Flow Matching for Conditional Text Generation in a Few Sampling Steps

Vincent Tao Hu[1,2] Di Wu[2] Yuki M Asano[2] Pascal Mettes[2]
Basura Fernando[3] Björn Ommer[1,†] Cees G. M. Snoek[2,†]

[1]LMU, Munich, DE
[2]University of Amsterdam, NL
[3]Institute of High-Performance Computing, Agency for Science, Technology and Research, Singapore

Abstract

Diffusion models are a promising tool for high-quality text generation. However, current models face multiple drawbacks including slow sampling, noise schedule sensitivity, and misalignment between the training and sampling stages. In this paper, we introduce FlowSeq, which bypasses all current drawbacks by leveraging flow matching for conditional text generation. FlowSeq can generate text in a few steps by training with a novel anchor loss, alleviating the need for expensive hyperparameter optimization of the noise schedule prevalent in diffusion models. We extensively evaluate our proposed method and show competitive performance in tasks such as question generation, open-domain dialogue, and paraphrasing.

1 Introduction

Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021b) have demonstrated impressive generative performances across many domains and have started to gain traction in the NLP domain as well. Diffusion models shine in their ability to generate diverse and high-quality samples but require many sampling steps leading to a long generation time (Li et al., 2022). Furthermore, they require careful tuning of hyperparameters such as the noise schedule (Gao et al., 2022; Yuan et al., 2022; Ye et al., 2023; Hoogeboom et al., 2023), time-step interval (Chen et al., 2023; Lin et al., 2023), and sampling algorithm (Tang et al., 2023). A suboptimal choice can drastically degrade the performance.

A recently proposed generative model, Flow Matching (Lipman et al., 2023; Liu et al., 2023; Neklyudov et al., 2022; Albergo et al., 2023; Tong et al., 2023), represents a superclass of diffusion models and has seen successful adoption to different applications such as image generation (Lipman et al., 2023; Hu et al., 2024b; Dao et al., 2023; Hu et al., 2023), video prediction (Davtyan et al., 2022), human motion generation (Hu et al., 2024a; Mehta et al., 2024) point cloud generation (Wu et al., 2022), and generative modelling on manifolds (Chen and Lipman, 2023). This model directly learns a vector field between the trajectory of instance data and random noise, formulating a nearly straight trajectory between data and noise which can significantly accelerate the sampling speed. In this paper, we propose FlowSeq, a flow matching model for sample-efficient text generation.

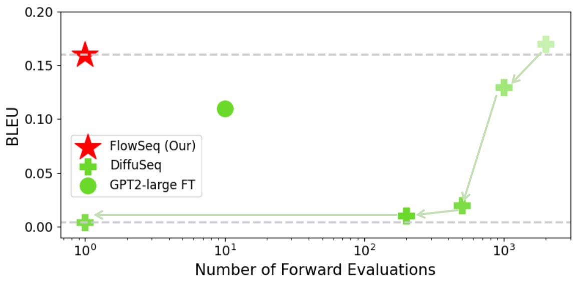

Figure 1: **FlowSeq:** Competitive BLEU scores using a single sampling step, demonstrated here for question generation on the Quasar-T dataset.

We perform generation at the continuous embedding space (Li et al., 2022; Dieleman et al., 2022; Gao et al., 2022) instead of the discrete space (Hoogeboom et al., 2021; Chen et al., 2023), for improved controllability (Li et al., 2022) and more flexibility (Strudel et al., 2022). To this end, a continuous vector field is learned to form a direct trajectory between the text embeddings and Gaussian noise. Additionally, we formulate an anchor loss to facilitate direct sampling from noisy data at any time step, thus further increasing sampling speed. As a result, our method yields strongly competitive performance on open domain dialogue, question generation, and paraphrasing tasks when compared to various baselines. As demonstrated in fig. 1, despite relying on as few as a *single*-step sampling, we manage to secure competitive results

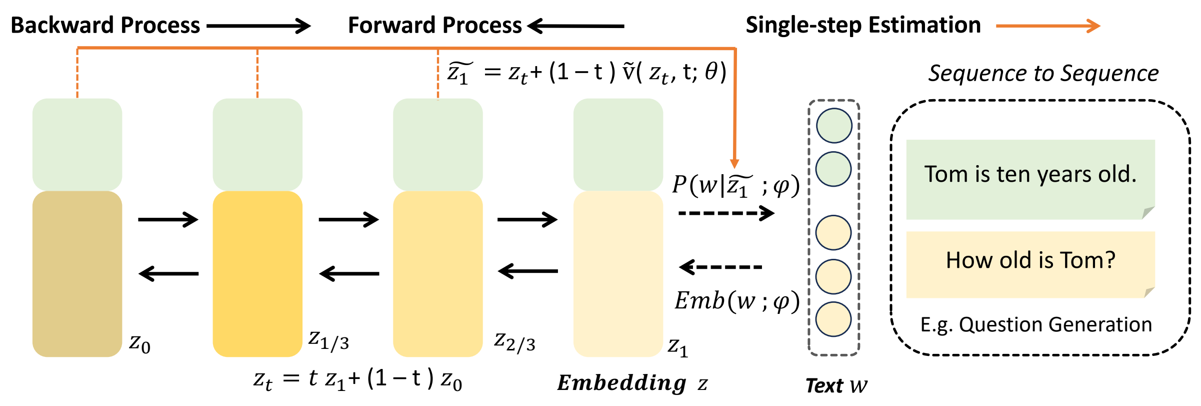

Figure 2: **FlowSeq framework .** We convert text representations to embeddings to facilitate flow matching on a continuous state. During the forward process, we uniformly corrupt the embeddings. We learn a vector field that can recover the original embeddings by a backward pass. Finally, we generate text using an `argmax` operation.

on the question generation task.

