ngram-OAXE : Phrase-Based Order-Agnostic Cross Entropy for Non-Autoregressive Machine Translation

Cunxiao Du
Singapore Management University
cnsdunm@gmail.com

Zhaopeng Tu[*]
Tencent AI Lab
zptu@tencent.com

Longyue Wang
Tencent AI Lab
vinnylywang@tencent.com

Jing Jiang
Singapore Management University
jingjiang@smu.edu.sg

Abstract

Recently, a new training OAXE loss (Du et al., 2021) has proven effective to ameliorate the effect of multimodality for non-autoregressive translation (NAT), which removes the penalty of word order errors in the standard cross-entropy loss. Starting from the intuition that reordering generally occurs between phrases, we extend OAXE by only allowing reordering between ngram phrases and still requiring a strict match of word order within the phrases. Extensive experiments on NAT benchmarks across language pairs and data scales demonstrate the effectiveness and universality of our approach. Further analyses show that *ngram*-OAXE indeed improves the translation of ngram phrases, and produces more fluent translation with a better modeling of sentence structure.[1]

1 Introduction

Fully non-autoregressive translation (NAT) has received increasing attention for its efficient decoding by predicting every target token in parallel (Gu et al., 2018; Ghazvininejad et al., 2019). However, such advantage comes at the cost of sacrificing translation quality due to the *multimodality* problem: there exist many possible translations of the same sentence, while vanilla NAT models may consider them at the same time due to the independent predictions, which leads to multi-modal outputs in the form of token repetitions (Gu et al., 2018).

Recent works have incorporated approaches to improving the standard cross-entropy (XE) loss to ameliorate the effect of multimodality. The motivation for these works is that modeling word order is difficult for NAT, since the model cannot condition on its previous predictions like its autoregressive counterpart. Starting from this intuition, a thread of research relaxes the word order restriction based on the **monotonic alignment** assumption (Libovický and Helcl, 2018; Ghazvininejad et al., 2020; Saharia et al., 2020). Du et al. (2021) take a further step by removing the penalty of word order errors with a novel order-agnostic cross entropy (OAXE) loss, which enables NAT models to handle **word reordering** – a common source of multimodality problem. Accordingly, OAXE achieves the best performance among these model variants.

However, OAXE allows reordering between every two words, which is not always valid in practice. For example, the reordering of the two words "this afternoon" is not correct in grammar. The reordering generally occurs between ngram phrases, such as "I ate pizza" and "this afternoon". Starting from this intuition, we extend OAXE by constraining the reordering between ngrams and requiring a strict match of word order within each ngram (i.e., *ngram*-OAXE). To this end, we first build the probability distributions of ngrams in the target sentence using the word probabilities produced by NAT models. Then we find the best ordering of target ngrams to minimize the cross entropy loss. We implement the *ngram*-OAXE loss in an efficient way, which only adds one more line of code on top of the source code of OAXE. Accordingly, *ngram*-OAXE only marginally increases training time (e.g., 3% more time) over OAXE.

Experimental results on widely-used NAT benchmarks show that *ngram*-OAXE improves translation performance over OAXE in all cases. Encouragingly, *ngram*-OAXE outperforms OAXE by up to +3.8 BLEU points on raw data (without knowledge distillation) for WMT14 En-De translation (Table 1), and narrows the performance gap between training on raw data and on distilled data. Further analyses show that *ngram*-OAXE improves over OAXE on the generation accuracy of ngram phrases and modeling reordering between ngram phrases, which makes *ngram*-OAXE handle long sentences better, especially on raw data. The

[*]Zhaopeng Tu is the corresponding author.
[1]The codes and models are in https://github.com/tencent-ailab/machine-translation/COLING22_ngram-OAXE/.

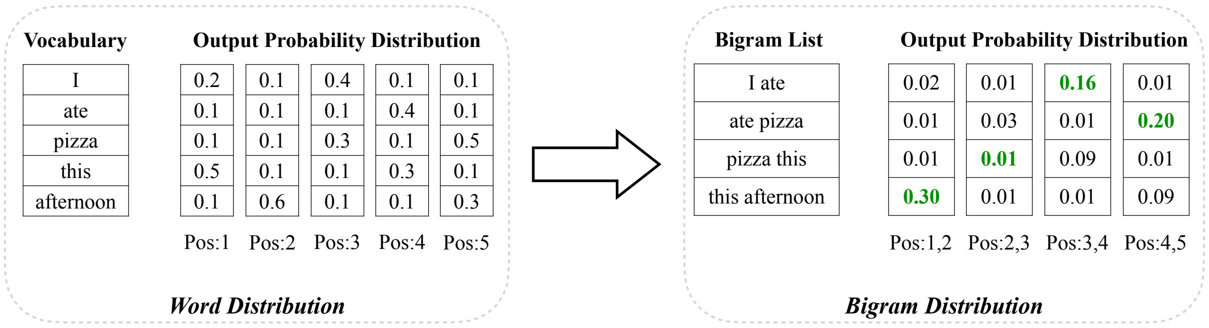


Figure 1: Illustration of the proposed *ngram*-OAXE loss with $N = 2$ (i.e., bigram-OAXE). We only show the probabilities of the target words and bigrams for better illustration. Firstly, *ngram*-OAXE transforms the word probability distributions to the bigram distributions by multiplying the word probabilities at the corresponding positions. For example, P("I ate" | Pos:1,2) = P("I" | Pos:1) * P("ate" | Pos:2) = 0.2*0.1=0.02. Then, we select the ngrams (highlighted in bold) for each neighbouring positions using the efficient Hungarian algorithm.

strength of *ngram*-OAXE on directly learning from the complex raw data indicates the potential to train NAT models without knowledge distillation.

