A Proof of Equation 13, Equation 15, and Equation 16

Equation 13 and and Equation 15 are the special cases of Equation 16 when n is 1 or 2, so we directly provide the proof of Equation 16:

$$C_g^s(\theta) = \sum_{i_1=1}^{T_a} \sum_{i_2=1}^{T_a} \cdots \sum_{i_n=1}^{T_a} p_{i_1}(g_1|x,\theta) \times \prod_{k=2}^{n} A_{i_{k-1},i_k} p_{i_k}(g_k|x,\theta), g \in G_n.$$

We begin with the definition of $C_g^s(\theta)$, which accumulates the n-gram g from all alignments:

$$C_g^s(\theta) = \sum_{a \in \mathcal{Y}^*} p(a|x,\theta) C_g(\beta_s^{-1}(a)),$$

where $C_g(\beta_s^{-1}(a))$ is the occurrence count of n-gram g in the collapsed sentence $\beta_s^{-1}(a)$. We use $i_{1:n} = \{i_1, ..., i_n\}$ to denote the possible position of $g = \{g_1, ..., g_n\} \in G_n$. There is an n-gram $g \in G_n$ in the position $i_{1:n}$ if and only if $a_{i_k} = g_k$ for all $k \in \{1, ..., n\}$ and all other words between them are blank tokens. By traversing all possible positions, we can obtain the n-gram count as follows:

$$C_g(\beta_s^{-1}(a)) = \sum_{i_1=1}^{T_a} \sum_{i_2=1}^{T_a} \cdots \sum_{i_n=1}^{T_a} 1(a_{i_1} = g_1) \times \prod_{k=2}^{n} 1(i_{k-1} < i_k, a_{i_k} = g_k, a_{i_{k-1}+1:i_k-1} = \epsilon), g \in G_n.$$

where $1(\cdot)$ is the indicator function that takes value 1 when the inside condition holds, otherwise it takes value 0. Recall that A_{ij} is the probability that all positions between i and j are blank tokens, which satisfies the following equation:

$$A_{i_{k-1},i_k} = p(i_{k-1} < i_k, a_{i_{k-1}+1:i_k-1} = \epsilon|x,\theta).$$

Using the above equations, we can rewrite $C_g^s(\theta)$ as:

$$\begin{aligned}
C_g^s(\theta) &= \sum_a p(a|x,\theta) C_g(\beta_s^{-1}(a)) \\
&= \sum_{i_1=1}^{T_a} \sum_{i_2=1}^{T_a} \cdots \sum_{i_n=1}^{T_a} \sum_a p(a|x,\theta) 1(a_{i_1} = g_1) \times \prod_{k=2}^{n} 1(i_{k-1} < i_k, a_{i_k} = g_k, a_{i_{k-1}+1:i_k-1} = \epsilon) \\
&= \sum_{i_1=1}^{T_a} \sum_{i_2=1}^{T_a} \cdots \sum_{i_n=1}^{T_a} p(a_{i_1} = g_1|x,\theta) \times \prod_{k=2}^{n} p(a_{i_k} = g_k|x,\theta) p(i_{k-1} < i_k, a_{i_{k-1}+1:i_k-1} = \epsilon) \\
&= \sum_{i_1=1}^{T_a} \sum_{i_2=1}^{T_a} \cdots \sum_{i_n=1}^{T_a} p_{i_1}(g_1|x,\theta) \times \prod_{k=2}^{n} A_{i_{k-1},i_k} p_{i_k}(g_k|x,\theta), g \in G_n. \ \square
\end{aligned}$$

B The $\mathcal{O}(nT_a^2)$ algorithm for Equation 16

Here we give the $\mathcal{O}(nT_a^2)$ algorithm to calculate $C_g^s(\theta)$ according to Equation 16. Notice that Equation 16 has a similar form with the n-step transition probability of a Markov chain, so we can try a similar method as calculating the transition probability for a Markov chain. Specifically, we introduce a set of state vectors s_k , where $k \in \{1, 2, ..., T_a\}$ and the dimension of each vector s_k is T_a . We define the state vectors as follows:

$$s_k(i_k) = \begin{cases} p_{i_k}(g_k|x,\theta), & k = 1 \\ \sum_{i_{k-1}=1}^{T_a} s_{k-1}(i_{k-1}) A_{i_{k-1},i_k} p_{i_k}(g_k|x,\theta), & k > 1 \end{cases}.$$

