Enable Fast Sampling for Seq2Seq Text Diffusion

Pan Liu, Xiaohua Tian, Zhouhan Lin*
Shanghai Jiao Tong University
{wslp1999, xtian}@sjtu.edu.cn lin.zhouhan@gmail.com

Abstract

Diffusion models exhibit promising capacity for generating high-quality text. However, owing to the curved nature of the generation path, they necessitate traversing numerous steps to guarantee high text quality. In this paper, we propose an efficient model FMSeq[1], which utilizes flow matching to straighten the generation path, thereby enabling fast sampling for diffusion-based seq2seq text generation. Specifically, we construct transport flow only on the target sequences to adapt the diffusion-based model to flow matching. Furthermore, we explore different settings and identify target-parameterization, self-conditioning, and time-difference as three effective techniques to improve the generation quality under a few steps. Experiments on four popular tasks demonstrate that FMSeq generates texts of comparable quality to the SOTA diffusion-based DiffuSeq in just 10 steps, achieving a 200-fold speedup.

1 Introduction

Sequence-to-sequence (seq2seq) text generation (Wu et al., 2016; Sutskever et al., 2014; Bahdanau et al., 2014) is a fundamental setting in NLP, encompassing a wide range of practical downstream tasks including machine translation (Bahdanau et al., 2014; Wu et al., 2016), text summarization (Rush et al., 2015; Nallapati et al., 2016), and conversational modeling (Shao et al., 2017). Existing text generation methods predominantly utilize the autoregressive (AR) approach (Vaswani et al., 2017; Radford et al., 2019), which generates target tokens one by one. While this method captures sequential dependencies and hence produces high-quality texts, it is time-consuming. Therefore, non-autoregressive (NAR) generation methods have been proposed. These methods generate all tokens in parallel, significantly reducing inference latency (Gu et al., 2017, 2019). However, NAR models generally underperform AR ones on text generation accuracy due to the issue of conditional total correlation (Huang et al., 2022).

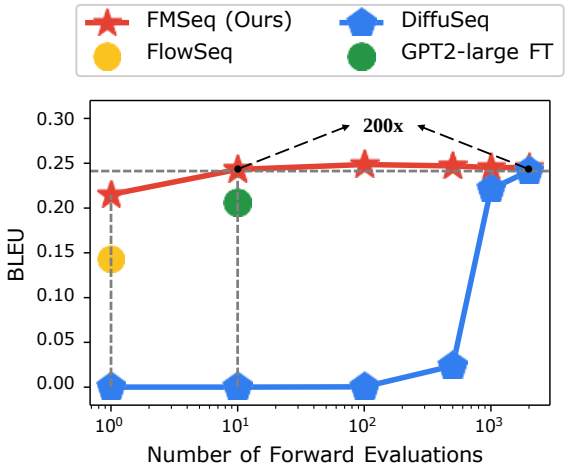

Figure 1: BLEU Scores of different models for the paraphrase task on the QQP dataset. Our FMSeq beats all the models when using a single sampling step and achieves comparable performance to DiffuSeq (2000 steps) with only 10 steps.

More recently, a new NAR generative paradigm, diffusion models (Ho et al., 2020; Song et al., 2020) has exhibited remarkable generative performance across various domains (Dhariwal and Nichol, 2021; Rombach et al., 2022; Chen et al., 2020; Hu et al., 2024b) and are increasingly gaining recognition in the field of NLP (Li et al., 2022; Gong et al., 2022). DiffuSeq (Gong et al., 2022) represents a pioneering effort in leveraging diffusion models for seq2seq text generation, demonstrating that diffusion model-based methods can achieve generation performance on par with AR models owing to their robust modeling capabilities.

However, as a class of NAR methods, diffusion models' sampling speed is not fast. To achieve a higher BLEU score than GPT2-large (Radford

*Corresponding author
[1]Code is released at https://github.com/Peacer68/FMSeq.git.

Findings of the Association for Computational Linguistics: EMNLP 2024, pages 8495–8505

et al., 2019), DiffuSeq requires 1000 sampling steps, which makes the number of tokens generated per second only half of GPT2-large. A recently proposed method, flow matching (Lipman et al., 2022; Liu et al., 2022) tries to speed up the sampling via straightening the generation path and has greatly improved the generation efficiency of diffusion-based models across multiple fields, including image generation (Lipman et al., 2022; Hu et al., 2024c; Dao et al., 2023), point cloud generation (Wu et al., 2023), audio synthesis (Guo et al., 2023) and beyond (Hu et al., 2023; Gui et al., 2024). However, in the field of text generation, when the standard diffusion process of DiffuSeq is directly replaced with flow matching, the quality of the generated texts significantly declines, rendering the flow matching-based model unusable, which is evidenced in Hu et al. (2024a).

In this paper, we find that the root of the above issue is the objective of recovering source sequences, which is not suitable for flow matching (see Section 3). Therefore, we eliminate this objective and construct transport flow only on the target sequences. Then we design a combined loss to fit the transport flow and simultaneously keep the trainable embedding from collapsing. To further improve the generation quality of FMSeq with a few steps, we explore different settings and identify target-parameterization, self-conditioning (Chen et al., 2022), and time-difference as three effective techniques. Experiments on four popular tasks demonstrate that FMSeq enables faster sampling compared to other models while generating texts of comparable quality. As shown in Fig. 1, in the paraphrase task FMSeq beats all the models on BLEU score when using a single sampling step and achieves comparable results with DiffuSeq (2000 steps) with only 10 steps.

