XLM-D: Decorate Cross-lingual Pre-training Model as Non-Autoregressive Neural Machine Translation

Yong Wang[1*] **Shilin He**[2] **Guanhua Chen**[3†] **Yun Chen**[4] **Daxin Jiang**[2†]
[1]Tencent Corporation [2]Microsoft Corporation
[3]Southern University of Science and Technology
[4]Shanghai University of Finance and Economics
seaywang@gmail.com shilin.he@microsoft.com
ghchen08@gmail.com yunchen@sufe.edu.cn djiang@microsoft.com

Abstract

Pre-training language models have achieved thriving success in numerous natural language understanding and autoregressive generation tasks, but non-autoregressive generation in applications such as machine translation has not sufficiently benefited from the pre-training paradigm. In this work, we establish the connection between a pre-trained masked language model (MLM) and non-autoregressive generation on machine translation. From this perspective, we present XLM-D, which seamlessly transforms an *off-the-shelf* cross-lingual pre-training model into a non-autoregressive translation (NAT) model with a lightweight yet effective decorator. Specifically, the decorator ensures the representation consistency of the pre-trained model and brings only one additional trainable parameter. Extensive experiments on typical translation datasets show that our models obtain state-of-the-art performance while realizing the inference speed-up by 19.9×. One striking result is that on WMT14 En⇒De, our XLM-D obtains 29.80 BLEU points with multiple iterations, which outperforms the previous mask-predict model by 2.77 points.

1 Introduction

As a dominant pre-training paradigm in natural language processing (NLP), masked language models such as BERT and its variants (Devlin et al., 2019; Liu et al., 2019), were initially proposed and have achieved state-of-the-art performance on various natural language understanding tasks. For generation tasks, previous studies either leverage the pre-trained BERT as an external component for representation fusing (Zhu et al., 2019) or simply initialize the generation models with a pre-trained model (Ma et al., 2020). Although straightforward, these methods suffer from either heavy computation costs during inference or only supporting autoregressive generation with sequential dependency. Additionally, these approaches result in inconsistency from two perspectives: 1) mismatch of architectures between pre-trained models and the generation model; 2) disagreement of training objectives between pre-training and autoregressive generation tasks.

Different from previous studies, we reexamine the problem from another point of view. We find that the training objective in pre-trained MLM can align well with the one in non-autoregressive generation, which is an emergent generation paradigm due to its excellent inference speed-up (Gu et al., 2018; Lee et al., 2018). Intuitively, both training objectives are formalized as a series of independent token predictions in the output sequence. Based on this observation, we incorporate a pre-trained MLM into the non-autoregressive generation. Specifically, in this work, we focus on the classic non-autoregressive machine translation task. Albeit studied for ages, NAT models have not performed very well, lagging behind the autoregressive translation counterpart.

We propose to adapt the cross-lingual pre-training model (XLMR) into NAT models with a lightweight, effective and user-configurable decorator. Following Occam's razor principle and to better leverage the capability of pre-training models, we design the decorator component based on two key criteria: 1) involve additional trainable parameters as few as possible, e.g., parameter-free; 2) keep the model intermediate representation consistent after using the decorator. Guided by the two criteria, the decorator consists of a *distance-based latent transformation* module and a *position-wise add and scale* module, which contains only one trainable scalar parameter. Moreover, the decorator can be flexibly incorporated into a user-specified layer of XLMR to balance the translation performance and inference speed. We use the connectionist temporal classification (CTC) (Graves et al.,

[*]Work done while at Microsoft Corporation.
[†]Corresponding authors.

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 6934 - 6946
December 7-11, 2022

2006) loss as the training objective.

To validate the effectiveness of our approach, we systematically conduct the evaluation on several widely-used translation datasets. Extensive experiments demonstrate that our proposed model significantly and consistently improves the translation performance and achieves new state-of-the-art results in both single-step and iterative NAT models. Results show that our single-step XLM-D model achieves $27.46/34.70$ BLEU points on WMT14 En⇒De and WMT16 En⇒Ro translation tasks with $19.9\times$ speed-up. Encouragingly, our iterative XLM-D model obtains $29.80/35.65$ points on both tasks and outperforms previous well-performed models (CMLM) (Ghazvininejad et al., 2019) by a large margin, i.e., $2.77/2.57$ points. Further analyses reveal that our approach enhances the capability in long sentence translation and can be user-configured to balance the trade-off between the translation quality and inference speed.

2 Approach

2.1 Problem Formulation

A common part of the masked language model (MLM) and non-autoregressive generation is that the prediction of each output token is made independently. In this section, we compare the training objective in pre-training MLM with the one in non-autoregressive generation and identify their inherent yet unnoticed connections.

Masked Language Model Given an input sentence $\mathbf{x} = \{x_1, \ldots, x_I\}$, a masked language model randomly masks out this sequence by replacing words with a special token [MASK]. Conditioned on the manipulated sequence, the model predicts the masked tokens in parallel. Formally, we denote $\mathbf{x}_{mask}$ as a set of output tokens and $\mathbf{x}_{obs} = \mathbf{x} \setminus \mathbf{x}_{mask}$ as the input sentence. The minimized training objective can be formulated as:

$$\mathcal{L}_{\text{MLM}}(\theta') = -\sum_{n=1}^{N} \log P(\mathbf{x}^n_{mask}|\mathbf{x}^n_{obs}; \theta') \quad (1)$$
$$= -\sum_{n=1}^{N}\sum_{i=1}^{I} \mathbb{1}(x^n_i = [\text{MASK}]) \log P(x^n_i|\mathbf{x}^n_{obs}; \theta'),$$

where N is the number of training examples and θ' is a set of trainable parameters in pre-training models. The training objective has been successfully applied to the state-of-the-art pre-training BERT model and its variants (Devlin et al., 2019; Liu et al., 2019). In particular, Conneau et al. (2020) extends this to monolingual data with one hundred languages for cross-lingual understanding, which attributes pre-training MLM models with cross-lingual properties of word, syntax and semantics.