From the above equation, the time complexity to calculate s_k given s_{k-1} is $\mathcal{O}(T_a^2)$ and can be efficiently calculated in matrix form. Therefore, the time complexity to sequentially calculate $s_{1:n}$ is $\mathcal{O}(nT_a^2)$. Finally, we can easily obtain $C_g^s(\theta)$ by summing the last vector s_n:

$$
\begin{aligned}
C_g^s(\theta) &= \sum_{i_1=1}^{T_a}\sum_{i_2=1}^{T_a}\cdots\sum_{i_n=1}^{T_a} p_{i_1}(g_1|x,\theta) \times \prod_{k=2}^{n} A_{i_{k-1},i_k} p_{i_k}(g_k|x,\theta) \\
&= \sum_{i_n=1}^{T_a}\cdots\sum_{i_2=1}^{T_a}\sum_{i_1=1}^{T_a} s_1(i_1) \prod_{k=2}^{n} A_{i_{k-1},i_k} p_{i_k}(g_k|x,\theta) \\
&= \sum_{i_n=1}^{T_a}\cdots\sum_{i_3=1}^{T_a}\sum_{i_2=1}^{T_a} s_2(i_2) \prod_{k=3}^{n} A_{i_{k-1},i_k} p_{i_k}(g_k|x,\theta) \\
&= ..... \\
&= \sum_{i_n=1}^{T_a}\sum_{i_{n-1}=1}^{T_a} s_{n-1}(i_{n-1}) \times A_{i_{n-1},i_n} p_{i_n}(g_n|x,\theta) \\
&= \sum_{i_n=1}^{T_a} s_n(i_n), g \in G_n. \quad \square
\end{aligned}
$$

C Proof of Equation 20

Here we provide the proof of Equation 20:

$$
R_g(\theta) = C_g^s(\theta) - C_g(\theta) = \sum_{t=1}^{T_a-1} p_t(g_1|x,\theta) p_{t+1}(g_1|x,\theta), g \in G_1,
$$

which is equivalent to proving the following equation:

$$
C_g(\theta) = p_1(g_1|x,\theta) + \sum_{t=2}^{T_a} p_t(g_1|x,\theta)(1 - p_{t-1}(g_1|x,\theta)), g \in G_1.
$$

We begin with the definition of $C_g(\theta)$, which accumulates the n-gram g from all alignments:

$$
C_g(\theta) = \sum_{a \in \mathcal{Y}^*} p(a|x,\theta) C_g(\beta^{-1}(a)).
$$

$C_g(\beta^{-1}(a))$ is the occurrence count of n-gram g in the collapsed sentence $\beta^{-1}(a)$. Besides blank tokens, repeated words will also be removed by β^{-1}. Therefore, we call a word in the alignment a valid word when it does not repeat the word before it. We can calculate the 1-gram count by collecting all valid words, which will not be removed by the collapsing function:

$$
C_g(\beta^{-1}(a)) = 1(a_1 = g_1) + \sum_{t=2}^{T_a} 1(a_t = g_1, a_{t-1} \neq g_1), g \in G_1.
$$

Using the above equation, we can rewrite $C_g(\theta)$ as:

$$
\begin{aligned}
C_g(\theta) &= \sum_a p(a|x,\theta) C_g(\beta^{-1}(a)) = p(a_1 = g_1|x,\theta) + \sum_{t=2}^{T_a} p(a_t = g_1, a_{t-1} \neq g_1|x,\theta) \\
&= p_1(g_1|x,\theta) + \sum_{t=2}^{T_a} p_t(g_1|x,\theta)(1 - p_{t-1}(g_1|x,\theta)), g \in G_1. \quad \square
\end{aligned}
$$

D Analysis

To better understand our method, we present a qualitative analysis to provide some insights into the improvements of NMLA. We analyze the generated outputs of the WMT14 En-De test set in the following experiments.

NMLA Handles Long Sequences Better We first investigate the performance of AT and NAT models for different sequence lengths. We split the test set into different buckets based on target sequence length and calculate BLEU for each bucket. We report the results in Table 6. The first observation is that the performance of Vanilla-NAT drops drastically as the sequence length increases. It is a well-known property of NAT and can be explained as the increased probability of observing misalignment between model output and target as the target sequence becomes longer. The misalignment in long sequences causes the inaccuracy of loss and therefore harms the translation quality. We have a similar observation when comparing CTC with NMLA. The performance of CTC is close to NMLA on short sequences, but the gap becomes larger as the sequence length increases. Our explanation is that the longer the sequence, the more likely the monotonic assumption fails due to the global word reordering, which causes the inaccuracy of loss and therefore harms the translation quality. NMLA models non-monotonic alignments and alleviates the inaccuracy of loss on long sequences, so it achieves strong improvements on long sequences and even outperforms AT by more than 2 BLEU, which suffers from the error accumulation on long sequences.

Table 6: The performance of AT and NAT models with respect to the target sequence length. 'Vanilla' means Vanilla-NAT. 'N' denotes the target sequence length. 'AT' means autoregressive Transformer.