2 Methodology

2.1 Preliminaries: NAT

Cross Entropy (XE) Standard NAT models (Gu et al., 2018) are trained with the cross entropy loss:

$$\mathcal{L}_{XE} = -\log P(Y|X) = -\sum_{y_n} \log P(y_i|X), \quad (1)$$

where (X, Y) with $Y = \{y_1, \ldots, y_I\}$ is a bilingual training example, and $P(y_i|X)$ is calculated independently by the NAT model. XE requires a strict match of word order between target tokens and model predictions, thus will heavily penalize hypotheses that are semantically equivalent to the target but different in word order.

Order-Agnostic Cross Entropy (OAXE) Du et al. (2021) remove the word order restriction of XE, and assign loss based on the best alignment between target tokens and model predictions. They define the ordering space $\mathbf{O} = \{O^1, \ldots, O^J\}$ for Y, where O^j is an ordering of the set of target tokens $(y_1, \ldots, y_I)$. The OAXE objective is defined as finding the best ordering O^j to minimize the cross entropy loss:

$$\mathcal{L}_{\text{OAXE}} = \min_{O^j \in \mathbf{O}} \left(-\log P(O^j|X)\right), \quad (2)$$

where $-\log P(O^i|X)$ is the cross entropy loss for ordering O^i, which is calculated by Equation 1.

2.2 *ngram*-OAXE Loss

Figure 1 illustrates the two-phase calculation of *ngram*-OAXE : 1) constructing the probability distributions of the ngrams in the target sentence; 2) searching the best ordering of the considered ngrams to minimize the cross entropy loss.

Formulation Given the target $Y = \{y_1, \ldots, y_I\}$, we define the target ngrams G^N of size N as all the N continuous tokens in Y: $\{y_{1:N}, \cdots, y_{I-N+1:I}\}$. The output ngram distributions P_G is defined as:

$$P_G(y_{i:i+N-1}|X) = \prod_{t=i}^{i+N-1} P(y_t|X), \quad (3)$$

where $P(y_t|X)$ is the prediction probability of NAT models for the token y_t in position t of the target sentence, and N is the size of ngrams.

The *ngram*-OAXE objective is defined as finding the best ordering O^j to minimize the cross entropy loss of the considered ngrams in target sentence Y:

$$\mathcal{L}_{ngram\text{-OAXE}} = \min_{O^j \in \mathbf{O}} \left(-\log P_G(O^j|X)\right). \quad (4)$$

Ideally, the best ordering O^j should meet the following conditions:

1. The ngrams in O^j should not be overlapped (e.g., "I ate" and "ate pizza" should not occur simultaneously in one O).
2. O^j is a mixture of ngrams with different sizes (e.g., "I ate pizza" and "this afternoon").

However, it is computationally infeasible to search the best ngram segmentation of the target sentence with highest probabilities. Given a target sentence with length I, there are 2^I ngram segmentation (i.e, each token can be labeled as the end of a ngram or not). For each ngram segmentation with expected length I/2, the time complexity is $O((I/2)^3)$ using the efficient Hungarian algorithm. In this way, the total computational complexity of the original two conditions is $O(2^I I^3)$.

For computational tractability, we loosen the conditions by:

1. We consider all ngrams in the target sentence to avoid searching the ngram segmentation. In other words, each word is allowed to occur in multiple ngrams in one ordering O.
2. We only consider ngrams with a fixed size N (e.g., only bigrams), which enables us to cast this problem as Maximum Bipartite Matching and leverage the efficient Hungarian algorithm, as done in (Du et al., 2021).

By loosening the conditions, there are (I-N+1) ngrams of size N in the sentence, and the computational complexity is $O(I^3)$. Accordingly, the loss of the ordering O^j is computed as:

$$P_G(O^j|X) = \prod_{y_{i:i+N-1} \in O^j} P_G(y_{i:i+N-1}|X). \quad (5)$$

Figure 1 shows the calculation of bigram-OAXE loss for the target sentence "I ate pizza this afternoon". We consider all bigrams in the sentence (see "Bigram List"), and obtain the probability distribution of the considered bigrams. We construct the bipartite graph $G = (U, V, E)$ where the first part of vertices U is the set of N-1 neighbouring positions (e.g., the first two positions"Pos:1,2"), and the second part of vertices V is the list of N-1 target bigrams. Each edge in E is the prediction log probability for the bigram in the corresponding position. We can follow Du et al. (2021) to leverage the efficient Hungarian algorithm (Kuhn, 1955) for fast calculation of *ngram*-OAXE (see the assigned probabilities for the consider bigrams).