2 Related works

Diffusion Models for Text Modeling. Diffusion models have been widely applied to NLP in a non auto-regressive way (Zou et al., 2023; Li et al., 2023). They can be categorized into two classes: discrete diffusion, which performs the diffusion process in token levels (Hoogeboom et al., 2021; Chen et al., 2023), and embedding diffusion, which performs sequence-level diffusion (Li et al., 2022; Dieleman et al., 2022; Gao et al., 2022). Embedding diffusion is more favorable than token diffusion, due to the better parallel generation, text interpolation and token-level controls, robustness to input corruption (Zou et al., 2023). To adapt continuous diffusion models to discrete space, an embedding is typically learned between the discrete and continuous states. Then, diffusion is conducted on the continuous state using standard continuous diffusion models. Without specific restrictions on the design space, this approach can result in trivial embeddings. The Anchor loss in Difformer (Gao et al., 2022) is key to avoiding this collapse. Our method is inspired by these approaches but differs in its focus on increasing the sampling speed while minimizing performance degradation. We achieve this by utilizing flow matching models.

Fast sampling for generative models. Despite the emergence of diffusion models in NLP, thousands of steps are still required to achieve the desired quality. DDIM (Song et al., 2021a) formulates the sampling trajectory process as an ODE. FastDPM (Kong and Ping, 2021) bridges the gap between discrete and continuous timesteps, reducing the sampling process to hundreds of steps. Recently, several other works (Luhman and Luhman, 2021; Salimans and Ho, 2022; Gu et al., 2023; Song et al., 2023; Tong et al., 2023) have been proposed that use distillation to further boost the sampling speed. In contrast, we propose to utilize ODE sampling from flow matching (Lipman et al., 2023; Liu et al., 2023; Neklyudov et al., 2022). This method can formulate a straight trajectory between the Gaussian distribution and the real data distribution, potentially enabling faster sampling speed.

3 Method

Problem Statement. We address sequence-to-sequence text generation tasks. In particular, given a source sequence $\mathbf{w}^x = \{w_1^x, ..., w_m^x\}$ of length M, we aim to learn a generative model that can produce a target sequence $\mathbf{w}^y = \{w_1^y, ..., w_n^y\}$ of length N conditioned on the source sequence.

Flow Model for Text Generation. Next, we introduce our text generation framework. At a high-level, our approach views text generation as a process of transporting a noisy version of the text embedding to the corresponding clean version by following a learned trajectory. We model these trajectories with an Ordinary Differential Equation (ODE) that specifies the shortest transport path and utilize a neural network to fit the vector field of the ODE. Our proposed training pipeline is shown in fig. 2. To generate samples, one can start with a random Gaussian noise sample and then solve the ODE in reverse starting from the noise.

Text is inherently discrete but modeling discrete distributions with flow-based models can be challenging and may require compromises that lose some of the benefits like fast sampling. Inspired by prior studies (Li et al., 2022; Gong et al., 2022), we choose to model the problem in the continuous text embedding space. We define the text embedding function $\textsc{Emb}(\cdot;\phi)$ with the word embeddings $\phi \in \mathbb{R}^{V\times D}$ and a vocabulary size of V. Then, we map the discrete text $\mathbf{w}$ to continuous embeddings $\mathbf{z}$. The word embeddings ϕ can be learnable or fixed. To map continuous embeddings back to the discrete words, we choose the one with the most similar embedding $\mathbf{w}^z$ =argmax $(\tilde{\mathbf{z}}_1 \cdot \phi^T)$.

Before we insert the text embeddings into the neural network, we concatenate both the source and target sequences $\mathbf{z}{:=}\mathbf{x} \oplus \mathbf{y}$, $\mathbf{x} = \textsc{Emb}(\mathbf{w}^x;\phi) \in \mathbb{R}^{M\times D}$, $\mathbf{y} = \textsc{Emb}(\mathbf{w}^y;\phi) \in \mathbb{R}^{N\times D}$, $\mathbf{z} \in \mathbb{R}^{L\times D}$, and $L{=}M{+}N$.

Training. Our objective is to construct a transport flow that moves the text embedding from the Gaussian distribution to the data distribution. Formally, we denote $\mathbf{z}_0 \in \mathbb{R}^{L\times D}$ as Gaussian noise and $\mathbf{z}_1 \in \mathbb{R}^{L\times D}$ as the continuous text embedding derived from the discrete text. We denote with $\mathbf{v}_\theta$ the velocity field neural network of the ODE:

$$\mathrm{d}\mathbf{z}_t = \mathbf{v}(\mathbf{z}_t, t; \theta)\mathrm{d}t, \tag{1}$$

where $t \in [0,1]$, and $\mathbf{z}_t$ represents the intermediate states at time t. The velocity field $\mathbf{v}(\mathbf{z}_t, t;\theta) : \mathbb{R}^{L\times D} \times [0,1] \to \mathbb{R}^{L\times D}$ is modeled by a neural network with parameters θ. Given a noisy version of the text embedding $\mathbf{z}_t$ at time $t < 1$ the velocity field $\mathbf{v}_\theta$ should move it towards the true data $\mathbf{z}_1$. The shortest path for doing so is the straight line between $\mathbf{z}_1$ and $\mathbf{z}_0$. Thus, we encourage our velocity field to also follow the optimal path that is described by the ODE $\mathrm{d}\mathbf{z}_t{=}(\mathbf{z}_1{-}\mathbf{z}_0)\mathrm{d}t$, which leads to the training objective:

$$\min_\theta \int_0^1 \mathbb{E}\left[\|\mathbf{v}(\mathbf{z}_t, t;\theta) - (\mathbf{z}_1 - \mathbf{z}_0)\|^2\right]\mathrm{d}t, \tag{2}$$

$$\text{where} \quad \mathbf{z}_t = t\mathbf{z}_1 + (1-t)\mathbf{z}_0 \quad t \in [0,1]. \tag{3}$$

