Directed Acyclic Transformer for Non-Autoregressive Machine Translation

Fei Huang [1 2 *] **Hao Zhou** [3] **Yang Liu** [2] **Hang Li** [3] **Minlie Huang** [1 2]

Abstract

Non-autoregressive Transformers (NATs) significantly reduce the decoding latency by generating all tokens in parallel. However, such independent predictions prevent NATs from capturing the dependencies between the tokens for generating multiple possible translations. In this paper, we propose Directed Acyclic Transfomer (DA-Transformer), which represents the hidden states in a Directed Acyclic Graph (DAG), where each path of the DAG corresponds to a specific translation. The whole DAG simultaneously captures multiple translations and facilitates fast predictions in a non-autoregressive fashion. Experiments on the raw training data of WMT benchmark show that DA-Transformer substantially outperforms previous NATs by about 3 BLEU on average, which is the first NAT model that achieves competitive results with autoregressive Transformers without relying on knowledge distillation.

1. Introduction

Transformer has been the most popular architecture for sequence-to-sequence learning, especially for machine translation (Vaswani et al., 2017). Vanilla Transformer adopts the autoregressive approach for generation, which obtains strong results but is inefficient in inference due to its sequential decoding. To tackle the problem, Non-autoregressive Transformers (NATs, Gu et al., 2018; Gu et al., 2019; Ma et al., 2019;Ding et al., 2021a; Gu & Kong, 2021) have been proposed, which significantly reduce the inference latency by predicting all tokens in parallel and achieve reasonably high performances in translation. Notably, an NAT-based system obtain the highest BLEU score in German to English translation of WMT21 (Qian et al., 2021b; Akhbardeh et al., 2021), even better than a line of Autoregressive Transformer (AT) systems.

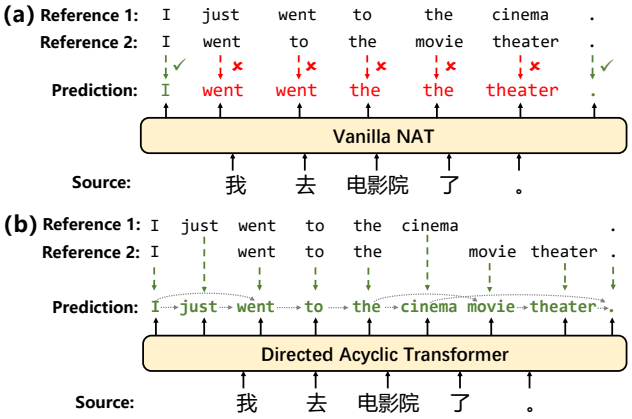

Figure 1. (a) The multi-modality problem in vanilla NAT. Multiple possible references provide inconsistent labels at some positions, leading to an implausible prediction that mixes several translations. (b) The proposed *DA-Transformer*. Tokens from different translations are assigned to distinct vertices to avoid the inconsistent labels. In inference, we follow the transitions to recover the output translation.

However, current NATs severely suffer from the *multi-modality problem* (Gu et al., 2018) in both training and inference.[1] Intuitively, in training, as shown in Fig.1(a), NAT models are trained to predict each token independently, where one position may have several possible tokens as labels from several different translation references. In such a case, an NAT model may learn to generate an implausible output mixing multiple translations. Additionally, in inference, the NAT still cannot sample fluent translations even if it captures multi-modal information in training. Since the NAT model generates all tokens simultaneously, no effective sampling approach can be used on top of it. In contrast, ATs do not have the same problem because of their left-to-right generation, where the multi-modality problem for a later position is not so severe since its prefix has been given.

[*]This work is partially done during Fei Huang's internship at ByteDance AI Lab. [1]The CoAI group, Tsinghua University, China. [2]Institute for Artificial Intelligence, State Key Lab of Intelligent Technology and Systems, Beijing National Research Center for Information Science and Technology, Department of Computer Science and Technology, Tsinghua University, China. [3]ByteDance AI Lab. Correspondence to: Fei Huang <f-huang18@mails.tsinghua.edu.cn>, Hao Zhou <haozhou0806@gmail.com>, Yang Liu <liuyang2011@tsinghua.edu.cn>, Hang Li <lihang.lh@bytedance.com>, Minlie Huang <aihuang@tsinghua.edu.cn>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

[1]The *multi-modality* here refers to the fact that there are multiple possible translations for a single source sentence.

Currently, the main solution to address the multi-modality problem is to reduce the data modalities by knowledge distillation (KD, Kim & Rush, 2016; Gu et al., 2018), namely, replacing the original training targets with predicted sentences from an AT teacher. KD is simple yet effective, which always leads to significant BLEU improvements, e.g., about 8 BLEU points on WMT14 En-De for vanilla NATs.

However, we argue that current state-of-art NAT models heavily rely on KD, which has two crucial disadvantages. a) Training NAT models by distilling from AT makes the training process redundant. We need to train an AT model first and then regenerate the whole training data. Such complex pre-processing prevents NATs from being practically used. b) Generally, the student model in KD cannot outperform its teacher model with a large margin. In such a case, KD restricts NAT's performance by imposing an upper bound (*not strict*), which seriously hurts the potential of further developing NAT models.

In this paper, we propose *Directed Acyclic Transformer* (DA-Transformer) for Non-Autoregressive Machine Translation, which directly captures many translation modalities via a proposed *Directed Acyclic Decoder*, instead of indirectly reducing modalities by KD. Specifically, different from decoders of ATs or vanilla NATs, our proposed decoder organizes the hidden states as a Directed Acyclic Graph (DAG) rather than a sequence. As shown in Fig.1(b), the DAG has multiple paths, each of which corresponds to a specific sentence.[2] In training, the DAG structure enables DA-Transformer to capture multiple translation modalities simultaneously, which avoids the inconsistent labels in vanilla NAT training. In inference, it can generate sentences along predicted paths, which not only avoids incorrect outputs mixing multiple translations but also enables the generation of diverse translations by sampling different paths.

Notice that DA-Transformer predicts all translation words in parallel, and the whole model is trained in an end-to-end fashion, which enjoys all merits of NAT models. We propose an objective that does not require multiple references in training, making it applicable to most translation benchmarks. In inference, we propose several sampling methods to decode a translation from DA-Transformer, which provides flexible quality-latency tradeoff in generation.

Experimental results show that DA-Transformer significantly reduces the gap between NATs and ATs while preserving the inference latency (7x $\sim$ 14x speedup over ATs). Especially on WMT17 Zh-En, our best model outperforms autoregressive Transformer by 0.6 BLEU without the help of knowledge distillation. To our best knowledge, it is the first time that a *non-iterative* NAT model achieves competitive results with AT models without KD. DA-Transformer outperforms existing NATs (including iterative approaches) with a large margin on the raw data of standard En$\leftrightarrow$DE and En$\leftrightarrow$Zh benchmarks, which sufficiently shows the effectiveness of our proposed model.

[2]The DAG is similar to the concept of word lattice (Richardson et al., 1995). The words are represented by edges instead of vertices in the word lattice, and in contrast, each vertex of the DAG in our model represents a word distribution rather than concrete words.

2. Related Work

Non-autoregressive Machine Translation Gu et al. (2018) propose NAT models to reduce the latency in generation or decoding, but there exists a gap in translation quality between NAT and AT models. To bridge the gap, iterative NATs manage to repeatedly refine the generated outputs (Lee et al., 2018; Ghazvininejad et al., 2019; Guo et al., 2020). However, as shown in Kasai et al. (2021), most iterative NATs are not advantageous against ATs in the quality-latency tradeoff. Non-iterative NATs are much faster, whose improvements mainly come from alignment-based objectives (Libovický & Helcl, 2018; Ghazvininejad et al., 2020a; Du et al., 2021), or incorporating extra decoder inputs (Shu et al., 2020; Qian et al., 2021a; Bao et al., 2021). Nevertheless, these NATs heavily rely on knowledge distillation (KD, Gu et al., 2018), which is found very effective in reducing the data modalities (Zhou et al., 2020). A recent study (Huang et al., 2022b) provides a unified perspective showing that most existing methods actually modify targets or inputs to reduce the token dependencies in the data distribution, which eases the NAT training but introduces data distortion.

Unlike existing NATs, our method retains multiple translations instead of dropping the multi-modal information in NAT training. It turns out that our method can effectively tackle the multi-modality problem without modifying the training data and not rely on KD to achieve a good translation performance.

Lattice-based Model in Machine Translation *Word lattices* have a long history in Statistic Machine Translation. A word lattice is a directed acyclic graph (DAG) with edges labeled with a token and weight, which can represent an exponential number of sentences in the a compact structure. A phrase-based translation system can generate a word lattice during decoding (Ueffing et al., 2002; Och & Ney, 2004). Some models take word lattices as inputs to alleviate input errors brought by word segmentation or speech recognition (Dyer et al., 2008; Koehn et al., 2007; Dong et al., 2014). There are also studies that combine multiple system outputs into a single lattice (Rosti et al., 2007; Feng et al., 2009) and decode a good translation from it (Tromble et al., 2008).

