lead to the same conclusion.

	De–En		
Metrics	BLEU	Meteor	Comet
Vanilla Transformer (TX)	27.57	49.72	75.01
Rec Only	27.23	49.29	75.27
Δ (-TX)	**-0.34**	**-0.43**	**0.26**
CRec Only	27.82	49.85	75.40
Δ (-TX)	**0.25**	**0.13**	**0.39**
TCL	28.48	50.20	75.55
Δ (-TX)	**0.91**	**0.48**	**0.54**

Table 7: Ablation tests. *Rec Only* (recovery) and *CRec Only* (constraining recovery) models compared to the vanilla Transformer and normal TCL models.

6 Conclusion

Current token-level methods addressing exposure bias may have a side effect: A sequence with errors may have a larger probability than the ground truth. Consequently, the generated sequence may deviate from the ground truth. Our experiments verify this side effect. We discover a missing objective behind this side effect that can explicitly constrain the recovery in a scope that does not impact the ground truth. We propose token-level contrastive learning to coordinate three objectives in the loss function: the original MLE, recovery from errors, and constraining the recovery in a scope not to exceed the ground truth. Experimental results on three language pairs show that our method outperforms the vanilla Transformer and five methods aiming at mitigating exposure bias. Empirical analysis demonstrates that this method achieves a Pareto optimality compared with the vanilla Transformer. It is also verified that each component in our loss function effectively improves the model in training.

References

Arora, Kushal, Layla El Asri, Hareesh Bahuleyan, and Jackie Chi Kit Cheung. 2022. Why exposure bias matters: An imitation learning perspective of error accumulation in language generation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 700–710.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bengio, Samy, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28.

Chiang, Ting-Rui and Yun-Nung Chen. 2021. Relating neural text degeneration to exposure bias. In *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 228–239.

Choshen, Leshem, Lior Fox, Zohar Aizenbud, and Omri Abend. 2019. On the weaknesses of reinforcement learning for neural machine translation. In *International Conference on Learning Representations*.

Edunov, Sergey, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Classical structured prediction losses for sequence to sequence learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 355–364.

French, Robert M. 1999. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4).

Goodman, Sebastian, Nan Ding, and Radu Soricut. 2020. Teaforn: Teacher-forcing with n-grams. In *Proceedings of the 2020 Conference on Empirical*

Methods in Natural Language Processing (EMNLP), pages 8704–8717.

He, Tianxing, Jingzhao Zhang, Zhiming Zhou, and James Glass. 2021. Exposure bias versus self-recovery: Are distortions really incremental for autoregressive text generation? In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5087–5102.

Huszár, Ferenc. 2015. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*.

Kiegeland, Samuel and Julia Kreutzer. 2021. Revisiting the weaknesses of reinforcement learning for neural machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1673–1681.

Korakakis, Michalis and Andreas Vlachos. 2022. Improving scheduled sampling with elastic weight consolidation for neural machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 7247–7258.

Liu, Yijin, Fandong Meng, Yufeng Chen, Jinan Xu, and Jie Zhou. 2021. Confidence-aware scheduled sampling for neural machine translation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2327–2337.

Liu, Yixin, Pengfei Liu, Dragomir Radev, and Graham Neubig. 2022. Brio: Bringing order to abstractive summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2890–2903.

Mihaylova, Tsvetomila and André FT Martins. 2019. Scheduled sampling for transformers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 351–356.

Ott, Myle, 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 the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53.

Pan, Xiao, Mingxuan Wang, Liwei Wu, and Lei Li. 2021. Contrastive learning for many-to-many multilingual neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 244–258.

Post, Matt. 2018. A call for clarity in reporting bleu scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191.

Ranzato, Marc'Aurelio, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016*.

Sennrich, Rico, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Shen, Shiqi, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1692.

Su, Yixuan, Tian Lan, Yan Wang, Dani Yogatama, Lingpeng Kong, and Nigel Collier. 2022. A contrastive framework for neural text generation. *Advances in Neural Information Processing Systems*, 35:21548–21561.

Sun, Shichao and Wenjie Li. 2021. Alleviating exposure bias via contrastive learning for abstractive text summarization. *arXiv preprint arXiv:2108.11846*.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, Chaojun and Rico Sennrich. 2020. On exposure bias, hallucination and domain shift in neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3544–3552.

Wu, Lijun, Xu Tan, Di He, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018. Beyond error propagation in neural machine translation: Characteristics of language also matter. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3602–3611.

Yang, Zonghan, Yong Cheng, Yang Liu, and Maosong Sun. 2019. Reducing word omission errors in neural machine translation: A contrastive learning approach. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6191–6196.

Zhang, Wen, Yang Feng, Fandong Meng, Di You, and Qun Liu. 2019. Bridging the gap between training and inference for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4334–4343.

A Ablation tests for En–Ru

Table 8 shows the ablation tests for En–Ru. Both *Rec Only* and *CRec Only* have some gains. The normal TCL that combines these components gets extra improvement.

	En–Ru		
Metrics	BLEU	Meteor	Comet
Vanilla Transformer (TX)	15.87	29.13	63.97
Rec Only	16.33	29.71	65.05
Δ (-TX)	**0.46**	**0.58**	**1.08**
CRec Only	16.65	30.86	65.92
Δ (-TX)	**0.78**	**1.73**	**1.95**
TCL	17.33	31.77	67.02
Δ (-TX)	**1.46**	**2.64**	**3.05**

Table 8: Ablation tests. *Rec Only* (recovery) and *CRec Only* (constraining recovery) models compared to the vanilla Transformer and normal TCL models.