In practice, we do not optimize the loss in eq. (2) directly because of the nontrivial integral over $t \in [0,1]$. Instead, for each data sample $\mathbf{z}_1$, we randomly draw a $\mathbf{z}_0$ from Gaussian noise, and a time t in $[0,1]$ and minimize the following loss with equivalent optimum:

$$\min_\theta \mathbb{E}_{t,\mathbf{z}_0}\|\mathbf{v}(\mathbf{z}_t, t;\theta) - (\mathbf{z}_1 - \mathbf{z}_0)\|^2.$$

Anchor loss. Motivate by (Gao et al., 2022), to prevent the word embeddings from collapsing and to encourage straighter paths between $\mathbf{z}_1$ and $\mathbf{z}_t$ – which will reduce the sampling steps needed – we introduce an extra term to the training objective. The anchor loss is a cross-entropy loss based on the estimation of $\mathbf{z}_1$ using $\tilde{\mathbf{z}}_1 : -\log p(\mathbf{w}^{x\oplus y}|\tilde{\mathbf{z}}_1;\phi)$. This loss backpropagates gradients to both the word embeddings ϕ and the neural network parameters θ of the velocity field. The full training objective then becomes:

$$\begin{aligned}\min_{\theta,\phi} \mathbb{E}_{t,\mathbf{z}_0}\big[&||\mathbf{v}(\mathbf{z}_t, t;\theta) - (\mathbf{z}_1 - \mathbf{z}_0)||^2 \\ &- \log p(\mathbf{w}^{x\oplus y}|\tilde{\mathbf{z}}_1;\phi)\big],\end{aligned} \tag{4}$$

where $\tilde{\mathbf{z}}_1$ is an approximation of data $\mathbf{z}_1$ from starting point $\mathbf{z}_t$ with estimated velocity $\mathbf{v}(\mathbf{z}_t, t;\theta)$:

$$\tilde{\mathbf{z}}_1 = \mathbf{z}_t + (1-t)\mathbf{v}(\mathbf{z}_t, t;\theta). \tag{5}$$

During training, to achieve the conditioning operation, we only add noise to $\mathbf{y}$ instead of $\mathbf{x}$ (see appendix A).

Sampling. We focus on single-step sampling for maximum efficiency, the anchor loss is designed to facilitate one-step generation by ODE:

$$\tilde{\mathbf{z}}_1 := \mathbf{z}_0 + \tilde{\mathbf{v}}(\mathbf{z}_0, 0;\theta).$$

The discrete token id is retrieved by argmax$(\tilde{\mathbf{z}}_1 \cdot \phi^T)$.

Algorithm 1 Single-Step Flow for Conditional Text Generation

```
Input: dataset $\mathcal{D}$, a neural velocity field $\mathbf{v}_\theta$ with parameter $\theta$.
Discrete to Continuous state:
$z_0 = \textsc{Emb}(\mathbf{w}_0, \phi)$
Training velocity flow model: randomly sample $\mathbf{z}_0 \sim \mathcal{N}(0, I)$ and $\mathbf{z}_1 \sim \mathcal{D}$, and train $\mathbf{v}_\theta$ follows the objective function eq. (6) to convergence.
Sampling (output): Randomly sample from $\mathbf{y}_0 \sim \mathcal{N}(0, I)$, $\mathbf{z}_0 = \mathbf{x}_1 \oplus \mathbf{y}_0$ and output the desired text $\mathbf{z}_1$ with $\mathbf{z}_1 = \mathbf{z}_0 + \mathbf{v}_\theta(\mathbf{z}_0, 0)$.
Continuous to discrete state:
$\mathbf{w}^z$ = argmax $(\mathbf{z}_1 \cdot \phi^T)$.
```

4 Experiments

Experimental Details. We focus on the task of Question Generation, Paraphrasing, and Open-domain Dialogue. We evaluate our approach on

Tasks	Methods	NFE↓	BLEU↑	R-L↑	Score↑	dist-1↑	selfB↓	div-4↑	Len
Open Domain Dialogue	Transformer-base	–	**0.018**	0.104	0.478	0.750	0.370	0.647	19.50
	GPT2-large FT	–	0.013	0.100	**0.529**	0.924	0.021	0.994	16.80
	GPVAE-T5	–	0.011	0.101	0.432	0.563	0.356	0.555	20.10
	NAR-LevT	–	0.016	0.055	0.476	**0.973**	0.710	0.142	4.11
	DiffuSeq	2,000	**0.014**	0.106	0.513	0.947	0.014	0.997	13.60
	FLOWSEQ (Ours)	**1**	0.011	**0.119**	0.345	0.709	0.027	**0.999**	30.70
Question Generation	Transformer-base	–	0.166	0.344	0.631	0.931	0.327	0.772	10.30
	GPT2-large FT	–	0.111	0.322	**0.635**	**0.967**	0.291	0.806	9.96
	GPVAE-T5	–	0.125	0.339	0.631	0.938	0.357	0.728	11.4
	NAR-LevT	–	0.093	0.289	0.549	0.891	0.983	0.478	6.93
	DiffuSeq	2,000	**0.173**	0.366	0.612	0.905	**0.279**	**0.810**	11.50
	DiffuSeq	500	0.016	0.120	0.334	0.543	0.321	0.435	11.50
	FLOWSEQ (Ours)	**1**	0.162	**0.370**	0.573	0.833	0.460	0.497	11.80
Paraphrase	Transformer-base	–	**0.272**	0.575	0.838	0.975	0.448	0.734	11.20
	GPT2-large FT	–	0.206	0.542	0.836	**0.982**	0.733	0.502	9.53
	GPVAE-T5	–	0.241	**0.589**	**0.847**	0.969	0.561	0.617	9.60
	NAR-LevT	–	0.227	0.580	0.834	0.979	0.999	0.333	8.85
	DiffuSeq	2,000	0.241	0.588	0.837	0.981	0.273	**0.864**	11.20
	FLOWSEQ (Ours)	**1**	0.143	0.461	0.669	0.862	**0.191**	0.781	11.90

Table 1: **Results for sequence-to-sequence text generation** on different tasks. Benchmarking autoregressive transformers, finetuned large pre-trained language models, and non-autoregressive methods. NFE denotes the number of neural forward evaluations. Len refers to the length of the generated tokens.