In summary, our work makes the following contributions: (a) we utilize flow matching in seq2seq text generation and propose an efficient generative model FMSeq; (b) we explore different settings and propose three simple but effective techniques to further enhance the quality of generated texts with only a few sampling steps; (c) experiments show that FMSeq generates texts of comparable quality to DiffuSeq in just 10 steps, achieving a 200-fold speedup.

2 Related Work

2.1 Diffusion Models for Text Generation

The primary difficulty in integrating diffusion models into NLP lies in the discrete nature of text, which contrasts with the continuous space in which diffusion models operate. There are two primary approaches to tackle this issue: discrete diffusion models (Hoogeboom et al., 2021; Austin et al., 2021) and embedding diffusion models (Li et al., 2022). Discrete diffusion models generalize the diffusion process to a discrete state space. In the forward process, each token has a probability of being corrupted to an absorbing or random token, and the reverse process refines the noisy token sequences. In contrast, embedding diffusion models operate within a continuous embedding space. These embeddings are trained end-to-end, as opposed to being inherited from pre-trained language models, resulting in improved generation performance (Li et al., 2022; Gong et al., 2022). Compared to discrete diffusion models, embedding diffusion models leverage the benefits arising from advancements in continuous diffusion techniques within the realm of image processing, thereby enhancing controllability and flexibility (Strudel et al., 2022). Hence, we also operate on learnable continuous embedding space.

2.2 Flow Matching for Text Generation

To accelerate the sampling speed of diffusion models, flow matching (Lipman et al., 2022; Liu et al., 2022) reformulates the standard diffusion process. Instead of diffusing the data distribution through either SDE or intricate ODE (Song et al., 2023), flow matching transforms noise distribution into data distribution with probability flow that is as straight as possible: the transformation path follows linear interpolation between the data and noise. With a straighter generation path, flow matching requires only a few Euler discretization steps to produce a data sample of satisfactory quality.

Flow matching has been utilized in multiple fields to improve the generation efficiency of diffusion-based models (Lipman et al., 2022; Wu et al., 2023; Guo et al., 2023) and FlowSeq (Hu et al., 2024a) is the first work to integrate it into text generation. Building upon the foundation laid by DiffuSeq (Gong et al., 2022), FlowSeq introduces a combination of velocity estimation loss on both the source and target sequences and anchor loss (Gao et al., 2022a) to make flow matching-

based models generate readable texts. Although FlowSeq requires only a single step to generate texts, the quality of generation on some tasks is much lower than DiffuSeq, e.g., paraphrase (14.30 BLEU vs 24.13 BLEU). Moreover, when we increase the number of generation steps to 10, the quality of generation becomes worse unexpectedly (5.00 BLEU). Different from FlowSeq, we propose to construct transport flow only on the target sequences and the final obtained model FMSeq can generate texts of comparable quality to DiffuSeq on all the tasks and support step-quality trade-off.

3 Preliminary and Analyses

3.1 Preliminary

Diffusion models consist of two processes: in the forward process they incrementally introduce noise into the data using Gaussian perturbations and in the reverse process they generate samples by removing the noise through iterative denoising steps. Denoting the data distribution as $p_{\text{data}}(\mathbf{z})$, diffusion models start by diffusing $p_{\text{data}}(\mathbf{z})$ with a stochastic differential equation (SDE) (Song et al., 2020; Karras et al., 2022):

$$\mathrm{d}\mathbf{z}_t = \mu(\mathbf{z}_t, t)\mathrm{d}t + \sigma(t)\mathrm{d}\varepsilon_t, \tag{1}$$

where $t \in [0, T]$, $T > 0$ is a fixed constant, $\mu(\cdot, \cdot)$ and $\sigma(\cdot)$ are the drift and diffusion coefficients respectively, and $\{\varepsilon_t\}_{t\in[0,T]}$ denotes the standard Brownian motion. Let the distribution of $\mathbf{z}_t$ be denoted as $p_t(\mathbf{z})$ and a generation path in the reverse process can be solved with the reverse form of 1 or Probability Flow ordinary differential equation (PFODE) (Song et al., 2020):

$$\mathrm{d}\mathbf{z}_t = \left[\mu(\mathbf{z}_t, t) - \frac{1}{2}\sigma(t)^2\nabla \log p_t(\mathbf{z}_t)\right]\mathrm{d}t. \tag{2}$$

The generation path of both SDE and PFODE is highly curved, and hence the model necessitates traversing numerous steps to mitigate path distortion and ensure the quality of generated data.

Flow matching tries to accelerate sampling via straightening the generation path. It reformulates the standard diffusion process and exploits linear interpolation to model the probability flow between noise distribution and data distribution:

$$\mathbf{z}_t = (1 - t)\mathbf{z}_0 + t\mathbf{z}_1, \tag{3}$$

where $t \in [0, 1]$, $\mathbf{z}_0 \sim p_{\text{data}}(\mathbf{z})$ and $\mathbf{z}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The generation path follows the ODE $\mathrm{d}\mathbf{z}_t = (\mathbf{z}_1 - \mathbf{z}_0)\mathrm{d}t$, which is non-causal since updating $\mathbf{z}_t$ requires the information of the destination $\mathbf{z}_0$. By setting the velocity field $\mathbf{v}(\mathbf{z}_t, t)$ to drive the flow to follow the direction $(\mathbf{z}_1 - \mathbf{z}_0)$ as much as possible, the linear path follows a new ODE flow $\mathrm{d}\mathbf{z}_t = \mathbf{v}(\mathbf{z}_t, t)\mathrm{d}t$, eliminating the dependency on $\mathbf{z}_1$ and rendering it causal.