Non-Autoregressive Machine Translation Recently, non-autoregressive models, which predict the target sentence in parallel conditioned on the input sentence, have obtained considerable attention in machine translation. Specifically, given an input sentence $\mathbf{x} = \{x_1, \ldots, x_I\}$ and an output sentence $\mathbf{y} = \{y_1, \ldots, y_J\}$, a standard non-autoregressive framework (Gu et al., 2018) breaks the probabilistic factorization with a product of independent probability for each output token:

$$\mathcal{L}_{\text{NAT}}(\theta) = -\sum_{n=1}^{N} \log P(\mathbf{y}^n|\mathbf{x}^n; \theta)$$
$$= -\sum_{n=1}^{N}\sum_{j=1}^{J} \log P(y^n_j|\mathbf{x}^n; \theta), \quad (2)$$

where θ is a set of trainable parameters. Typically, NAT models have an auxiliary length predictor, which is used to determine the translation length J.

From equation (1) and (2), we observe that although the output sequences are different in MLM and NAT models, the output tokens are independently predicted. In essence, we find that a masked language model could be endowed with non-autoregressive generation inherently. To utilize available pre-training models effectively, we propose to decorate the existing cross-lingual pre-training model (XLMR) as non-autoregressive machine translation. Specifically, we present a lightweight and effective approach by reducing the incorporation of additional learnable parameters and maintaining the consistency of representations in pre-training models.

2.2 XLM-D

In this section, we introduce the proposed approach of decorating XLMR as non-autoregressive machine translation, termed as XLM-D, which is shown in Figure 1. First of all, we use the latent alignment loss as the training objective. Meanwhile, we augment the standard XLMR model with a *distance-based latent transformation* module, which transforms the hidden states of a source sentence into the representations with extended length. Besides, we incorporate a *position-wise*

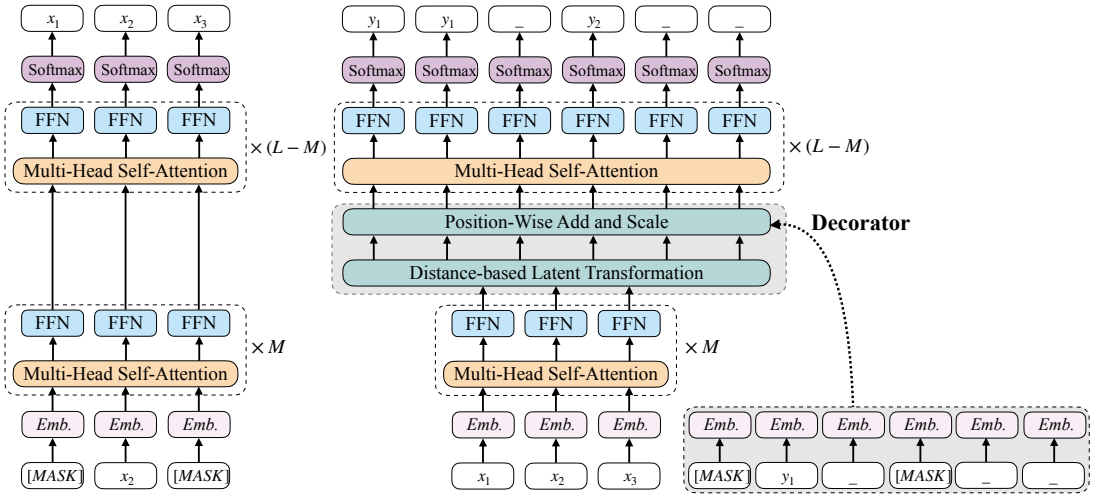


Figure 1: An illustration of the proposed approach, which decorates the pre-training XLMR model as a NAT model. The left figure shows a decomposable view of XLMR, and the right figure presents our proposed architecture. "Decorator" (in the gray background) denotes the incorporated modules while all other components derive directly from pre-trained XLMR. "M" indicates the user-configurable layer where we insert our decorator. For simplicity, we omit some model details, including residual connections and layer normalization.

add and scale component into our model to equip the presented approach with iterative refinement.

Latent Alignment as Training Objective In this work, we train the model with the connectionist temporal classification (CTC) (Graves et al., 2006; Chan et al., 2020) loss, a.k.a. latent alignment. The benefits are twofold. Firstly, utilizing latent alignment is effective in dealing with the token repetition problem (Libovický and Helcl, 2018; Ghazvininejad et al., 2020; Saharia et al., 2020) in the output sentence. In addition, CTC loss does not need the prediction of target length. Specifically, the predicted output has the length $s \times I$, where s is the predefined factor and I is the source length. Formally, let $\boldsymbol{a}$ denote an aligned sequence, we define the collapsing function $\Gamma^{-1}(\boldsymbol{a})$ as collapsing consecutive repeated tokens and removing blank tokens. Given the conditional independence assumption, CTC models the alignment distribution and marginalizes the following log-likelihood:

$$\log P(\mathbf{y}|\mathbf{x};\theta) = \log \sum_{\boldsymbol{a}\in\Gamma(\mathbf{y})} P(\boldsymbol{a}|\mathbf{x};\theta). \quad (3)$$

Distance-based Latent Transformation Suppose the source sentence of length I has hidden states $\mathbf{h}^M_{1:I} = \{\mathbf{h}^M_1, \ldots, \mathbf{h}^M_I\}$ and the predicted output is of length $s \times I$. To bridge the length gap, we transform the hidden states into $s \times I$ vectors. Inspired from the attention mechanism (Bahdanau et al., 2015), we employ a monotonic distance-based attention (Shu et al., 2020) to produce the transformed latent variables. We denote the resulting vectors of latent transformation as $\mathbf{z}_{1:s\times I} = \{\mathbf{z}_1, ..., \mathbf{z}_{s\times I}\}$. Each output vector is calculated with a weighted sum of the hidden states $\mathbf{h}^M_{1:I}$:

$$\mathbf{z}_j = \sum_{i=1}^{I} w_i^j \mathbf{h}_i^M, \quad w_i^j = \frac{\exp(\alpha_i^j)}{\sum_{i'=1}^{I}\exp(\alpha_{i'}^j)},$$
$$\alpha_{i'}^j = -\frac{1}{2\sigma^2}(i' - \frac{I}{s\times I}j)^2,$$

where the weight is calculated with a softmax function over a set of relative distances to the source sentence. In this distance-based attention mechanism, the variance σ is the only trainable parameter.