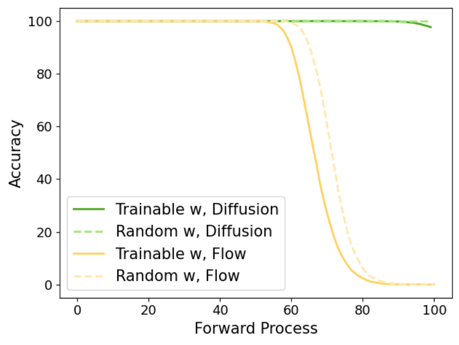

Figure 3: **Forward process classification** accuracy of embeddings from $\mathbf{x}_1$ estimation. 1). Both the noise schedule of FLOWSEQ and the joint training with the embedding can more evenly distribute the corruption, thereby aiding the denoising process. 2). Our empirical findings suggest that learnable embeddings are superior because they can more uniformly corrupt the data, and resulting in worse accuracy.

Method	Iters	BLEU↑	R-L ↑	Score↑
FLOWSEQ	40k	0.162	0.345	0.573
w/o anchor loss	40k	0.001	0.001	0.143

Table 2: **Ablation study on Question Generation task.** The anchor loss is important.

the widely used datasets: Quasar-T (Dhingra et al., 2017), QQP dataset[1], and the Commonsense Conversation Dataset (Zhou et al., 2018).

We report the BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), and BERTScore (Zhang et al., 2019) for string and sentence-based similarity. To evaluate the diversity, we report unique unigram (dist-1), sentence-level self-BLEU (Zhu et al., 2018), and diverse 4-gram (div-4) (Deshpande et al., 2019). We use a transformer architecture to model $\mathbf{v}(\mathbf{z}_t, t; \theta)$. More details about architecture structure, evaluation, and training details are given in the appendix B.

Main Results. We compare our model to three groups of baselines, which cover both autoregressive (AR) and non-autoregressive (NAR) architectures. The first group of methods use an encoder-decoder-based Transformer (Vaswani et al., 2017). The second group uses a finetuned large-scale pretrained language model, i.e., GPT2 (Radford et al., 2019). We also compare our model to GPVAE (Du et al., 2022), which augments a pretrained T5 (Raffel et al., 2020) with VAE to improve generation diversity. For the last group of baselines, we consider two strong iterative NAR models: LevT (Gu et al., 2019) and DiffuSeq (Gong et al., 2022). The results of the baseline are from (Gong et al., 2022).

The results in table 1 show that our approach is competitive when compared to the three strong baseline groups but requires much less compute.

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

Importantly, our model only uses a *single* sampling step, which is a significant reduction in compute when compared to DiffuSeq (Gong et al., 2022), which requires 2,000 steps to perform well. For example, when we reduce the number of sampling steps for DiffuSeq (Gong et al., 2022) to 500 on the Question Generation dataset, we observe a significant performance drop. Running 2,000 sampling steps required 520 seconds, while our single-step sample merely required 0.26 seconds per sentence[2]. Overall, our method does not exhibit obvious performance advantage over the baseline model. Our goal is to enhance the balance between sampling speed and generation capability (see fig. 1). This could unlock new potential for non-autoregressive models, as the number of network forwards is significantly smaller compared to related non-autoregressive baselines. In some cases, BLEU and BertScore metrics do not align, highlighting well-known inconsistencies in evaluating surface-level (BLEU) versus representation-level (BertScore) aspects (Freitag et al., 2022).

Sampling steps. We primarily compare the sampling steps with DiffuSeq, Difformer (Gao et al., 2022) in table 3. Our findings indicate that our method achieves strong results while Difformer fails. Furthermore, we consistently observe gains as the number of sampling steps increases.

Amortizing the corruption by flow. In fig. 3, we contrast the classification accuracy of embeddings from the $\mathbf{x}_1$ estimation conditioned on $\mathbf{x}_t$, as well as under different embeddings in the forward pass. For diffusion models, we apply the `linear` noise schedule as in DDPM (Ho et al., 2020). We consider 64 sentences, resulting in a total of 64×128 tokens. During the forward pass over 100 steps, we note that the corruption of our FLOWSEQ is significantly more intense than that of the diffusion model, indicating a more uniformly distributed noising process. When comparing the curve in the flow model with and without trainable embedding, we can observe that trainable embedding can lead to a higher level of corruption. This emphasizes the significance of training the embedding along with the network.

Anchor loss. In table 2, we evaluate the impact of anchor loss (Gao et al., 2022). The results emphasize the critical need to use anchor loss in preventing the collapse of word embeddings. Simply applying flow matching loss is insufficient for efficient network performance. We hypothesize that the anchor loss implicitly improves stability and performance, thereby facilitating a more direct training trajectory in the ODE. This enhancement could potentially lead to increased sampling speed.

[2]with batch size=128 on single A5000 GPU.

Method	Steps	BLEU↑	R-L ↑	Score↑
DiffuSeq	1	0.08	0.141	0.412
Difformer	5	0.00	0.01	0.000
Difformer	1	0.00	0.000	0.412
FLOWSEQ	1	0.14	0.461	0.669

Table 3: **Ablation study about few-step sampling on Paraphrase task.**

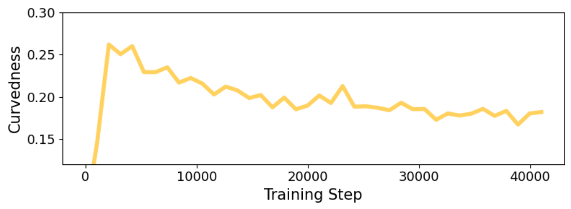

Figure 4: **Curvedness**. The Curvedness will gradually decrease after a short peak. This suggests that the trajectory becomes increasingly straight as training progresses.