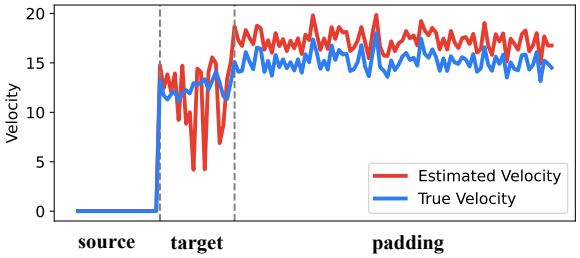

Figure 2: Estimated velocity on three parts of the input sequences. On the target sequences the estimation is near to zero, which is far from the true velocity.

3.2 Problem Statement

In sequence-to-sequence text generation, there is a source sequence $\mathbf{w}^x = \{w_1^x, ..., w_m^x\}$, and a target sequence $\mathbf{w}^y = \{w_1^y, ..., w_n^y\}$. The objective is to generate $\mathbf{w}^y$ given $\mathbf{w}^x$.

3.3 Analyses

DiffuSeq models this problem with a classifier-free diffusion process and learns the joint distribution of $\mathbf{w}^x$ and $\mathbf{w}^y$ in its implementation[2]. In the forward process, only $\mathbf{w}^y$ is noised and $\mathbf{w}^x$ remains unchanged, and so the model only learns to easily repeat the input $\mathbf{w}^x$, while the learning of target sequences is much more difficult, which requires estimating the distribution of $\mathbf{w}^y$.

When we apply this setting to flow matching, the model needs to estimate the velocity of $\mathbf{w}^x$ and $\mathbf{w}^y$. The velocity of $\mathbf{w}^x$ is zero since it just needs to stay on the spot but the velocity of $\mathbf{w}^y$ is nonzero and difficult to learn, which transforms the noise distribution into the data distribution conditioned on $\mathbf{w}^x$. The estimation of zero velocity makes the model degenerate and reduces the model's representation capacity. In Fig. 2, we probe the estimated velocity when the target sequences are still noisy and observe that it is very close to the true velocity on the source sequences and the padding with simple patterns, but on the target sequences with complex patterns, it is still close to zero, which is far from the true velocity. Therefore, the objective of source

[2]https://github.com/Shark-NLP/DiffuSeq.git

sequence recovery is not suitable for flow matching.

4 Method

We propose FMSeq to utilize flow matching into seq2seq text generation, the workflow of which is shown in Fig. 3.

4.1 FMSeq

We employ an encoder-only Transformer (Vaswani et al., 2017) to model the conditional distribution $p(\mathbf{w}^y|\mathbf{w}^x)$ and the sample is $\mathbf{w}^z = \mathbf{w}^{x \oplus y}$, i.e., the concatenation of $\mathbf{w}^x$ and $\mathbf{w}^y$. Denoting the embedding function as $\textsc{Emb}(\cdot)$ and the embedding matrix ϕ, we operate on $\mathbf{z}_0 = \textsc{Emb}(\mathbf{w}^z)$ and the generated embedding of target part $\hat{\mathbf{y}}$ is rounded into discrete tokens by multiplying ϕ^T.

In contrast to DiffuSeq and FlowSeq, the transport flow is only constructed on $\mathbf{y}_0 = \textsc{Emb}(\mathbf{w}^y)$:

$$\mathbf{y}_t = (1-t)\mathbf{y}_0 + t\mathbf{y}_1, \tag{4}$$

where $t \in [0,1]$ and $\mathbf{y}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. With such a flow, $\mathbf{y}_0$ can be recovered starting from $\mathbf{y}_1$ using the ODE $\mathrm{d}\mathbf{y}_t = \mathbf{v}_t \mathrm{d}t$, where $\mathbf{v}_t = \mathbf{y}_1 - \mathbf{y}_0$ is constant. We encourage the model to estimate the velocity according to $\mathbf{z}_t = \mathbf{x}_0 \oplus \mathbf{y_t}$, where the condition $\mathbf{x}_0 = \textsc{Emb}(\mathbf{w}^x)$ remains unchanged. Denote the model as $\mathbf{u}_\theta$ and then the flow matching loss is:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t,\mathbf{y}_1,(\mathbf{x}_0,\mathbf{y}_0)} ||\mathbf{u}_\theta(\mathbf{z}_t, t) - (\mathbf{y}_1 - \mathbf{y}_0)||^2. \tag{5}$$

Flow matching loss backpropagates gradients to both the embedding matrix and the model parameters, which can easily cause the embedding matrix to collapse since when all word vectors are the same, the data distribution becomes a Dirac distribution that is very easy to model. Therefore, we additionally introduce a rounding loss to regularize the embedding learning. (Li et al., 2022):

$$\mathcal{L}_{\text{R}} = -\mathbb{E}_{\mathbf{z}_0} \log p_\phi(\mathbf{w}^z|\mathbf{z}_0). \tag{6}$$

The rounding loss is minimal when all the word embeddings are orthogonal to each other, and hence this approach can effectively alleviate embedding collapse. The final end-to-end training loss is:

$$\begin{aligned}\mathcal{L}_{\text{E2E}} = \mathbb{E}_{t,\mathbf{y}_1,(\mathbf{x}_0,\mathbf{y}_0)}[&||\mathbf{u}_\theta(\mathbf{z}_t, t) - (\mathbf{y}_1 - \mathbf{y}_0)||^2 \\ &- \log p_\phi(\mathbf{w}^z|\mathbf{z}_0)].\end{aligned} \tag{7}$$

4.2 Training and Sampling Methods

Parameterization. A variant of $\mathbf{v}$-parameterized flow matching is $\mathbf{y}_0$-parameterized, and there is no essential difference between them since they can be obtained from each other linearly ($\mathbf{y}_0 = \mathbf{y}_t - t\mathbf{v}_t$). We test both variants and find that the $\mathbf{y}$-parameterized model performs better, which is adopted in the experiments.

