Flexible-length Text Infilling for Discrete Diffusion Models

Andrew Zhang **Anushka Sivakumar** **Chiawei Tang** **Chris Thomas**
Department of Computer Science
Virginia Tech
{azhang42,anushkas01,cwtang,christhomas}@vt.edu

Abstract

Discrete diffusion models are a new class of text generation models that offer advantages such as bidirectional context, parallelizable generation, and flexible prompting compared to autoregressive models. However, a critical limitation has been the inability to perform flexible-length or flexible-position text infilling without access to ground-truth positional data. We introduce **DDOT**[1] (**D**iscrete **D**iffusion with **O**ptimal **T**ransport Position Coupling), a discrete diffusion model that overcomes this limitation by *jointly* denoising token values and token positions using a novel sample-level optimal transport coupling. This coupling preserves relative token ordering while dynamically adjusting the positions and lengths of infilled segments. DDOT is orthogonal to existing discrete text diffusion methods and is compatible with various pretrained text denoisers. On text-infilling benchmarks such as One-Billion-Word and Yelp, DDOT outperforms naive diffusion baselines and achieves performance on par with state-of-the-art non-autoregressive models, while improving training efficiency and prompting flexibility.

1 Introduction

While autoregressive (AR) models are effective, they also involve highly sequential sampling, cannot use bidirectional context, and constrain architectures by requiring a decoder mask. In contrast, discrete diffusion models (Lou et al., 2024; Sahoo et al., 2024) can parallelize generation by denoising multiple tokens simultaneously, use bidirectional context, do not need a decoder mask, and allow for more controllable generation. Previous work (Gat et al., 2024; Shi et al., 2024) has demonstrated that discrete diffusion models can handle prompts at arbitrary locations, whereas autoregressive models are only capable of left-to-right text completion.

[1]Project page: https://andrew-zhang.github.io/ddot-page

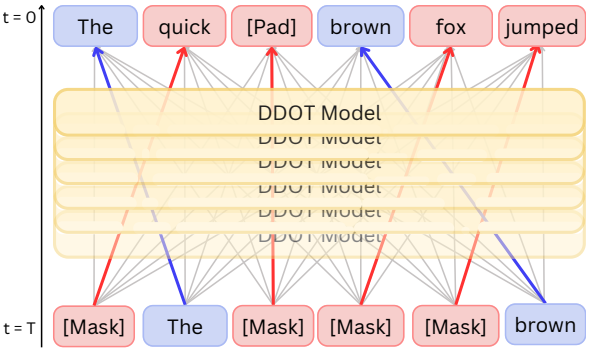


Figure 1: Our diffusion across token positions enables dynamic token movement for infilling. Unlike prior methods, DDOT learns to move masked tokens to appropriate locations, such as to the right of "brown," even if that position was not initially masked. The OT coupling (colored lines) simplifies this learning by drastically reducing the set of possible permutations.

However, this advantage has a significant limitation: existing diffusion models cannot alter the distances between these prompt tokens. Consequently, existing text diffusion models cannot generate the ground-truth sample without access to the oracle positions of the prompt and infilled text.

We solve this issue by enabling discrete diffusion models to *learn where to move tokens*. Specifically, we design a diffusion process that operates across token positions (in addition to token values), allowing the model to vary the positions and lengths of infilled spans (Figure 1). Furthermore, given the importance of token positioning in preserving semantic meaning (He et al., 2020), we incorporate sample-level OT coupling (Tong et al., 2024) to maintain relative token ordering throughout the diffusion process. Even minor positional changes can dramatically alter meaning, as seen in phrases like "The child's green coat" and "The green child's coat." DDOT's OT coupling preserves this relative ordering throughout the diffusion pro-

cess while supporting flexible-length text infilling. Our OT coupling prevents such swaps and drastically improves DDOT's training efficiency and downstream performance across all studied benchmarks and metrics. Extensive experiments show that DDOT outperforms naive diffusion baselines and achieves performance on par with state-of-the-art non-autoregressive (NAR) models.