Curvedness. In fig. 4, we quantitatively show the change in curvedness during training. Curvedness is a metric used to evaluate the curvedness of the ODE trajectory, and its definition can be found in Appendix eq. (8). We can see that, after a short temporary peak, the curvedness continues to decrease, indicating that the trajectory becomes progressively more straight. This trajectory straightening is crucial for generating high-quality samples with fewer steps.

5 Conclusion and Future Work

In this paper, we propose a flow-based method for sequence-to-sequence generation. We utilize the regression of the vector field and an anchor loss to encourage single-step generation. Our approach achieves competitive results compared to several autoregressive and non-autoregressive baselines. Importantly, it achieves a remarkable 2,000-fold acceleration in sampling speed relative to the recent diffusion-based baseline. In the future, we may opt to apply flow matching directly in discrete space instead of continuous space. Alternatively, we might explore generative models in a token-free manner, such as char-level generation.

6 Limitations

Our work is constrained by a few limitations. Primarily, due to limited computational resources, we could not validate the performance on large-scale datasets. Additionally, our method involves random sampling of $\mathbf{x}_0$ and $\mathbf{x}_1$ in an independent manner. An optimized matching approach (Tong et al., 2023; Pooladian et al., 2023) could potentially be employed prior to the vector field regression, which is a point of investigation we defer to future work. In the last, Furthermore, a rectification (Liu et al., 2023; Albergo et al., 2023) based on flow matching may yield a more direct trajectory and accelerate the sampling speed.

From an ethical standpoint, the generated sentences have the potential to contain inappropriate content that may require further review by human inspectors.

References

Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. 2023. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv*.

Ricky TQ Chen and Yaron Lipman. 2023. Riemannian flow matching on general geometries. *arXiv*.

Ting Chen, Ruixiang ZHANG, and Geoffrey Hinton. 2023. Analog bits: Generating discrete data using diffusion models with self-conditioning. In *ICLR*.

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

Aram Davtyan, Sepehr Sameni, and Paolo Favaro. 2022. Randomized conditional flow matching for video prediction. *arXiv*.

Aditya Deshpande, Jyoti Aneja, Liwei Wang, Alexander G Schwing, and David Forsyth. 2019. Fast, diverse and accurate image captioning guided by part-of-speech. In *CVPR*.

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

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, et al. 2022. Continuous diffusion for categorical data. *arXiv*.

Wanyu Du, Jianqiao Zhao, Liwei Wang, and Yangfeng Ji. 2022. Diverse text generation via variational encoder-decoder models with gaussian process priors. In *ACL*.

Markus Freitag, David Grangier, Qijun Tan, and Bowen Liang. 2022. High quality rather than high model probability: Minimum bayes risk decoding with neural metrics. *TACL*.

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

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

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. *NeurIPS*.

Jiatao Gu, Shuangfei Zhai, Yizhe Zhang, Lingjie Liu, and Josh Susskind. 2023. Boot: Data-free distillation of denoising diffusion models with bootstrapping. *arXiv*.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *NeurIPS*.

Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. 2023. simple diffusion: End-to-end diffusion for high resolution images. *arXiv*.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. 2021. Argmax flows and multinomial diffusion: Learning categorical distributions. In *NeurIPS*.

Vincent Tao Hu, Yunlu Chen, Mathilde Caron, Yuki M. Asano, Cees G.M. Snoek, and Björn Ommer. 2023. Guided diffusion from self-supervised diffusion features. In *Arxiv*.

Vincent Tao Hu, Wenzhe Yin, Pingchuan Ma, Yunlu Chen, Basura Fernando, Yuki M. Asano, Efstratios Gavves, Pascal Mettes, Björn Ommer, and Cees G.M. Snoek. 2024a. Motion flow matching for human motion synthesis and editing. In *Arxiv*.

Vincent Tao Hu, David W Zhang, Pascal Mettes, Meng Tang, Deli Zhao, and Cees G.M. Snoek. 2024b. Latent space editing in transformer-based flow matching. In *AAAI 2024*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *EMNLP*.

Zhifeng Kong and Wei Ping. 2021. On fast sampling of diffusion probabilistic models. *arXiv*.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. 2022. Diffusion-lm improves controllable text generation. *NeurIPS*.

Yifan Li, Kun Zhou, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Diffusion models for non-autoregressive text generation: A survey. *arXiv*.

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

Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. 2023. Common diffusion noise schedules and sample steps are flawed. *arXiv*.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. 2023. Flow matching for generative modeling. *ICLR*.

Xingchao Liu, Chengyue Gong, and Qiang Liu. 2023. Flow straight and fast: Learning to generate and transfer data with rectified flow. *ICLR*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *ICLR*.

Eric Luhman and Troy Luhman. 2021. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv*.

Shivam Mehta, Ruibo Tu, Simon Alexanderson, Jonas Beskow, Éva Székely, and Gustav Eje Henter. 2024. Unified speech and gesture synthesis using flow matching. In *ICASSP*.

Kirill Neklyudov, Daniel Severo, and Alireza Makhzani. 2022. Action matching: A variational method for learning stochastic dynamics from samples. *arXiv*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky Chen. 2023. Multisample flow matching: Straightening flows with minibatch couplings. *ARXIV*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*.

Tim Salimans and Jonathan Ho. 2022. Progressive distillation for fast sampling of diffusion models. *arXiv*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*.

Jiaming Song, Chenlin Meng, and Stefano Ermon. 2021a. Denoising diffusion implicit models. In *ICLR*.

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

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2021b. Score-based generative modeling through stochastic differential equations. In *ICLR*.

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*.

Zecheng Tang, Pinzheng Wang, Keyan Zhou, Juntao Li, Ziqiang Cao, and Min Zhang. 2023. Can diffusion model achieve better performance in text generation? bridging the gap between training and inference! *ACL*.

Alexander Tong, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Kilian Fatras, Guy Wolf, and Yoshua Bengio. 2023. Conditional flow matching: Simulation-free dynamic optimal transport. *arXiv*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *NeurIPS*.

