Viterbi Decoding of Directed Acyclic Transformer for Non-Autoregressive Machine Translation

Chenze Shao1,2, Zhengrui Ma1,2, Yang Feng1,2∗
1 Key Laboratory of Intelligent Information Processing
Institute of Computing Technology, Chinese Academy of Sciences
2 University of Chinese Academy of Sciences
{shaochenze18z, mazhengrui21b, fengyang}@ict.ac.cn

Abstract

Non-autoregressive models achieve significant decoding speedup in neural machine translation but lack the ability to capture sequential dependency. Directed Acyclic Transformer (DA-Transformer) was recently proposed to model sequential dependency with a directed acyclic graph. Consequently, it has to apply a sequential decision process at inference time, which harms the global translation accuracy. In this paper, we present a Viterbi decoding framework for DA-Transformer, which guarantees to find the joint optimal solution for the translation and decoding path under any length constraint. Experimental results demonstrate that our approach consistently improves the performance of DA-Transformer while maintaining a similar decoding speedup.1

1 Introduction

Non-autoregressive translation (Gu et al., 2018) models achieve a significant decoding speedup but suffer from performance degradation, which is mainly attributed to the multi-modality problem. Multi-modality refers to the scenario where the same source sentence may have multiple translations with a strong cross-correlation between target words. However, non-autoregressive models generally hold the conditional independence assumption on target words, which prevents them from capturing the multimodal target distribution.

Recently, Directed Acyclic Transformer (Huang et al., 2022) was proposed to model sequential dependency with a directed acyclic graph consisting of different decoding paths that enable the model to capture multiple translation modalities. Although it has been proven effective, it cannot directly find the most probable translation with the argmax operation. Therefore, DA-Transformer has to apply a sequential decision process at inference time, which harms the global translation accuracy.

In this paper, we propose a Viterbi decoding (Viterbi, 1967) framework for DA-Transformer to improve the decoding accuracy. Using the Markov property of decoding path, we can apply Viterbi decoding to find the most probable path, conditioned on which we can generate the translation with argmax decoding. Then, we further improve this decoding algorithm to perform a simultaneous search for decoding paths and translations, which guarantees to find the joint optimal solution under any length constraint. After Viterbi decoding, we obtain a set of translations with different lengths and rerank them to obtain the final translation. We apply a length penalty term in the reranking process, which prevents the generation of empty translation (Stahlberg and Byrne, 2019) and enables us to control the translation length flexibly.

Experimental results on several machine translation benchmark tasks (WMT14 En↔De, WMT17 Zh↔En) show that our approach consistently improves the performance of DA-Transformer while maintaining a similar decoding speedup.

2 Preliminaries: DA-Transformer

2.1 Model Architecture

DA-Transformer is formed by a Transformer encoder and a directed acyclic decoder. The encoder and layers of the decoder are the same as vanilla Transformer (Vaswani et al., 2017). On top of the decoder, the hidden states are organized as a directed acyclic graph, whose edges represent transition probabilities between hidden states.

Given a source sentence $X = \{x_1, \cdots, x_N\}$ and a target sentence $Y = \{y_1, \cdots, y_M\}$, the decoder length $L$ is set to $\lambda \cdot N$, where $\lambda$ is a hyperparameter. The translation probability from $X$ to
Y is formulated as:
\[
P_\theta(Y|X) = \sum_{A \in \Gamma} P_\theta(A|X)P_\theta(Y|X,A),
\]  
where \(A = \{a_1, \cdots, a_M\}\) is a translation path for the target sentence \(Y\) and \(a_i\) represents the position of \(y_i\) in the decoder. \(\Gamma\) contains all possible translation paths with \(1 = a_1 < \cdots < a_M = L\).

The probability of translation path \(A\) is formulated based on the Markov hypothesis:
\[
P_\theta(A|X) = \prod_{i=1}^{M-1} P_\theta(a_{i+1}|a_i, X) = \prod_{i=1}^{M-1} E_{a_i,a_{i+1}},
\]  
where \(E \in \mathbb{R}^{L \times L}\) is the transition matrix obtained by self-attention, and \(E_{a_i,a_{i+1}}\) represents the transition probability from position \(a_i\) to position \(a_{i+1}\). \(E\) is masked by a lower triangular matrix to ensure that the translation path is acyclic.

Conditioned on \(X\) and the translation path \(A\), the translation probability of \(Y\) is formulated as:
\[
P_\theta(Y|A, X) = \prod_{i=1}^{M} P_\theta(y_i|a_i, X),
\]  
where \(P_\theta(y_i|a_i, X)\) represents the translation probability of word \(y_i\) on the position \(a_i\) of decoder.