Implementation Algorithm 1 shows the pseudo-code of *ngram*-OAXE with $N = 2$. The implementation of *ngram*-OAXE is almost the same with that of OAXE, except that we add one more line (in red color) for constructing the probability distribution of ngrams. We implement *ngram*-OAXE on top of the source code of OAXE, and leverage the same recipes (i.e., loss truncation and XE pretrain) to effectively restrict the free-order nature of OAXE.

Algorithm 1 Bigram-OAXE Loss

```
Input: Ground truth $Y$, NAT output log $P$
$bs$, $len$ = $Y$.size()
$Y$ = $Y$.repeat(1, $len$).view($bs$, $len$, $len$)
$costM$ = -log $P$.gather(index=$Y$, dim=2)
$costM$ = $costM$[:, :-1, :-1] +$costM$[:, 1:, 1:]
for $i = 0$ to $bs$ do
    $bestMatch$[i] = HungarianMatch($costM$[i])
end for
Return:$costM$.gather(index=$bestMatch$)
```

Since both *ngram*-OAXE and OAXE only modify the training of NAT models, their inference latency is the same with the CMLM baseline (e.g., 15.3x speed up over the AT model). Concerning the training latency, OAXE takes 36% more training time over the CMLM baseline, and our *ngram*-OAXE takes 40% more training time, which is almost the same to OAXE since we only add one more line of code.

Discussion Some researchers may doubt that the *ngram*-OAXE loss is not an intuitively understandable "global" loss, since some words are counted multiple times. We use the example in Figure 1 to dispel the doubt. Firstly, except for the first and last words (i.e., "I" and "afternoon"), the *ngram*-OAXE loss equally counts the other words twice, which would not introduce the count bias.

Secondly, we follow Du et al. (2021) to start with an initialization pre-trained with the XE loss, which ensures that the NAT models can produce reliable token probabilities to compute ngram probabilities. We also use the **loss truncation** technique (Kang and Hashimoto, 2020) to drop invalid ngrams with low probabilities (e.g., "pizza this" | Pos:2,3) in the selected ordering O^j.

Thirdly, the overlapped ngrams can help to produce more fluent translations by modeling global context in a manner of ngram LM. For example, the high-probability overlapped token in position 4 "ate" (i.e., P(ate | Pos:4) = 0.4) will guide NAT models to assign high probabilities to the neighbouring ngrams ("I ate" | Pos:3,4) and ("ate pizza" | Pos:4,5), which form a consistent clause ("I ate pizza | Pos:3,4,5"). In contrast, *ngram*-OAXE would not simultaneously assign high probabilities to the phrases ("this afternoon" | Pos:1,2) and ("pizza this" | Pos:2,3), since the two phrases require NAT models to assign high probabilities to two different words (i.e., "afternoon" and "pizza") in the overlapped position 2.

The $\mathcal{L}_{ngram\text{-OAXE}}$ loss with $N = 2$ in Figure 1 is calculated as:

$$\log P(\text{“}this\ afternoon\text{”}|Pos:1,2) +$$
$$\log P(\text{“}I\ ate\text{”}|Pos:3,4) +$$
$$\log P(\text{“}ate\ pizza\text{”}|Pos:4,5)$$

where the low-probability bigram (“pizza this” | Pos:2,3) is truncated. In this way, *ngram*-OAXE carries out operation at the ngram granularity: *ngram*-OAXE requires exact match of the word order within the ngram phrases, and allows reordering between phrases (e.g., “I ate pizza | Pos:3,4,5” and “this afternoon | Pos:1,2").

3 Experiment

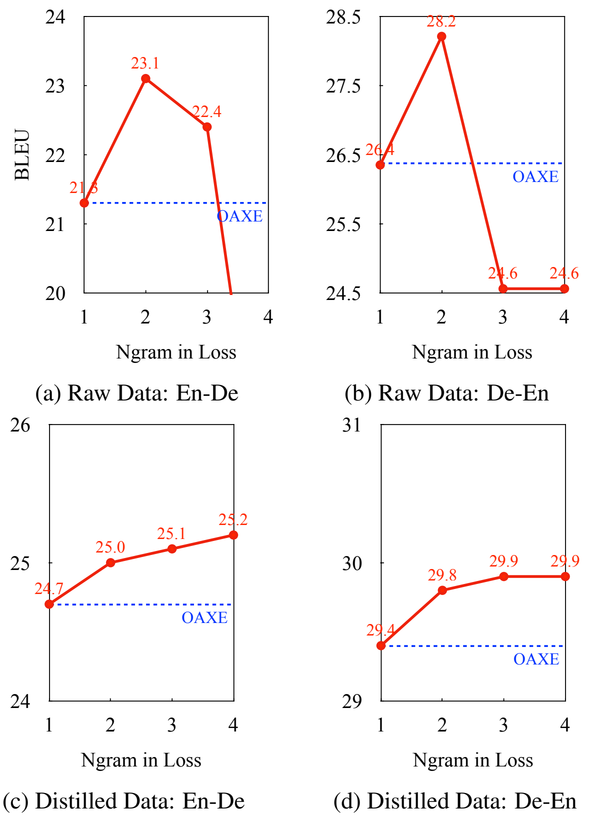
3.1 Experimental Setup

Data We conducted experiments on major benchmarking datasets that are widely-used in previous NAT studies (Ma et al., 2019; Saharia et al., 2020): WMT14 English⇔German (En⇔De, 4.5M sentence pairs) and WMT17 English⇔Chinese (En⇔Zh, 20.0M sentence pairs). We preprocessed the datasets with a joint BPE (Sennrich et al., 2016) with 32K merge operations for the En⇔De and En⇔Zh datasets. For fair comparison with prior work, we reported the Sacre BLEU (Post, 2018)[2] on the En-Zh task, and the compound BLEU (Papineni et al., 2002) on the other tasks.