In summary, our contributions are as follows:

- We propose **DDOT**, the first discrete text diffusion method for infilling arbitrary text sequences without ground-truth span lengths.
- We provide **extensive experiments** on DDOT that show it outperforms diffusion baselines and achieves performance on par with state-of-the-art NAR models.
- We provide detailed ablations and visualizations that **verify DDOT's effectiveness** in adjusting the positions and lengths of infilled text spans and provide insights into our novel sample-level OT coupling. The OT coupling significantly outperforms naive diffusion baselines across all tested benchmarks and metrics.

2 Related Work

2.1 Lexically Constrained Generation

Constrained text generation has been explored through a variety of approaches, including AR and NAR methods (Zhang et al., 2020; Iso, 2024; He, 2021; Stern et al., 2019; Lu et al., 2022). POINTER (Zhang et al., 2020) enables flexible token generation through iterative insertion, though it still depends on sequential token prediction and multiple forward passes. AutoTemplate (Iso, 2024) approaches constrained text generation by feeding the prompt tokens into an encoder–decoder-style model. CBART (He, 2021) extends the POINTER architecture by moving to an encoder–decoder model. Autoregressive methods, while effective for their specific use cases, inherit fundamental limitations: they require sequential generation that scales linearly with sequence length, and their causal attention masks prevent the full use of bidirectional context during generation. Most critically, for text infilling tasks, these approaches struggle to consider both past and future contexts simultaneously when generating intermediate content (Cao et al., 2023). Furthermore, to allow for flexible positions, autoregressive methods often regenerate the entire sequence instead of only inserting tokens, leading to wasted compute (Iso, 2024).

2.2 Discrete Diffusion Models

Discrete diffusion models offer an innovative approach to text generation, addressing key limitations of autoregressive methods (Lou et al., 2024; Ren et al., 2024; Gong et al., 2024; Sahoo et al., 2024). These models denoise corrupted text sequences, enabling parallel updates of multiple tokens rather than the token-by-token process of autoregressive methods, thereby reducing the number of forward passes. Additionally, their bidirectional nature allows them to leverage both past and future context, in contrast to the causal masking that constrains autoregressive models. Early frameworks like D3PM (Austin et al., 2021) adapted continuous diffusion to discrete tokens by assigning probability that a token corrupts by either turning into a mask token or random token.

Recent work on score-based discrete diffusion has further advanced the field by providing analytical solutions for the denoising process. Instead of directly modeling transition probabilities, SEDD (Lou et al., 2024) uses a score-based approach that learns the gradient of the log-probability and introduces an entropy-based loss function, narrowing the gap to autoregressive models.

However, despite these advantages, current discrete diffusion models face a significant limitation: they require fixed token positions throughout the generation process. This constraint makes them unsuitable for flexible-length text infilling, where the length of the generated text might differ from the original masked region.

2.3 Optimal Transport Coupling for Flexible Generation

OT (Villani et al., 2009) coupling has been well studied in continuous diffusion. Tong et al. (2024) introduce *minibatch couplings*—either independent (random) or determined by OT—that regularize training and encourage near-straight trajectories. We adopt the term "coupling" from Tong et al. (2024) to denote a matching between an source and target distribution. By incentivizing straighter paths, minibatch OT coupling yields practical benefits such as faster sampling, a pattern also observed in OT-inspired methods like Rectified Flow (Liu et al., 2023) and Stochastic Interpolants (Albergo and Vanden-Eijnden, 2023).

