FUZZY ALIGNMENTS IN DIRECTED ACYCLIC GRAPH FOR NON-AUTOREGRESSIVE MACHINE TRANSLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Non-autoregressive translation (NAT) reduces the decoding latency but suffers from performance degradation due to the multi-modality problem. Recently, the structure of Directed Acyclic Graph has achieved great success in NAT, which tackles the multi-modality problem by introducing dependency between vertices. However, training it with Negative Log Likelihood loss implicitly requires a strict alignment between reference tokens and vertices, weakening its ability to handle multiple translation modalities. In this paper, we hold the view that all paths in the graph are fuzzily aligned with the reference sentence. We do not require the exact alignment but train the model to maximize a fuzzy alignment score between the graph and reference, which takes captured translations in all modalities into account. Extensive experiments on major WMT benchmarks show that our method substantially improves translation performance and increases prediction confidence, setting a new state of the art for NAT on the raw training data.[1]

1 INTRODUCTION

Non-autoregressive translation (NAT) (Gu et al., 2018) reduces the decoding latency by generating all target tokens in parallel. Compared with the autoregressive (AT) counterpart (Vaswani et al., 2017), NAT often suffers from performance degradation due to the severe *multi-modality problem* (Gu et al., 2018), which refers to the fact that one source sentence may have multiple translations in the target language. NAT models are usually trained with the cross entropy loss, which strictly aligns model predictions with target tokens. The strict alignment does not allow multi-modality such as position shifts and word reorderings, so proper translations are likely to be wrongly penalized. The inaccurate training signal makes NAT tend to generate a mixture of different translations rather than a consistent translation, which typically contains many repeated tokens in generated results.

Many efforts have been devoted to address the above problem (Libovický & Helcl, 2018; Shao et al., 2020; Ghazvininejad et al., 2020a; Du et al., 2021; Huang et al., 2022b). Among them, Directed Acyclic Transformer (DA-Transformer) (Huang et al., 2022b) introduces a Directed Acyclic Graph (DAG) on top of the NAT decoder, where decoder hidden states are organized as a graph rather than a sequence. By modeling the dependency between vertices, DAG is able to capture multiple translation modalities simultaneously by assigning tokens in different translations to distinct vertices. In this way, DA-Transformer does not heavily rely on knowledge distillation (KD) (Kim & Rush, 2016; Zhou et al., 2020) to reduce data modalities and can achieve superior performance on raw data.

Despite the success of DA-Transformer, training it with Negative Log Likelihood (NLL) loss, which marginalizes all the joint probability of DAG paths and references, is sub-optimal in the scenario of NAT. It implicitly introduces a strict monotonic alignment between reference tokens and vertices on all paths. Although DAG enables the model to capture different translations in different transition paths, only paths that are aligned verbatim with reference with a large probability will be well calibrated by NLL (See Section 2.3 for detailed analysis). It weakens DAG's ability to handle data multi-modality, making the model less confident in generating outputs and requiring a large graph size to achieve satisfying performance.

[1] We will release the source code upon publication.

In this paper, we extend the verbatim alignment between reference and DAG path to a fuzzy alignment, aiming to better handle the multi-modality problem. Specifically, we do not require the exact alignment but hold the view that all paths in DAG are fuzzily aligned with the reference sentence. To indicate the quality of alignment, an alignment score is assigned to each DAG path based on the expectation of n-gram overlapping. We further define an alignment score between the whole DAG and reference as the expected alignment score of all its paths. The model is trained to maximize the alignment score, which takes captured translations in all modalities into account.

Experiments on major WMT benchmarks show that our method substantially improves the translation quality of DA-Transformer. It achieves comparable performance to the autoregressive Transformer without the help of knowledge distillation and beam search decoding, setting a new state of the art for NAT on the raw training data.

2 BACKGROUND

2.1 NON-AUTOREGRESSIVE MACHINE TRANSLATION

Non-autoregressive translation (Gu et al., 2018) is proposed to reduce the decoding latency. It abandons the assumption of autoregressive dependency between output tokens and generates all tokens simultaneously. Given a source sentence $\boldsymbol{x} = \{x_1, ..., x_N\}$, NAT factorizes the joint probability of target tokens $\boldsymbol{y} = \{y_1, ..., y_M\}$ as,

$$P_\theta(\boldsymbol{y}|\boldsymbol{x}) = \prod_{i}^{M} P_\theta(y_i|\boldsymbol{x}), \tag{1}$$

where θ is the model parameter and $P_\theta(y_i|\boldsymbol{x})$ denotes the translation probability of y_i at position i.

In vanilla NAT, the decoder length is set to reference length during the training and determined by a trainable length predictor during the inference. Standard NAT model is trained with the cross entropy loss, which strictly requires the generation of word y_i at position i:

$$\mathcal{L}_{CE} = -\sum_{i}^{M} \log P_\theta(y_i|\boldsymbol{x}). \tag{2}$$

2.2 DIRECTED ACYCLIC TRANSFORMER

Vanilla NAT model suffers two major drawbacks, including inflexible length prediction and disability to handle multi-modality. DA-Transformer (Huang et al., 2022b) addresses these problems by stacking a Directed Acyclic Graph (DAG) on the top of NAT decoder, where hidden states and transitions between states represent vertices and edges in DAG.

Formally, given a bilingual pair $\boldsymbol{x} = \{x_1, ..., x_N\}$ and $\boldsymbol{y} = \{y_1, ..., y_M\}$, DA-Transformer sets the decoder length $L = \lambda \cdot N$ and models the translation probability by marginalizing over all valid paths in DAG:

$$P_\theta(\boldsymbol{y}|\boldsymbol{x}) = \sum_{\boldsymbol{a} \in \Gamma_{\boldsymbol{y}}} P_\theta(\boldsymbol{y}|\boldsymbol{a}, \boldsymbol{x}) P_\theta(\boldsymbol{a}|\boldsymbol{x}), \tag{3}$$

where $\boldsymbol{a} = \{a_1, ..., a_M\}$ is a path represented by a sequence of vertex indexes with the bound $1 = a_1 < ... < a_M = L$ and $\Gamma_{\boldsymbol{y}}$ contains all paths with the same length as the target sentence $\boldsymbol{y}$. $P_\theta(\boldsymbol{a}|\boldsymbol{x})$ and $P_\theta(\boldsymbol{y}|\boldsymbol{a}, \boldsymbol{x})$ indicate the probability of path $\boldsymbol{a}$ and the probability of target sentence $\boldsymbol{y}$ conditioned on path $\boldsymbol{a}$ respectively. DAG factorizes the path probability $P_\theta(\boldsymbol{a}|\boldsymbol{x})$ based on the Markov hypothesis:

$$P_\theta(\boldsymbol{a}|\boldsymbol{x}) = \prod_{i=1}^{M-1} P_\theta(a_{i+1}|a_i, \boldsymbol{x}) = \prod_{i=1}^{M-1} \mathbf{E}_{a_i, a_{i+1}}, \tag{4}$$

where $\mathbf{E} \in \mathbb{R}^{L \times L}$ is the row-normalized transition matrix. Due to the unidirectional inherence of DAG, the lower triangular part of $\mathbf{E}$ is masked to zeros. Once path $\boldsymbol{a}$ is determined, token y_i can be

generated conditioned on the decoder hidden state with index a_i:

$$P_\theta(\boldsymbol{y}|\boldsymbol{a},\boldsymbol{x}) = \prod_{i=1}^{M} P_\theta(y_i|a_i,\boldsymbol{x}). \tag{5}$$

DA-Transformer is trained to minimize the Negative Log Likelihood loss via dynamic programming:

$$\mathcal{L} = -\log P_\theta(\boldsymbol{y}|\boldsymbol{x}) = -\log \sum_{\boldsymbol{a}\in\Gamma_{\boldsymbol{y}}} P_\theta(\boldsymbol{y}|\boldsymbol{a},\boldsymbol{x}) P_\theta(\boldsymbol{a}|\boldsymbol{x}). \tag{6}$$

The structure of DAG helps NAT model token dependency through transition probability between vertices while almost zero overhead of sequential operation is paid in decoding.