Self-Conditioning. Self-conditioning (SED) is first proposed in Analog Bits (Chen et al., 2022) to improve the sample quality of Bit diffusion models, and the following research finds that it also yields improved results for continuous diffusion models. We find that it is also helpful to FMSeq. In the $\mathbf{y}_0$-parameterized case, we directly take the estimated $\hat{\mathbf{y}}$ as the self-condition and add another input $\mathbf{z}_c = \mathbf{x}_0 \oplus \hat{\mathbf{y}}$ into the model. The approximation error of $\hat{\mathbf{y}}$ always exists, especially in the early stages of training, and hence we skip the estimation and set $\mathbf{z}_c = \mathbf{x}_0 \oplus \mathbf{0}$ with some probability (e.g., 50%), which is equivalent to the model without self-condition and can also help to remain computationally efficient.

Time Difference. It is worth noting that when t is close to zero, the noisy $\mathbf{y}_0$ can be perfectly rounded to $\mathbf{w}^y$, so the model cannot learn enough effective information at small time steps. As shown in Fig. 4, we can see that with initialized embedding, the word error rate (wer) after rounding is always zero until $t = 0.5$. Even with the learned embedding, wer starts to be greater than zero after $t = 0.16$. This results in that during sampling, the forward evaluations at small t are of little use. To tackle this issue, we propose a time difference mechanism: replace the input t with $t + \Delta t$ to fully utilize the ability of the model at large time steps.

Putting it together. We summarize the training and sampling process of FMSeq in Algorithm 1 and 2.

5 Experiments

5.1 Experimental Setup

Tasks and Datasets. Seq2seq text generation covers a wide range of tasks, and we choose four typical and popular tasks:

- Paraphrase aims to generate an alternative surface form in the same language to express the same semantic content. We adopt the widely

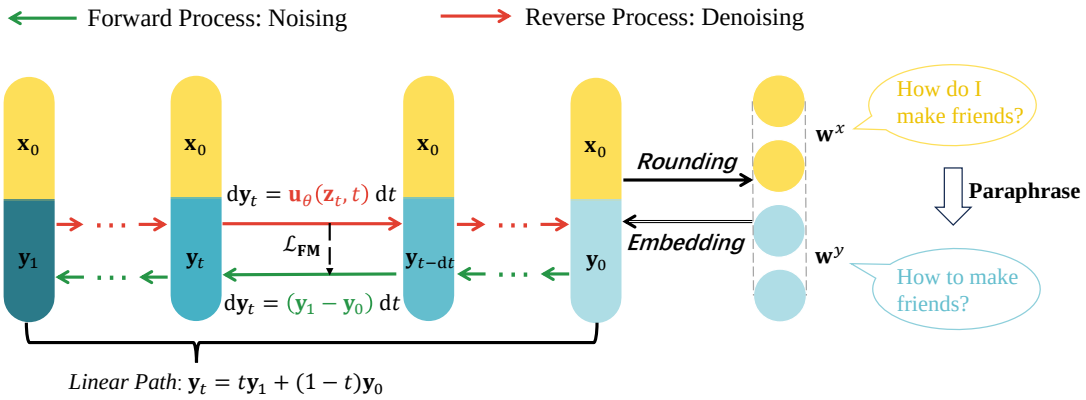

Figure 3: Workflow of FMSeq. We utilize embedding to map the discrete token space into a continuous space. The forward process diffuses the target embedding along a linear path, and the model fits the velocity of the target part conditioned on clean source embedding and noisy target embedding.

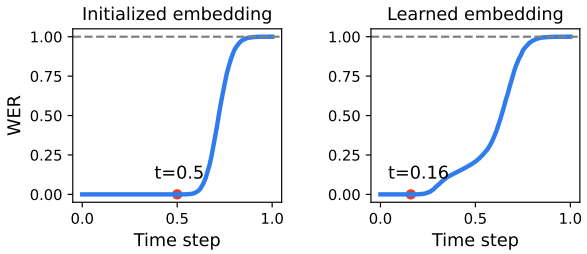

Figure 4: Word error rate (WER) at different time steps in the forward process. The lower the WER is, the closer to lossless the rounded noisy embeddings are.

used QQP dataset [3], sourced from the community question answering forum Quora, with 147K positive pairs.

- Question generation aims to generate questions given an answer as input. We use the dataset Quasar-T (Dhingra et al., 2017) preprocessed by (Gong et al., 2022), consisting of 119k training samples and 10k testing samples.

- Open domain dialogue aims to generate informative responses given a dialogue context. We use Commonsense Conversation Dataset (Zhou et al., 2018), which is extracted from Reddit single-round dialogues, with over 3 million conversational pairs.

- Text simplification aims to revise the complex text into sequences with simplified grammar and word choice. We use the dataset proposed by Jiang et al. (2020), which consists of 677K complex-simple sentences with revision alignment.