Knowledge Distillation We closely followed previous works on NAT to apply sequence-level knowledge distillation (Kim and Rush, 2016) to reduce the modes of the training data. Specifically, we obtained **distilled data** by replacing the target side of the original training data (i.e., **raw data**) with translation produced by an external AT teacher. Consistent with previous works (Ghazvininejad et al., 2019, 2020; Du et al., 2021), we employed Transformer-BIG (Vaswani et al., 2017) as the AT teacher for knowledge distillation.

NAT Models We validated our approach on the representative NAT model – CMLM (Ghazvininejad et al., 2019), which uses the conditional mask LM (Devlin et al., 2019) to generate the target sequence from the masked input. The NAT model shares the same architecture as Transformer-BASE (Wang and Tu, 2020): 6 layers for both the

[2]SacreBLEU hash : BLEU+case.mixed+lang.en-zh + numrefs.1+smooth.exp+test.wmt17+tok.zh+version.1.4.2



(a) Raw Data: En-De (b) Raw Data: De-En (c) Distilled Data: En-De (d) Distilled Data: De-En

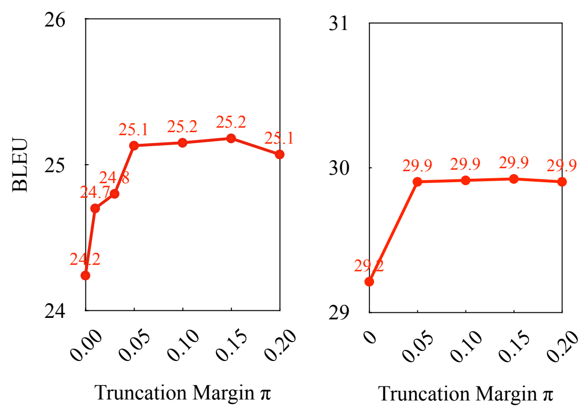
Figure 2: Impact of N-gram choice in *ngram*-OAXE loss (i.e., N in Equation 3). OAXE can be viewed as a special case of *ngram*-OAXE with $N = 1$.

encoder and decoder, 8 attention heads, 512 model dimensions. We chose the CMLM models with the vanilla XE loss (Ghazvininejad et al., 2019) and the OAXE loss (Du et al., 2021) as our two main baselines. To keep consistent with main baselines, we set 5 as length candidates for all CMLM models during inference.

We generally followed the hyperparameters used in (Ghazvininejad et al., 2019). We trained batches of approximately 128K tokens using Adam (Kingma and Ba, 2015). The learning rate warmed up to 5×10^{-4} in the first 10K steps, and then decayed with the inverse square-root schedule. We trained all models for 300k steps, measured the validation BLEU at the end of each epoch, and averaged the 5 best checkpoints. We followed (Li et al., 2019; Sun and Yang, 2020; Saharia et al., 2020; Du et al., 2021) to use de-duplication trick to remove repetitive tokens in the generated output.

3.2 Ablation Study

In this section, we investigated the impact of different components for *ngram*-OAXE on the WMT14 En⇔De validation sets.



(a) Distilled Data: En-De (b) Distilled Data: De-En

Figure 3: Impact of the truncation margin π for NAT models trained on the distilled data.

Impact of Ngram Size We first investigated the impact of different N in the *ngram*-OAXE loss on the translation performance. Figure 2 shows the results for both raw data and distilled data. As seen, the bigram-OAXE achieves the best performance on raw data, while 4gram-OAXE performs best on the distilled data. We attribute the different behaviors to the difficulty of the dataset: raw data contains more modes than distilled data (Gu et al., 2018), thus it is more difficult to learn larger ngrams from the complicated raw data. In the following experiments, we set $N = 2$ for raw data, and $N = 4$ for distilled data.

Impact of Truncation Margin Figure 3 shows the impact of truncation margin π, which is searched from $\{0, 0.05, 0.10, 0.15, 0.20\}$. Intuitively, higher π drops more ngrams. When π increases from 0 to 0.05, we achieved 0.7$\sim$0.9 BLEU improvement by dropping likely invalid ngrams. *ngram*-OAXE is robust to the truncation margin π: when π further increases, the performance does not vary too much. We follow Du et al. (2021) to use $\pi = 0.15$ for all language pairs and datasets in the following experiments.

3.3 Translation Performance

In this section, we conduct comprehensive experiments to validate the effectiveness of the proposed *ngram*-OAXE model. First, we use multiple semantically equivalent references to better evaluate the multimodality nature of generated translation, which serves as the main results for analyses in the following sections. Then we compare our approach with previous work on the benchmarking testsets with single reference.