2.3 Multi-modality in DAG

DA-Transformer alleviates the multi-modality problem by arranging tokens in different translations to distinct vertices, and the learned vertex transition probability prevents them from co-occurring in the same DAG path. Despite DAG's ability to capture different translations simultaneously, only paths that are aligned verbatim with reference with a large probability will be well calibrated by the NLL loss. It can be demonstrated by inspecting the gradients:

$$\frac{\partial}{\partial\theta}\mathcal{L} = \sum_{\boldsymbol{a}\in\Gamma_y} P_\theta(\boldsymbol{a}|\boldsymbol{y},\boldsymbol{x}) \frac{\partial}{\partial\theta}\mathcal{L}_{\boldsymbol{a}}, \tag{7}$$

where $\mathcal{L}_{\boldsymbol{a}} = -\log P_\theta(\boldsymbol{y},\boldsymbol{a}|\boldsymbol{x})$. Equation 7 shows that NLL loss will assign paths with small posterior probability smaller gradient weights, making paths corresponding to translations in other modalities poorly calibrated. Considering that only one reference is provided in major translation benchmarks, training DAG with the NLL loss inevitably hurts its potential to handle translation multi-modality.

3 Methodology

In this section, we extend the verbatim alignment between reference and DAG path to an n-gram based fuzzy alignment. In Section 3.1 and 3.2, we explore a metric to calibrate DAG under fuzzy alignment. We develop an efficient algorithm to calculate the alignment score in Section 3.3, and introduce a training strategy with fuzzy alignment in Section 3.4.

3.1 Fuzzy Alignment

We do not require the exact alignment but hold the view that all paths in DAG are fuzzily aligned with the reference sentence to some extent. In order to allow multi-modality such as position shifts and word reorderings, we measure the quality of fuzzy alignment by the order-agnostic n-gram overlapping, which is a widely used factor in machine translation evaluation (Papineni et al., 2002).

Given a target sentence $\boldsymbol{y} = \{y_1, ..., y_M\}$ and an integer $n \geq 1$, we denote the set of all its non-repeating n-grams as $G_n(\boldsymbol{y})$, the elements of which are mutually exclusive. For any n-gram $\boldsymbol{g} \in G_n(\boldsymbol{y})$, the number of its appearances in $\boldsymbol{y}$ is denoted as $C_{\boldsymbol{g}}(\boldsymbol{y})$. We measure the alignment quality by clipped n-gram precision of model output $\boldsymbol{y}'$ against reference $\boldsymbol{y}$:

$$p_n(\boldsymbol{y}',\boldsymbol{y}) = \frac{\sum_{\boldsymbol{g}\in G_n(\boldsymbol{y})} \min(C_{\boldsymbol{g}}(\boldsymbol{y}'), C_{\boldsymbol{g}}(\boldsymbol{y}))}{\sum_{\boldsymbol{g}\in G_n(\boldsymbol{y}')} C_{\boldsymbol{g}}(\boldsymbol{y}')}. \tag{8}$$

Considering that the DAG path is composed of consecutive vertices, and each vertex represents a word distribution rather than a concrete word, we assign an alignment score to each DAG path with the expected n-gram precision:

$$p_n(\boldsymbol{a},\boldsymbol{y}) = \mathbb{E}_{\boldsymbol{y}'\sim P_\theta(\boldsymbol{y}'|\boldsymbol{a},\boldsymbol{x})}[p_n(\boldsymbol{y}',\boldsymbol{y})]. \tag{9}$$

3.2 ESTIMATING FUZZY ALIGNMENT OF DAG

Directed Acyclic Graph simultaneously retains multiple translations in different paths, requiring us to measure the alignment between generated graph and reference sentence by the averaged alignment score of all its possible transition paths:

$$p_n(\theta, \boldsymbol{y}) = \mathbb{E}_{\boldsymbol{a} \sim P_\theta(\boldsymbol{a}|\boldsymbol{x})}[p_n(\boldsymbol{a}, \boldsymbol{y})]. \tag{10}$$

However, the search spaces for translations and paths are both exponentially large, making Equation 10 intractable. Alternatively, we turn to calculate the ratio of the clipped expected count of n-gram matching to expected number of n-grams, which can be considered as an approximation of $p_n(\theta, \boldsymbol{y})$:

$$p'_n(\theta, \boldsymbol{y}) = \frac{\sum_{\boldsymbol{g} \in G_n(\boldsymbol{y})} \min(\mathbb{E}_{\boldsymbol{y}'}[C_{\boldsymbol{g}}(\boldsymbol{y}')], C_{\boldsymbol{g}}(\boldsymbol{y}))}{\mathbb{E}_{\boldsymbol{y}'}[\sum_{\boldsymbol{g} \in G_n(\boldsymbol{y}')} C_{\boldsymbol{g}}(\boldsymbol{y}')]}. \tag{11}$$

Though $p'_n(\theta, \boldsymbol{y})$ is much simplified, the direct calculation will still suffer from exponential time complexity due to the intractable expectation terms in it. In the following section, we will focus on developing an efficient algorithm to calculate $\mathbb{E}_{\boldsymbol{y}'}[C_{\boldsymbol{g}}(\boldsymbol{y}')]$ and $\mathbb{E}_{\boldsymbol{y}'}[\sum_{\boldsymbol{g} \in G_n(\boldsymbol{y}')} C_{\boldsymbol{g}}(\boldsymbol{y}')]$.