[3]https://www.kaggle.com/c/quora-question-pairs

Algorithm 1 Training

repeat
  $(\mathbf{w}^x, \mathbf{w}^y) \sim p_{\text{data}}$, padded and embedded into $(\mathbf{x}_0, \mathbf{y}_0)$
  $t \sim \text{Uniform}([0, 1])$
  $\mathbf{y}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $r \sim \text{Uniform}([0, 1])$
  $\mathbf{z}_t = \mathbf{x}_0 \oplus [(1-t)\mathbf{y}_0 + t\mathbf{y}_1]$
  $\hat{\mathbf{y}} = \mathbf{u}_\theta(\mathbf{z}_t, \mathbf{x}_0 \oplus \mathbf{0}, t)$ if $r > 0.5$ else $\mathbf{0}$
  Take gradient descent step on
  $(||\mathbf{u}_\theta(\mathbf{z}_t, \mathbf{x}_0 \oplus \hat{\mathbf{y}}, t) - \mathbf{y}_0||^2) - \log p_\phi(\mathbf{w}^z|\mathbf{z}_0)$
until converged

Algorithm 2 Sampling

Input: N steps, $\mathbf{x}_0$, $\mathbf{y}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, time difference Δt
$\hat{\mathbf{y}} = \mathbf{u}_\theta(\mathbf{x}_0 \oplus \mathbf{y}_1, \mathbf{x}_0 \oplus \mathbf{0}, 1)$, $\hat{\mathbf{y}}_t = \mathbf{y}_1$, $t' = 1$
for $t = \frac{N-1}{N}, \ldots, \frac{1}{N}$ **do**
  $\hat{\mathbf{y}}_t = \hat{\mathbf{y}}_t - \frac{1}{N}\frac{\hat{\mathbf{y}}_t - \hat{\mathbf{y}}}{t'}$, $t' = t$
  $\hat{\mathbf{y}} = \mathbf{u}_\theta(\mathbf{x}_0 \oplus \hat{\mathbf{y}}_t, \mathbf{x}_0 \oplus \hat{\mathbf{y}}, min(1, t + \Delta t))$
end for
Output: $\hat{\mathbf{y}}$

Baselines. We compare FMSeq to two groups of baselines, covering both autoregressive (AR) and non-autoregressive (NAR) architectures. The first group of methods adopts encoder-decoder Transformer (Vaswani et al., 2017) which is well-studied for seq2seq tasks and another decoder-only GPT2-large (Radford et al., 2019) which is a finetuned pre-trained language model (PLM) demonstrating great success in almost all seq2seq tasks. For the NAR group of baselines, we consider three iterative NAR models: Levenshtein distance-based LevT (Gu et al., 2019), diffusion model-based DiffuSeq (Gong et al., 2022) and a recently proposed flow matching-based FlowSeq (Hu et al., 2024a).

Evaluation. We evaluate the generated sequences from two aspects: quality and diversity. To evaluate quality, standard BLEU (Papineni et al.,

2002) and ROUGE (Lin, 2004) are utilized for string-based similarity and BERTScore (Zhang et al., 2019) for sentence-based similarity. As for diversity, we adopt widely used unique unigram (dist-1).

Implementation Details. Our model adopts the same architecture as DiffuSeq and FlowSeq, which is based on the 12 layers of encoder-only Transformer with 12 attention heads. The time step embedding is incorporated in a similar way to the position embedding. The maximum length of $\mathbf{z}$ is 128, with embedding dimension $d = 128$, diffusion steps $T = 2,000$. We train the model for the same number of iterations with a smaller batch size of 1024.

To improve the quality of generation, we adopt the widely used Minimum Bayes Risk (MBR) decoding strategy (Koehn, 2004). We first generate 10 candidate samples from different random noise and select the best output sequence that achieves the minimum expected risk under the negative BLEU score.

5.2 Main Results

As presented in Table 1, it can be concluded that FMSeq achieves comparable or even higher generation quality than other baselines with only 10 sampling steps. Furthermore, FMSeq outperforms several baselines on three of the four tasks, even when the number of sampling steps is reduced to one.

As we can see from Table 1, compared to DiffuSeq (2000 steps), FMSeq (10 steps) outperforms in at least one quality metric across all four tasks. The performance gap between them across all tasks is small. It is worth noting that in the question generation task, FMSeq (10 steps) even achieves better overall performance than DiffuSeq (2000 steps). These results demonstrate that FMSeq achieves a 200-fold speedup to DiffuSeq. Compared to the conventional NAR model LevT, FMSeq (10 steps) significantly outperforms it, achieving relative improvements of over 50% in both the BLEU score for the question generation task and the ROUGE-L score for the open-domain dialogue task. These results highlight the substantial potential of flow-matching-based NAR models. Furthermore, even when compared with AR models, FMSeq (10 steps) still delivers comparable performance and achieves better overall performance in question generation and text simplification tasks.

When we focus on single step generation, it can be seen that FMSeq is comparable to FlowSeq (wins on question generation and paraphrase tasks, but loses on open-domain dialogue and text simplification tasks). Compared to other baselines, FMSeq (1 step) still outperforms NAR-LevT in question generation and text simplification tasks (BLEU and R-L).

Generation diversity is an additional advantage of the diffusion-based model, and from the value of dist-1, we can see that FMSeq, based on flow matching, inherits this. We further demonstrate this point with some generated samples in Table 2.

5.3 Sampling Steps vs Generation Quality

We further study the generation quality of DiffuSeq, FlowSeq and FMSeq with different sampling steps, and the results in the paraphrase task are shown in Table 3. We can see that even when the number of steps is increased to 100, the generation quality of DiffuSeq is still very poor. It is worth noting that the generation quality of FlowSeq inconsistently decreases when the sampling steps increase, the root of which is that its proposed anchor loss encourages the model to infer the original dataset by a single step and more steps lead to error accumulation. Among these models, FMSeq is the only model to support efficiency-quality trade-off with a few steps.

5.4 Ablation Study

Parameterization and Self-Conditiong. Table 4 shows the effectiveness of $\mathbf{y}_0$ parameterization and the self-conditioning mechanism. Across all evaluated tasks, the best setting is $\mathbf{y}_0$-parameterization with self-conditioning, and alterations to $\mathbf{v}$-parameterization or the removal of self-conditioning result in a degradation in generation quality, particularly pronounced in the paraphrase task.