### 2.2 Training and Inference

The training objective of DA-Transformer is to maximize the log-likelihood \(\log P_\theta(Y|X)\), which requires marginalizing all paths \(A\). Using the Markov property of translation path, DA-Transformer employs dynamic programming to calculate the translation probability. Besides, it applies glancing training (Qian et al., 2021) with a hyper-parameter \(\tau\) to promote learning.

During inference, the objective is to find the most probable translation \(\arg\max_Y P_\theta(Y|X)\). However, there is no known tractable decoding algorithm for this problem. Huang et al. (2022) proposed three approximate decoding strategies to find high-probability translations. The intuitive strategy is greedy decoding, which sequentially takes the most probable transition as the decoding path and generates a translation according to the conditional probabilities. Lookahead decoding improves greedy decoding by taking the most probable combination of transition and prediction as follows:
\[
y^*_i, a^*_i = \arg\max_{y_i, a_i} P_\theta(y_i|a_i, X)P_\theta(a_i|a_{i-1}, X).
\]  
Beam search decoding is a more accurate method that merges the paths of the same prefix, which approximates the real translation probability and better represents the model’s preference. Beam search can be optionally combined with an n-gram language model to improve the performance further. However, the speed of beam search is much lower than greedy and lookahead decoding.

### 3 Methodology

This section presents a Viterbi decoding framework for DA-Transformer to improve decoding accuracy. We first develop a basic algorithm to find the optimal decoding path and then improve it to find the joint optimal solution of the translations and decoding paths. Finally, we introduce the technique to rerank the Viterbi decoding outputs.

#### 3.1 Optimal Decoding Path

Recall that the greedy decoding strategy sequentially takes the most probable transition as the decoding path, which may not be optimal since the greedy strategy does not consider long-term profits. In response to this problem, we propose a Viterbi decoding framework for DA-Transformer that guarantees to find the optimal decoding path \(\arg\max_A P_\theta(A|X)\) under any length constraint.

Specifically, we consider decoding paths of length \(i\) that end in position \(a_i = t\), and use \(\alpha(i, t)\) to represent the maximum probability of these paths. By definition, we set the initial state \(\alpha(1, 1) = 1\) and \(\alpha(1, t > 1) = 0\). The Markov property of decoding paths enables us to sequentially calculate \(\alpha(i, \cdot)\) from its previous step \(\alpha(i - 1, \cdot)\):
\[
\alpha(i, t) = \max_{t'} \alpha(i - 1, t') \cdot E_{t,t'},
\]  
\[
\psi(i, t) = \arg\max_{t'} \alpha(i - 1, t') \cdot E_{t,t'},
\]  
where \(E\) is the transition matrix defined in Equation 2 and \(\psi(i, t)\) is the backtracking index pointing to the previous position. After \(L\) iterations, we obtain the score for every possible length, and then we can find the optimal length with the argmax function:
\[
M = \arg\max_i \alpha(i, L).
\]  
After determining the length \(M\), we can trace the best decoding path along the backtracking index starting from \(a_M = L\):
\[
a_i = \psi(i + 1, a_{i+1}).
\]  

4420
Finally, conditioning on the optimal path $A$, we can generate the translation with argmax decoding:

$$y_i = \arg\max_{y_i} P_\theta(y_i|a_i, X).$$

(8)

3.2 Joint Optimal Solution

The decoding algorithm described above can be summarized as the following process:

$$A^* = \arg\max_A P_\theta(A|X),$$

$$Y^* = \arg\max_Y P_\theta(Y|X, A^*).$$

(9)

Even though the algorithm now finds the optimal decoding path, the translation on this path may have low confidence, resulting in a low joint probability $P_\theta(A, Y|X)$. We further improve the decoding algorithm to search for both decoding paths and translations, which guarantees to find the joint optimal solution:

$$A^*, Y^* = \arg\max_{A,Y} P_\theta(A, Y|X).$$

(10)

Notice that when the path $A$ is given, we can easily find the most probable translation $Y$ with argmax decoding. Let $Y^A$ denote the argmax decoding result under path $A$, where

$$y_i^A = \arg\max_{y_i} P_\theta(y_i|a_i, X)$$

is the $i$-th word of $Y^A$. Then we can simplify our objective with $Y^A$:

$$\max_{A,Y} P_\theta(A, Y|X) = \max_{A,Y} P_\theta(A|X)P_\theta(Y|X, A)$$

$$= \max_A P_\theta(A|X)\max_Y P_\theta(Y|X, A)$$

$$= \max_A P_\theta(A|X)\max_Y P_\theta(Y^A|X, A)$$

$$= \max_A P_\theta(y_1^A|a_1, X)\prod_{i=1}^{M-1} E_{a_i, a_{i+1}} P_\theta(y_{i+1}^{a_{i+1}}|a_{i+1}, X)$$