3.3 EFFICIENT CALCULATION OF FUZZY ALIGNMENT

Fortunately, the Markov property of DAG makes it possible to simplify the calculation of Equation 11. We first consider an n-vertex subpath $v_1, ..., v_n$, which is bounded by $1 \leq v_1, ..., v_n \leq L$ [2]. An indicator function $\mathbb{1}(v_1, ..., v_n \in \boldsymbol{a})$ is introduced to indicate whether $v_1, ..., v_n$ is a part of path $\boldsymbol{a}$. Then we can define the passing probability of a subpath by enumerating all possible paths $\boldsymbol{a}$:

$$P(\mathbb{1}(v_1, ..., v_n)|\boldsymbol{x}) = \sum_{\boldsymbol{a}} P_\theta(\boldsymbol{a}|\boldsymbol{x}) \mathbb{1}(v_1, ..., v_n \in \boldsymbol{a}). \tag{12}$$

With the definition of passing probability, we can transform the summation operation from enumerating all translations to enumerating all the n-vertex subpaths, which largely narrows the search space. We directly give the following theorem and refer readers to Appendix A for detailed derivation:

$$\mathbb{E}_{\boldsymbol{y}'}[\sum_{\boldsymbol{g} \in G_n(\boldsymbol{y}')} C_{\boldsymbol{g}}(\boldsymbol{y}')] = \sum_{v_1, ..., v_n} P(\mathbb{1}(v_1, ..., v_n)|\boldsymbol{x}), \tag{13}$$

$$\mathbb{E}_{\boldsymbol{y}'}[C_{\boldsymbol{g}}(\boldsymbol{y}')] = \sum_{v_1, ..., v_n} P(\mathbb{1}(v_1, ..., v_n)|\boldsymbol{x}) \prod_{i=1}^{n} P_\theta(g_i|v_i), \tag{14}$$

where $P_\theta(g_i|v_i)$ denotes the probability that vertex with index v_i generates the i-th token in n-gram g. By narrowing the search space, we can find that those expected counts are actually the sum and weighted sum of passing probabilities of n-vertex subpaths. It is worth noting that the n-vertex subpath $v_1, ..., v_n$ forms a Markov chain, leading us to factorize the passing probability into the product of transitions. We reuse the transition matrix $\mathbf{E}$ introduced in Section 2.2 for notation, where $\mathbf{E}_{v_i, v_j}$ is the transition probability from v_i to v_j. The passing probability of an n-vertex subpath can be formulated as:

$$P(\mathbb{1}(v_1, ..., v_n)|\boldsymbol{x}) = P(\mathbb{1}(v_1)|\boldsymbol{x}) \prod_{i=1}^{n-1} \mathbf{E}_{v_i, v_{i+1}}, \tag{15}$$

where $P(\mathbb{1}(v)|\boldsymbol{x})$ denotes the passing probability of vertex with index v. For any $1 \leq v \leq L$, $P(\mathbb{1}(v)|\boldsymbol{x})$ can be calculated efficiently via dynamic programming:

$$P(\mathbb{1}(v)|\boldsymbol{x}) = \sum_{v' < v} P(\mathbb{1}(v')|\boldsymbol{x}) \mathbf{E}_{v', v}, \tag{16}$$

[2] Note that there is no need to require $1 = v_1 < ... < v_n = L$. In fact, the upper triangular transition matrix only considers paths with monotonic order and sets the probability of others to 0.

Algorithm 1 Calculation of $\mathbb{E}_{\boldsymbol{y}'}[C_{\boldsymbol{g}}(\boldsymbol{y}')]$ and $\mathbb{E}_{\boldsymbol{y}'}[\sum_{\boldsymbol{g}\in G_n(\boldsymbol{y}')} C_{\boldsymbol{g}}(\boldsymbol{y}')]$

Input: Transition matrix $\mathbf{E}$, Token probability matrix $\mathbf{G}$, Graph size L, n-gram order N.
Output: Expected counts $\mathbb{E}_{\boldsymbol{y}'}[C_{\boldsymbol{g}}(\boldsymbol{y}')]$ and $\mathbb{E}_{\boldsymbol{y}'}[\sum_{\boldsymbol{g}\in G_n(\boldsymbol{y}')} C_{\boldsymbol{g}}(\boldsymbol{y}')]$.

$p_1 \leftarrow 1$
for $i \leftarrow 2$ to L **do**
  $p_i \leftarrow \sum_{j=1}^{i-1} p_j \mathbf{E}_{j,i}$
$\boldsymbol{p} \leftarrow [p_1, p_2, ..., p_L]$
$\boldsymbol{c}^{(1)} \leftarrow \boldsymbol{p}^T$; $\boldsymbol{c}^{(2)} \leftarrow \boldsymbol{p}^T \odot \mathbf{G}_{1,:}$
for $i \leftarrow 2$ to N **do**
  $\boldsymbol{c}^{(1)} \leftarrow \boldsymbol{c}^{(1)} \cdot \mathbf{E}$
  $\boldsymbol{c}^{(2)} \leftarrow (\boldsymbol{c}^{(2)} \cdot \mathbf{E}) \odot \mathbf{G}_{i,:}$
$\mathbb{E}_{\boldsymbol{y}'}[C_{\boldsymbol{g}}(\boldsymbol{y}')] \leftarrow \|\boldsymbol{c}^{(2)}\|_1$; $\mathbb{E}_{\boldsymbol{y}'}[\sum_{\boldsymbol{g}\in G_n(\boldsymbol{y}')} C_{\boldsymbol{g}}(\boldsymbol{y}')] \leftarrow \|\boldsymbol{c}^{(1)}\|_1$
return $\mathbb{E}_{\boldsymbol{y}'}[C_{\boldsymbol{g}}(\boldsymbol{y}')]$ and $\mathbb{E}_{\boldsymbol{y}'}[\sum_{\boldsymbol{g}\in G_n(\boldsymbol{y}')} C_{\boldsymbol{g}}(\boldsymbol{y}')]$.

with the boundary condition that the passing probability of the first vertex in the graph is equal to 1. For convenience, we introduce a passing probability vector $\boldsymbol{p} \in \mathbb{R}^L$, where $\boldsymbol{p}_v = P(\mathbb{1}(v)|\boldsymbol{x})$. Equation 15 reminds us that the sum of n-vertex passing probability is actually the sum of $(n-1)$-hop transition probability in a Markov chain with initial probability $\boldsymbol{p}$ and transition matrix $\mathbf{E}$. On the basis of Equation 13, we have:

$$\mathbb{E}_{\boldsymbol{y}'}[\sum_{\boldsymbol{g}\in G_n(\boldsymbol{y}')} C_{\boldsymbol{g}}(\boldsymbol{y}')] = \sum_{v_1,\ldots,v_n} P(\mathbb{1}(v_1)|\boldsymbol{x}) \prod_{i=1}^{n-1} \mathbf{E}_{v_i,v_{i+1}} = \|\boldsymbol{p}^T \cdot \mathbf{E}^{n-1}\|_1. \tag{17}$$

By rearranging the order of products, we can handle the weighted sum of passing probabilities in Equation 14 in a similar way. We introduce $\mathbf{G} \in \mathbb{R}^{n\times L}$, with the ith row of $\mathbf{G}$ representing the token probability of g_i at each vertex, i.e., $\mathbf{G}_{i,v} = P_\theta(g_i|v)$. Then we can obtain:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{y}'}[C_{\boldsymbol{g}}(\boldsymbol{y}')] &= \sum_{v_1,\ldots,v_n} P(\mathbb{1}(v_1)|\boldsymbol{x}) \prod_{i=1}^{n-1} \mathbf{E}_{v_i,v_{i+1}} \prod_{i=1}^{n} P_\theta(g_i|v_i) \\
&= \sum_{v_1,\ldots,v_n} (P(\mathbb{1}(v_1)|\boldsymbol{x}) P_\theta(g_1|v_1)) \prod_{i=1}^{n-1} \left(\mathbf{E}_{v_i,v_{i+1}} P_\theta(g_{i+1}|v_{i+1})\right) \\
&= \|\boldsymbol{p}^T \odot \mathbf{G}_{1,:} \cdot \mathbf{E} \odot \mathbf{G}_{2,:} \cdot \ldots \cdot \mathbf{E} \odot \mathbf{G}_{n,:}\|_1,
\end{aligned} \tag{18}$$

where $\odot$ denotes position-wise product, and $\mathbf{G}_{i,:}$ should be broadcast when $i > 1$. We refer readers to Appendix B for detailed derivation.