Lemeng Wu, Dilin Wang, Chengyue Gong, Xingchao Liu, Yunyang Xiong, Rakesh Ranjan, Raghuraman Krishnamoorthi, Vikas Chandra, and Qiang Liu. 2022. Fast point cloud generation with straight flows. *arXiv*.

Jiasheng Ye, Zaixiang Zheng, Yu Bao, Lihua Qian, and Mingxuan Wang. 2023. Dinoiser: Diffused conditional sequence learning by manipulating noises. *arXiv*.

Hongyi Yuan, Zheng Yuan, Chuanqi Tan, Fei Huang, and Songfang Huang. 2022. Seqdiffuseq: Text diffusion with encoder-decoder transformers. *arXiv*.

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

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

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *SIGIR*.

Hao Zou, Zae Myung Kim, and Dongyeop Kang. 2023. Diffusion models in nlp: A survey. *arXiv*.

A Method Details

Partial noising to conditioning on x. Since our goal is to generate a target text conditioned on the source text, we can simply replace $\mathbf{x}_t$ with its clean version $\mathbf{x}_0$ both during training and sampling. Hence, it is only necessary to add noise to the target sequence part $\mathbf{y}$ and we can remove the terms corresponding to the source sequence from the loss in eq. (4):

$$\min_{\theta,\phi} \mathbb{E}_{t,\mathbf{z}_0} \big[||\tilde{\mathbf{v}}(\mathbf{y}_t, t; \theta) - (\mathbf{y}_1 - \mathbf{y}_0)||^2 \\ - \log p(\mathbf{w}^y|\tilde{\mathbf{y}}_1; \phi)\big], \tag{6}$$

here we use $\tilde{\mathbf{v}}(\mathbf{y}_t, t; \theta)$ to denote the velocity field of $\mathbf{y}_t$ conditioned on $\mathbf{x}_1$.

Single-step sampling. The anchor loss is designed to facilitate one-step generation. First, we randomly select $\mathbf{y}_0$, and concatenate it with the source text $\mathbf{x}_1$ to form $\mathbf{z}_0 = \mathbf{x}_1 \oplus \mathbf{y}_0$. Following this, we proceed to sample:

$$\tilde{\mathbf{z}}_1 := \mathbf{z}_0 + \tilde{\mathbf{v}}(\mathbf{z}_0, 0; \theta).$$

The discrete token id is retrieved by argmax($\tilde{\mathbf{z}}_1 \cdot \phi^T$). We focus on single-step sampling for maximum efficiency, but our approach generalizes to multi-stage sampling, as shown in the appendix A.

To perform SEQ2SEQ generation, we initiate by randomly sampling $\mathbf{y}_0 \sim \mathcal{N}(0, I)$. Starting from $\mathbf{y}_0$, we append $\mathbf{x}_1$ to form $\mathbf{z}_0 = \mathbf{x}_1 \oplus \mathbf{y}_0$. Following this, samples are generated by discretizing the ODE process using an Euler solver, as described in eq. (1), into N steps (e.g., $N = 1000$), as shown below,

$$\mathbf{z}'_{(\hat{t}+1)/N} \longleftarrow \mathbf{z}'_{\hat{t}/N} + \frac{1}{N}\, \mathbf{v}_\theta(\mathbf{z}'_{\hat{t}/N}, \frac{\hat{t}}{N}), \tag{7}$$

the integer time step $\hat{t}$ is defined as $\hat{t} \in \{0, 1, \cdots, N-1\}$. Here $\mathbf{z}'_1$ denotes our generated samples and $\mathbf{z}'_0 = \mathbf{z}_0$. We summarize the overall algorithm of training and sampling in algorithm 1.

Padding tokens. We pad the sequence to a fixed length. The flow matching model will learn when to generate PADDING tokens based on the distribution learning process. This way, our method can generate sentences of diverse lengths. It's worth noting that the potential issue of padding is not exclusive to the diffusion models in sentence generation, but is a general problem for non-autoregressive generation as well.

B Experimental Details

In evaluating the generated sequences, we regard both quality and diversity. For quality, we employ standard metrics like BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004). However, as string-similarity-based metrics can sometimes be inadequate for open-ended generation tasks, we also resort to BERTScore (Zhang et al., 2019), which assesses the semantic similarity between the generated sentences and the references. Higher BLEU, ROUGE, and BERTScore values indicate better performance.

Regarding diversity, we use distinct unigram (dist-1) to measure the intra-sentence diversity, where lower dist-1 denotes more word repetition in the generated sentence. For the sentence-level diversity evaluation, we apply self-BLEU (Zhu et al., 2018) to measure the n-gram overlap within the set of outputs related to one source sentence. Furthermore, we utilize diverse 4-gram (div-4) (Deshpande et al., 2019) to estimate the ratio of distinct 4-grams in the set of outputs per source sentence. Lower self-BLEU and higher div-4 imply greater diversity in generation.

To enhance the generation quality, we apply the widely used Minimum Bayes Risk (MBR) decoding strategy (Koehn, 2004). We first generate a set of candidate samples $\mathcal{S}$ from different random seeds and select the output sequence that yields the minimum expected risk under a meaningful loss function (e.g., BLEU or other less expensive metrics like precision). We compute the accuracy metrics using MBR with a candidate sample size of $|\mathcal{S}| = 10$.

Our method incorporates a Transformer with 12 layers and 12 attention heads, treating the time step embedding in a manner similar to the position embedding. We set the maximum sequence length at 128, with an embedding dimension of $d = 128$. To minimize out-of-vocabulary generation, we apply Byte Pair Encoding (Sennrich et al., 2016) for vocabulary construction.

The learning rate is set at 0.0001 and undergoes annealing during the training process. We use AdamW (Loshchilov and Hutter, 2019) for optimization. The experiments are conducted on NVIDIA A100 Tensor Core GPUs, utilizing 4 GPUs for training and a single GPU for sampling. We maintain the same parameter count as the DifFuSeq baseline.