Time difference. We vary the time difference parameter Δt from 0.0 to 1.0 linearly and $\Delta t = 0.0$ means that we eliminate this mechanism in the sampling process. The generation quality for different values of Δt is shown in Table 5 and we can see that when $\Delta t = 0.0$, the performance on all the metrics is almost the worst. The generation quality reaches the best around $\Delta t = 0.5$, which is consistent with the observation in section 4.2. The initialized random embedding makes the noisy $\mathbf{y}_t$ be rounded into the original $\mathbf{w}^y$ losslessly until

Tasks	Methods	NFE↓	BLEU↑	R-L↑	Score↑	dist-1↑	Len
Question Generation	Transformer-base	-	0.1663	0.3441	0.6307	0.9309	10.3
	GPT2-large FT	-	0.1110	0.3215	0.6346	0.9670	10.0
	NAR-LevT	-	0.0930	0.2893	0.5491	0.8914	6.9
	DiffuSeq	2000	0.1731 (2)	0.3665 (3)	0.6123 (4)	0.9056 (3)	11.5
	FMSeq (Ours)	10	0.1810 (1)	0.3669 (2)	0.6189 (3)	0.8966 (4)	12.8
	FlowSeq	1	0.1620	0.3700	0.5730	0.8330	11.8
	FMSeq (Ours)	1	0.1630	0.3544	0.5954	0.8951	12.3
Paraphrase	Transformer-base	-	0.2722	0.5748	0.8341	0.9748	11.2
	GPT2-large FT	-	0.2059	0.5415	0.8363	0.9819	9.5
	NAR-LevT	-	0.2268	0.5795	0.8344	0.9790	8.9
	DiffuSeq	2000	0.2413 (3)	0.5880 (1)	0.8365 (1)	0.9807 (2)	11.2
	FMSeq (Ours)	10	0.2430 (2)	0.5514 (4)	0.8115 (5)	0.9670 (5)	11.8
	FlowSeq	1	0.1430	0.4610	0.6690	0.8620	11.9
	FMSeq (Ours)	1	0.2150	0.5333	0.7468	0.9244	9.4
Open Domain Dialogue	Transformer-base	-	0.0189	0.1039	0.4781	0.7493	19.5
	GPT2-large FT	-	0.0125	0.1002	0.5293	0.9244	16.8
	NAR-LevT	-	0.0158	0.0550	0.4760	0.9726	4.1
	DiffuSeq	2000	0.0139 (3)	0.1056 (3)	0.5131 (2)	0.9467 (3)	13.6
	FMSeq (Ours)	10	0.0136 (4)	0.1113 (2)	0.4786 (3)	0.9065 (5)	21.5
	FlowSeq	1	0.0110	0.1190	0.3450	0.7090	30.7
	FMSeq (Ours)	1	0.0011	0.0495	0.3157	0.9991	2.75
Text Simplification	Transformer-base	-	0.2693	0.4907	0.7381	0.8886	18.5
	GPT2-large FT	-	0.2693	0.5111	0.7882	0.9464	15.4
	NAR-LevT	-	0.2052	0.4402	0.7254	0.9715	8.3
	DiffuSeq	2000	0.3622 (1)	0.5849 (1)	0.8126 (1)	0.9264 (4)	17.7
	FMSeq (Ours)	10	0.3190 (2)	0.5590 (2)	0.7916 (2)	0.9333 (3)	17.6
	FlowSeq	1	0.2628	0.5130	0.7075	0.8897	17.3
	FMSeq (Ours)	1	0.2270	0.4580	0.6636	0.8026	16.1

Table 1: The overall results of different methods on different seq2seq tasks. The first group of methods is autoregressive and the second group is non-autoregressive, including classic LevT and diffusion-based models. We rank the performance of DiffuSeq and FMSeq (10 steps) for ease of comparison. NFE means the number of forward evaluations.

Statement: *The Japanese yen is the official and only currency recognized in Japan.* ***Question***: *What is the Japanese currency?*	
GPT2-large FT * What is the basic unit of currency for Japan? * What is the Japanese currency * What is the basic unit of currency for Japan?	**NAR-LevT** * What is the basic unit of currency for Japan ? * What is the basic unit of currency for Japan ? * What is the basic unit of currency for Japan ?
DiffuSeq * What is the Japanese currency * Which country uses the “yen yen” in currency * What is the basic unit of currency?	**FMSeq** * What is the basic unit of currency for Japan? * Which country uses the "yen" for currency * What is the Japanese currency

Table 2: Sample outputs in Quasar-T test set, conditioned on the same **x**.

Steps	1	10	20	50	100
DiffuSeq	0/0/0.262	0/0/0.260	0/0/0.262	0/0/0.260	0/0.002/0.258
FlowSeq	0.143/0.461/0.669	0.050/0.181/0.459	0.060/0.191/0.472	0.058/0.184/0.471	0.051/0.182/0.464
FMSeq	0.215/0.533/0.745	0.243/0.551/0.811	0.244/0.551/0.809	0.247/0.555/0.812	0.249/0.559/0.819

Table 3: Sampling Steps vs Generation Quality in the paraphrase task. BLEU↑/ R-L↑/ Score↑ are listed.

Task	Paraphrase	Question Generation	Text Simplification
$\mathbf{y}_0$-parameterization w/ SED	0.243/0.551/0.811	0.181/0.367/0.619	0.319/0.559/0.792
$\mathbf{v}$-parameterization w/ SED	0.211/0.534/0.768	0.155/0.345/0.572	0.303/0.534/0.768
$\mathbf{y}_0$-parameterization w/o SED	0.215/0.523/0.779	0.178/0.359/0.606	0.309/0.545/0.775

Table 4: Ablation study of parameterization and self-conditioning. BLEU↑/ R-L↑/ Score↑ are listed.