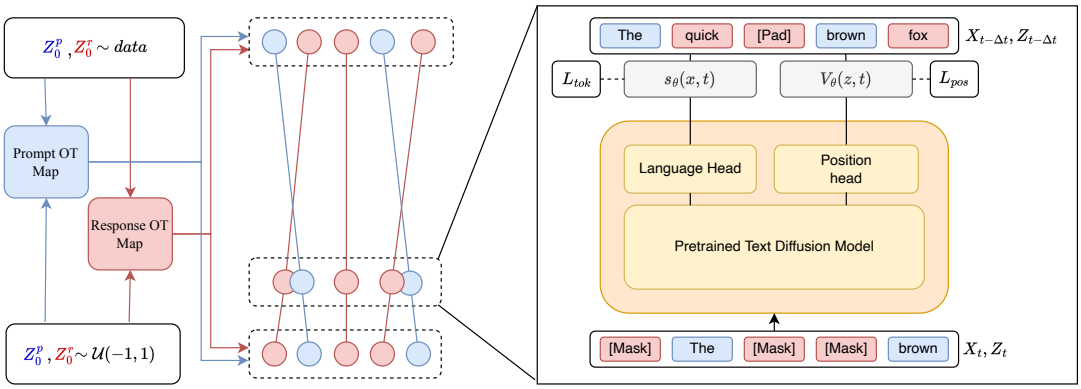


Figure 2: DDOT learns to vary infilled span lengths and positions, unlike prior fixed-position diffusion methods. *(Left)* We compute two separate intra-set OT couplings within the prompt positions and the response positions, which drastically simplifies the set of possible permutations. *(Right)* Given a time step t, we predict the token and position.

Our OT coupling differs in scope and granularity: we construct a *sample-level* coupling *within each sequence*, aligning token positions between an initial state and a target ordering. We instantiate this coupling via OT (and compare to independent coupling in ablations), which preserves relative order and guides tokens along predictable paths while still allowing the inter-set crossings needed for flexible-length infilling. Unlike minibatch couplings aimed primarily at accelerating continuous diffusion models, our coupling enables joint optimization of *content* and *placement* of tokens for discrete text diffusion, retaining parallel and bidirectional conditioning and permitting dynamic adjustment of span lengths and token spacing.

3 Background: Masked Text Diffusion

Discrete diffusion adapts the continuous nature of diffusion processes to text. The simplest and most performant version of discrete diffusion is masked diffusion (Lou et al., 2024; Shi et al., 2024; Sahoo et al., 2024). Rather than adding Gaussian noise to continuous values such as pixels, masked text diffusion assigns a probability of masking tokens throughout the forward diffusion process. For the purposes of this paper, masked diffusion can be seen as a masked language model (like BERT (Devlin et al., 2019)) that operates at progressively decreasing masking ratios to generate text. Specifically, our text diffusion process follows Score Entropy Discrete Diffusion (SEDD) (Lou et al., 2024), which models a score function over a support of N states (token values). The forward diffusion process is described by a continuous-time Markov chain (CTMC):

$$\frac{dp_t}{dt} = Q_t p_t \quad p_0 \approx p_{\text{data}}, \tag{1}$$

where $p_t \in \mathbb{R}^N$ is the column vector of state probabilities at time $t \in [0, T]$, $Q_t \in \mathbb{R}^{N \times N}$ is the (time-dependent) transition matrix (its columns sum to 0), and $p_{\text{data}} \in \mathbb{R}^N$ is the empirical token distribution at $t = 0$. In masked diffusion we include the mask token among the N states and make it absorbing: each non-mask column has -1 on its diagonal and $+1$ in the mask row, while the mask column is all zeros, so columns sum to 0 and probability flows from non-mask tokens into the mask token.

SEDD reverses the text diffusion dynamics by learning the score $s_\theta(x)_y$ of a transition from token x to token y via an entropy-based loss:

$$\mathcal{L}_{\text{tok}} = \int_0^T \mathbb{E}_{x_t \sim p_{t|0}(\cdot|x_0)} \sum_{y \neq x_t} Q_t(x_t, y) \; (s_\theta(x_t, t)_y - \frac{p_{t|0}(y|x_0)}{p_{t|0}(x_t|x_0)} \log s_\theta(x_t, t)_y + K\left(\frac{p_{t|0}(y|x_0)}{p_{t|0}(x_t|x_0)}\right)\Big) \; dt \tag{2}$$

where x_0 is the start token, x_t is the token at time t under the forward CTMC with marginal $p_{t|0}(\cdot|x_0)$, $Q_t(x_t, y)$ denotes the (row x_t, column y) entry of Q_t, and $K(a) = a(\log a - 1)$ is a scalar regularizer; Finally, to simulate the reverse diffusion process, we either take Euler steps or use an analytical denoiser (Lou et al., 2024).