Position-Wise Add and Scale To support iterative refinement, we introduce the conditional masked strategy (Ghazvininejad et al., 2019) into our model, which selects a subset of tokens to mask and then predicts them in parallel, by using a *position-wise add and scale* component. Let $\epsilon \in \mathbb{R}^{|V|\times d}$ denote the word embeddings borrowed from pre-trained XLMR, which does not introduce additional trainable parameters. The input alignment $\boldsymbol{a} = \{a_1, \ldots, a_{s\times I}\}$ is represented as $\mathbf{e}_{1:s\times I} = \{\mathbf{e}_1, \ldots, \mathbf{e}_{s\times I}\} = \{\epsilon[a_1], \ldots, \epsilon[a_{s\times I}]\}$. Inspired by previous works (Dou et al., 2018; Bapna and Firat, 2019), we propose to combine the transformed latent variables with alignment representations. Specifically, we employ the position-wise add and scale component to generate the latent

representations:

$$\mathbf{z}'_i = (\mathbf{z}_i + \mathbf{e}_i) \times \gamma + \mathbf{p}_i, \quad (4)$$

where $\mathbf{p}_i$ is a positional embedding and we normalize the output by a factor γ to maintain the consistency of representations with the pre-trained XLMR model.

2.3 Training and Inference

Algorithm 1 Training for XLM-D

Input: (1) Parallel translation data $\{\mathcal{D}_n\}_{n=1}^{N}$, (2) User-specified layer M.
procedure TRAINING($\{\mathcal{D}_n\}_{n=1}^{N}$, M)
  while not convergence **do**
    $\boldsymbol{a}' \leftarrow \{[MASK], ..., [MASK]\}$ ▷ $\boldsymbol{a}' = \{a'_1, ..., a'_{s\times I}\}$
    $\boldsymbol{a} \leftarrow \arg\max_{\boldsymbol{a}\in\Gamma(\mathbf{y})} P(\boldsymbol{a}|\mathbf{x}, \boldsymbol{a}'; \theta)$ ▷ using Viterbi algorithm
    $\bar{\boldsymbol{a}} \leftarrow mask(\boldsymbol{a})$ ▷ masked alignment $\bar{\boldsymbol{a}}$ is obtained with a mask policy
    $\theta \leftarrow \arg\max P(\mathbf{y}|\mathbf{x}, \bar{\boldsymbol{a}}; \theta)$ ▷ update θ using $\mathbf{x}$ and masked alignment $\bar{\boldsymbol{a}}$
  end while
end procedure

In this section, we describe the overall training algorithm. As shown in Algorithm 1, we provide the model with masked alignments during training in XLM-D by using Viterbi algorithm to produce the best alignment, namely $\boldsymbol{a} = \arg\max_{\boldsymbol{a}\in\Gamma(\mathbf{y})} P(\boldsymbol{a}|\mathbf{x}, \boldsymbol{a}'; \theta)$. For the masked alignment, we use a block-wise mask policy. Specifically, we divide the sequence into blocks with equal length B and mask the tokens within each block randomly. For the loss computation, we force the output in non-masked positions of alignment $\boldsymbol{a}$ to be the tokens of input alignment $\bar{\boldsymbol{a}}$, termed as constrained CTC loss. For efficiency consideration, we implement the Viterbi algorithm and constrained CTC loss computation using CUDA programming with C++ language extension.

During inference, the translation is produced with a constant number of generative steps. Specifically, we initialize the equal-size blocks with all masked-out tokens. Then, we obtain a new alignment output of the model by selecting the tokens with the largest predicted probability, and merging the alignment output with the previous one. We iteratively update the translation for each block by masking the tokens with low predicted probability.

2.4 Configurability

Considering that the introduced decorator is tied to neither a specific layer nor a specific type of masked language model, our approach is flexible and user-configurable. Moreover, M controls the trade-off between the translation performance and inference speed. Intuitively, a larger M will lead to fewer layers accumulatively that a translation iteration needs during inference. In the analyses, we will explore the influence of M on translation performance and decoding speed. In the above illustration, we mainly elaborate our approach in an iterative scenario. In fact, our model can be simplified to a single-step NAT model by removing the *position-wise add and scale* component. During training in our single-step XLM-D model, we eliminate the block-wise mask policy and only use vanilla CTC loss to train the model.

3 Experiments

3.1 Setup

Datasets We evaluate the effectiveness of our approach on two widely adopted benchmark datasets: WMT14 English-German (En-De)[1] and WMT16 English-Romanian (En-Ro)[2], which consist of 4.0M and 610K sentence pairs respectively. We also show the generality of our approach on four other datasets, including WMT14 English-French (En-Fr) and WMT20 Japanese-English (Ja-En) and IWSLT17 Korean/Arabic-English (Ko/Ar-En). For the WMT14 En-De task, newstest2013 and newstest2014 are used as the validation and test set respectively. For the WMT16 En-Ro task, we use newsdev2016 and newstest2016 as the validation and test set. We follow the dataset configurations of previous works (Gu et al., 2018; Lee et al., 2018) strictly. For our model, we segment each word into tokens with a sentencepiece model (Kudo and Richardson, 2018), learned on the full CC-100 data, that includes 250K subword tokens in XLMR (Conneau et al., 2020). To keep a consistent comparison with previous researches, we apply the sentencepiece model to the preprocessed data directly. For evaluation, we report tokenized case-sensitive BLEU scores (Papineni et al., 2002) for En-De, En-Ro and En-Fr, while using SacreBLEU (Post, 2018) for Ja-En, Ko-En and Ar-En.