Unlike previous studies that construct the word lattices with a search algorithm, our model predicts the whole DAG with multiple translations simultaneously. Moreover, DA-Transformer's training does not require ground-truth word

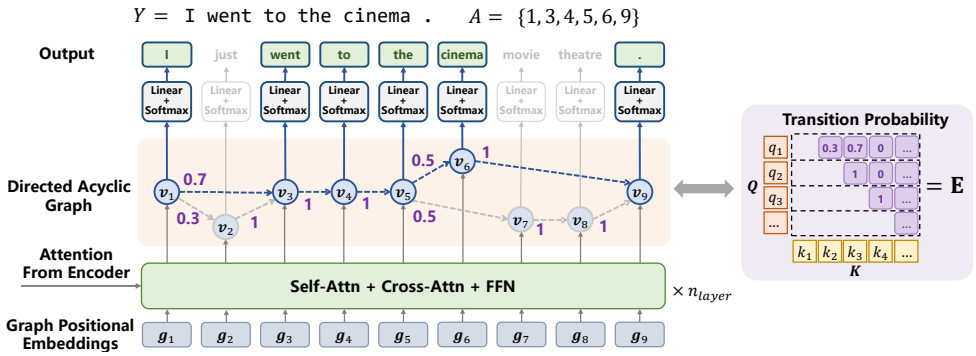

Figure 2. Overview of Directed Acyclic Decoder. The decoder organizes the hidden states $\mathbf{V} = [\mathbf{v}_1, \cdots, \mathbf{v}_L]^T$ in a directed acyclic graph (DAG) structure, which stores representations of multiple translations. The transition probabilities in $\mathbf{E}$ are predicted based on the vertex states. To generate one of the translations, a path A is first sampled from the DAG structure, and the selected vertices predict the whole sentence in parallel. The other vertices and their predicted tokens are skipped in the output.

lattices for supervision, making it applicable to most translation benchmarks.

3. Our Proposed Method

In this section, we describe our proposed DA-Transformer in detail. Intuitively, to facilitate the explicitly modeling of multiple modalities, we propose to replace the original Non-autoregressive Transformer decoder with a directed acyclic decoder, whose topological structure is a DAG. Each path of the DAG forms a sequence of hidden states that stores a possible translation, and the whole DAG store multiple translations in different paths. DA-Transformer still generates in a non-autoregressive fashion.

We will first introduce the network structures in Section 3.1, which presents how to construct the DA-Transformer to parameterize the conditional probability. Then in Section 3.2, we will elaborate on the training of DA-Transformer, including how to train it with one reference and the efficient implementation of traversing possible paths. Finally, in Section 3.3, we provide several decoding approaches, aiming to sample fluent sentences efficiently given the well-trained DA-Transformer.

3.1. Architecture of DA-Transformer

DA-Transformer consists of a Transformer encoder and a *directed acyclic decoder*. The encoder is the same as vanilla Transformer while the decoder organizes its hidden states as a DAG. As shown in Fig.2, hidden states correspond to vertices of the DAG, which model word distributions in specific positions; and edges of the DAG are transitions between hidden states, which organize generated words into a final sentence.

Intuitively, given a source sentence X, the directed acyclic decoder generates a sentence in three steps: (1) receiving the position embeddings as inputs and producing hidden states as vertices; (2) calculating the transition probabilities between the vertices based on the vertex states; (3) sampling a path from the DAG following the transitions, and then predicting target tokens using the vertex states on the path.

Formally, the probability of a target sentence $Y = \{y_1, y_2, \cdots, y_M\}$ is formulated as

$$P_\theta(Y|X) = \sum_{A \in \Gamma} P_\theta(Y, A|X) = \sum_{A \in \Gamma} P_\theta(A|X) P_\theta(Y|A, X), \quad (1)$$

where $A = \{a_1, a_2, \cdots, a_M\}$ is a path represented by a sequence of vertex indexes, and Γ contains all paths with the same length of the target sentence Y.

Vertex The directed acyclic decoder utilizes the Transformer layers (Vaswani et al., 2017) to predict the vertex states. Unlike the autoregressive decoder that generates tokens from left to right, it generates the vertex states in parallel.

Specifically, we use graph positional embeddings $\mathbf{G} = \{\mathbf{g}_1, \cdots, \mathbf{g}_L\}$ as the decoder inputs, which is identical to the learnable positional embeddings in vanilla Transformer but represents the vertex indexes instead of the token positions. Note that L is the graph size, where we set L to λ times the source length N and tune λ as a hyper-parameter. The decoder then produces the vertex states $\mathbf{V} = [\mathbf{v}_1, \cdots, \mathbf{v}_L]^T$, which is defined as

$$[\mathbf{v}_1, \cdots, \mathbf{v}_L] = \text{Transformer-Blocks}(\mathbf{g}_1, \cdots, \mathbf{g}_L).$$

Transition Each edge of the DAG is assigned the transition probability between the connecting vertices. The transition probabilities are locally normalized, i.e., the probabilities of outgoing edges sum to one. Formally, the probability of path A is defined as

$$P_\theta(A|X) = \prod_{i=1}^{M-1} P_\theta(a_{i+1}|a_i, X) = \prod_{i=1}^{M-1} \mathbf{E}_{a_i, a_{i+1}},$$

where $\mathbf{E} \in \mathbb{R}^{L\times L}$ is the transition matrix normalized by rows. Specifically, the transition matrix is obtained by

$$\mathbf{E} = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}), \tag{2}$$
$$\mathbf{Q} = \mathbf{V}\mathbf{W}_{\text{Q}}, \quad \mathbf{K} = \mathbf{V}\mathbf{W}_{\text{K}},$$

where d is the hidden size, $\mathbf{W}_{\text{Q}}$ and $\mathbf{W}_{\text{K}}$ are learnable parameters. To ensure that there is no cycle in the DAG, we apply lower triangular masking on $\mathbf{E}$, which only allows transitions from vertices with small indexes to large indexes. Note that the matrix $\mathbf{E}$ can be calculated in parallel, thereby facilitating fast sampling of paths.

Token Prediction Conditioned on the vertex states in $\mathbf{V}$ and the selected path A, the decoder predicts the target tokens in parallel. Formally, we have

$$P_\theta(Y|A,X) = \prod_{i=1}^{M} P_\theta(y_i|a_i,X) = \prod_{i=1}^{M} \text{softmax}(\mathbf{W}_{\text{P}}\mathbf{v}_{a_i}),$$

where $\mathbf{W}_{\text{P}}$ are learnable weights, and $\mathbf{v}_{a_i}$ is the representation of the i-th vertex on the path A.

In the implementation, we actually calculate the distributions on all vertices and then skip the vertices not appearing on the chosen path. Specifically, we obtain

$$\mathbf{P} = \text{softmax}(\mathbf{V}\mathbf{W}_{\text{P}}^T), \tag{3}$$

where $\mathbf{P} \in \mathbb{R}^{L\times|\mathbb{V}|}$ is the matrix containing the token distributions on the L vertices, and $P_\theta(y_i|a_i,X) = \mathbf{P}_{a_i,y_i}$. The matrix $\mathbf{P}$ facilitates fast calculation for multiple paths since the shared vertices are not calculated twice, which is significant in training and inference introduced later.

3.2. Training

To capture multiple translation modalities in training, the proposed directed acyclic decoder arranges words from different modalities in different vertex states of the decoder, which can effectively reduces the inconsistent problem in training. In this section, we will elaborate on training details of DA-Transformer, including training with one reference, efficient implementation of marginalizing all paths in the DAG, and modified glancing training techniques according to the graph structures.

Training DA-Transformer with One Reference Although DA-Transformer retains multiple translations in the DAG, its training objective only requires one reference per sample, which facilitates efficient training on most translation benchmarks. Specifically, it directly maximizes the log-likelihood $\log P(Y|X)$ by marginalizing all possible paths A, which can be formulated as follows,

$$\mathcal{L} = -\log P_\theta(Y|X) = -\log \sum_{A\in\Gamma} P_\theta(Y,A|X), \tag{4}$$

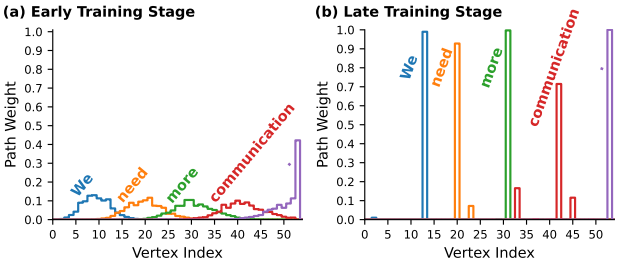

Figure 3. The cumulative weight w_A of paths that pass through each vertex, where lines represent token labels for the vertex.[3] The weights are sparse in the late stage of training, indicating that only several vertices are updated to fit the sample. Source: "我们 需要 更多 的 交流 。" Target: "*We need more communication* ."

where Γ contains all paths with $1 = a_1 < \cdots < a_M = L$.

To understand why a single reference is adequate for the DAG learning, we analyze the training process by inspecting the gradients. Intuitively, we find that the objective assigns a single reference to several paths, where the vertices on the chosen paths are updated to generate the reference tokens, and the other vertices remain unchanged. The sparse assignment is the key to the successful training, which avoids inconsistent labels in token predictions and preserves the unseen translations stored on the unchanged paths. In such a way, the DAG can be learned across different training instances, each of which only provides a single reference, not requiring an instance with multiple references.

Specifically, we inspect the gradient of $\mathcal{L}$ and find that

$$\frac{\partial}{\partial\theta}\mathcal{L} = \sum_{A\in\Gamma} w_A \left(\frac{\partial}{\partial\theta}\mathcal{L}_A\right), \tag{5}$$

where

$$\mathcal{L}_A = -\log P_\theta(Y,A|X), \tag{6}$$
$$w_A = \frac{P_\theta(Y,A|X)}{\sum_{A'\in\Gamma} P_\theta(Y,A'|X)}. \tag{7}$$

$\mathcal{L}_A$ maximizes the likelihood of sampling Y with the path A, and w_A is the weight of $\mathcal{L}_A$. Eq(5) indicates that the weights of paths are assigned according to the probability that Y appears on A. If a path A is more probable for the target Y, then a larger weight will be used in optimizing $\mathcal{L}_A$, which further strengthens its dominance. In contrast, an unlikely path A will get a negligible weight, indicating that the vertices on A are not affected in the update.