Based on the discussion above, we finally avoid the exponential time complexity in calculating $p'_n(\theta, \boldsymbol{y})$. Instead, it can be implemented with $\mathcal{O}(n + L)$ parallel operations, making the proposed fuzzy alignment objective practical for the training. We summarize the calculation process in Algorithm 1.

3.4 Training Strategy

We have found a way to train DA-Transformer to maximize the fuzzy alignment score $p'_n(\theta, \boldsymbol{y})$ efficiently. However, it may bias the model to favor shorter translations. To this end, we adopt the idea of *brief penalty* (Papineni et al., 2002) for regularization. Specifically, we define the brief penalty of DAG based on the ratio of reference length to the expected translation length:

$$BP = \min(\exp(1 - \frac{T_{\boldsymbol{y}}}{\mathbb{E}_{\boldsymbol{y}'}[T_{\boldsymbol{y}'}]}), 1), \tag{19}$$

where $T_{\boldsymbol{y}}$ denotes the length of $\boldsymbol{y}$. As the expected translation length is equal to the expected number of 1-grams in outputs, it is convenient to calculate $\mathbb{E}_{\boldsymbol{y}'}[T_{\boldsymbol{y}'}]$ efficiently with the help of Equation 17. Finally, we train the model with the loss:

$$\mathcal{L} = -BP \times p'_n(\theta, \boldsymbol{y}). \tag{20}$$

We first pretrain DA-Transformer with the NLL loss to provide a good initialization, and then finetune the model with our fuzzy alignment objective.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets We conduct experiments on two major benchmarks that are widely used in previous studies: WMT14 English↔German (EN↔DE, 4M) and WMT17 Chinese↔English (ZH↔EN, 20M). For WMT EN↔DE, we use *newstest2013* as the validation set and *newstest2014* as the test set. For WMT ZH↔EN, we use *devtest2017* as the validation set and *newstest2017* as the test set. We apply BPE (Sennrich et al., 2016) to learn a joint subword vocabulary for EN↔DE and separate vocabularies for ZH↔EN on the tokenized data. For fair comparison, we evaluate our method with SacreBLEU (Post, 2018)[3] for WMT ZH-EN task and tokenized BLEU (Papineni et al., 2002) for other benchmarks. Considering BLEU may be biased, we also measure the models with BERTScore (Zhang et al., 2020). The decoding speedup is measured with a batch size of 1.

Implementation Details We adopt Transformer-*base* (Vaswani et al., 2017) as our autoregressive baseline. For the architecture of our model, we strictly follow the settings of DA-Transformer, where we set the decoder length to 8 times the source length ($\lambda = 8$) and use graph positional embeddings as decoder inputs unless otherwise specified. During both pretraining and finetuning, we set dropout rate to 0.1, weight decay to 0.01 and no label smoothing is applied. On pretraining, all models are trained for 300k updates with the batch size of 64k tokens. The learning rate warms up to $5 \cdot 10^{-4}$ within 10k steps. In finetuning, we use the batch size of 192k tokens to stabilize the gradients and train models for 5k updates. The learning rate warms up to $2 \cdot 10^{-4}$ on WMT EN↔DE within 500 steps and to $3 \cdot 10^{-4}$ on WMT ZH↔EN within 700 steps. We evaluate BLEU scores on the validation set and average the best 5 checkpoints for the final model. We implement our models with open-source toolkit `fairseq` (Ott et al., 2019). All the experiments are conducted on GeForce RTX 3090 GPUs.

Glancing Glancing (Qian et al., 2021a) is a promising strategy to alleviate data multimodality while training. We apply the same glancing strategy as Huang et al. (2022b), which assigns the decoder vertex inputs with the most probable path: $\widetilde{\boldsymbol{a}} = \arg\max_{\boldsymbol{a}} P_\theta(\boldsymbol{a}|\boldsymbol{x}, \boldsymbol{y})$. In the pretraining, we linearly anneal the unmasking ratio τ from 0.5 to 0.1. In the finetuning, we fix τ to 0.1.

Decoding We apply beam search with beam size of 5 for autoregressive baselines and argmax decoding for vanilla NAT to generate translations. Following Huang et al. (2022b), we find the translation of DAG with *Greedy* and *Lookahead* decoding algorithm. The former only searches paths greedily and then uses argmax decoding on vertices on the path:

$$a_i^* = \arg\max_{a_i} P_\theta(a_i|a_{i-1}, \boldsymbol{x}),\ y_i^* = \arg\max_{y_i} P_\theta(y_i|a_i, \boldsymbol{x}). \tag{21}$$

And the latter jointly searches the transitions and tokens in a greedy way:

$$a_i^*, y_i^* = \arg\max_{a_i, y_i} P_\theta(y_i|a_i, \boldsymbol{x}) P_\theta(a_i|a_{i-1}, \boldsymbol{x}). \tag{22}$$

Parallel argmax operations can be performed to obtain tokens and transitions for all vertices. The only needed sequential operation is collecting tokens along the chosen path. Both decoding methods can realize a comparable speedup with fully NAT models.

4.2 PRELIMINARY EXPERIMENT: EFFECTS OF n

We first study the effects of n on our fuzzy alignment objective. In this experiment, we set the decoder length to 4 times the source length ($\lambda = 4$) and apply Lookahead decoding to generate outputs. We compare different settings of n and report BLEU scores and BERTScores on the raw WMT14 EN-DE dataset in Table 1.

As shown in Table 1, the performance of our fuzzy alignment objective differs as n varies. We find that using a large n is helpful for DAG to generate better translations. Specifically, it improves the

[3]SacreBLEU signature: BLEU+case.mixed+numrefs.1+smooth.exp+tok.zh+version.1.5.1

Table 1: Effects of n on our fuzzy alignment objective on **raw** WMT14 EN-DE. Decoder length is set to 4 times source length ($\lambda = 4$).

	BLEU	**BERTScore**
DA-Transformer	26.16	60.57
+1-gram	25.71	60.69
+2-gram	26.98	61.68
+3-gram	26.81	61.56

Table 2: BLEU scores on **raw** WMT14 EN↔DE and WMT17 ZH↔EN dataset. Results of NAT baselines are quoted from Huang et al. (2022b). * and ** indicate the improvement over DA-Transformer is statistically significant ($p < 0.05$ and $p < 0.01$, respectively).