Model Configuration For model hyperparameters, we mainly follow the XLMR-base configurations in (Conneau et al., 2020). Specifically, for all translation tasks, we use the

[1]https://www.statmt.org/wmt14/translation-task
[2]https://www.statmt.org/wmt16/translation-task

Model	Iter.	Speed	WMT14		WMT16	
			En⇒De	De⇒En	En⇒Ro	Ro⇒En
Autoregressive Models						
Transformer (Vaswani et al., 2017)	N	1.0×	27.74	31.09	34.28	33.99
Iterative Non-Autoregressive Models						
LaNMT (Shu et al., 2020)	4	5.7×	26.30	–	–	29.10
Iter-NAT (Lee et al., 2018)	10	1.5×	21.61	25.48	29.32	30.19
LevT (Gu et al., 2019)	Adv.	4.0×	27.27	–	–	33.26
DisCO (Kasai et al., 2020)	Adv.	3.5×	27.34	31.31	33.22	33.25
CMLM (Ghazvininejad et al., 2019)	10	1.7×	27.03	30.53	33.08	33.31
CMLM-LFR (Ding et al., 2021b)	10	1.5×	27.80	–	–	33.90
Our Work						
XLM-D (M=0)	2	10.5×	28.69	32.08	35.15	35.10
	4	5.4×	29.41	32.73	35.47	35.52
	8	2.8×	**29.80**	32.88	35.34	35.50
XLM-D (M=6)	2	12.5×	28.91	32.59	35.38	35.37
	4	7.6×	29.37	32.99	35.65	35.63
	8	4.1×	29.59	**33.28**	**35.65**	**35.84**

Table 1: Evaluation of translation performance on the test sets of WMT14 En-De and WMT16 En-Ro with iterative decoding models. The speed-up is measured on the WMT14 En-De test set. "Iter." indicates the number of iterations at inference time. "–" means not reported and "Adv." means adaptive.

hyper-parameters ($d_{\text{model}} = 768$, $d_{\text{hidden}} = 3072$, $n_{\text{layer}} = 12$, $n_{\text{head}} = 12$, $p_{\text{dropout}} = 0.1$). We employ $t_{\text{warmup}} = 10000$ as the warm-up learning rate schedule. In our implementation, the upsampling ratio (s) of the *distance-based latent transformation* module and the scale factor (γ) of the *position-wise add and scale* module are set to 2 and 0.5 respectively. We use weight decay 0.01 and learning rate 0.0002. For more implementation details, please refer to Appendix A.1.

Knowledge Distillation As a key component in NAT models, knowledge distillation (KD) (Zhou et al., 2020) has been proven to effectively reduce the complexity of target data, which is beneficial to the training in NAT models. We strictly follow previous work (Gu et al., 2018) and utilize sequence-level knowledge distillation (Kim and Rush, 2016) to produce the training data. Specifically, for each sentence pair in the training corpus, we replace the target sentence with the translation generated by a pre-trained autoregressive model.

3.2 Results

Table 1 and 2 show the translation quality and inference speed of our approach and baselines on WMT14 En-De and WMT16 En-Ro datasets. For a fair comparison, we divide the results into two types (including iterative and single-step NAT models) based on the number of iterations.

Iterative NAT Models Table 1 shows that CMLM (Ghazvininejad et al., 2019) achieves comparable translation performance with the autoregressive *Transformer-base* models while the inference speed-up is still unsatisfactory. As seen, our iterative models outperform the baseline model (CMLM) by 2.77 and 2.75 BLEU points on WMT14 En⇒De and De⇒En respectively, while maintaining a faster inference speed. Besides, our models achieve significant improvements by up to 2.57/2.53 points on both WMT16 En-Ro tasks. In particular, our approach outperforms the autoregressive *Transformer-base* models by a very large margin. Specifically, on WMT14 En⇒De, our best model achieves 29.80 BLEU points with 2.8× inference speed-up. These results clearly indicate the tremendous potential of our approach.

Single-step NAT Models As presented in Table 2, our approach achieves significant and consistent improvements over all previous baselines across all translation tasks. Specifically, on WMT14 En⇒De and De⇒En, our single-step models outperform the strong baseline model (GLAT) by 2.25 and 0.84 BLEU points respectively. In addition, on the WMT16 En⇒Ro translation task, our model achieves a significant improvement by up to 3.51 BLEU points. Encouragingly, our approach outperforms the *Transformer-base* models by 0.42/0.33 BLEU points on the

Model	Iter.	Speed	WMT14		WMT16	
			En⇒De	De⇒En	En⇒Ro	Ro⇒En
Autoregressive Models						
Transformer (Vaswani et al., 2017)	N	1.0×	27.74	31.09	34.28	33.99
Single-step Non-Autoregressive Models						
Vanilla-NAT (Gu et al., 2018)	1	15.6×	17.69	21.47	27.29	29.06
FCL-NAT (Guo et al., 2020a)	1	16.0×	25.75	29.50	–	–
ReorderNAT (Ran et al., 2021)	1	16.1×	22.79	27.28	29.30	29.50
Flowseq (Ma et al., 2019)	1	1.1×	23.72	28.39	29.73	30.72
AXE (Ghazvininejad et al., 2020)	1	15.3×	23.53	27.90	30.75	31.54
Bag-of-ngrams (Shao et al., 2020)	1	10.0×	20.90	24.60	28.30	29.30
GLAT (Qian et al., 2021)	1	15.3×	25.21	29.84	31.19	32.04
CNAT (Bao et al., 2021)	1	10.4×	25.56	29.36	–	–
latent-GLAT (Bao et al., 2022)	1	11.3×	26.64	29.93	–	–
Our Work						
XLM-D (M=0)	1	19.5×	26.91	30.34	33.90	34.11
XLM-D (M=6)	1	19.6×	27.01	30.62	34.23	**34.32**
XLM-D with LT only	1	19.9×	**27.46**	**30.68**	**34.70**	34.29

Table 2: Evaluation of translation performance on the test sets of WMT14 En-De and WMT16 En-Ro with single-step decoding models. "XLM-D with LT only" means that we only use the *distance-based latent transformation* component by removing the *position-wise add and scale* module.