A real example is shown in Fig.3. In the early stage, training with one sample will affect all vertices in the DAG. In the late stage, only some vertices are updated, reserving the other vertices for storing unseen translations.

[3]For example, the orange line (the second token, *need*) on the vertex v_{20} is the sum of w_A for the paths $A = \{a_1, a_2, \cdots, a_5\}$ satisfying $a_2 = 20$. w_A is defined in Eq(7).

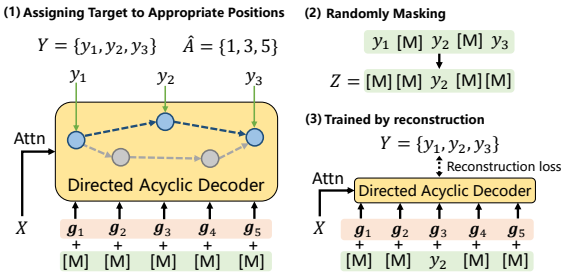

Figure 4. Glancing training for DA-Transformer, which is similar to the masked language model for promoting representation learning. Glancing training requires two forward passes of Directed Acyclic Decoder: The first pass assigns target tokens to appropriate positions. The second pass calculates the reconstruction loss. [M] indicates a masked token whose embeddings are all zeros.

Marginalizing A with Dynamic Programming The objective $\mathcal{L}$ requires marginalizing all paths A, which is expensive due to the numerous paths. Similar to Graves et al. (2006), we employ dynamic programming to tackle the issue.

Generally, we recurrently calculate the probability sum of path prefixes that end at the vertex u and generate the target prefix $Y_{\leq i}$, denoted as $f_{i,u}$. Since the path prefixes that end at the vertex u should pass through a vertex v satisfying $v < u$, so $f_{i,u}$ can be obtained from $f_{i-1,v}$. By recurrently calculating $f_{i,u}$, we finally obtain the probability sum of all valid paths and the training objective $\mathcal{L} = -\log f_{M,L}$. Our algorithm reduces the time complexity to $\mathcal{O}(ML^2)$ and can be implemented by $\mathcal{O}(M)$ PyTorch operations. The detailed formulation is presented in Appendix A.

Glancing Training on Graph Previous work shows that glancing training (Qian et al., 2021a) can significantly improve the translation quality of non-iterative NATs. Here we present a modified glancing training technique on DAG, which still requires only one reference per sample.

Specifically, to improve the training of DA-Transformer via glancing training, we add a masked target to the decoder input and train the model by reconstruction, which promotes the learning of dependency between vertices. Formally, the objective of glancing is defined as

$$\mathcal{L}_{\text{glancing}} = -\log P_\theta(Y|X, Z), \tag{8}$$

where $Z = [z_1, \cdots, z_L]$ is a randomly masked target provided as an extra decoder input, and $P_\theta(Y|X, Z)$ is similarly defined as Eq(4).

The glancing training follows three steps, as shown in Fig.4. (1) We assign the target tokens to appropriate vertices since the decoder input is longer than the target sentence. The assignment follows the most probable path $\hat{A} = \arg\max_{A \in \Gamma} P_\theta(Y, A|X)$, which requires a forward pass of the decoder and dynamic programming. (2) We obtain Z by masking some tokens. We utilize the masking strategy proposed by GLAT (Qian et al., 2021a), which decides the number of unmasked tokens according to the prediction accuracy.[4] (3) We add Z to the decoder input and train the model by minimizing Eq(8).

Algorithm 1 Greedy / Lookahead Decoding in Pytorch-like Parallel Pseudocode

Input: Graph Size L, Transition Matrix $\mathbf{E} \in \mathbb{R}^{L \times L}$, Token Distributions $\mathbf{P} \in \mathbb{R}^{L \times |\mathbb{V}|}$
if Using Lookahead **then**
  $\mathbf{E} := \mathbf{E} \otimes$ [$\mathbf{P}$.MAX(dim=1).UNSQUEEZE(dim=0)]
  # $\mathbf{E}$ now jointly considers $\mathbf{P}$ and $\mathbf{E}$
  # $\otimes$ is element-wise multiplication
end if
tokens := $\mathbf{P}$.ARGMAX(dim=1) # shape: (L)
edges := $\mathbf{E}$.ARGMAX(dim=1) # shape: (L)
$i := 1$, *output* := [*tokens*[1]]
repeat
  $i :=$ *edges*[i] # jumping along the transition
  output.APPEND(*tokens*[i])
until $i = L$

3.3. Inference

In inference, DA-Transformer constructs a DAG that stores multiple translations, where we aim to find the most probable one. Compared with existing NATs, DA-Transformer utilizes transitions to distinguish different candidates, which improves fluency and avoids errors like repeated tokens. We propose three decoding strategies to find high-quality translations while keeping low latency.

Greedy The simplest strategy is to take the most likely choices for the transitions and tokens. Specifically, we perform parallel argmax operations to obtain the most likely transition and token for each vertex. Then, we generate the translation by collecting the predicted tokens along the chosen path. The greedy decoding is highly efficient that only uses two parallel operations, as shown in Algo.1.

Lookahead Lookahead decoding improves the greedy strategy by jointly considering the transitions and the tokens. Specifically, we rearrange $P_\theta(Y, A|X)$ into

$$P_\theta(y_1|a_1, X) \prod_{i=2}^{M} P_\theta(a_i|a_{i-1}, X) P_\theta(y_i|a_i, X), \tag{9}$$

which becomes a sequential decision problem of choosing a_i and y_i in order. We simultaneously obtain

$$y_i^*, a_i^* = \arg\max P_\theta(y_i|a_i, X) P_\theta(a_i|a_{i-1}, X), \tag{10}$$

which can be still implemented in parallel with almost zero overhead, as presented in Algo.1.

BeamSearch BeamSearch is a more accurate method for solving the above decoding problem. Following Gu & Kong (2021), we combine an n-gram language model to

[4] The number of unmasked token $t = \tau \sum_{i=1}^{M} [y_i \neq \hat{y}_i]$, where $\hat{y}_i = \arg\max P_\theta(\cdot|a_i, X)$, and $\tau \in [0, 1]$ is a hyper-parameter.

Table 1. Results on WMT14 En↔De and WMT17 Zh↔En. We present DA-Transformer' results with mean and standard deviation of three runs with different random seeds. Best performance of non-iterative NATs (iter=1) are **bolded**. Average Gap is the gap of BLEU against the best AT model, excluding the missing values. * indicates results of our re-implementation. Our autoregressive transformer is better than previously reported results because we use the same training setting as NATs (300k steps, 64k tokens/batch; previous results use 100k steps, 32k tokens/batch). † uses reranking methods in NAT decoding (LPD, Wei et al., 2019).

Model	Iter #	WMT14 En-De Raw	WMT14 En-De KD	WMT14 De-En Raw	WMT14 De-En KD	WMT17 En-Zh Raw	WMT17 En-Zh KD	WMT17 Zh-En Raw	WMT17 Zh-En KD	Average Gap ↓ Raw	Average Gap ↓ KD	Speedup
Transformer (Vaswani et al., 2017)	M	27.6	27.8	31.4	31.3	34.3	34.4	23.7	24.0	0.45	0.49	1.0x
Transformer (Ours)	M	28.07*	28.54*	31.94*	31.54*	34.89*	34.69*	23.89*	24.68*	0	0	1.0x
CMLM (Ghazvininejad et al., 2019)	10	24.61	27.03	29.40	30.53	-	33.19	-	23.21	3.00	1.37	2.2x
SMART (Ghazvininejad et al., 2020b)	10	25.10	27.65	29.58	31.27	-	34.06	-	23.78	2.67	0.67	2.2x
DisCo (Kasai et al., 2020)	≈4	25.64	27.34	-	31.31	-	34.63	-	23.83	2.43	0.59	3.5x
Imputer (Saharia et al., 2020)	8	25.0	28.2	-	31.8	-	-	-	-	3.07	0.04	2.7x
CMLMC (Huang et al., 2022c)	10	26.40	28.37	30.92	31.41	-	-	-	-	1.35	0.15	1.7x
Vanilla NAT (Gu et al., 2018)	1	11.79*	19.99*	16.27*	25.77*	18.92*	25.84*	8.69*	14.81*	15.78	8.26	15.3x
CTC (Libovický & Helcl, 2018)	1	18.42*	25.52	23.65*	28.73	26.84*	31.39*	12.23*	19.93*	9.41	3.47	14.6x
AXE† (Ghazvininejad et al., 2020a)	1	20.40	23.53	24.90	27.90	-	30.88	-	19.79	7.36	4.34	14.2x
GLAT (Qian et al., 2021a)	1	19.42*	25.21	26.51*	29.84	29.79*	32.22*	18.88*	21.84*	6.05	2.59	15.3x
OaXE† (Du et al., 2021)	1	22.4	26.1	26.8	30.2	-	32.9	-	22.1	5.4	2.0	14.2x
CTC + GLAT (Qian et al., 2021a)	1	25.02*	26.39	29.14*	29.54	30.65*	32.51*	19.92*	23.11*	3.52	1.98	14.6x
CTC + DSLP (Huang et al., 2022a)	1	24.81	27.02	28.33	31.61	-	-	-	-	3.44	0.73	14.0x
DA-Transformer + Greedy (Ours)	1	26.08$_{\pm.25}$	27.31$_{\pm.08}$	30.48$_{\pm.18}$	31.30$_{\pm.06}$	33.27$_{\pm.12}$	33.80$_{\pm.11}$	22.66$_{\pm.12}$	24.04$_{\pm.09}$	1.58	0.75	14.0x
+ Lookahead	1	26.57$_{\pm.21}$	27.49$_{\pm.05}$	30.68$_{\pm.24}$	31.37$_{\pm.06}$	33.83$_{\pm.13}$	34.08$_{\pm.13}$	22.82$_{\pm.20}$	24.23$_{\pm.14}$	1.22	0.57	13.9x
+ BeamSearch	1	27.02$_{\pm.15}$	27.78$_{\pm.07}$	31.24$_{\pm.18}$	31.80$_{\pm.03}$	34.21$_{\pm.21}$	**34.35**$_{\pm.12}$	24.22$_{\pm.10}$	24.90$_{\pm.16}$	0.53	0.16	7.1x
+ BeamSearch + 5-gram LM	1	**27**.25$_{\pm.12}$	**27**.91$_{\pm.07}$	**31.54**$_{\pm.20}$	**31**.95$_{\pm.06}$	**34.23**$_{\pm.17}$	34.27$_{\pm.05}$	**24.49**$_{\pm.06}$	**25.01**$_{\pm.18}$	**0.32**	**0.08**	7.0x