Δt	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
BLEU↑	0.2265	0.2316	0.2444	0.2438	**0.2487**	0.2485	0.2395	0.2354	0.2386	0.2385	0.2378
R-L↑	0.5326	0.5413	0.5561	0.5521	0.5565	**0.5594**	0.5576	0.5527	0.5547	0.5534	0.5538
Score↑	0.7975	0.8075	0.8140	0.8130	0.8150	**0.8188**	0.8155	0.8123	0.8118	0.8120	0.8114
dist-1↑	0.9646	0.9684	0.9705	0.9673	0.9697	0.9713	0.9702	0.9702	0.9706	**0.9716**	0.9671

Table 5: The generation quality with different time difference parameters in the paraphrase task.

$t = 0.5$, and so the model capacity is insufficient at these small time steps. Since the objective of the model is always to estimate velocity or target, both of which are constant, we can fully utilize the model at large time steps to improve the generation quality. Note that the generation quality does not keep increasing with increased Δt and a possible reason is that the model tries to recover the target sequences only based on the source sequences when t is close to 1 and starts to refine the generated texts when t is relatively small ($\mathbf{y}_t$ cannot be rounded losslessly).

5.5 Anisotropy of Embedding

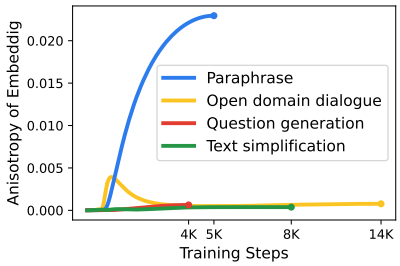

Figure 5: The anisotropy of embedding during training in all the tasks. Low anisotropy means that all the token embeddings are discriminative from each other.

To validate the effectiveness of rounding loss, we calculate the anisotropy of embedding, which is the mean of cosine similarity across all word pairs (Gao et al., 2022b). The lower the anisotropy, the better the embedding, making different words more distinguishable from each other. We show the change of embedding anisotropy during training in Fig. 5 and the result demonstrates that after training, the anisotropy score in all these tasks is near zero and the largest value is only around 0.025 (in paraphrase task). Therefore, the rounding loss effectively alleviates embedding collapse.

5.6 Generation Path

The key factor for FMSeq's superior performance is that the generation path is straightened. We demonstrate this by visualizing the generation paths of FMSeq and DiffuSeq. Since a point on the path represents all the token embeddings of a sentence, and these token embeddings are high-dimensional, we employ t-SNE (Van der Maaten and Hinton, 2008) to project them into a two-dimensional space, and the result is shown in Fig. 6, confirming the effectiveness of our proposed method.

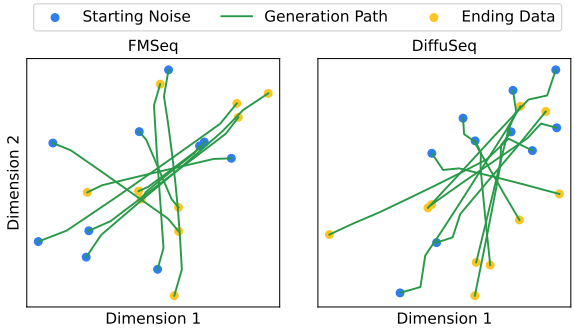

Figure 6: Differences between generation paths of FMSeq (ours) and DiffuSeq.

6 Conclusion

In this paper, we propose FMSeq, a novel seq2seq text generation model based on flow matching. We find that the objective of recovering the source sequences adopted in the diffusion-based model is unsuitable to flow matching and construct transport flow only on the target sequences. To fit the transport flow and simultaneously keep the trainable embedding from collapsing, we design a combined loss. To further improve the generation quality of FMSeq under a few steps, we explore different settings and identify target-parameterization, self-conditioning, and time-difference as three effective

techniques. Experiments in four popular NLP tasks demonstrate that FMSeq generates texts of comparable quality to DiffuSeq in just 10 steps, achieving a 200-fold speedup.

Limitations

Due to the random matching of the data and noise, the occurrence of intersections among linear paths is inevitable. The final learned velocity fields rewire the individual trajectories passing through the intersection points to avoid crossing, while also inducing curvature in the generation path. To further straighten the generation path and improve the performance of the one-step model, rectification (Liu et al., 2022) and other methods (Lee et al., 2023) can be utilized in our work. Therefore, flow matching-based models have the potential to outperform classic NAR models (Gu et al., 2019) on both quality and speed.

From an ethical perspective, the generated sentences may include inappropriate content that may potentially necessitate additional review by human inspectors.

Acknowledgment

This work was sponsored by the National Key Research and Development Program of China (No. 2023ZD0121402, 2020YFB1708700) and National Natural Science Foundation of China (NSFC) grant (No.62106143, 61922055, 61872233, 62272293).

References

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. 2021. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. 2020. Wavegrad: Estimating gradients for waveform generation. *arXiv preprint arXiv:2009.00713*.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. 2022. Analog bits: Generating discrete data using diffusion models with self-conditioning. *arXiv preprint arXiv:2208.04202*.

Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. 2023. Flow matching in latent space. *arXiv preprint arXiv:2307.08698*.

Prafulla Dhariwal and Alexander Nichol. 2021. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794.

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017. Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904*.

Z Gao, J Guo, X Tan, Y Zhu, F Zhang, J Bian, and L Difformer Xu. 2022a. Empowering diffusion model on embedding space for text generation. *arXiv preprint arXiv:2212.09412*.