WMT16 En-Ro dataset. We attribute this to the incorporation of additional language knowledge benefited from the pre-training model. These results clearly demonstrate the effectiveness of decorating cross-lingual pre-training models as non-autoregressive machine translation.

Inference Speed To evaluate the decoding speed of our approach, we run all models with one sentence at a time on a single GPU and calculate the inference speed-up on the WMT14 En-De task (for more computation details, please refer to Appendix A.2). As demonstrated in Table 2, our single-step NAT models achieve 19.9× inference speed over the *Transformer-base* counterpart and exceed all previous single-step models. We credit this to removing the overhead caused by cross-attention modules in the conventional encoder-decoder architecture. Moreover, our iterative models also achieve a good acceleration as the number of iterations increases, especially for $M = 6$. The results reveal that our model is lightweight and efficient.

Translation Quality on Different Datasets Table 3 shows results on four other datasets: WMT14 En⇒Fr, WMT20 Ja⇒En, IWSLT17 Ko⇒En and IWSLT17 Ar⇒En, covering large-scale and small-scale training data (i.e. 35.8M, 16.8M, 0.23M and 0.23M). We follow the widely used translation directions (Vaswani et al., 2017; Liu et al., 2020; Gu and Kong, 2021). As seen, the superiority of our approach holds across different language pairs and data sizes, demonstrating the universality of the proposed approach. In addition, the results in Table 3, show that the gains of our approach on small-scale data (Ko⇒En and Ar⇒En) are more significant than on large-scale data (En⇒Fr and Ja⇒En). This observation suggests that low-resource language pairs benefit more from pre-training models, which is also verified by Liu et al. (2020).

Model	Iter.	En⇒Fr (35.8M)	Ja⇒En (16.8M)	Ko⇒En (0.23M)	Ar⇒En (0.23M)
Transformer	N	41.23	19.38	15.15	32.54
CMLM	4	39.26	16.47	12.96	28.45
	10	39.69	18.53	13.91	29.46
XLM-D (M=0)	2	40.06	17.46	15.46	34.35
	4	40.97	19.01	**16.18**	34.79
	8	**41.11**	**19.29**	16.17	**34.97**

Table 3: Evaluation of translation quality on different datasets varied in language pair and size.

3.3 Analyses

In this section, we perform extensive analyses on the WMT14 En-De task to better demonstrate the effectiveness of our model in terms of: 1) translation quality on original data, 2) effects of upsampling ratio, 3) effects of different M, 4) effects of pre-training, and 5) effects of the sentence length. For more analyses, please refer to Appendix A.3.

Model	Iter.	Speed	Original	Distill
Transformer	N	1.0×	27.74	27.86
CMLM	10	1.7×	24.61	27.03
XLM-D (M=0)	1	19.5×	20.28	26.91
	2	10.5×	25.26	28.69
	4	5.4×	26.95	29.41
	8	2.8×	**27.49**	**29.80**

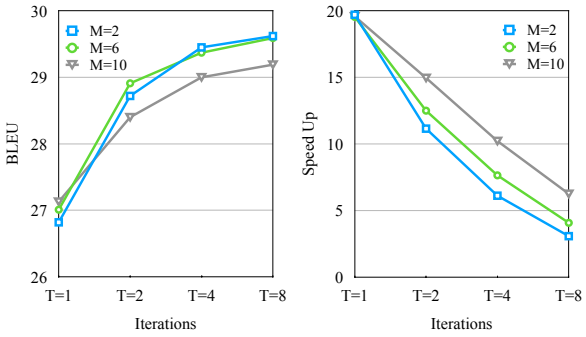
Table 4: Translation performance on WMT14 En⇒De with original training data and distilled data.

Translation Quality on Original Data Previous works (Gu et al., 2018; Lee et al., 2018) reported that it was necessary to train NAT models on distillation data, which is generated with a well-trained autoregressive model. To evaluate our model's dependence on this process, we also present the results of translation performance with *Transformer-base*, CMLM and our approach on WMT14 En⇒De original training data, which are shown in Table 4. Overall, our approach (XLM-D) performs better than the CMLM baseline model significantly. For instance, our model obtains 27.49 BLEU scores, which outperforms the CMLM model by 2.88 points while remaining a faster inference speed with 2.8×. In particular, our model can achieve comparable results (27.49 vs. 27.74) with autoregressive *Transformer-base*, which is a very promising result to compensate for the performance gap on the original data. These results confirm the robustness and effectiveness of our approach.

Iter.	$s = 1.5$	$s = 2.0$	$s = 2.5$	$s = 3.0$
1	25.53	26.91	27.10	27.32
2	28.08	28.69	28.68	28.65
4	28.97	29.41	29.45	29.61
8	29.20	29.80	29.70	29.76

Table 5: Comparison of translation performance on WMT14 En⇒De with different upsampling ratios.

Effects of Upsampling Ratio In Table 5, we study the influence of different upsampling ratios (s) in our model. We find that a higher upsampling value achieves a significant improvement in the translation performance with a single decoding iteration. For instance, the translation performance is improved by 1.38 points when increasing s from 1.5 to 2.0. Besides, the upsampling ratio does not influence the performance notably with multiple iterations. This reveals that multiple iterations can remedy the translation performance effectively.