Model		**Iter.**	**Speedup**	**WMT14**		**WMT17**	
				EN-DE	**DE-EN**	**ZH-EN**	**EN-ZH**
Transformer(Vaswani et al., 2017)		N	1.0×	27.54	31.55	24.23	35.19
CMLM(Ghazvininejad et al., 2019)		10	2.2×	24.61	29.40	-	-
SMART(Ghazvininejad et al., 2020b)		10	2.2×	25.10	29.58	-	-
DisCo(Kasai et al., 2020)		10	3.5×	25.64	-	-	-
Imputer(Saharia et al., 2020)		8	2.7×	25.00	-	-	-
CMLMC(Huang et al., 2022c)		10	1.7×	26.40	30.92	-	-
Vanilla-NAT (Gu et al., 2018)		1	15.3×	11.79	16.27	8.69	18.92
CTC(Libovický & Helcl, 2018)		1	14.6×	18.42	23.65	12.23	26.84
AXE (Ghazvininejad et al., 2020a)		1	14.2×	20.40	24.90	-	-
GLAT (Qian et al., 2021a)		1	15.3×	19.42	26.51	18.88	29.79
OaXE (Du et al., 2021)		1	14.2×	22.40	26.80	-	-
CTC + GLAT (Qian et al., 2021b)		1	14.6×	25.02	29.14	19.92	30.65
CTC + DSLP (Huang et al., 2022a)		1	14.0×	24.81	28.33	-	-
DA-Transformer	+ *Greedy*	1	14.2×	26.08	30.48	22.66	33.27
(Huang et al., 2022b)	+ *Lookahead*	1	14.0×	26.57	30.68	22.82	33.83
DA-Transformer	+ *Greedy*	1	14.2×	26.07	30.69	22.35	33.58
(Our implementation)	+ *Lookahead*	1	14.0×	26.56	30.81	22.65	33.62
FA-DAT	+ *Greedy*	1	14.2×	27.49**	31.36**	23.78**	33.97*
	+ *Lookahead*	1	14.0×	**27.53****	**31.37****	**23.81****	**34.02****

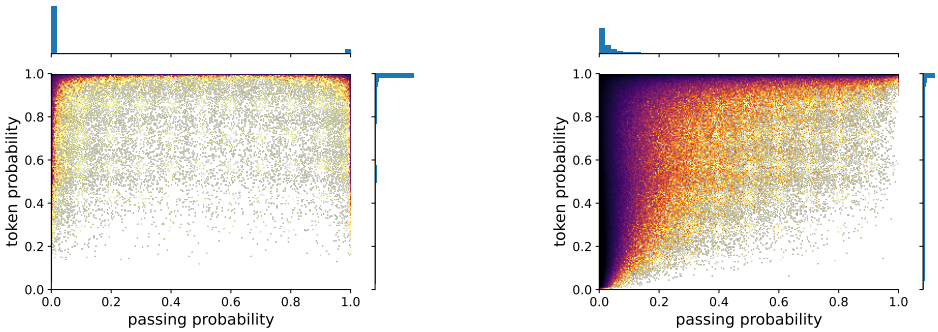
DA-Transformer baseline by 0.82 BLEU and 1.11 BERTScore when $n = 2$ and by 0.65 BLEU and 0.99 BERTScore when $n = 3$. We speculate the reason for performance degradation with $n = 1$ is that the 1-gram alignment objective thoroughly breaks the order dependency among tokens in a sentence, which will encourage the model to output bag-of-words instead of a fluent sentence. It is also noteworthy that 2-gram alignment performs better than 3-gram. We attribute the reason to the Markov property of DAG, which only models the dependency between adjacent tokens, thus making the 2-gram alignment more appropriate. In the following experiments, we will apply $n = 2$ as the default setting and name it as **F**uzzy **A**ligned **D**irected **A**cyclic **T**ransformer (FA-DAT).

4.3 Main Results

We compare our FA-DAT with the AT baseline and previous NAT approaches in Table 2 and 3. As shown in Table 2, FA-DAT substantially improves the translation quality of the DA-Transformer baseline by a large margin under both Greedy and Lookahead decoding. Notably, it achieves comparable performance with the autoregressive baseline without knowledge distillation and beam search decoding. The average gap between AT and NAT model on raw data is further reduced to 0.09 BLEU on WMT EN↔DE and 0.80 BLEU on WMT ZH↔EN, while maintaining 14.0 times decoding speedup. We observe similar trending in Table 3. Compared with DA-Transformer, FA-DAT achieves the improvement of 1.09 BERTScore on average under Lookahead decoding, which further demonstrates its effectiveness.

Table 3: BERTScores on **raw** WMT14 EN↔DE and WMT17 ZH↔EN dataset.

		EN-DE	DE-EN	ZH-EN	EN-ZH
DA-Transformer	+*Greedy*	60.86	66.91	56.70	61.64
	+*Lookahead*	60.89	67.00	56.97	61.62
FA-DAT	+*Greedy*	62.16	67.48	58.50	62.80
	+*Lookahead*	62.18	67.50	58.51	62.66



(a) Vertex distribution of FA-DAT

(b) Vertex distribution of DA-Transformer

Figure 1: The distribution of vertices' passing probabilities and max token probabilities on test set of WMT14 EN-DE. Passing probability and max token probability refer to the probability of a vertex appearing on a sampled path and the probability of its most probable token respectively. Darker area indicates dense distribution of vertices. Marginal distributions are given on the top and right.

4.4 Analysis and Discussion

Generation Confidence We note that the performance of Greedy decoding is on par with that of Lookahead decoding in FA-DAT while left far behind in DA-Transformer in Table 2. To conduct the analysis, we collect the probabilities of path ($P_\theta(\boldsymbol{a}|\boldsymbol{x})$) and translation given path ($P_\theta(\boldsymbol{y}|\boldsymbol{a},\boldsymbol{x})$) generated by both Greedy and Lookahead decoding and further calculate the probability of generated translation ($P_\theta(\boldsymbol{y}|\boldsymbol{x})$) via dynamic programming. Statistics are shown in Table 4. We observe that $-\log P_\theta(\boldsymbol{y}|\boldsymbol{a},\boldsymbol{x})$ is extremely low in FA-DAT, suggesting every vertex in DAG is assigned with a particular token with high confidence, making Greedy and Lookahead decoding comparable. Additionally, paths searched in FA-DAT has a larger probability and the model is more confident about its output translation. We owe it to FA-DAT's ability to well calibrate vertices in different modalities. It can be verified by checking the distribution of vertices' passing probabilities and max token probabilities in Figure 1. Vertices of FA-DAT have passing probabilities either close to 0 or to 1 and token probabilities all close to 1. It shows fuzzy alignment objective reduces model perplexity and improves generation confidence.

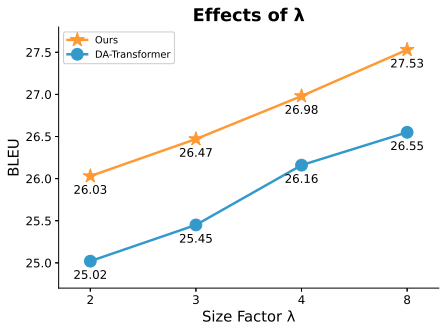


Figure 2: Effects of λ on **raw** WMT14 EN-DE. Decoder length is set to λ times source length.

Effects of Graph Size We further study how the size of the graph affects our method. We vary the graph size λ from 2 to 8 and measure the performance of the proposed FA-DAT and DA-Transformer baseline with BLEU on WMT14 EN-DE. As shown in Figure 2, FA-DAT consistently outperforms DA-Transformer by 0.96 BLEU on average across all sizes. Notably, FA-DAT does not necessarily

Table 4: Statistics of translations on **raw** WMT14 EN-DE dataset.