Zhujin Gao, Junliang Guo, Xu Tan, Yongxin Zhu, Fang Zhang, Jiang Bian, and Linli Xu. 2022b. Difformer: Empowering diffusion models on the embedding space for text generation. *arXiv preprint arXiv:2212.09412*.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and LingPeng Kong. 2022. Diffuseq: Sequence to sequence text generation with diffusion models. *arXiv preprint arXiv:2210.08933*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2017. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. *Advances in neural information processing systems*, 32.

Ming Gui, Johannes S Fischer, Ulrich Prestel, Pingchuan Ma, Dmytro Kotovenko, Olga Grebenkova, Stefan Andreas Baumann, Vincent Tao Hu, and Björn Ommer. 2024. Depthfm: Fast monocular depth estimation with flow matching. *arXiv preprint arXiv:2403.13788*.

Yiwei Guo, Chenpeng Du, Ziyang Ma, Xie Chen, and Kai Yu. 2023. Voiceflow: Efficient text-to-speech with rectified flow matching. *arXiv preprint arXiv:2309.05027*.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. 2021. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in Neural Information Processing Systems*, 34:12454–12465.

Vincent Hu, Di Wu, Yuki Asano, Pascal Mettes, Basura Fernando, Björn Ommer, and Cees Snoek. 2024a. Flow matching for conditional text generation in a few sampling steps. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 380–392.

Vincent Tao Hu, Stefan Andreas Baumann, Ming Gui, Olga Grebenkova, Pingchuan Ma, Johannes S Fischer, and Björn Ommer. 2024b. Zigma: A dit-style zigzag mamba diffusion model. *arXiv preprint arXiv:2403.13802*.

Vincent Tao Hu, Wenzhe Yin, Pingchuan Ma, Yunlu Chen, Basura Fernando, Yuki M Asano, Efstratios Gavves, Pascal Mettes, Bjorn Ommer, and Cees GM Snoek. 2023. Motion flow matching for human motion synthesis and editing. *arXiv preprint arXiv:2312.08895*.

Vincent Tao Hu, Wei Zhang, Meng Tang, Pascal Mettes, Deli Zhao, and Cees Snoek. 2024c. Latent space editing in transformer-based flow matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 2247–2255.

Fei Huang, Tianhua Tao, Hao Zhou, Lei Li, and Minlie Huang. 2022. On the learning of non-autoregressive transformers. In *International Conference on Machine Learning*, pages 9356–9376. PMLR.

Chao Jiang, Mounica Maddela, Wuwei Lan, Yang Zhong, and Wei Xu. 2020. Neural crf model for sentence alignment in text simplification. *arXiv preprint arXiv:2005.02324*.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. 2022. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 388–395.

Sangyun Lee, Beomsu Kim, and Jong Chul Ye. 2023. Minimizing trajectory curvature of ode-based generative models. In *International Conference on Machine Learning*, pages 18957–18973. PMLR.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. 2022. Diffusion-lm improves controllable text generation. *Advances in Neural Information Processing Systems*, 35:4328–4343.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. 2022. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*.

Xingchao Liu, Chengyue Gong, and Qiang Liu. 2022. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Louis Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. 2017. Generating high-quality and informative conversation responses with sequence-to-sequence models. *arXiv preprint arXiv:1701.03185*.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. 2023. Consistency models. *arXiv preprint arXiv:2303.01469*.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2020. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.

Robin Strudel, Corentin Tallec, Florent Altché, Yilun Du, Yaroslav Ganin, Arthur Mensch, Will Grathwohl, Nikolay Savinov, Sander Dieleman, Laurent Sifre, et al. 2022. Self-conditioned embedding diffusion for text generation. *arXiv preprint arXiv:2211.04236*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Lemeng Wu, Dilin Wang, Chengyue Gong, Xingchao Liu, Yunyang Xiong, Rakesh Ranjan, Raghuraman Krishnamoorthi, Vikas Chandra, and Qiang Liu. 2023. Fast point cloud generation with straight flows. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9445–9454.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018. Commonsense knowledge aware conversation generation with graph attention. In *IJCAI*, pages 4623–4629.

Appendix

A Inference Speed

	running time (second/batch)	inference speed (tokens/second)
FMSeq with 10 steps	1.05	6481.32
FMSeq with 100 steps	7.58	923.89
DiffuSeq with 2000 steps	254.48	24.26

Table 6: The running time and the inference speed of FMSeq and DiffuSeq.

In diffusion models, the input and output sizes remain consistent at each inference step, making the practical runtime nearly linear with the number of forward evaluations (NFE). Since FMSeq uses the same architecture as DiffuSeq (and FlowSeq), we can conclude that FMSeq with 10 inference steps achieves a 200-fold speedup compared to DiffuSeq with 2000 steps, without the need to compare running time directly. To further address any concerns, we test the practical running time and inference speed of FMSeq with 10 steps and DiffuSeq with 2000 steps, and the result is shown in Table 6.

B Further Ablation Study

	BLEU↑	R-L↑	Bert-Score↑
FMSeq	0.243	0.551	0.811
w/o partial flow	0.067	0.332	0.601
w/o targte-parameterization	0.211	0.534	0.768
w/o SED	0.215	0.523	0.779
w/o time-difference	0.227	0.533	0.798

Table 7: The effects of each part in FMSeq (NFE $= 10$ in the paraphrase task).

We conduct further ablation study to further reveal which part contributes most to the success of FMSeq, and the results are shown in Table 7. The results demonstrates that partial flow contributes most to the success of FMSeq, which is introduced in detail in section 3.3 and 4.1, while the other three techniques including target-parameterization, SED, and time-difference are introduced to further enhance the performance of FMSeq. In another words, partial flow makes flow matching work and the other three techniques make flow matching work better.

C Machine Translation Task

The detailed result can be found at https://github.com/Peacer68/FMSeq.git.