$$= \max_A P_\theta(y_1^A|a_1, X)\prod_{i=1}^{M-1} E'_{a_i, a_{i+1}}$$

(11)

where we introduce a new transition matrix $E'$ with $E_{a_i, a_{i+1}} = E_{a_i, a_{i+1}} P_\theta(y_{i+1}^{a_{i+1}}|a_{i+1}, X)$. Compared to $\max_A P_\theta(A|X)$, the major difference is the transition matrix $E'$, which considers both the transition probability and the prediction probability. Therefore, we can still apply the Viterbi decoding framework to find the optimal joint solution.

We use ‘Viterbi’ to represent the Viterbi decoding algorithm proposed in section 3.1, and use ‘Joint-Viterbi’ to represent the improved algorithm in this section that finds the joint optimal solution. It is worth noting that Viterbi and Joint-Viterbi can be regarded as improvements to greedy decoding and lookahead decoding, respectively. Both greedy decoding and lookahead decoding consider the one-step probability and find the next token with $\arg\max_{a_i} P_\theta(a_i|X, a_{i-1})$ and $\arg\max_{y_i, a_i} P_\theta(y_i|a_i, X)P_\theta(a_i|a_{i-1}, X)$, respectively. In comparison, Viterbi and Joint-Viterbi consider the whole decoding path and guarantee to find the global optimal solution $\arg\max_{A,Y} P_\theta(A|X)$ and $\arg\max_{A,Y} P_\theta(A, Y|X)$, respectively.

3.3 Reranking with Length Penalty

After Viterbi decoding, we have a set of translations of different lengths that can be ranked to obtain the most probable one. However, argmax decoding is biased toward short translations and may even degenerate to an empty translation, as also observed in Stahlberg and Byrne (2019).

To solve this problem, we introduce the hyper-parameter $\beta$ for length normalization in Wu et al. (2016) and modify Equation 6 to divide by the length penalty term:

$$M = \max_i \frac{\alpha(i, L)}{i^\beta}.$$  

(12)

By changing the length penalty $\beta$ to different values, we now have the flexibility to control the translation length with little additional overhead, which is another appealing feature of our approach.

4 Experiments

4.1 Settings

We conduct experiments on WMT14 English–German (En→De, 4.5M pairs) and WMT17 Chinese–English (Zh→En, 20M pairs). These datasets are all encoded into subword units (Sennrich et al., 2016). We use the same preprocessed data and train/dev/test splits as Kasai et al. (2020). The translation quality is evaluated with sacreBLEU (Post, 2018) for WMT17 En-Zh and tokenized BLEU (Papineni et al., 2002) for other benchmarks. We use GeForce RTX 3090 to train models and measure translation latency. Our models are implemented based on the open-source toolkit of fairseq (Ott et al., 2019).

We strictly follow the hyper-parameter settings of Huang et al. (2022) to reimplement DA-Transformer. We adopt Transformer-base (Vaswani et al., 2017) as the model architecture. We set dropout to 0.1, weight decay to 0.01, and label smoothing to 0.1 for regularization. We use $\lambda = 8$
for the graph size and linearly anneal $\tau$ from 0.5 to 0.1 for the glancing training. For fair comparisons, we tune the length penalty in $[0.95, 1.05]$ to obtain a similar translation length as lookahead. We train all models for 300K steps, where each batch contains approximately 64K source tokens. All models are optimized by Adam (Kingma and Ba, 2014) with $\beta = (0.9, 0.999)$ and $\epsilon = 10^{-8}$. The learning rate warms up to $5 \times 10^{-4}$ and then begins to anneal it after 10K steps with the inverse square-root schedule. We calculate the validation BLEU scores every epoch and obtain the final model by taking an average of the best five checkpoints.

### 4.2 Main Results

As shown in Table 1, both Viterbi and Joint-Viterbi improve over their corresponding baseline. Joint-Viterbi achieves the best performance, which outperforms the previous lookahead strategy by 0.33 BLEU. Besides, it is worth noting that the Viterbi decoding process is highly parallelizable, which does not bring much overhead in the decoding and only reduces the speedup by less than $1 \times$.

### 4.3 Results with Knowledge Distillation

In this section, we evaluate the performance of our method with sequence-level knowledge distillation (Hinton et al., 2015; Kim and Rush, 2016), where the target side of the training set is replaced by the output of an autoregressive teacher model. Experimental results in Table 2 show that the differences between decoding strategies are relatively small.