		$-\log P_\theta(\boldsymbol{a}\|\boldsymbol{x})$	$-\log P_\theta(\boldsymbol{y}\|\boldsymbol{a},\boldsymbol{x})$	$-\log P_\theta(\boldsymbol{y}\|\boldsymbol{x})$
DA-Transformer	+*Greedy*	10.27	4.51	11.47
	+*Lookahead*	10.49	3.84	11.06
FA-DAT	+*Greedy*	2.99	1.03	2.03
	+*Lookahead*	3.00	1.00	2.02

require a larger graph size to perform competitively. It makes use of vertices in DAG more efficiently, realizing comparable performance with $\lambda = 3$ to DA-Transformer with $\lambda = 8$ (26.47 vs 26.55). We attribute the success to that fuzzy alignment considers all translation modalities during training. Vertices corresponding to tokens in all modalities are incorporated in the backward flow with non-negligible gradients. Oppositely, NLL loss only trains paths that are strictly aligned with the reference. As shown in Figure 1, most vertices' token probabilities are close to 1 in FA-DAT while are scattered in DA-Transformer. It suggests all the vertices in DAG are well calibrated with the fuzzy alignment objective. Since model complexity is quadratic to graph size, massive overhead will be introduced with a larger graph size. FA-DAT achieves a better trade-off between graph size and translation quality, making the structure of DAG more appealing in practical application.

5 Related Work

Gu et al. (2018) first proposes non-autoregressive translation to reduce the decoding latency. Compared with the autoregressive counterpart, NAT suffers from the performance degradation due to the multi-modality problem and heavily relies on knowledge distilled from AT (Kim & Rush, 2016; Zhou et al., 2020; Tu et al., 2020; Ding et al., 2021; Shao et al., 2022). To mitigate the performance gap, some researchers introduce semi-autoregressive decoding (Wang et al., 2018; Ran et al., 2020; Ghazvininejad et al., 2020b; Wang et al., 2021) or iterative decoding mechanisms to refine NAT outputs (Lee et al., 2018; Ghazvininejad et al., 2019; Gu et al., 2019; Kasai et al., 2020; Saharia et al., 2020; Huang et al., 2022c). However, these approaches inevitably weakens the advantage of fast inference (Kasai et al., 2021). To this end, some researchers focus on improving the training procedure of fully NAT models. Qian et al. (2021a) introduces glancing training to NAT, which helps the model to calibrate outputs by feeding partial targets. Huang et al. (2022a) further applies a similar idea to middle layers with deep supervision. In another direction, researchers are looking for flexible training objectives that alleviate strict position-wise alignment required by the naive NLL loss. Libovický & Helcl (2018) proposes latent alignment model with CTC loss (Graves et al., 2006). Wang et al. (2019) introduces two auxiliary regularization terms to improve the quality of decoder hidden representations. Shao et al. (2020; 2021) introduce sequence-level training objectives with reinforcement learning and bag-of-ngrams difference. Ghazvininejad et al. (2020a) trains NAT model with the best monotonic alignment found by dynamic programming and Du et al. (2021) further extends it to order-agnostic cross entropy loss. Recently, Huang et al. (2022b) introduces DA-Transformer, which alleviates multi-modality by modeling dependency among vertices in DAG. Despite the success of DA-Transformer, it still implicitly requires monotonic one-to-one alignment between target tokens and vertices in the training, which weakens its ability to handle multi-modality.

6 Conclusion

In this paper, we introduce a fuzzy alignment objective between the directed acyclic graph and reference based on n-gram matching. Our proposed objective can better handle predictions with position shift, word reordering or length variation, which are critical sources of translation multi-modality. Experiments demonstrate that our method facilitates training of DA-Transformer, achieving comparable performance to autoregressive baseline with fully parallel decoding and sets new state of the art for NAT on the raw training data.

REFERENCES

Liang Ding, Longyue Wang, Xuebo Liu, Derek F. Wong, Dacheng Tao, and Zhaopeng Tu. Understanding and improving lexical choice in non-autoregressive translation. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=ZTFeSBIX9C.

Cunxiao Du, Zhaopeng Tu, and Jing Jiang. Order-agnostic cross entropy for non-autoregressive machine translation. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 2849–2859. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/du21c.html.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 6112–6121, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1633. URL https://aclanthology.org/D19-1633.

Marjan Ghazvininejad, Vladimir Karpukhin, Luke Zettlemoyer, and Omer Levy. Aligned cross entropy for non-autoregressive machine translation. In *International Conference on Machine Learning*, pp. 3515–3523. PMLR, 2020a.

Marjan Ghazvininejad, Omer Levy, and Luke Zettlemoyer. Semi-autoregressive training improves mask-predict decoding. *CoRR*, abs/2001.08785, 2020b. URL https://arxiv.org/abs/2001.08785.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pp. 369–376, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933832. doi: 10.1145/1143844.1143891. URL https://doi.org/10.1145/1143844.1143891.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1l8BtlCb.

Jiatao Gu, Changhan Wang, and Junbo Zhao. Levenshtein transformer. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 11179–11189, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/675f9820626f5bc0afb47b57890b466e-Abstract.html.

Chenyang Huang, Hao Zhou, Osmar R. Zaïane, Lili Mou, and Lei Li. Non-autoregressive translation with layer-wise prediction and deep supervision. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pp. 10776–10784. AAAI Press, 2022a. URL https://ojs.aaai.org/index.php/AAAI/article/view/21323.

Fei Huang, Hao Zhou, Yang Liu, Hang Li, and Minlie Huang. Directed acyclic transformer for non-autoregressive machine translation. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9410–9428. PMLR, 2022b. URL https://proceedings.mlr.press/v162/huang22m.html.

Xiao Shi Huang, Felipe Perez, and Maksims Volkovs. Improving non-autoregressive translation models without distillation. In *International Conference on Learning Representations*, 2022c. URL https://openreview.net/forum?id=I2Hw58KHp8O.

Jungo Kasai, James Cross, Marjan Ghazvininejad, and Jiatao Gu. Non-autoregressive machine translation with disentangled context transformer. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5144–5155. PMLR, 2020. URL http://proceedings.mlr.press/v119/kasai20a.html.

Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah A. Smith. Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=KpfasTaLUpq.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In Jian Su, Xavier Carreras, and Kevin Duh (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pp. 1317–1327. The Association for Computational Linguistics, 2016. doi: 10.18653/v1/d16-1139. URL https://doi.org/10.18653/v1/d16-1139.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1173–1182, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1149. URL https://www.aclweb.org/anthology/D18-1149.

Jindřich Libovický and Jindřich Helcl. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3016–3021, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1336. URL https://aclanthology.org/D18-1336.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. Facebook fair's WMT19 news translation task submission. In Ondrej Bojar, Rajen Chatterjee, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno-Yepes, Philipp Koehn, André Martins, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana L. Neves, Matt Post, Marco Turchi, and Karin Verspoor (eds.), *Proceedings of the Fourth Conference on Machine Translation, WMT 2019, Florence, Italy, August 1-2, 2019 - Volume 2: Shared Task Papers, Day 1*, pp. 314–319. Association for Computational Linguistics, 2019. doi: 10.18653/v1/w19-5333. URL https://doi.org/10.18653/v1/w19-5333.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In Waleed Ammar, Annie Louis, and Nasrin Mostafazadeh (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Demonstrations*, pp. 48–53. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-4009. URL https://doi.org/10.18653/v1/n19-4009.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040.