Model	**W14 En-De**		**NIST Zh-En**	
	BLEU	**Δ**	**BLEU**	**Δ**
Raw Data				
Transformer	71.4	-	41.7	-
CMLM	28.1	-	12.1	-
+OAXE	57.5	+29.4	36.5	+24.4
+*ngram*-OAXE	61.3$^{\uparrow\Uparrow}$	+33.2	38.6$^{\uparrow\Uparrow}$	+26.5
Distilled Data				
Transformer	72.7	-	42.0	-
CMLM	50.7	-	23.7	-
+OAXE	68.0	+17.3	40.4	+16.7
+*ngram*-OAXE	68.9$^{\uparrow\Uparrow}$	+18.2	41.2$^{\uparrow\Uparrow}$	+17.5

Table 1: BLEU scores on test sets with multiple references. "Δ" denotes the improvement over CMLM. "$\uparrow$" and "$\Uparrow$" denotes significantly better than CMLM and OAXE with $p < 0.05$, respectively. The Zh-En NMT model is trained on the WMT17 Zh-En data.

Multiple References We follow Du et al. (2021) to use two test sets with multiple references: 1) the dataset released by Ott et al. (2018) that consists of ten human translations for 500 sentences taken from the WMT14 En-De test set; and 2) the combination of NIST02-08 Zh-En test sets that consists of 7497 sentences with four references. The translation models are trained on the WMT14 En-De and WMT17 Zh-En training data, respectively.

Table 1 lists the translation performance. Encouragingly, *ngram*-OAXE narrows the performance gaps between:

- *NAT models trained on raw data and on distilled data*: Take W14 En-De as an example, knowledge distillation brings an improvement of 22.6 BLEU points over raw data for XE (i.e., from 28.1 to 50.7). OAXE narrows the gap to 10.5 BLEU points (i.e., 57.5 vs. 68.0), and our *ngram*-OAXE further narrows the gap to 7.6 BLEU points (i.e., 61.3 vs. 68.9), moving toward training NAT models without distillation.

- *NAT and AT models trained on raw data*: For independent NAT models without distillation, OAXE reduces the performance gap from 43.3 to 13.9 for En-De, and from 29.6 BLEU to 5.2 BLEU for Zh-En. *Ngram*-OAXE further reduces the gaps to 10.1 and 3.1 BLEU points, indicating the potential of NAT to become a practical system without relying on external resources.

Model	WMT14	
	En-De	De-En
Autoregressive Transformer	27.6	31.4
Non-Autoregressive		
CTC Loss (Libovický and Helcl, 2018)	17.7	19.8
Flowseq (Ma et al., 2019)	18.6	23.4
Imputer (Saharia et al., 2020)	15.6	-
CMLM (Ghazvininejad et al., 2019)	10.6	15.1
+AXE (Ghazvininejad et al., 2020)	20.4	24.9
+Correction (Huang et al., 2022)	20.6	25.4
+OAXE (Du et al., 2021)	22.4	26.8
+*ngram*-OAXE (*Ours*)	**23.6**$^{\uparrow\Uparrow}$	**27.9**$^{\uparrow\Uparrow}$

Table 2: BLEU scores on testsets with single reference for NAT models trained on the **raw data**.

Model	WMT14		WMT17	
	En-De	De-En	En-Zh	Zh-En
Autoregressive Transformer	27.8	31.3	34.4	24.0
Non-Autoregressive				
Bag-of-ngrams (Shao et al., 2020)	20.9	24.6	-	-
Flowseq (Ma et al., 2019)	21.5	26.2	-	-
Bigram CRF (Sun et al., 2019)	23.4	27.2	-	-
Imputer (Saharia et al., 2020)	25.8	28.4	-	-
CMLM (Ghazvininejad et al., 2019)	18.1	21.8	24.2	13.6
+AXE (Ghazvininejad et al., 2020)	23.5	27.9	30.9	19.8
+GLAT (Qian et al., 2021)	25.2	29.8	-	-
+CTC+VAE (Gu and Kong, 2021)	27.5	31.1	-	-
+OAXE (Du et al., 2021)	26.1	30.2	32.9	22.1
+*ngram*-OAXE (*Ours*)	**26.5**$^{\uparrow\Uparrow}$	**30.5**$^{\uparrow\Uparrow}$	**33.2**$^{\uparrow\Uparrow}$	**22.8**$^{\uparrow\Uparrow}$

Table 3: BLEU scores on testsets with single reference for NAT models trained on the **distilled data**.

Benchmarks with Single Reference We also evaluated the performance of fully NAT models on benchmarks with single reference. In addition to the closely related XE variants (e.g., AXE and OAXE), we also compare against several strong baseline models: 1) CTC Loss – a NAT model with latent alignments (Libovický and Helcl, 2018); 2) Flowseq – a latent variable model based on generative flow (Ma et al., 2019); 3) Imputer – an extension of CTC with the use of distillation during training (Saharia et al., 2020); 4) Corretion – a NAT model with error correction mechanism (Huang et al., 2022); 5) Bigram CRF – the CRF-based semi-autoregressive model (Sun et al., 2019); 6) GLAT – Glancing-based training (Qian et al., 2021); 7) CTC+VAE – combining the CTC loss and latent variables (VAE) (Gu and Kong, 2021).