(a) BLEU Score (b) Speed Up

Figure 2: Translation performance and inference speed on the WMT14 En⇒De test set with respect to different M. The left figure denotes the BLEU scores vs. the number of iterations while the right figure indicates the speed up vs. the number of iterations.

Effects of Different M Previous researches demonstrate that to obtain reliable performance, NAT models usually sacrifice the inference speed by multiple refinement iterations. In our model, we introduce an extra user-configurable hyper-parameter M to balance the translation quality and inference speed. To verify its efficacy, we study the influence of different M on translation performance and decoding latency. As shown in Figure 2, we can find that a higher M value brings more speed-up by multiple iterations without damaging much translation performance. Although the literature shows that the translation quality and inference speed is irreconcilable, our results suggest that the proposed approach can achieve their balance by modifying the user-configurable M.

Model	Iter.	Pre-train	Reset
XLM-D with LT only	1	27.46	25.80
XLM-D (M=0)	1	26.91	26.04
	2	28.69	27.98
	4	29.41	28.56
	8	29.80	28.83
XLM-D (M=6)	1	27.01	25.75
	2	28.91	27.64
	4	29.37	28.24
	8	29.59	28.57

Table 6: Effects of utilizing pre-training parameters vs. resetting parameters on WMT14 En⇒De.

Effects of Pre-training To study the influence of additional knowledge brought by pre-training, we conduct an ablation analysis on the WMT14 En⇒De translation task. Specifically, we follow

our approach but reset the parameters by the strategy of random initialization in XLMR. As shown in Table 6, our model with pre-trained parameters achieves significant and consistent improvements over the counterparts with parameters reset across all model variants. For instance, in single-step NAT models, our system outperforms the "Reset" model by 1.66 BLEU points (27.46 vs. 25.80). Besides, our iterative NAT models achieve improvements by over 1.0 BLEU points as the increase of multiple iterations. These results confirm that the pre-training provides additional language knowledge, which helps non-autoregressive machine translation.

Effects of Sentence Length To explore the benefits of our iterative models, we investigate the translation results of our approach on WMT14 En⇒De with respect to different lengths of target sentences. Specifically, based on the respective lengths of target sentences, translations are allocated into different buckets. The evaluation results of BLEU scores for each bucket are shown in Table 7. As expected, our approach achieves improvements with the increase of iterations across all buckets. In addition, the improvements of translation performance are limited when the lengths of target sentences are constrained (≤ 10). In particular, for longer sentences (≥ 40), our model improves the performance by 2.70 BLEU points with $T = 2$. This evidence suggests that our model clearly benefits long-distance dependencies in NAT models.

Bucket	$T=1$	$T=2$	$T=4$
$1 \leq N < 10$	21.90	22.64	22.51
$10 \leq N < 20$	26.07	27.47	28.42
$20 \leq N < 30$	27.09	28.92	29.61
$30 \leq N < 40$	27.11	29.57	30.02
$40 \leq N$	26.95	29.65	30.23

Table 7: Translation performance on WMT14 En⇒De with respect to different lengths of target sentences (N). "T" denotes the number of iterations.

4 Related Work

Pre-training Models Unsupervised representation learning (Peters et al., 2018; Devlin et al., 2019; Liu et al., 2019) has achieved remarkable success in natural language understanding. Peters et al. (2018) first introduced a general approach for learning high-quality context-dependent representations, which encode syntactic and semantic information efficiently. By using the objective of masked language model, Devlin et al. (2019) introduced a new bidirectional representation model and obtained state-of-the-art results on various tasks. Most recently, Conneau et al. (2020) extended pre-training models to one hundred languages by using more than two terabytes of data. Different from these studies, we observe that cross-lingual pre-trained masked language models are endowed with the attribute of non-autoregressive generation inherently. Hence, we propose to decorate a cross-lingual pre-training model as non-autoregressive translation.

Non-Autoregressive Neural Machine Translation Many previous works investigate reducing the decoding latency by empowering the parallel sequence generation capability (Gu et al., 2018; Lee et al., 2018). Gu et al. (2018) pioneered to propose a non-autoregressive translation model by modeling fertility as latent variables. Although accelerating the inference process significantly, NAT models suffer from the serious multi-modality problem, which results in considerably worse performance than their autoregressive counterparts. To alleviate this issue, extensive methods have been investigated in vanilla non-autoregressive training (Kaiser et al., 2018; Sun et al., 2019; Bao et al., 2021; Du et al., 2021). Another line of researches (Lee et al., 2018; Ghazvininejad et al., 2019; Gu et al., 2019; Ding et al., 2021a, 2022; Huang et al., 2022) employ an iterative refinement process to maintain the translation quality. Ghazvininejad et al. (2019) used a masked language modeling to predict any subset of target words by conditioning on a partially masked target translation.

5 Conclusion

We propose to decorate cross-lingual pre-training models (XLMR) as non-autoregressive machine translation in a lightweight, effective and flexible way. Firstly, we bridge pre-training masked language models with non-autoregressive generation, revealing that a pre-trained masked language model can be directly used to produce sequences in parallel. Moreover, we propose to incorporate *distance-based latent transformation* and *position-wise add and scale* components into a user-specified layer of XLMR. Empirical results on a variety of language pairs demonstrate the effectiveness and universality of our approach. Further analyses confirm that the proposed method benefits translations for long sentences and achieves a better balance between translation quality and inference speedup.

Limitations

This work has several limitations which are expected to explore sufficiently in future work. Firstly, we only conduct experiments on non-autoregressive machine translation. Although our method significantly improves translation performance and obtains state-of-the-art results on translation tasks, we believe that the proposed approach is generally applicable to other generation tasks as well, including text summarization (See et al., 2017) and dialogue generation (Vinyals and Le, 2015). Future explorations on these tasks are needed to extend the application scope of XLM-D.