Intuitively, we attribute this phenomenon to the improvement of model confidence. As knowledge distillation reduces the multi-modality of the dataset (Zhou et al., 2020; Sun and Yang, 2020), the model may become more confident in predicting target sentences, which makes the greedy strategy more likely to reach the optima. To verify this, we measure the average entropy of transition and prediction probabilities and evaluate the percentage of lookahead outputs that match the optima $\arg\max_{A,Y} P(A,Y \mid X)$ under their length. As Table 3 shows, DA-Transformer with distillation has smaller entropies and a larger percentage of optimal translations, which confirms our intuition.

![Image](image_url)
Method Lookahead Joint-Viterbi
Log-prob −4.39 −4.14
Percentage 24.4% 41.6%

Table 4: Probability analysis of on Lookahead and Joint-Viterbi decoding on WMT14 En-De test set.

by optimizing the joint probability.

4.5 Effect of Length Penalty

Viterbi decoding is capable of flexibly controlling the output length with the length penalty $\beta$. To show the effect of the length penalty, we change the value of $\beta$ in Joint-Viterbi to decode the WMT17 Zh-En test set and report the corresponding BLEU scores and average output lengths in Figure 1. It shows that the length penalty can almost linearly control the output length, which can help us obtain satisfactory translations. Generally, Viterbi decoding can obtain better performance when the output length is closer to the reference length. If there is no length penalty, only finding outputs with the maximum joint probability will break the translation quality with extremely small output lengths.

Figure 1: The effect of length penalty $\beta$ measured on WMT17 Zh-En test set.

5 Related Works

Most non-autoregressive models can directly find the most probable output with argmax decoding, which is the fastest decoding algorithm. However, models of this type usually suffer from the multi-modality problem (Gu et al., 2018), leading to severe performance degradation. A relatively more accurate method is noisy parallel decoding, which requires generating multiple translation candidates and greatly increases the amount of computation.

Many efforts have been made to address the multi-modality problem, including latent models (Kaiser et al., 2018; Ma et al., 2019; Shu et al., 2020; Bao et al., 2021, 2022), alignment-based models (Gu et al., 2018; Ran et al., 2021; Song et al., 2021), and better training objectives (Shao et al., 2019, 2020; Shan et al., 2021; Ghazvininejad et al., 2020; Du et al., 2021; Shao et al., 2021). However, these techniques are still not powerful enough, which heavily rely on knowledge distillation (Kim and Rush, 2016).

Some researchers seek iterative decoding approaches to improve translation quality. Work in this area includes semi-autoregressive decoding (Wang et al., 2018), iterative refinement (Lee et al., 2018), mask-predict decoding (Ghazvininejad et al., 2019), Levenshtein Transformer (Gu et al., 2019), multi-thread decoding (Ran et al., 2020), Imputer (Saharia et al., 2020), and rewriting (Geng et al., 2021). Although their translations are of better quality, they are criticized for being slow at inference time (Kasai et al., 2021).

Recently, latent alignment models like CTC (Libovický and Helcl, 2018; Saharia et al., 2020) and DA-Transformer (Huang et al., 2022) achieved impressive performance and received a lot of attention. Beam search is an useful decoding strategy for latent alignment models (Kasner et al., 2020; Gu and Kong, 2020; Zheng et al., 2021; Shao et al., 2022; Huang et al., 2022; Shao and Feng, 2022). It brings considerable improvements but also reduces the decoding speed.

Viterbi decoding has also been used in non-autoregressive models. In CRF-based NAT models, Viterbi decoding is applied to find the most probable output (Sun et al., 2019; Sun and Yang, 2020).

6 Conclusion

The current decoding strategies of DA-Transformer need to apply a sequential decision process, which harms the global translation accuracy. In this paper, we propose a Viterbi decoding framework for DA-Transformer to find the joint optimal solution of the translation and decoding path and further demonstrate its effectiveness on multiple benchmarks.

7 Acknowledgement

We thank the anonymous reviewers for their insightful comments. We thank Fei Huang for helping us open source code.

Limitations

The major limitation of our method is that it cannot find the most probable translation
argmax\textsubscript{Y} P(Y|X) but alternatively finds the joint optimal solution argmax\textsubscript{A,Y} P(A,Y|X). However, as we show in section 4.4, outputs with higher joint probability usually also have higher translation probability, suggesting that optimizing the joint probability is helpful.

Another limitation is that the improvements of our method are smaller in the knowledge distillation setting. However, the main advantage of DA-Transformer is that it does not heavily rely on knowledge distillation and achieves superior performance on raw data, which makes the impact of this limitation small.

References