Table 2 lists the results on the **raw data** that do not rely on any external resources (e.g., AT models for KD). CMLMs trained by *ngram*-OAXE improves over the XE-trained baseline by 12.8 BLEU points on average, and outperforms the strong OAXE by +1.0 BLEU points. These results indicate that the ngram supervision helps to better capture the complicated patterns from the raw data.

Table 3 lists the BLEU scores on the **distilled data**. Our approach consistently improves over the strong OAXE loss in all cases, demonstrating the effectiveness and universality of the proposed *ngram*-OAXE loss. Our approach also outperforms all existing NAT using a single technique (exclude Gu and Kong (2021) with two techniques).

Model	**Ngram Size**			
	1	2	3	4
Raw Data				
CMLM	81.8	48.8	30.1	20.4
+OAXE	86.0	64.3	48.0	35.6
+*ngram*-OAXE	88.2	69.8	54.7	42.3
Distilled Data				
CMLM	86.0	63.3	47.1	36.0
+OAXE	90.6	75.4	62.0	51.1
+*ngram*-OAXE	90.5	76.1	62.9	52.0

Table 4: Accuracy (%) of the generated ngram phrases in the model outputs.

Refer.	The Vollmaringen Male Voice Choir got things running with atmospheric songs such as "Im Weinparadies" and "Lustig, ihr Brüder".
CMLM	The MGV Vollmaringen opened with atmospheric songs songs as "Im Weinparapara" and "Lustig, her brothers".
OAXE	The MGV Vollmaringen opened with atmospheric songs such "" "Im Weinpara" and "Lustig, her brothers".
Ours	The MGV Vollmaringen opened with atmospheric songs such as "Im Weinparadies" and "Lustig, ihr Brüder"

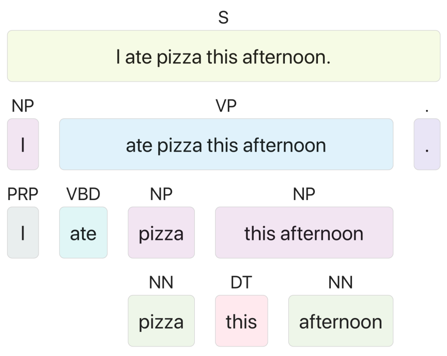
Table 5: Examples of De-En translation for NAT models trained on distilled data. OAXE model often mistakenly translates some ngram phrases (in red color), and *ngram*-OAXE can correctly translate them (in blue color).

4 Analysis

In this section, we provide some insights where *ngram*-OAXE improves over CMLM (i.e., XE) and OAXE from different perspectives. Otherwise stated, we report results on the WMT14 En-De test set with multiple references (i.e., the column "W14 En-De" in Table 1).

4.1 Analysis of Ngram Translation

We first investigate whether the proposed *ngram*-OAXE improves the generation of phrases in the output. To this end, we use the individual ngram scores as the accuracy of generating ngrams of the corresponding size. An individual ngram score is the evaluation of just matching ngrams of a specific size, such as unigram and bigram.[3] As shown in Table 4, our *ngram*-OAXE consistently outperforms the OAXE counterparts in all ngram levels and the improvement goes up with the increase of ngram, demonstrating that *ngram*-OAXE indeed raises the ability of NAT model on capturing the patterns of ngram phrases.



(a) An example constituent tree.

Level	**Syntactic Sequence**
3	NP VP .
2	PRP VBD NP NP .
1	PRP VBD NN DT NN .

(b) Syntactic sequences at different levels.

Figure 4: Constituent tree of the sentence "I ate pizza this afternoon." (a), and the corresponding syntactic sequences at different levels (b, from bottom to up). "2-level" denotes the syntactic sequence at the last but one level of the constituent tree.

Case Study Table 5 shows an translation example on the WMT14 De-En testset. The vanilla CMLM model mistakenly generates the ngram phrase "*songs songs as*" with repeated words "*songs*". Although the OAXE model remedies the repetition problem, it fails to generate the phrase "*such as*". In addition, both the CMLM and OAXE models fail to generate the names of the two songs "*Im Weinparadies*" and "*Lustig, ihr Brüder*". Our *ngram*-OAXE successfully generate all the three ngram phrases.