Length	**Vanilla**	**CTC**	**AT**	**NMLA**
$1 \leq N < 20$	21.27	24.83	25.68	25.55
$20 \leq N < 40$	19.49	26.81	28.05	27.82
$40 \leq N < 60$	16.21	26.54	26.71	27.74
$60 \leq N$	10.69	26.69	26.44	28.89
All	19.32	26.34	27.54	27.57

NMLA Improves Model Confidence NAT suffers from the multi-modality problem, which makes the model consider many possible translations at the same time and become less confident in generating outputs. We investigate whether our method alleviates the multi-modality problem by measuring how confident the model is during the decoding. We use the information entropy to measure the confidence, which is calculated as $H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x)$, and lower entropy indicates higher confidence. We calculate the entropy in every position when decoding the WMT14 En-De test set and use the average entropy to represent the model confidence. We do not report the entropy of AT since it varies with the decoding algorithm. The average entropy scores of NAT models are reported in Table 7. We can see that the Vanilla-NAT has low prediction confidence, which is substantially improved by CTC. NMLA further reduces the average entropy to 0.061, showing that the model confidence can be improved by modeling non-monotonic latent alignments.

Table 7: The average prediction entropy of NAT models. 'Vanilla' means Vanilla-NAT.

	Vanilla	**CTC**	**NMLA**
Entropy	2.098	0.260	0.061

NMLA Improves Generation Fluency Since NAT generates target words independently, generation fluency is a main weakness of NAT. The multi-modality problem makes NAT consider many possible translations at the same time and generate an inconsistent output, which further reduces the generation fluency. We investigate whether our method improves the generation fluency with an n-gram language model [20] trained on the target side of WMT14 En-De training data. We use the language model to calculate the perplexity scores of model outputs and report the results in Table 8. We can see that Vanilla-NAT and CTC have high perplexity which indicates low fluency. NMLA substantially reduces the perplexity score, which is only 42.6 higher than the gold reference. For AT, its perplexity score is even lower than the gold reference, which is consistent with prior works that autoregressive NMT generally improves fluency at the cost of adequacy [8, 10].

Table 8: The perplexity scores of AT and NAT models on WMT14 En-De test set.

	Vanilla	**CTC**	**NMLA**	**AT**	**Gold**
PPL	597.0	315.5	258.8	197.0	216.2

E Batch Speedup

For CTC-based models including NMLA, a common concern is that they require a larger number of calculations than the autoregressive model due to the longer decoder length. Though there is a significant speedup when decoding sentence by sentence, this advantage may disappear when we use a larger decoding batch. In response to this concern, we measure the decoding speedup on WMT14 En-De test set under different batch sizes and report both BLEU and speedup in Figure 3. Under the memory limit, the maximum batch size is 400. From Figure 3, NMLA still maintains $2.4\times$ speedup under the maximum batch size while achieving comparable performance to the autoregressive model. For NMLA+beam&lm, it simultaneously achieves superior translation quality and speed to the autoregressive model, which addresses the concern on batch speedup for CTC-based models.

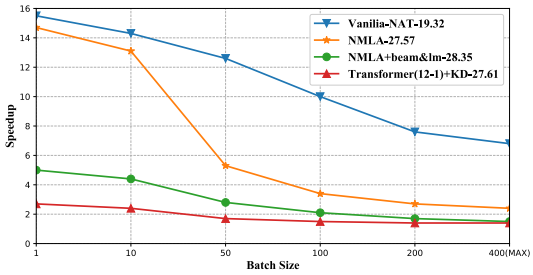

Figure 3: BLEU and speedup of NAT and AT models measured under different batch sizes on WMT14 En-De test set.

F Combination of N-grams

In Table 1, we only finetune the baseline with different n-gram matching objectives respectively. Here we combine different n-gram matching objectives to see whether the combination can improve the model performance. First, we linearly combine 1-gram and 2-gram matching with a hyperparameter α:

$$\mathcal{L}(\theta) = \alpha \cdot \mathcal{L}_1(\theta) + (1-\alpha) \cdot \mathcal{L}_2(\theta), \quad \mathcal{L}^s(\theta) = \alpha \cdot \mathcal{L}_1^s(\theta) + (1-\alpha) \cdot \mathcal{L}_2^s(\theta).$$

We train CTC and SCTC baselines with different α and report the results in Table 9, which shows that the combination of 1-gram and 2-gram underperforms simply using 2-gram matching.

Table 9: BLEU scores of SCTC and CTC under different α on WMT14 En-De validation set.

α	0	0.25	0.5	0.75	1
SCTC	25.09	24.98	24.85	24.70	24.54
CTC	25.90	25.74	25.66	25.51	25.42

As $n > 2$ is allowed in SCTC, we further try to combine higher rank n-grams. We consider $n \in \{1, 2, 3, 4\}$ and combine them with arithmetic average and geometric average respectively. The BLEU scores of arithmetic average and geometric average are respectively 24.94 BLEU and 24.88 BLEU on WMT14 En-De validation set, which also underperforms simply using 2-gram matching. Therefore, we conclude that simply combining different n-gram matching objectives cannot improve the model performance.