improve the performance. Specifically, we search in beam to approximately find the optimal Y^* that maximizes

$$\frac{1}{|Y|^\alpha}\left[\log P_\theta(Y|X) + \gamma \log P_{n\text{-gram}}(Y)\right], \qquad (11)$$

where α, γ are hyper-parameters for length penalty and language model scores. Note that Y can appear on multiple paths, where we obtain the probability sum of these paths in BeamSearch to obtain $P_\theta(Y|X)$. More details are shown in Appendix B.

It should be noticed that BeamSearch requires sequential operations and does not preserve the non-autoregressive nature. However, such sequential operations do not involve deep network computations and can still be very efficient, with about 7 times speedups compared with AT models.[5]

4. Experiments

Dataset We conduct experiments on two benchmarks, WMT14 En↔De (4.5M) and WMT17 Zh↔En (20M), where we follow Zhou et al. (2020); Kasai et al. (2020) for pre-processing. For knowledge distillation, we follow Du et al. (2021) to use Transformer-*big* as our teacher model and generate the distilled data with a beam size of 5.

Metrics For fair comparisons with previous work, we use tokenized BLEU (Papineni et al., 2002) for all benchmarks except WMT17 En-Zh, where we use sacreBLEU (Post, 2018). The latency speedup is evaluated on WMT17 En-De test set with a batch size of 1.

Hyper-parameters Our models generally use the hyper-parameters of transformer-*base* (Vaswani et al., 2017). For regularization, we set dropout to 0.1, weight decay to 0.01, and label smoothing to 0.1. All models, including ATs, are trained for 300k updates with a batch of 64k tokens. The learning rate warms up to $5 \cdot 10^{-4}$ within 10k steps and then decays with the inverse square-root schedule. We evaluate the BLEU scores on the validation set every epoch and average the best 5 checkpoints for the final model. For DA-Transformer, we use $\lambda = 8$ and Lookahead Decoding unless otherwise specified. We linearly anneal τ from 0.5 to 0.1 for glancing training. For BeamSearch, we set beam size to 200, γ to 0.1, and tune α from $[1, 1.4]$ on the validation set. The training lasts approximately 32 hours on 16 Nvidia V100-32G GPUs.

4.1. Main Results

As shown in Table 1, DA-Transformer substantially improves the translation quality and outperforms strong baselines by a large margin. Our model alleviates the multi-modality problem by capturing multiple translations within a DAG, which avoids inconsistent labels in training and reduces the errors of mixing translations in inference. We highlight the empirical advantages of our method:

1) Better translation quality compared with non-iterative NATs. As a non-iterative NAT, our model achieves new SoTA results in translation quality while preserving compet-

[5] We will release an efficient C++ implementation at https://github.com/thu-coai/DA-Transformer.

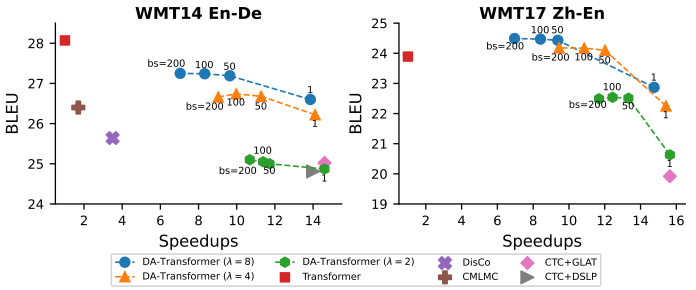

Figure 5. Quality-latency tradeoff on WMT14 En-De and WMT17 Zh-En with varying graph sizes and beam sizes. The graph size is λ times of the source length. We use Beamsearch + 5-gram LM with the beam size (bs) of $200, 100, 50$. bs $= 1$ indicates Lookahead Decoding.

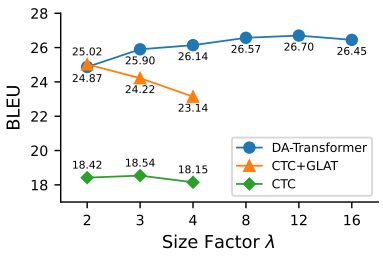

Figure 6. Effects of λ on WMT14 En-De. The graph size (DA-Transformer) or the output length (CTC) is λ times of the source length.

itive speedups. Unlike existing NATs which heavily rely on KD, our model with Lookahead outperforms the best baselines on the raw data by 2.2 BLEU on average, verifying that DA-Transformer can effectively alleviate the multi-modality problem without simplifying the training data.

2) Lower inference latency compared with ATs and iterative NATs. DA-Transformer achieves 7x$\sim$14x speedups over ATs, where the remaining BLEU gaps are about 0.32 on average. Especially on WMT17 Zh-En, our best model with BeamSearch outperforms ATs by 0.6 BLEU. Moreover, DA-Transformer dominates all iterative NATs on both BLEU and latency for all benchmarks except WMT14 En-DE with KD, which shows the great potential of our model.

3) Flexible quality-latency tradeoff. Comparing the decoding strategies of our method, we find that Lookahead Decoding consistently outperforms Greedy Decoding, and the n-gram LM usually benefits BeamSearch, with almost zero overheads. To better show the quality-latency tradeoff, we tune the graph size and beam size with our decoding strategies. As shown in Fig.5, our method significantly outperforms existing NATs and provides flexible quality-latency tradeoff for non-autoregressive translation.

4.2. Ablation Study

In this section, we investigate the effects of the graph size and training methods on the raw data of WMT14 En-De.

Graph Size DA-Transformer utilizes a DAG with L vertices, which is empirically set to λ times of the source length. A large DAG can model more translations. However, it also makes the transition predictions difficult. We manually tune λ from 2 to 16, as shown in Fig.6.

The results show that larger graphs improve the translation quality until λ exceeds 12, where λ is not sensitive around its best value. We compare our methods against CTC, which also utilizes a similar hyper-parameter to determine the output length (Libovický & Helcl, 2018). Although CTC+GLAT has a similar performance with DA-Transformer when $\lambda = 2$, the BLEU score does not increase for a larger λ. We attribute the problem to the inconsistent label problem: a longer output sequence does not help CTC to reduce the inconsistent labels in training, where DA-Transformer benefits from larger graphs by assigning different tokens to distinct vertices. Considering the performance and computation cost, we choose $\lambda = 8$ and apply it to all other datasets.

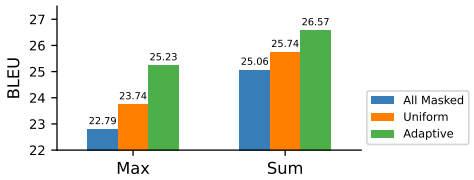

Figure 7. Ablation study of training objectives on WMT14 En-De. **Max/Sum** represent using the max operation or the sum operation in marginalizing all paths in Eq(4). We compare three masking strategies in obtaining the decoder input: **All Masked**, **Uniform** (Ghazvininejad et al., 2019), and **Adaptive** (Qian et al., 2021a).

Training Objectives DA-Transformer is trained with a glancing objective that only requires one reference for each sample, where we investigate two important designs: *First*, we marginalize all possible paths to obtain $\mathcal{L}$, which is equivalent to optimizing the paths with different weights as discussed in Sec.3.2. We compare it with the objective only optimizing the most probable path, i.e., replacing the sum operation by the max operation in Eq(4). *Second*, we use glancing training with a masked target as inputs, where the masked tokens are adaptively chosen according to the prediction accuracy (Qian et al., 2021a). We compare it with two other strategies: masking all inputs (i.e., do not use glancing training), uniform random masking (Ghazvininejad et al., 2019).

The results are shown in Fig.7. *First*, marginalizing all paths (Sum) outperforms choosing the most probable path (Max). One possible reason is that the max operation makes sharp weight assignments in the early training, leading to a premature convergence in which only several paths are

Table 2. Token accuracy under *the best assignment*. An assignment matches each reference token with a predicted token, which is called an alignment in CTC or a path in DA-Transformer.

Model	WMT14 En-De		WMT17 Zh-En	
	Train	Valid	Train	Valid
Vanilla NAT	29.7	29.6	39.8	22.4
CTC + GLAT	50.2	51.7	47.1	32.3
DA-Transformer	**69.3**	**69.9**	**80.1**	**67.0**

used. *Second*, the glancing training (Uniform or Adaptive) is better than the vanilla training (All Masked), which improves the translation quality by promoting representation learning. Moreover, the adaptive strategy can further boost performance by choosing the masking ratio dynamically.

4.3. Analysis

This section verifies that DA-Transformer benefits from assigning tokens to vertices in training and explicitly considers the transitions in inference. It also shows some cases of learned DAGs. We present more analyses in the appendix, including the translation performance on different lengths (Appendix C.1), performance with controlled training time (Appendix C.2), and some statistics of DAGs (Appendix E).