At the same time, we hypothesize that a cross-lingual pre-training model can be seamlessly transformed as a non-autoregressive translation model and take bilingual machine translation as a testbed in this work. Actually, a multilingual pre-training model, such as XLMR (Conneau et al., 2020), can be employed in the multilingual scenario (Aharoni et al., 2019; Freitag and Firat, 2020) and has great potential for building multilingual/zero-shot non-autoregressive machine translation (Johnson et al., 2017; Zhang et al., 2020, 2022). XLM-D is bilingual and only evaluated on the language pair involved in training. It is interesting to further extend XLM-D to the multilingual scenario that can benefit from effective cross-lingual transfer and improved serving efficiency. We leave more explorations on this direction as future work.

Acknowledgement

We thank the anonymous reviewers for their insightful feedback on this work. Yun Chen is supported by National Natural Science Foundation of China (No. 62106138).

References

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. In *NAACL*.

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

Yu Bao, Shujian Huang, Tong Xiao, Dongqi Wang, Xinyu Dai, and Jiajun Chen. 2021. Non-autoregressive translation by learning target categorical codes. In *NAACL*.

Yu Bao, Hao Zhou, Shujian Huang, Dongqi Wang, Lihua Qian, Xinyu Dai, Jiajun Chen, and Lei Li. 2022. latent-glat: Glancing at latent variables for parallel text generation. In *ACL*.

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. In *EMNLP*.

William Chan, Chitwan Saharia, Geoffrey Hinton, Mohammad Norouzi, and Navdeep Jaitly. 2020. Imputer: Sequence modelling via imputation and dynamic programming. In *ICML*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Édouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *ACL*.

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

Liang Ding, Longyue Wang, Xuebo Liu, Derek F Wong, Dacheng Tao, and Zhaopeng Tu. 2021a. Progressive multi-granularity training for non-autoregressive translation. In *ACL*.

Liang Ding, Longyue Wang, Xuebo Liu, Derek F Wong, Dacheng Tao, and Zhaopeng Tu. 2021b. Rejuvenating low-frequency words: Making the most of parallel data in non-autoregressive translation. In *ACL*.

Liang Ding, Longyue Wang, Shuming Shi, Dacheng Tao, and Zhaopeng Tu. 2022. Redistributing low-frequency words: Making the most of monolingual data in non-autoregressive translation. In *ACL*.

Zi-Yi Dou, Zhaopeng Tu, Xing Wang, Shuming Shi, and Tong Zhang. 2018. Exploiting deep representations for neural machine translation. In *EMNLP*.

Cunxiao Du, Zhaopeng Tu, and Jing Jiang. 2021. Order-agnostic cross entropy for non-autoregressive machine translation. In *ICML*.

Markus Freitag and Orhan Firat. 2020. Complete multilingual neural machine translation. In *WMT*.

Marjan Ghazvininejad, Vladimir Karpukhin, Luke Zettlemoyer, and Omer Levy. 2020. Aligned cross entropy for non-autoregressive machine translation. In *ICML*.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *EMNLP*.

Alex Graves, Santiago Fernández, Faustino J Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ICML*.

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

Jiatao Gu and Xiang Kong. 2021. Fully non-autoregressive neural machine translation: Tricks of the trade. In *ACL-Findings*.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *NeurIPS*.

Junliang Guo, Xu Tan, Linli Xu, Tao Qin, Enhong Chen, and Tie-Yan Liu. 2020a. Fine-tuning by curriculum learning for non-autoregressive neural machine translation. In *AAAI*.

Junliang Guo, Zhirui Zhang, Linli Xu, Hao-Ran Wei, Boxing Chen, and Enhong Chen. 2020b. Incorporating bert into parallel sequence decoding with adapters. In *NeurIPS*.

Xiao Shi Huang, Felipe Pérez, and Maksims Volkovs. 2022. Improving non-autoregressive translation models without distillation. In *ICLR*.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. In *TACL*.

Lukasz Kaiser, Samy Bengio, Aurko Roy, Ashish Vaswani, Niki Parmar, Jakob Uszkoreit, and Noam Shazeer. 2018. Fast decoding in sequence models using discrete latent variables. In *ICML*.

Jungo Kasai, James Cross, Marjan Ghazvininejad, and Jiatao Gu. 2020. Non-autoregressive machine translation with disentangled context transformer. In *ICML*.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. In *EMNLP*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *EMNLP (System Demonstrations)*.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *EMNLP*.

Jindřich Libovickỳ and Jindřich Helcl. 2018. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *EMNLP*.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. In *TACL*.

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. In *arXiv preprint arXiv:1907.11692*.

Shuming Ma, Jian Yang, Haoyang Huang, Zewen Chi, Li Dong, Dongdong Zhang, Hany Hassan Awadalla, Alexandre Muzio, Akiko Eriguchi, Saksham Singhal, et al. 2020. Xlm-t: Scaling up multilingual machine translation with pretrained cross-lingual transformer encoders. In *arXiv preprint arXiv:2012.15547*.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. 2019. Flowseq: Non-autoregressive conditional sequence generation with generative flow. In *EMNLP*.

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: Demonstrations*.

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

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Matt Post. 2018. A call for clarity in reporting bleu scores. In *WMT*.

Lihua Qian, Hao Zhou, Yu Bao, Mingxuan Wang, Lin Qiu, Weinan Zhang, Yong Yu, and Lei Li. 2021. Glancing transformer for non-autoregressive neural machine translation. In *ACL*.

Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. 2021. Guiding non-autoregressive neural machine translation decoding with reordering information. In *AAAI*.

Chitwan Saharia, William Chan, Saurabh Saxena, and Mohammad Norouzi. 2020. Non-autoregressive machine translation with latent alignments. In *EMNLP*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*.

Chenze Shao, Jinchao Zhang, Yang Feng, Fandong Meng, and Jie Zhou. 2020. Minimizing the bag-of-ngrams difference for non-autoregressive neural machine translation. In *AAAI*.