Matt Post. A call for clarity in reporting BLEU scores. In Ondrej Bojar, Rajen Chatterjee, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno-Yepes, Philipp Koehn, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana L. Neves, Matt Post, Lucia Specia, Marco Turchi, and Karin Verspoor (eds.), *Proceedings of the Third Conference on Machine Translation: Research Papers, WMT 2018, Belgium, Brussels, October 31 - November 1, 2018*, pp. 186–191. Association for Computational Linguistics, 2018. doi: 10.18653/v1/w18-6319. URL https://doi.org/10.18653/v1/w18-6319.

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

59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 1993–2003, Online, August 2021a. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.155. URL https://aclanthology.org/2021.acl-long.155.

Lihua Qian, Yi Zhou, Zaixiang Zheng, Yaoming Zhu, Zehui Lin, Jiangtao Feng, Shanbo Cheng, Lei Li, Mingxuan Wang, and Hao Zhou. The volctrans GLAT system: Non-autoregressive translation meets WMT21. In *Proceedings of the Sixth Conference on Machine Translation*, pp. 187–196, Online, November 2021b. Association for Computational Linguistics. URL https://aclanthology.org/2021.wmt-1.17.

Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. Learning to recover from multi-modality errors for non-autoregressive neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 3059–3069, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.277. URL https://www.aclweb.org/anthology/2020.acl-main.277.

Chitwan Saharia, William Chan, Saurabh Saxena, and Mohammad Norouzi. Non-autoregressive machine translation with latent alignments. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1098–1108, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.83. URL https://aclanthology.org/2020.emnlp-main.83.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016. doi: 10.18653/v1/p16-1162. URL https://doi.org/10.18653/v1/p16-1162.

Chenze Shao, Jinchao Zhang, Yang Feng, Fandong Meng, and Jie Zhou. Minimizing the bag-of-ngrams difference for non-autoregressive neural machine translation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 198–205. AAAI Press, 2020. URL https://ojs.aaai.org/index.php/AAAI/article/view/5351.

Chenze Shao, Yang Feng, Jinchao Zhang, Fandong Meng, and Jie Zhou. Sequence-Level Training for Non-Autoregressive Neural Machine Translation. *Computational Linguistics*, pp. 1–35, 10 2021. ISSN 0891-2017. doi: 10.1162/coli_a_00421. URL https://doi.org/10.1162/coli_a_00421.

Chenze Shao, Xuanfu Wu, and Yang Feng. One reference is not enough: Diverse distillation with reference selection for non-autoregressive translation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3779–3791, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.277. URL https://aclanthology.org/2022.naacl-main.277.

Lifu Tu, Richard Yuanzhe Pang, Sam Wiseman, and Kevin Gimpel. ENGINE: Energy-based inference networks for non-autoregressive machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2819–2826, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.251. URL https://aclanthology.org/2020.acl-main.251.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Chunqi Wang, Ji Zhang, and Haiqing Chen. Semi-autoregressive neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 479–488, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1044. URL https://www.aclweb.org/anthology/D18-1044.

Qiang Wang, Heng Yu, Shaohui Kuang, and Weihua Luo. Hybrid-regressive neural machine translation, 2021. URL https://openreview.net/forum?id=jYVY_piet7m.

Yiren Wang, Fei Tian, Di He, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. Non-autoregressive machine translation with auxiliary regularization. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, 2019*, pp. 5377–5384. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33015377. URL https://doi.org/10.1609/aaai.v33i01.33015377.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=SkeHuCVFDr.

Chunting Zhou, Jiatao Gu, and Graham Neubig. Understanding knowledge distillation in non-autoregressive machine translation. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BygFVAEKDH.

A PROOF OF EQUATION 13 AND EQUATION 14

We denote the length of arbitrary translation $\boldsymbol{y}'$ and that of path $\boldsymbol{a}$ as $T_{\boldsymbol{y}'}$ and $T_{\boldsymbol{a}}$ respectively. Note that the averaged length of generated translations should be the same as the averaged length of DAG paths:

$$\sum_{\boldsymbol{y}'} P_\theta(\boldsymbol{y}'|\boldsymbol{x}) T_{\boldsymbol{y}'} = \sum_{\boldsymbol{a}} P_\theta(\boldsymbol{a}|\boldsymbol{x}) T_{\boldsymbol{a}}. \tag{23}$$

Then we have:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{y}'}[\sum_{\boldsymbol{g}\in G_n(\boldsymbol{y}')} C_{\boldsymbol{g}}(\boldsymbol{y}')] &= \sum_{\boldsymbol{y}'} P_\theta(\boldsymbol{y}'|\boldsymbol{x}) \sum_{\boldsymbol{g}\in G_n(\boldsymbol{y}')} C_{\boldsymbol{g}}(\boldsymbol{y}') \\
&= \sum_{\boldsymbol{y}'} P_\theta(\boldsymbol{y}'|\boldsymbol{x})(T_{\boldsymbol{y}'} - n + 1) \\
&= \sum_{\boldsymbol{a}} P_\theta(\boldsymbol{a}|\boldsymbol{x})(T_{\boldsymbol{a}} - n + 1) \\
&= \sum_{\boldsymbol{a}} P_\theta(\boldsymbol{a}|\boldsymbol{x}) \sum_{v_1,...,v_n} \mathbb{1}(v_1, ..., v_n \in \boldsymbol{a}) \\
&= \sum_{v_1,...,v_n} P(\mathbb{1}(v_1, ..., v_n)|\boldsymbol{x}),
\end{aligned} \tag{24}$$

We can handle $\mathbb{E}_{\boldsymbol{y}'}[C_{\boldsymbol{g}}(\boldsymbol{y}')]$ in a similar way. When a particular path $\boldsymbol{a}$ is given, it is convenient to calculate the averaged appearance of n-gram g through sliding a window of size n on token distributions along the path (Shao et al., 2020):

$$\mathbb{E}_{\boldsymbol{y}'|\boldsymbol{a}}[C_{\boldsymbol{g}}(\boldsymbol{y}')] = \sum_{\boldsymbol{y}'} P_\theta(\boldsymbol{y}'|\boldsymbol{a}, \boldsymbol{x}) C_{\boldsymbol{g}}(\boldsymbol{y}') = \sum_{i=1}^{T_a-n+1} \prod_{j=1}^{n} P_\theta(g_j|a_{i+j-1}). \tag{25}$$

Then we apply the indicator function $\mathbb{1}(v_1, ..., v_n \in \boldsymbol{a})$ to transform the enumerating space from the space of n adjacent vertices in path $\boldsymbol{a}$ to the space of arbitrary n vertices in graph.