DA-Transformer improves token accuracy. In training, we assign tokens of different translations to different vertices, which avoids the inconsistent labels in training and thus improves the token accuracy in inference. We compare our model against two baselines, Vanilla NAT and CTC+GLAT. Note that CTC utilizes an alignment-based objective, which also assigns the reference tokens to different positions of Transformer. We calculate the accuracy *under the best assignment* following two steps: We first obtain the most probable assignment that matches each reference token to a prediction. Then, we calculate the accuracy by comparing the predicted tokens on the best assignment (i.e., the best path in DA-Transformer) against the reference. [6]

As shown in Table 2, vanilla NAT suffers from label inconsistency problem, leading to low token accuracies. Comparing DA-Transformer and CTC, we find that our method is far more effective, especially on the syntax distant language pair such as WMT17 Zh-En. We conjecture that the advantage mainly comes from our flexible assignment method. CTC only avoids position mismatches by inserting empty or repeated tokens. It requires that the possible translations share similar lexical choices, which cannot handle highly diverse translations.

DA-Transformer facilitates diverse generation. In inference, DA-Transformer utilizes the transition matrix to avoid incorrect outputs caused by mixing multiple translations. We evaluate the ability to distinguish different translations by sampling diverse translations from the DAG. Specifically, we begin at the start vertex and repeatedly use Nucleus Sampling (top-p sampling, Holtzman et al., 2020) to choose the next vertex and token according to Eq.(9). We use $p = 0.8$ and vary the temperature from 0.4 to 1.0.

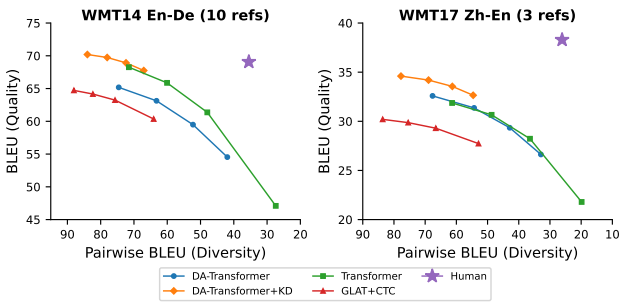

Figure 8. Quality-diversity tradeoff of sampled sentences from different models. Each curve has 4 points with sampling temperatures $t = [0.4, 0.6, 0.8, 1]$ (from left to right). We sample hypotheses with the same number of human-written references.

We follow Shen et al. (2019) to evaluate the quality and diversity by multi-reference BLEU and pairwise BLEU. We compare our model against AT and GLAT+CTC, the best non-iterative NAT baseline. We obtain the hypotheses from GLAT+CTC by replacing the argmax operations in decoding with Nucleus Sampling with the same p and temperature.

The results are shown in Fig.8. Compared with GLAT+CTC, DA-Transformer achieves a better tradeoff between quality and diversity. With the same temperature, the generated samples (without KD) by our model are far more diverse than GLAT+CTC. It shows that our model can learn multiple diverse translations and further decode them in inference. Compared with Transformer, DA-Transformer (without KD) is slightly less diverse but achieves a close tradeoff on WMT17 Zh-En, which shows the great potential of our model. Moreover, we find that applying KD to DA-Transformer improves the quality but sacrifices the diversity because KD reduces the data modalities.

Case Study We choose a test sample of WMT17 Zh-En and illustrate the DAG predicted by our model. For a clear presentation, we use $\lambda = 4$ for a small graph and further remove some useless vertices and edges. Specifically, we remove all vertices with passing probabilities smaller than 0.1, where the passing probabilities represent how likely the vertex will appear on a randomly sampled path. We only show the transitions in the top 90% of probabilities.

As shown in Fig.9, the predicted DAG is highly reasonable. Following the transitions, we can clearly distinguish translation expressions, which avoids the errors like repeated tokens shown in the vanilla NAT's output. We present the

[6] In CTC, a reference token may be matched with several predictions, so we average the accuracies for the reference token. The special empty tokens are not counted in the accuracy.

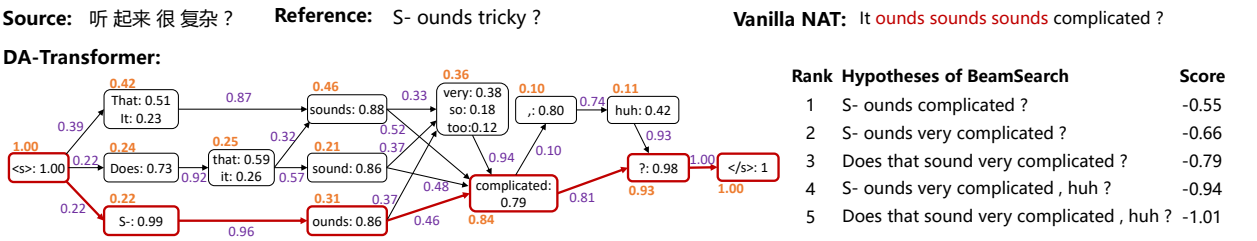

Figure 9. A test sample of WMT17 Zh-En with the DAG and BeamSearch results. We present the top candidates on each vertex and the transition probabilities between vertices. We also present the passing probabilities representing how likely a vertex will appear on a sampled path. We remove vertices/edges with small passing/transition probabilities for a clear presentation. The vanilla NAT mixes the tokens from different translations, which can be avoided in DA-Transformer inference. We use the BPE tokenizer (Sennrich et al., 2016), where a subword prefix is marked by -. See more examples in Appendix D.

top-5 hypotheses produced by BeamSearch, which are fluent and diverse.

However, we can still find errors in the predicted DAG, e.g., a possible incorrect translation "*Does that sounds ...*". Although the error does not easily occur in Lookahead or BeamSearch decoding, it shows that there is still space for improving the consistency between the tokens in our model.

5. Conclusion

In this paper, we propose DA-Transformer for non-autoregressive machine translation. Unlike previous NAT models relying on knowledge distillation, DA-Transformer tackles the multi-modality problem by capturing multiple translations with a directed acyclic decoder. Experimental results show that DA-Transformer outperforms all NAT baselines on raw training data and achieves competitive results with AT models. The best model of DA-Transformer even outperforms the autoregressive Transformer by 0.6 BLEU on Zh-En, which demonstrates the potential of the proposed approach.

Acknowledgments

This work was supported by the National Science Foundation for Distinguished Young Scholars (with No. 62125604) and the NSFC projects (Key project with No. 61936010 and regular project with No. 61876096). This work was also supported by the Guoqiang Institute of Tsinghua University, with Grant No. 2019GQG1 and 2020GQG0005, and sponsored by Tsinghua-Toyota Joint Research Fund.

References

Akhbardeh, F., Arkhangorodsky, A., Biesialska, M., Bojar, O., Chatterjee, R., Chaudhary, V., Costa-jussa, M. R., España-Bonet, C., Fan, A., Federmann, C., Freitag, M., Graham, Y., Grundkiewicz, R., Haddow, B., Harter, L., Heafield, K., Homan, C., Huck, M., Amponsah-Kaakyire, K., Kasai, J., Khashabi, D., Knight, K., Kocmi, T., Koehn, P., Lourie, N., Monz, C., Morishita, M., Nagata, M., Nagesh, A., Nakazawa, T., Negri, M., Pal, S., Tapo, A. A., Turchi, M., Vydrin, V., and Zampieri, M. Findings of the 2021 conference on machine translation (WMT21). In *Proceedings of the Sixth Conference on Machine Translation*, pp. 1–88, Online, November 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.wmt-1.1.

Bao, Y., Zhou, H., Feng, J., Wang, M., Huang, S., Chen, J., and Li, L. Non-autoregressive transformer by position learning. *CoRR*, abs/1911.10677, 2019. URL http://arxiv.org/abs/1911.10677.

Bao, Y., Huang, S., Xiao, T., Wang, D., Dai, X., and Chen, J. Non-autoregressive translation by learning target categorical codes. In Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tür, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., and Zhou, Y. (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pp. 5749–5759. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.458. URL https://doi.org/10.18653/v1/2021.naacl-main.458.

Ding, L., Wang, L., Liu, X., Wong, D. F., Tao, D., and Tu, Z. Understanding and improving lexical choice in non-autoregressive translation. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021a. URL https://openreview.net/forum?id=ZTFeSBIX9C.

Ding, L., Wang, L., Liu, X., Wong, D. F., Tao, D., and Tu, Z. Rejuvenating low-frequency words: Making the most of parallel data in non-autoregressive translation. In Zong, C., Xia, F., Li, W., and Navigli, R. (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Pro-*

cessing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021, pp. 3431–3441. Association for Computational Linguistics, 2021b. doi: 10.18653/v1/2021.acl-long.266. URL https://doi.org/10.18653/v1/2021.acl-long.266.

Dong, M., Cheng, Y., Liu, Y., Xu, J., Sun, M., Izuha, T., and Hao, J. Query lattice for translation retrieval. In Hajic, J. and Tsujii, J. (eds.), *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pp. 2031–2041. ACL, 2014. URL https://aclanthology.org/C14-1192/.

Du, C., Tu, Z., and Jiang, J. Order-agnostic cross entropy for non-autoregressive machine translation. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 2849–2859. PMLR, 2021. URL http://proceedings.mlr.press/v139/du21c.html.

Dyer, C., Muresan, S., and Resnik, P. Generalizing word lattice translation. In McKeown, K. R., Moore, J. D., Teufel, S., Allan, J., and Furui, S. (eds.), *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA*, pp. 1012–1020. The Association for Computer Linguistics, 2008. URL https://aclanthology.org/P08-1115/.