Raphael Shu, Jason Lee, Hideki Nakayama, and Kyunghyun Cho. 2020. Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior. In *AAAI*.

Zhiqing Sun, Zhuohan Li, Haoqing Wang, Di He, Zi Lin, and Zhihong Deng. 2019. Fast structured decoding for sequence models. In *NeurIPS*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *ICML Deep Learning Workshop*.

Biao Zhang, Ankur Bapna, Melvin Johnson, Ali Dabirmoghaddam, Naveen Arivazhagan, and Orhan Firat. 2022. Multilingual document-level translation enables zero-shot transfer from sentences to documents. In *ACL*.

Biao Zhang, Philip Williams, Ivan Titov, and Rico Sennrich. 2020. Improving massively multilingual neural machine translation and zero-shot translation. In *ACL*.

Chunting Zhou, Jiatao Gu, and Graham Neubig. 2020. Understanding knowledge distillation in non-autoregressive machine translation. In *ICLR*.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tieyan Liu. 2019. Incorporating bert into neural machine translation. In *ICLR*.

A Appendix

A.1 Implementation Details

We implement our approach with open-source toolkit - fairseq[3] (Ott et al., 2019). For WMT tasks, all models were trained for 200K updates on 8 NVIDIA Tesla A100 GPUs with a batch size of 128K tokens using Adam optimizer (Kingma and Ba, 2015). For IWSLT17 Ko-En and IWSLT17 Ar-En tasks, we trained the model for 100K updates on 1 GPU with a batch size of 8K tokens. Following previous studies (Gu et al., 2018; Lee et al., 2018), we evaluate the speed-up by averaging the decoding latency for each sentence on the WMT14 En-De test set with batch size 1 on a single NVIDIA Tesla A100 GPU for *Transformer-base* and our model.

A.2 Computation of Inference Speed

To compute the inference speed, we run our own baselines and perform 5 runs to reduce the potential noise for timing. Specifically, we run our model and *Transformer-base* model on a single GPU with the WMT14 En⇒De test set and evaluate the speed up by comparing the translation latency. The details are shown in Table 8.

Model	Iter.	Latency	Speed
Transformer	N	320.91ms	1.0×
XLM-D (M=0)	1	16.48ms	19.5×
XLM-D (M=6)	1	16.41ms	19.6×
XLM-D with LT only	1	16.15ms	19.9×

Table 8: Translation latency and inference speed on the WMT14 En⇒De task.

A.3 More Analyses

Effects of Different Decoding Strategies The block-wise mask policy is to ensure that for every decoding iteration, the tokens in each sentence block have the chance to be selected and predicted, and thereby improves the translation performance. As shown in Table 9, we systematically compare different decoding strategies on both WMT14 En⇒De and De⇒En translation tasks. As seen, the dynamic strategy performs better than the static one across all number of iterations and tasks. Therefore, we use the strategy of dynamic decoding throughout our experiments.

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

Model	Iter.	En⇒De		De⇒En	
		Static	Dynamic	Static	Dynamic
XLM-D (M=0)	2	28.57	28.69	31.88	32.08
	4	29.18	29.41	32.58	32.73
	8	29.80	29.80	32.88	32.88
XLM-D (M=6)	2	28.51	28.91	32.04	32.59
	4	29.25	29.37	32.85	32.99
	8	29.59	29.59	33.28	33.28

Table 9: Effects of different decoding strategies on WMT14 En-De. "Static" denotes that the block size is fixed as 8, while "Dynamic" indicates that it is equal to the number of iterations in iterative decoding.

Comparison with Pre-training Models Guo et al. (2020b) propose to incorporate BERT into parallel sequence decoding with adapters. Different from their work, our study builds the connection between the pre-training MLM and non-autoregressive generation. Furthermore, we propose a lightweight, effective and user-configurable approach to decorating XLMR as NAT models, achieving both inference speed-up and huge performance gain. We compare the translation quality and inference speed for both approaches, which are shown in Table 10. The results show that compared with AB-Net, our approach achieves more improvements (29.59 vs. 28.69) while remaining a faster inference speed (4.1× vs. 2.4×). This again demonstrates the superiority of our method.

Model	Iter.	Speed	BLEU
Transformer	N	1.0×	27.74
AB-Net-Enc	10	4.7×	28.08
AB-Net	10	2.4×	28.69
XLM-D (M=6)	2	12.5×	28.91
	4	7.6×	29.37
	8	4.1×	**29.59**

Table 10: Evaluation of translation quality with pre-training models on WMT14 En⇒De.

Case Study We carried out a case study to illustrate the performance of our method and baseline approach. Table 11 shows a translation example randomly selected from the test set in the WMT14 De⇒En task. As seen, XLM-D can produce more adequate and fluent translations. For instance, the German words "*befinden sich in einem Dauer-Kampf*" are mistranslated by baseline, while the XLM-D model accurately translates it into "*engaged in an ongoing fight*". Besides, the baseline

Source	Sowohl die US-Behörden als auch die mexikanischen Sicherheitskräfte befinden sich in einem Dauer-Kampf gegen die Drogenkartelle.
Target	Both the US authorities and **the Mexican security forces** are *engaged in an ongoing battle* against the drug cartels.
Baseline	Both the US authorities and **Mexico's forces** are in a long-term fight against the drug cartels.
XLM-D	Both the US authorities and **the Mexican security forces** are *engaged in an ongoing fight* against drug cartels.

Table 11: An example from the WMT14 De⇒En task. Phrases formatted as bold (or underline) indicate the problem of incomplete translations (or under translations) in the baseline but fixed by XLM-D.

tends to generate incomplete words (e.g., "*Mexico's forces*"), while our model corrects this issue. This demonstrates that our model improves the fluency and adequacy of translations in terms of words, phrases and patterns.