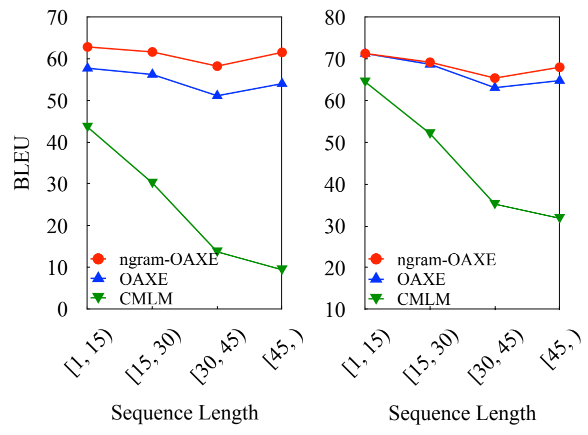
4.2 Analysis of Structure Modeling

Structure Ordering To assess the models' abilities of modeling reordering between ngram phrases, we follow Wang et al. (2021) to measure the precision of outputs at the syntactic level, which can reflect the structure ordering at phrase level. Specifically, we use the syntactic sequence at a certain

[3]The individual ngram BLEUs are generally produced by the BLEU script for the same model outputs, and do not refer to the model training. For example, given the output of *ngram*-OAXE trained on the WMT14 En-De raw data, the script outputs "$BLEU$ = 61.3 88.2/69.8/54.7/42.3". The final BLEU score is 61.3, and the individual 1gram, 2gram, 3gram, and 4gram BLEU scores are 88.2, 69.8, 54.7, and 42.3, respectively.

Model	Syntactic Level			
	1	2	3	4
Raw Data				
CMLM	18.3	11.5	13.4	17.4
OAXE	37.3	28.5	26.6	28.5
ngram-OAXE	39.9$^{\uparrow\Uparrow}$	30.9$^{\uparrow\Uparrow}$	29.4$^{\uparrow\Uparrow}$	30.0$^{\uparrow\Uparrow}$
Distilled Data				
CMLM	31.5	14.7	23.2	25.8
OAXE	41.9	33.0	30.4	31.2
ngram-OAXE	42.6$^{\uparrow\Uparrow}$	33.8$^{\uparrow\Uparrow}$	31.4$^{\uparrow\Uparrow}$	32.3$^{\uparrow\Uparrow}$

Table 6: BLEU scores of the syntactic sequence at different levels (from bottom to up).



(a) Raw Data (b) Distilled Data

Figure 5: Translation performance with respect to the length of the target sentence.

layer (Figure 4b) of the constituent tree of the generated outputs (Figure 4a). Generally, each tag at a higher syntactic level covers more words (e.g., "VP (ate pizza this afternoon)" at 3-level) and corresponds well to a ngram phrase. Accordingly, higher-level syntactic sequences denotes structure ordering at a larger granularity. We calculate the BLEU score for the syntactic sequences of models' outputs to measure the precision of the structure ordering at different granularities.

Table 6 shows the results for syntactic sequences at different levels. OAXE significantly improves the precision of structure order over the CMLM baseline by a large margin, which is consistent with the claim of Du et al. (2021) that OAXE is better at modeling word order. Our *ngram*-OAXE can further improve the precision of structure order, which we attribute to that *ngram*-OAXE models ordering at a larger granularity (i.e., ngrams).

Model	Repetition	PPLs
Gold Test Set	0.04%	90.9
Raw Data		
CMLM	31.11%	1820.2
+OAXE	3.14%	237.6
+*ngram*-OAXE	2.99%	199.9
Distilled Data		
CMLM	12.10%	1435.5
+OAXE	1.56%	240.8
+*ngram*-OAXE	0.98%	125.9

Table 7: Analyses of the generated outputs. Lower repeated token percentage ("Repetition") denotes lower multimodality in a model. Lower perplexities ("PPLs") denote better fluency.

Sequence Length We also investigate the model performance for different sequence lengths. We split the test sets into different buckets based on the reference sentence length, indicating whether a system does better or worse at shorter or longer sentences. Generally, longer sentences are more complex in linguistic structure. Figure 5 shows that results on the sampled WMT14 En-De test set with multiple references. As seen, the performance of XE drops rapidly when the sequence length increases, and OAXE can significantly improves performance on longer sentences with a better modeling of word order. Our *ngram*-OAXE can handle long sequences even better, which we attribute to the strength of *ngram*-OAXE on both translating longer ngrams and modeling structure ordering between ngram phrases.

4.3 Analysis of Generated Output

Token Repetition One widely-cited weakness of existing NAT models is the multimodality problem, in which a model may consider many possible translations at the same time due to the independent predictions of target words (Gu et al., 2018). Accordingly, the NAT output typically contains many repetitive tokens (e.g., "songs songs" in Table 5). We followed the common practices to use repeated token percentage for measuring multimodality in a NAT model, as listed in Table 7. While OAXE can mostly alleviate the repetition problem, the proposed *ngram*-OAXE can further reduce the repeated percentage over the very strong baseline (e.g., 0.98% vs. 1.56% on distilled data).

Generation Fluency We followed Du et al. (2021) to measure the generation fluency with language models released by Fairseq,[4] which are trained on the News Crawl corpus for the target language. To better evaluate the fluency of the generated output, we use a practical trick *de-duplication* (Saharia et al., 2020) to remove the repetitive tokens. Clearly, *ngram*-OAXE consistently improves fluency in all settings compared with OAXE. We attribute the fluency improvement to the strength of *ngram*-OAXE on both translating longer ngrams[5] and modeling sentence structures.