Feng, Y., Liu, Y., Mi, H., Liu, Q., and Lü, Y. Lattice-based system combination for statistical machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009, 6-7 August 2009, Singapore, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1105–1113. ACL, 2009. URL https://aclanthology.org/D09-1115/.

Ghazvininejad, M., Levy, O., Liu, Y., and Zettlemoyer, L. Mask-predict: Parallel decoding of conditional masked language models. In Inui, K., Jiang, J., Ng, V., and Wan, X. (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 6111–6120. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1633. URL https://doi.org/10.18653/v1/D19-1633.

Ghazvininejad, M., Karpukhin, V., Zettlemoyer, L., and Levy, O. Aligned cross entropy for non-autoregressive machine translation. 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. 3515–3523. PMLR, 2020a. URL http://proceedings.mlr.press/v119/ghazvininejad20a.html.

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

Graves, A., Fernández, S., Gomez, F. J., and Schmidhuber, J. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In Cohen, W. W. and Moore, A. W. (eds.), *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, volume 148 of *ACM International Conference Proceeding Series*, pp. 369–376. ACM, 2006. doi: 10.1145/1143844.1143891. URL https://doi.org/10.1145/1143844.1143891.

Gu, J. and Kong, X. Fully non-autoregressive neural machine translation: Tricks of the trade. In Zong, C., Xia, F., Li, W., and Navigli, R. (eds.), *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pp. 120–133. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.findings-acl.11. URL https://doi.org/10.18653/v1/2021.findings-acl.11.

Gu, J., Bradbury, J., Xiong, C., Li, V. O. K., and Socher, R. Non-autoregressive neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=B1l8BtlCb.

Gu, J., Wang, C., and Zhao, J. Levenshtein transformer. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (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.

Guo, J., Tan, X., He, D., Qin, T., Xu, L., and Liu, T. Non-autoregressive neural machine translation with enhanced decoder input. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence*

Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, pp. 3723–3730. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33013723. URL https://doi.org/10.1609/aaai.v33i01.33013723.

Guo, J., Xu, L., and Chen, E. Jointly masked sequence-to-sequence model for non-autoregressive neural machine translation. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. R. (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 376–385. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.36. URL https://doi.org/10.18653/v1/2020.acl-main.36.

Hannun, A. Y., Maas, A. L., Jurafsky, D., and Ng, A. Y. First-pass large vocabulary continuous speech recognition using bi-directional recurrent dnns. *CoRR*, abs/1408.2873, 2014. URL http://arxiv.org/abs/1408.2873.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. The curious case of neural text degeneration. 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=rygGQyrFvH.

Huang, C., Zhou, H., Zaïane, O. R., Mou, L., and Li, L. Non-autoregressive translation with layer-wise prediction and deep supervision. *The Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022*, 2022a. URL https://arxiv.org/abs/2110.07515.

Huang, F., Tao, T., Zhou, H., Li, L., and Huang, M. On the learning of non-autoregressive transformers. In *Proceedings of the 39th International Conference on Machine Learning, ICML 2022*, 2022b.

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

Kaiser, L., Bengio, S., Roy, A., Vaswani, A., Parmar, N., Uszkoreit, J., and Shazeer, N. Fast decoding in sequence models using discrete latent variables. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2395–2404. PMLR, 2018. URL http://proceedings.mlr.press/v80/kaiser18a.html.

Kasai, J., Cross, J., Ghazvininejad, M., and Gu, J. 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.

Kasai, J., Pappas, N., Peng, H., Cross, J., and Smith, N. A. 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.

Kim, Y. and Rush, A. M. Sequence-level knowledge distillation. In Su, J., Carreras, X., and Duh, K. (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.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. Moses: Open source toolkit for statistical machine translation. In Carroll, J. A., van den Bosch, A., and Zaenen, A. (eds.), *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic*. The Association for Computational Linguistics, 2007. URL https://aclanthology.org/P07-2045/.

Lee, J., Mansimov, E., and Cho, K. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In Riloff, E., Chiang, D., Hockenmaier, J., and Tsujii, J. (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 1173–1182. Association for Computational Linguistics, 2018. doi: 10.18653/v1/d18-1149. URL https://doi.org/10.18653/v1/d18-1149.

Lee, J., Shu, R., and Cho, K. Iterative refinement in the continuous space for non-autoregressive neural machine translation. In Webber, B., Cohn, T., He, Y., and Liu, Y. (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pp. 1006–1015. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.73. URL https://doi.org/10.18653/v1/2020.emnlp-main.73.

Libovický, J. and Helcl, J. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In Riloff, E., Chiang, D., Hockenmaier, J., and Tsujii, J. (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 3016–3021. Association for Computational Linguistics, 2018. doi: 10.18653/v1/d18-1336. URL https://doi.org/10.18653/v1/d18-1336.

Ma, X., Zhou, C., Li, X., Neubig, G., and Hovy, E. H. Flowseq: Non-autoregressive conditional sequence generation with generative flow. In Inui, K., Jiang, J., Ng, V., and Wan, X. (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 4281–4291. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1437. URL https://doi.org/10.18653/v1/D19-1437.

Och, F. J. and Ney, H. The alignment template approach to statistical machine translation. *Comput. Linguistics*, 30(4):417–449, 2004. doi: 10.1162/0891201042544884. URL https://doi.org/10.1162/0891201042544884.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. fairseq: A fast, extensible toolkit for sequence modeling. In Ammar, W., Louis, A., and Mostafazadeh, N. (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.

Papineni, K., Roukos, S., Ward, T., and Zhu, W. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pp. 311–318. ACL, 2002. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040/.

Post, M. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 186–191, Belgium, Brussels, October 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W18-6319.

Qian, L., Zhou, H., Bao, Y., Wang, M., Qiu, L., Zhang, W., Yu, Y., and Li, L. Glancing transformer for non-autoregressive neural machine translation. In Zong, C., Xia, F., Li, W., and Navigli, R. (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pp. 1993–2003. Association for Computational Linguistics, 2021a. URL https://aclanthology.org/2021.acl-long.155.

Qian, L., Zhou, Y., Zheng, Z., Zhu, Y., Lin, Z., Feng, J., Cheng, S., Li, L., Wang, M., and Zhou, H. 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.

Richardson, F., Ostendorf, M., and Rohlicek, J. R. Lattice-based search strategies for large vocabulary speech recognition. In *1995 International Conference on Acoustics, Speech, and Signal Processing, ICASSP '95, Detroit, Michigan, USA, May 08-12, 1995*, pp. 576–579. IEEE Computer Society, 1995. doi: 10.1109/ICASSP.1995.479663. URL https://doi.org/10.1109/ICASSP.1995.479663.

Rosti, A. I., Ayan, N. F., Xiang, B., Matsoukas, S., Schwartz, R. M., and Dorr, B. J. Combining outputs from multiple machine translation systems. In Sidner, C. L., Schultz, T., Stone, M., and Zhai, C. (eds.), *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, April 22-27, 2007, Rochester, New York, USA*, pp. 228–235. The Association for Computational Linguistics, 2007. URL https://aclanthology.org/N07-1029/.

Saharia, C., Chan, W., Saxena, S., and Norouzi, M. Non-autoregressive machine translation with latent alignments. In Webber, B., Cohn, T., He, Y., and Liu, Y. (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pp. 1098–1108. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.83. URL https://doi.org/10.18653/v1/2020.emnlp-main.83.

Sennrich, R., Haddow, B., and Birch, A. 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.

Shao, C., Zhang, J., Feng, Y., Meng, F., and Zhou, J. 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://aaai.org/ojs/index.php/AAAI/article/view/5351.

Shen, T., Ott, M., Auli, M., and Ranzato, M. Mixture models for diverse machine translation: Tricks of the trade. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5719–5728. PMLR, 2019. URL http://proceedings.mlr.press/v97/shen19c.html.

Shu, R., Lee, J., Nakayama, H., and Cho, K. Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior. 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. 8846–8853. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6413.

Sun, Z. and Yang, Y. An EM approach to non-autoregressive conditional sequence generation. 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. 9249–9258. PMLR, 2020. URL http://proceedings.mlr.press/v119/sun20c.html.

Sun, Z., Li, Z., Wang, H., He, D., Lin, Z., and Deng, Z. Fast structured decoding for sequence models. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (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. 3011–3020, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/74563ba21a90da13dacf2a73e3ddefa7-Abstract.html.

Tromble, R., Kumar, S., Och, F. J., and Macherey, W. Lattice minimum bayes-risk decoding for statistical machine translation. In *2008 Conference on Empirical Methods in Natural Language Processing, EMNLP 2008, Proceedings of the Conference, 25-27 October 2008, Honolulu, Hawaii, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 620–629. ACL, 2008. URL https://aclanthology.org/D08-1065/.

Ueffing, N., Och, F. J., and Ney, H. Generation of word graphs in statistical machine translation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing, EMNLP 2002, Philadelphia, PA, USA, July 6-7, 2002*, pp. 156–163, 2002. doi: 10.3115/1118693.1118714. URL https://aclanthology.org/W02-1021/.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (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.

Wei, B., Wang, M., Zhou, H., Lin, J., and Sun, X. Imitation learning for non-autoregressive neural machine translation. In Korhonen, A., Traum, D. R., and Màrquez, L. (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 1304–1312. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1125. URL https://doi.org/10.18653/v1/p19-1125.

Yang, K., Lei, W., Liu, D., Qi, W., and Lv, J. Pos-constrained parallel decoding for non-autoregressive generation. In Zong, C., Xia, F., Li, W., and Navigli, R. (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pp. 5990–6000. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-long.467. URL https://doi.org/10.18653/v1/2021.acl-long.467.