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{y}'}[C_{\boldsymbol{g}}(\boldsymbol{y}')] &= \sum_{\boldsymbol{a}} P_\theta(\boldsymbol{a}|\boldsymbol{x}) \mathbb{E}_{\boldsymbol{y}'|\boldsymbol{a}}[C_{\boldsymbol{g}}(\boldsymbol{y}')] \\
&= \sum_{\boldsymbol{a}} P_\theta(\boldsymbol{a}|\boldsymbol{x}) \sum_{i=1}^{T_a-n+1} \prod_{j=1}^{n} P_\theta(g_j|a_{i+j-1}) \\
&= \sum_{\boldsymbol{a}} P_\theta(\boldsymbol{a}|\boldsymbol{x}) \sum_{v_1,...,v_n} \mathbb{1}(v_1, ..., v_n \in \boldsymbol{a}) \prod_{j=1}^{n} P_\theta(g_j|v_j) \\
&= \sum_{v_1,...,v_n} \sum_{\boldsymbol{a}} P_\theta(\boldsymbol{a}|\boldsymbol{x}) \mathbb{1}(v_1, ..., v_n \in \boldsymbol{a}) \prod_{j=1}^{n} P_\theta(g_j|v_j) \\
&= \sum_{v_1,...,v_n} [\sum_{\boldsymbol{a}} P_\theta(\boldsymbol{a}|\boldsymbol{x}) \mathbb{1}(v_1, ..., v_n \in \boldsymbol{a})] \prod_{j=1}^{n} P_\theta(g_j|v_j) \\
&= \sum_{v_1,...,v_n} P(\mathbb{1}(v_1, ..., v_n)) \prod_{j=1}^{n} P_\theta(g_j|v_j)
\end{aligned} \tag{26}$$

We can find $\mathbb{E}_{\boldsymbol{y}'}[C_{\boldsymbol{g}}(\boldsymbol{y}')]$ is actually a weighted sum of passing probabilities of n-vertex subpaths.

B PROOF OF EQUATION 18

On the basis of Equation 14 and Equation 15, we first rewrite it as:

$$\mathbb{E}_{\boldsymbol{y}'}[C_{\boldsymbol{g}}(\boldsymbol{y}')] = \sum_{v_1,...,v_n} P(\mathbb{1}(v_1)|\boldsymbol{x}) \prod_{i=1}^{n-1} \mathbf{E}_{v_i,v_{i+1}} \prod_{i=1}^{n} P_\theta(g_i|v_i) \tag{27}$$

We can construct a similar form to Equation 17 by commutating and associating terms in the product:

$$\begin{aligned}\mathbb{E}_{\boldsymbol{y}'}[C_{\boldsymbol{g}}(\boldsymbol{y}')] &= \sum_{v_1,\dots,v_n} P(\mathbb{1}(v_1)|\boldsymbol{x}) \prod_{i=1}^{n-1} \mathbf{E}_{v_i,v_{i+1}} \prod_{i=1}^{n} P_\theta(g_i|v_i) \\ &= \sum_{v_1,\dots,v_n} [P(\mathbb{1}(v_1)|\boldsymbol{x}) P_\theta(g_1|v_1)] \prod_{i=1}^{n-1} [\mathbf{E}_{v_i,v_{i+1}} P_\theta(g_{i+1}|v_{i+1})].\end{aligned} \tag{28}$$

Equation 28 indicates that the weighted sum of passing probability shares the same form with the sum of $(n-1)$-hop transition probabilities of a Markov chain with unnormalized initial probability and step-variant transition matrix. The initial probability and the ith step transition probability are given as:

$$\begin{cases}\tilde{P}(\mathbb{1}(v)|\boldsymbol{x}) = P(\mathbb{1}(v)|\boldsymbol{x}) P_\theta(g_1|v) \\ \tilde{\mathbf{E}}^{(i)}_{u,v} = \tilde{\mathbf{E}}^{(i)}_{u,v} P_\theta(g_{i+1}|v)\end{cases} \tag{29}$$

With passing probability vector $\boldsymbol{p}$ and prediction matrix $\mathbf{G}$ introduced in Section 3.3, above equations be expressed in matrix form:

$$\begin{cases}\tilde{\boldsymbol{p}}^T = \boldsymbol{p}^T \odot \mathbf{G}_{1,:} \\ \tilde{\mathbf{E}}^{(i)} = \mathbf{E} \odot \mathbf{G}_{i+1,:}\end{cases} \tag{30}$$

where $\mathbf{G}_{i+1,:}$ should be broadcast to the size of $\mathbf{E}$. Then we can obtain the expression of Equation 28 in matrix form based on the sum of $(n-1)$-hop transition probabilities in Markov chain:

$$\begin{aligned}\mathbb{E}_{\boldsymbol{y}'}[C_{\boldsymbol{g}}(\boldsymbol{y}')] &= \|\tilde{\boldsymbol{p}}^T \cdot \prod_{i}^{n-1} \tilde{\mathbf{E}}^{(i)}\|_1 \\ &= \|(\boldsymbol{p}^T \odot \mathbf{G}_{1,:}) \cdot (\mathbf{E} \odot \mathbf{G}_{2,:}) \cdot \dots \cdot (\mathbf{E} \odot \mathbf{G}_{n,:})\|_1.\end{aligned} \tag{31}$$

Considering $\mathbf{G}_{i+1,:}$ is broadcast across rows, removing parentheses in Equation 31 will not change the results:

$$\mathbb{E}_{\boldsymbol{y}'}[C_{\boldsymbol{g}}(\boldsymbol{y}')] = \|\boldsymbol{p}^T \odot \mathbf{G}_{1,:} \cdot \mathbf{E} \odot \mathbf{G}_{2,:} \cdot \dots \cdot \mathbf{E} \odot \mathbf{G}_{n,:}\|_1. \tag{32}$$

C Fluency Analysis

We argue that FA-DAT improves NAT performance by better capturing multi-modality with the fuzzy alignment objective. It helps the model avoid generating a mixture of different translations. To demonstrate that, we measure the fluency of outputs from different models. The perplexity (PPL) score is calculated by a pretrained language model (Ng et al., 2019)[4] with context window size 128. Lookahead decoding is applied to DA-Transformer and FA-DAT.

Table 5: Perplexity scores on **raw** WMT14 EN-DE dataset.

	Vanilla-NAT	**DA-Transformer**	**FA-DAT**	**AT**	**Reference**
PPL	1909.9	495.8	468.6	411.7	395.8

As shown in Table 5, AT achieves comparable fluency with gold reference while vanilla-NAT suffers a significant drop due to the disability to handle multi-modality. DA-Transformer improves NAT fluency by a large margin and proposed FA-DAT further reduces the fluency gap between AT and NAT.

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

D RESULTS ON DISTILLED DATASET

Though FA-DAT is proposed to deal with the multi-modality problem which is severe on raw training data, we also conduct experiments to evaluate the performance of FA-DAT on distilled data, where the data distribution is much simplified. Following Gu et al. (2018), we use Transformer-*base* as the teacher model to generate the distilled dataset. We apply Lookahead decoding to both DA-Transformer and FA-DAT and report the results in Table 6.

Table 6: BLEU scores on **raw** and **distilled** WMT14 EN-DE dataset.

	DA-Transformer	**FA-DAT**
Raw	26.56	27.53
Distilled	26.94	27.17

We find that FA-DAT consistently outperforms DA-Transformer on both settings of training data. However, the improvement on the distilled dataset is relatively marginal. We attribute it to that multi-modality in distilled data is already alleviated due to the simplification by the autoregressive teacher. Interestingly, we note that FA-DAT performs better on raw training data by a margin of 0.36 BLEU, which shows a distinct feature in contrast to NAT models in the literature. We argue that traditional sequence-level KD limits the performance of NAT model by imposing an upper bound (AT's performance). It demonstrates that NAT model endowed with the ability to handle data multi-modality can benefit more from training on authentic data, showing the potential of further developing stronger NAT models.