For simplicity, we avoid applying Gaussian cor-

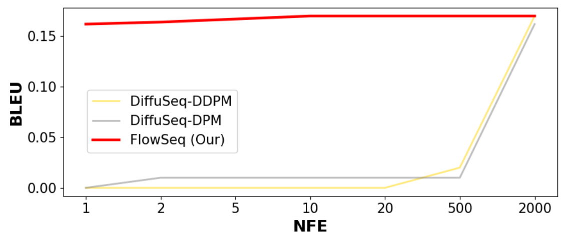

Figure 5: **The comparison of BLEU between our method and DiffSeq** under different Number of Function Evaluation (NFE).

ruption to the embedding, as performed in (Li et al., 2022).

For the details of training, please refer to table 4. For the details of the baseline, we follow the (Gong et al., 2022). For complexity, we list them in table 5.

C Extra Experiment

Curvedness. The curvedness of the trajectories can be quantified using the following formula:

$$\text{Curvedness} = \frac{1}{N}\sum_{\hat{t}=0}^{N-1}\Big[\parallel(\mathbf{z}_1 - \mathbf{z}_0) - \mathbf{v}_\theta(\mathbf{z}_{\hat{t}/N}, \hat{t}/N)\parallel^2\Big], \quad (8)$$

where $\hat{t}/N$ represents the discretized timestep ranging from 0 to N. We illustrate the change in curvedness as the training progresses in fig. 4.

Minimum Bayes Risk (MBR) . We further explore the impact of the number of candidates in MBR, as depicted in fig. 6. Our observations suggest that performance in terms of BLEU and Rouge-L improves as we incrementally increase the number of candidates.

The accuracy of $\mathbf{x}1$ estimation in 100-step backward process. We demonstrate it in fig. 7.

The embedding corruption visualization. We demonstrate the visualization in fig. 8.

Many-step sampling. We show the many step sampling result in table 6.

Training progress. The accuracy of the classification on the embedding from $\mathbf{x}_1$ estimation are shown in fig. 9 (training set) and fig. 10 (validation set).

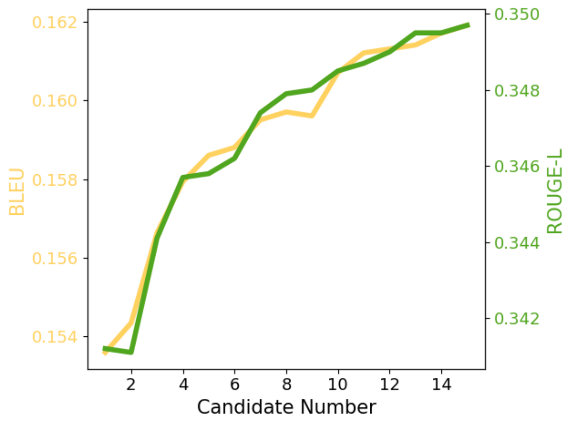

Figure 6: **Ablation of MBR.** As we progressively increase the number of candidates, we observe a slight increase in BLEU and Rouge-L scores. However, the rate of increase is not as significant. Notably, our method, FLOWSEQ, is less sensitive to changes in candidate number compared to DiffuSeq.

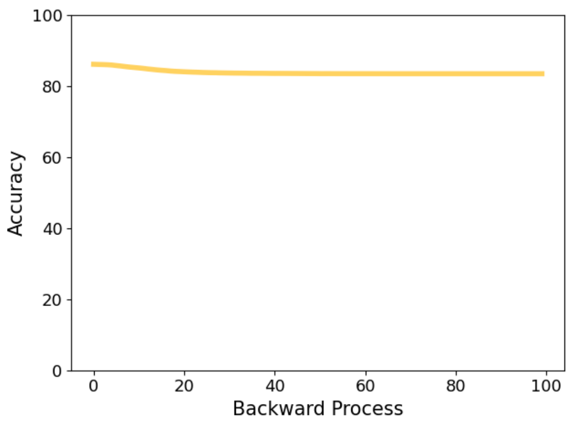

Figure 7: **Classification Accuracy of Embeddings.** The accuracy of embeddings from the $\mathbf{x}_1$ estimation, which is conditioned on $\mathbf{x}_0$, slightly decreases during the backward process over 100 steps. This decrease is anticipated, as our anchor loss primarily encourages single-step sampling, which can negatively impact the vector field prediction for shorter step sizes.

Task	**Question Generation**	**Paraphrasing**	**Open-domain Dialogue**
Dataset	Quasar-T (Dhingra et al., 2017)	QQP[3]	Commonsense Conversation (Zhou et al., 2018)
Dataset Size	117k	144k	3382k
Input shape	128×128	128×128	128×128
Transformer type	bert-base-uncased	bert-base-uncased	bert-base-uncased
Vocabulary Size	30,522	30,522	30,522
depth	12	12	12
embedding dim	768	768	768
num of head	12	12	12
Batch size	1,024	1,024	1,024
Micro Batch size	64	64	64
Training iterations	40k	50k	50k
Training Time	5 days	5 days	5 days
GPU	4 × A5000	4 × A5000	4 × A5000
Optimizer	AdamW	AdamW	AdamW
Learning rate	1e-4	1e-4	1e-4
Betas	(0.99, 0.999)	(0.99, 0.999)	(0.99, 0.999)

Table 4: **The training details of three tasks.** bert-base-uncased denotes a transformer type other than the pretrained BERT.

Models	# Parameters	Learning Paradigm	Diversity Source
Transformer-base	80M	encoder-decoder	Temperature/DBS
GPT2-large FT	774M	pretrain-finetune	Hybrid strategy
GPVAE-T5	220M	pretrain+VAE	Gaussian sampling
NAR-LevT	80M	non-autoregressive	-
DiffuSeq	91M	non-autoregressive	Gaussian sampling
FLOWSEQ	91M	non-autoregressive	Gaussian sampling

Table 5: **The comparison for different models.**

Step	1	10	500
DiffuSeq	0/0/0	0/0/0	0.02/0.12/0.33
FLOWSEQ	0.162/0.37/0.57	0.17/0.38/0.61	0.17/0.37/0.60

Table 6: **Ablation study** about sampling steps on Question Generation task. BLEU↑/ R-L ↑/ Score↑ are listed.