Zhou, C., Gu, J., and Neubig, G. Understanding knowledge distillation in non-autoregressive machine translation. 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=BygFVAEKDH.

A. Dynamic Programming for Training

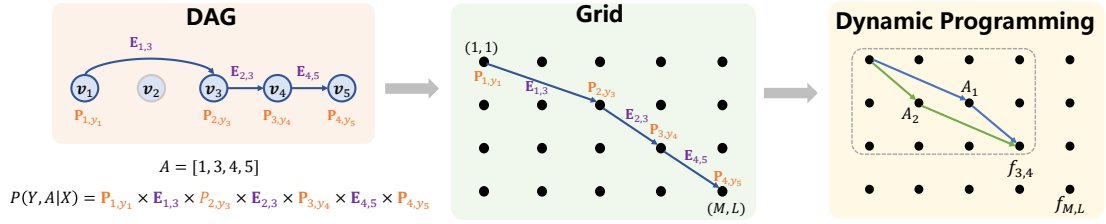

Figure 10. An overview of dynamic programming for DA-Transformer training. Each valid path in the DAG corresponds to a path in the grid that starts from $(1, 1)$ to (M, L). M is the target length, L is the graph size. $P(Y, A|X)$ can be calculated by multiplying the token probabilities and the transition probabilities. In Dynamic Programming, we recurrently calculate $f_{i,u}$, which is the probability sum of all paths that start from $(1, 1)$ and end at (i, u). E.g., $f_{3,4}$ is the sum of two paths' probabilities, A_1 and A_2. Our training objective $\mathcal{L}$ is equal to $-\log f_{M,L}$.

The training objective of DA-Transformer is formulated in Eq(4), which requires marginalizing all possible paths A. To avoid the expensive cost of enumerating the paths, we employ dynamic programming that reduces the time complexity to $\mathcal{O}(ML^2)$, where M is the target length, L is the graph size.

To utilize dynamic programming, we first represent the valid paths in a $M \times L$ grid, where each valid path of the DAG corresponds to a path in the grid that starts from the left-upper corner $(1, 1)$ and ends in the right-bottom corner (M, L), as shown in Fig.10. Formally, the path $A = \{a_1, \cdots, a_M\}$ satisfying $1 = a_1 < \cdots < a_M = L$ corresponds to a path in the grid that passes through $(1, a_1), (2, a_2), \cdots, (M, a_M)$.

Then we find that the probability on the path A, i.e., $P_\theta(Y, A|X)$, can be decomposed and calculated by multiplying the probabilities of the token predictions and transitions. Specifically, we have

$$P_\theta(Y, A|X) = \prod_{j=1}^{M} P_\theta(y_j|a_j, X) \prod_{j=2}^{M} P_\theta(a_j|a_{j-1}, X) \tag{12}$$

$$= \prod_{j=1}^{M} \mathbf{P}_{a_j, y_j} \prod_{j=2}^{M} \mathbf{E}_{a_{j-1}, a_j}. \tag{13}$$

where $\mathbf{P}_{a_j, y_j}$ can be regarded as the token probability on the point (j, a_j), and the $\mathbf{E}_{a_{j-1}, a_j}$ can be regarded as the transition probability on the edge connecting $(j-1, a_{j-1})$ with (j, a_j).

Recall that our objective requires the sum of the probabilities of all valid paths. We can recurrently calculate $f_{i,u}$, which is defined as the probability sum of the paths that start from $(1, 1)$ but end at (i, u). Since each valid path that ends at (i, u) must pass through a point $(i-1, v)$ where $v \in [1, u)$, we reach a recurrence formula that obtains $f_{i,u}$ from $f_{i-1,v}$:

$$f_{i,u} = \mathbf{P}_{u,y_i} \sum_{v=1}^{u-1} f_{i-1,v}\, \mathbf{E}_{v,u} \quad (2 \le i \le M, 1 \le u \le L), \tag{14}$$

where the boundary conditions are:

$$f_{1,1} = \mathbf{P}_{1,y_1}; \qquad f_{1,u} = 0 \quad (2 \le u \le L). \tag{15}$$

Algorithm 2 Dynamic Programming Algorithm in Pytorch-like Parallel Pseudocode

Input: Target Length M, Graph Size L, Target Sentence Y, Transition Matrix $\mathbf{E} \in \mathbb{R}^{L \times L}$, Token Distributions $\mathbf{P} \in \mathbb{R}^{L \times |\mathbb{V}|}$
Initialize a zero matrix $f \in \mathbb{R}^{M \times L}$
$f[1, 1] := 1$
for $i = 2, 3, \cdots, M$ **do**
  $f[i, :] := \mathbf{P}[:, y_i] \otimes (f[i-1, :] \times \mathbf{E})$ # $\otimes$ is the element-wise multiplication, $\times$ is the vector-matrix multiplication
end for
Update the model by minimizing $\mathcal{L} = -\log f[M, L]$.

Finally, the loss can be obtained by $\mathcal{L} = -\log f_{M,L}$.

Since the product-sum operations can be calculated by matrix multiplications, the above recurrent process can be implemented with $\mathcal{O}(M)$ parallel operations, as shown in Algorithm 2.

B. Implementation of Beam Search

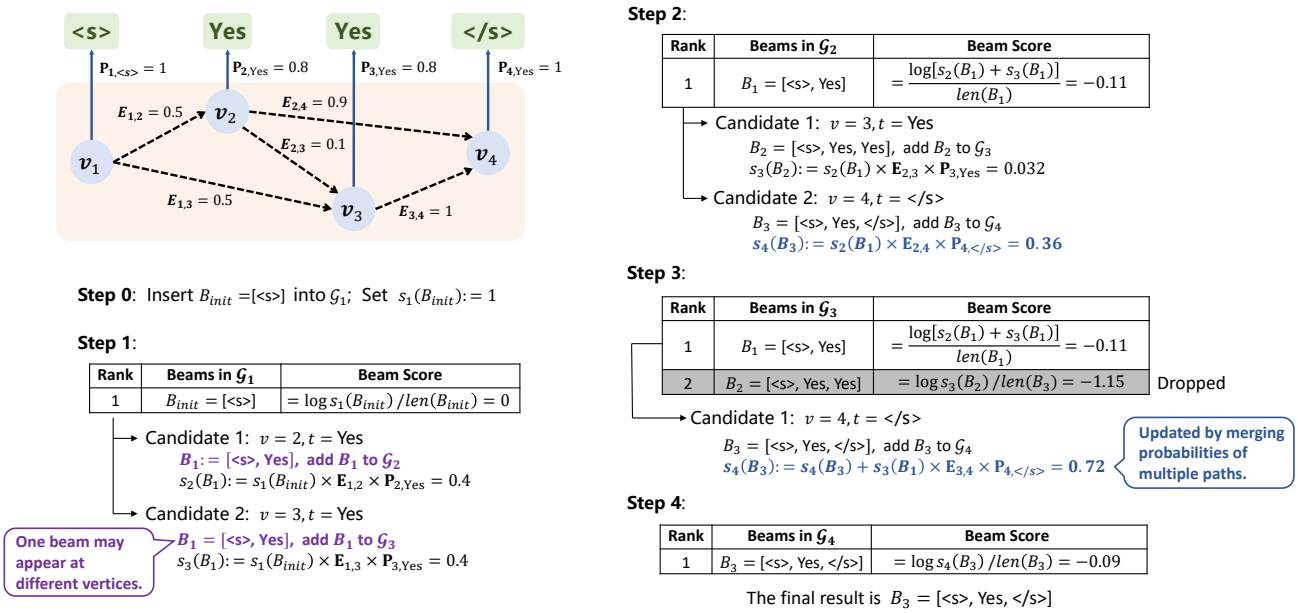

Figure 11. An step-by-step example of the beam search algorithm presented in Algorithm 3. One beam represents a translation prefix, which may appear on multiple paths. For demonstration, we only preserve the top-1 beam in each step, limit the candidate number when expanding the beams, and set $\alpha = 1$ for the length penalty and $\gamma = 0$ to disable the n-gram language model.

Our concept of beam is similar to the prefix beam search (Hannun et al., 2014), where a beam represents a translation prefix but may appear on multiple paths. E.g., in Fig.11, $[\langle s \rangle, \text{Yes}]$ is a beam that appears on two paths, $\{v_1, v_2\}$ and $\{v_1, v_3\}$. Our beam search aims to calculate the probability sum of all paths that produce the same translation, which approximates $P(Y|X)$ and works better than finding a single path that maximizes $P(Y, A|X)$.

To achieve an effective calculation of the scores, we maintain the probability sum for a beam B during the beam search. Specifically, we define $s_i(B)$ as the probability sum of the paths ends at vertex i. When sorting the beams, we use the beam score defined in Eq(11), where $P(Y|X)$ is equal to the probability sum of all paths, i.e., $\sum_{i=1}^{L} s_i(B)$.

Our algorithm is presented in Algorithm 3 with an example shown in Fig.11. We further apply some tricks to reduce the computation costs:

- Unlike vanilla beam search that all beams have the same length in each step, our algorithm may compare beams with different lengths. To avoid a length bias in the selected beams, we only preserve the top-10 for each length. If the total number of beams is still too large, we choose the top-200 beams.
- When expanding beams, we only use the top-5 candidates. A candidate is a $\langle v, t \rangle$ pair, indicating the next vertex and token, where we jointly consider their probabilities as Eq(10).