5 Related Work

Alleviating Multimodality Problem for NAT A number of recent efforts have explored ways to improve the NAT models' ability to handle multimodality. One thread of work iteratively refines the generated outputs with K decoding passes (Lee et al., 2018; Gu et al., 2019), which sacrifices the primary benefit of NAT models – fast inference (Kasai et al., 2021). To maintain the advantage of decoding efficiency, another thread of research aims to improve fully NAT models by building dependencies between target tokens (Ma et al., 2019; Shu et al., 2020), or improving the training loss to ameliorate the effect of multimodality (Ghazvininejad et al., 2020; Saharia et al., 2020; Du et al., 2021).

Knowledge distillation (Kim and Rush, 2016) is the preliminary step for the majority of NAT systems, which can effectively alleviate the multimodality problem by simplifying the training data (Zhou et al., 2020) and reducing the token dependency in target sequence (Ren et al., 2020). However, knowledge distillation relies on an external AT teacher, which prevents NAT models from self-completion. The ultimate goal is to train NAT models from scratch (Huang et al., 2022). Our work shows that augmenting NAT models the ability to handle the complex patterns of raw data (e.g., reordering patterns) with advanced training loss is a promising direction to accomplish the goal.

Incorporating Ngrams into NMT Previous studies have incorporated the ngram phrases as an external signal to guide the generation in AT models (Wang et al., 2017; Zhang et al., 2017; Zhao et al., 2018). Concerning NAT models, Guo et al. (2019) enhance decoder inputs with ngram phrases, Sun et al. (2019) use CRF to model bigram dependencies among target tokens to improve the decoding consistency. Kong et al. (2020) use LSTM to generate ngram chunks, which are then merged via heuristic searching algorithm. Closely related to our work, Ma et al. (2018) use bag of ngram phrases as additional training objective for AT models, and Shao et al. (2020) adapt this idea to NAT models. While Shao et al. (2020) require NAT models to fit all the possible orderings of ngrams, we compute the *ngram*-OAXE loss based on the best ordering of ngrams.

6 Conclusion

In this work, we extend OAXE by modeling ordering at the ngram phrase granularity, which can better ameliorate the effect of multimodality for NAT models. Benefiting from modeling translation at a larger granularity, the proposed *ngram*-OAXE loss performs better at translating phrases and long sentences, and improves the fluency of generated translations. Extensive experiments on representative NAT benchmarks show that *ngram*-OAXE consistently improves translation performance over OAXE, and is especially effective on raw data without distillation.

Acknowledgements

We thank all anonymous reviewers for their insightful comments. And the authors gratefully acknowledge the support by the Lee Kuan Yew Fellowship awarded by Singapore Management University.

References

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

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

Marjan Ghazvininejad, V. 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 Zettloyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *EMNLP*.

[4]`https://github.com/pytorch/fairseq/blob/master/examples/language_model/`

[5]The longer ngrams generally account for the fluency of the translation.

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 *Findings of ACL*.

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

Junliang Guo, Xu Tan, Di He, Tao Qin, Linli Xu, and Tie-Yan Liu. 2019. Non-autoregressive neural machine translation with enhanced decoder input. In *AAAI*.

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

Daniel Kang and Tatsunori Hashimoto. 2020. Improved natural language generation via loss truncation. In *ACL*.

Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah Smith. 2021. Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation. In *ICLR*.

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*.

Xiang Kong, Zhisong Zhang, and Eduard Hovy. 2020. Incorporating a local translation mechanism into non-autoregressive translation. *EMNLP*.

Harold W Kuhn. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly*.

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

Zhuohan Li, Zi Lin, Di He, Fei Tian, Tao Qin, Liwei Wang, and T. Liu. 2019. Hint-based training for non-autoregressive machine translation. In *EMNLP-IJCNLP*.

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

Shuming Ma, Xu Sun, Yizhong Wang, and Junyang Lin. 2018. Bag-of-words as target for neural machine translation. In *ACL*.

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

Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Analyzing uncertainty in neural machine translation. In *ICML*.

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

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*.

Yi Ren, Jinglin Liu, Xu Tan, Zhou Zhao, Sheng Zhao, and Tie-Yan Liu. 2020. A study of non-autoregressive model for sequence generation. In *ACL*.

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

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. 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, Zi Lin, Di He, and Zhi-Hong Deng. 2019. Fast structured decoding for sequence models. In *NeurIPS*.

Zhiqing Sun and Yiming Yang. 2020. An em approach to non-autoregressive conditional sequence generation. In *ICML*.

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

Shuo Wang, Zhaopeng Tu, Zhixing Tan, Shuming Shi, Maosong Sun, and Yang Liu. 2021. On the language coverage bias for neural machine translation. In *Findings of ACL*.

Wenxuan Wang and Zhaopeng Tu. 2020. Rethinking the value of transformer components. In *COLING*.

Xing Wang, Zhengdong Lu, Zhaopeng Tu, Hang Li, Deyi Xiong, and Min Zhang. 2017. Neural machine translation advised by statistical machine translation. In *AAAI*.

Jiacheng Zhang, Yang Liu, Huanbo Luan, Jingfang Xu, and Maosong Sun. 2017. Prior knowledge integration for neural machine translation using posterior regularization. In *ACL*.

Yang Zhao, Yining Wang, Jiajun Zhang, and Chengqing Zong. 2018. Phrase table as recommendation memory for neural machine translation. In *IJCAI*.

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