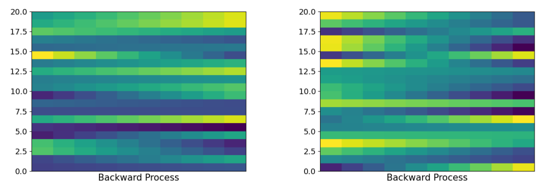

Figure 8: **Visualization Comparison of Embeddings.** The comparison visualizes the embeddings of a diffusion-based model (left) and a flow-based model (right) during the forward process. Both visualizations share the same colormap. Our FLOWSEQ method manages to corrupt the embedding more uniformly than diffusion models, significantly facilitating the denoising process.

The loss trend of the vector field regression loss and anchor loss are shown in fig. 11, fig. 14 respectively.

D Qualitative Result

We show our qualitative result of the question generation task in table 7.

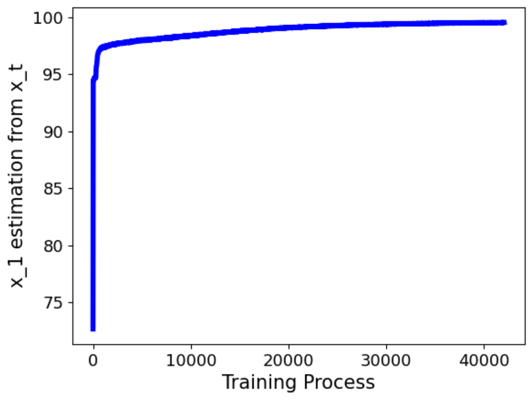

Figure 9: **Accuracy of Embedding Classification from x_1 Estimation.** The accuracy is measured using training batches for the task of question generation on the Quasar-T dataset.

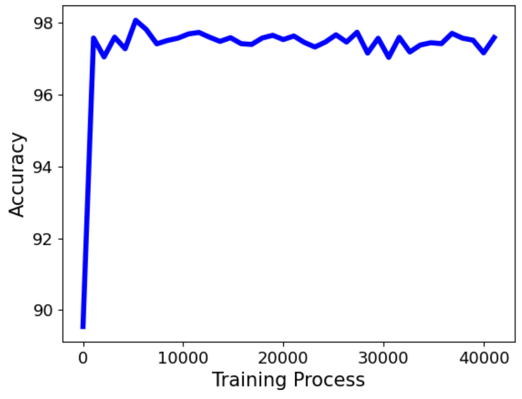

Figure 10: **Accuracy of Embedding Classification from x_1 estimation.** The accuracy is measured using validation batches for the task of question generation on the Quasar-T dataset.

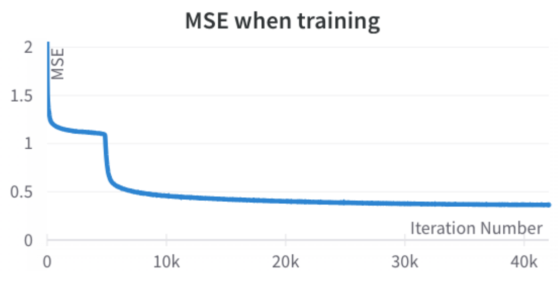

Figure 11: **Vector field regression loss trend for training set.**

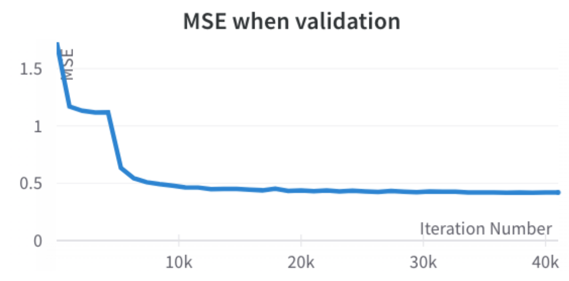

Figure 12: **Vector field regression loss trend for validation set.**

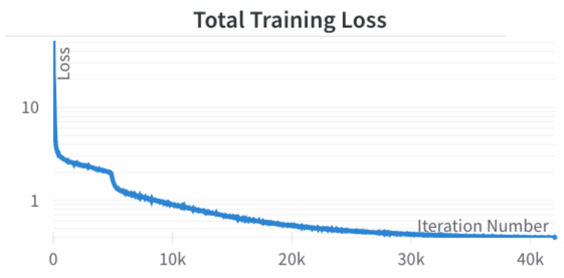

Figure 13: **Total loss trend for training set.**

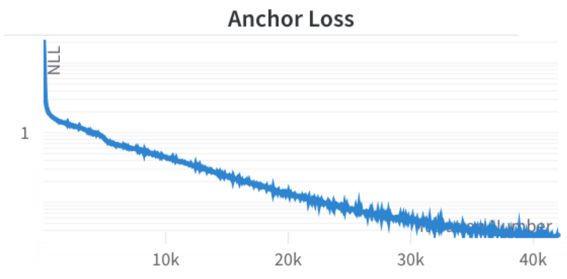

Figure 14: **Anchor loss trend.**

***Statement**: The Japanese yen is the official and only currency recognized in Japan.* ***Question**: What is the Japanese currency?*	
GPVAE-T5 * What is the japanese currency * What is the japanese currency * What is the japanese currency	**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 ?
GPT2-large finetune * What is the basic unit of currency for Japan? * What is the Japanese currency * 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?
FLOWSEQ * What is the basic unit for Japan currency? * What is the currency in Japanese? * What is the currency for Japanese?	

Table 7: **Sample outputs with different random seed in Question Generation test set.**