C. More Analyses

C.1. Translation Performance on Different Lengths

To investigate the translation performance on different lengths, we split the test set into 6 buckets according to reference lengths and evaluate the BLEU score in each bucket as shown in Fig.12. Compared with NAT baselines, DA-Transformer has

Algorithm 3 BeamSearch for DA-Transformer

Input: Graph Size L, Transition Matrix $\mathbf{E} \in \mathbb{R}^{L \times L}$, Token Distributions $\mathbf{P} \in \mathbb{R}^{L \times |\mathbb{V}|}$
For $i \in [1, L]$ and any possible beam B, initialize $s_i(B) := 0$ that stores the probability sum of all B's paths that end at vertex i
Initialize $\mathcal{G}_1, \cdots, \mathcal{G}_L$ as L empty sets that store the beams at the vertex i
Insert a beam B_{init} with the start token in $\mathcal{G}_1$. Set $s_1(B_{init}) := 1$
for $i = 1, 2, \cdots, L-1$ **do**
  # Filter the beams
  Sort the beams in $\mathcal{G}_i$ by the score defined in Eq(11)
  For each beam length in $\mathcal{G}_i$, preserve the top-10 beams and remove the other beams
  Considering all beams in $\mathcal{G}_i$, preserve the top-200 beams and remove the other beams
  # Expand the beams
  for each beam B in $\mathcal{G}_i$ **do**
    for $\langle v, t \rangle$ in B's top-5 next candidates **do**
      # v is the next vertex; t is the next token
      Insert $B' = B + \{t\}$ as a new beam into $\mathcal{G}_v$.
      Update $s_v(B') := s_v(B') + s_i(B) \times \mathbf{P}_{v,t} \times \mathbf{E}_{i,v}$
    end for
  end for
end for
Output the best beam in $\mathcal{G}_L$

a substantial improvement for sentences longer than 20. These long sentences usually have more modalities in translation, which are challenging in previous NATs but can be better handled in DA-Transformer.

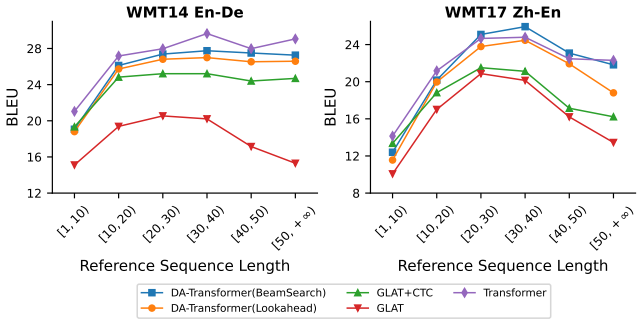

Figure 12. The BLEU score on WMT14 En-De and WMT17 Zh-En bucketed by the reference length.

C.2. Performance with Controlled Training Time

One update step of DA-Transformer's training is slower than many previous NATs because our Directed Acyclic Decoder has to process a longer sequence whose length is about 8 times of the original target. In Fig.13, we show that DA-Transformer still substantially outperforms strong NAT baselines when the training time is controlled. Moreover, we observe that our performance is more stable than the baselines during the training process.

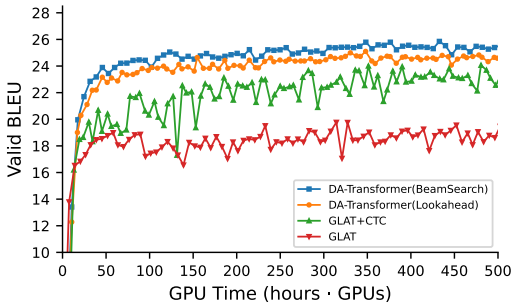

Figure 13. The valid BLEU on WMT14 En-De. We do not apply the checkpoint average trick. We evaluate the model approximately every 8 GPU-hours. The training costs 500 GPU-hours, which has about 300k, 490k, 970k updates for DA-Transformer, GLAT+CTC, GLAT, respectively.

D. More Cases

Two more test cases from WMT17 Zh-En are presented in Fig.14.

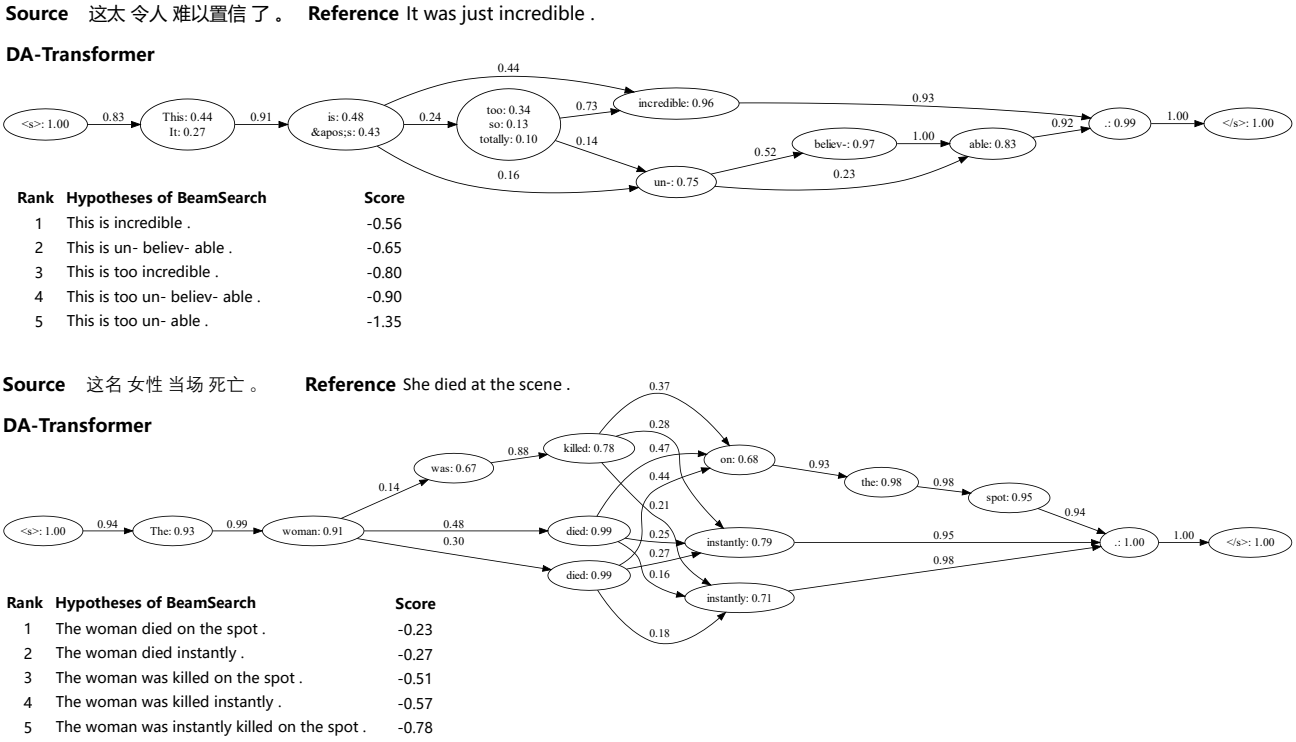

Figure 14. Two test samples of WMT17 Zh-En with the DAG and BeamSearch results. We show the top candidates on each vertex and the transition probabilities between vertices. We remove vertices/edges with small passing/transition probabilities for a clear presentation. We use the BPE tokenizer (Sennrich et al., 2016), where a subword prefix is marked by -.

E. Statistics of DAGs

For a better understanding of DA-Transformer, we collect some statistics of predicted DAGs on WMT17 Zh-En. We use a DA-Transformer with $\lambda = 4$.

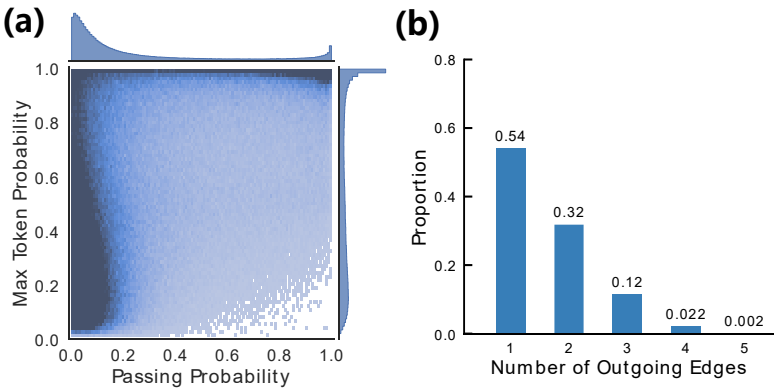

Figure 15. Statistics of DAGs. (a) The distribution of vertices' passing probabilities and max token probabilities. The passing probability represents how likely a vertex would appear on a randomly sampled path. The max token probability represents the probability of the most probable token on the vertex. (b) The distribution of numbers of vertices' outgoing edges. We only consider the most likely edges accounting for 80% of the transition probabilities and ignore the vertices with passing probabilities smaller than 0.2. We further merge the edges that are linked to vertices predicting the same token.

In Fig.15 (a), we present the distribution of vertices with passing probability and max token probability. We generally divide the vertices into three categories:

- Vertices with Passing Prob > 0.5 (accounting for 19.6%): They are very likely to appear in the generated translation. Since the average target length is about $\frac{1}{\lambda} = 25\%$ of the graph size, these vertices generate most of the tokens in the outputs.
- Vertices with Passing Prob < 0.5 and Max Token Prob > 0.5 (accounting for 40.2%): They have high confidence in predicting tokens but do not usually appear in the translation. They may contain some rare expressions.
- Vertices with Passing Prob < 0.2 and Max Token Prob < 0.2 (accounting for 15.4%): These vertices do not have specific meanings. We think the vertices are not well learned. It may be helpful if we encourage them to be more confident in generating some specific tokens.

In Fig.15 (b), we present the number of outgoing edges of vertices. We find that half of the vertices have only one outgoing edge, and the other half have multiple edges. The result shows that the predicted DAGs have complicated structures, which do not degenerate into chains.