4 Approach

Prior discrete diffusion models fix token positions during denoising and therefore require oracle positions to infill masked spans. We introduce **DDOT**, which jointly denoises discrete token values and

continuous token positions so infilled spans can move and change length during generation. This enables flexible-length, flexible-position infilling from arbitrary prompts while preserving the parallel, bidirectional nature of discrete diffusion. Figure 2 summarizes DDOT.

4.1 Token Value Diffusion

We partition token values into three disjoint subsets $x = (x^p, x^r, x^{\text{pad}})$, which are described as follows: (i) *Prompt* (x^p): these slots are clamped to their ground-truth tokens for all t; we never mask or resample them, and they serve purely as conditioning during denoising. (ii) *Response* (x^r): these are the real tokens to be infilled. They start at ground truth and follow the masked-diffusion forward process from section 3, progressively transitioning toward [MASK] as t increases; the reverse chain denoises them back to tokens. (iii) *Pad* (x^{pad}): these slots hold the vocabulary token [PAD] (distinct from [MASK]), are not part of x^r, and exist only to maintain a fixed length L. Under the forward process (section 3), [PAD] may corrupt to [MASK]; in the reverse process, [MASK] on these slots denoises back to [PAD].

4.2 Continuous Position Diffusion

To enable diffusion models to move masked tokens to regions that require infilling, DDOT also diffuses *positions*. Similar to token values, we partition positions into three disjoint subvectors $z_t = (z_t^p, z_t^r, z_t^{\text{pad}})$ for prompt, response, and pad positions. Let $\ell = |x^p| + |x^r|$ be the number of real tokens, with per-subset counts $\ell^p = |x^p|$ and $\ell^r = |x^r| = \ell - \ell^p$; there are $L - \ell$ pad slots.

The noisy positions are sampled i.i.d. as $z_T \sim \mathcal{U}(-1, 1)^L$ (so $z_T^p, z_T^r, z_T^{\text{pad}}$ are slices of a single length-L vector). We define the ground-truth positions for the *real* (non-pad) tokens as the length-ℓ subvector over the union of prompt and response indices, $z_0^{p\cup r} \in \mathbb{R}^\ell$ (equivalently $z_0^{p\cup r} = (z_0^p, z_0^r)$), evenly spaced and scaled to the true length:

$$z_{0,i}^{p\cup r} = \frac{\ell}{L}\,\frac{2i - (\ell - 1)}{\max\{1, \ell - 1\}}, \qquad i = 0, \ldots, \ell - 1. \tag{3}$$

so real tokens occupy the sub-interval $[-\ell/L,\, \ell/L] \subset [-1, 1]$.

Linear position paths. Following Albergo and Vanden-Eijnden (2023) and Liu et al. (2023), positions follow straight, noise-free interpolation:

$$z_t \;=\; (1 - t)\, z_0 + t\, z_T. \tag{4}$$

Pad tokens. Pad tokens are part of the ground-truth vocabulary (distinct from [MASK]). Pad positions remain stationary up to the same global scaling:

$$z_0^{\text{pad}} \;=\; \frac{\ell}{L}\, z_T^{\text{pad}} \tag{5}$$

Together with the $z^{p\cup r}$ construction in Equation 3, this induces the full length-L slot vector $z_0 = (z_0^p, z_0^r, z_0^{\text{pad}})$.

4.3 Positional Optimal Transport Coupling

Our preliminary experiments (Table 4) show that naively diffusing continuous token positions performs poorly. It induces a combinatorial explosion of permutations, weakens supervision from the ground truth, and leads to invalid infilling results. We therefore use *sample-level* optimal-transport (OT) coupling. In particular, we find an OT coupling independently within each set z^p and z^r. Our sample-level OT coupling has the following benefits:

(a) Reduces the permutation space. OT yields order-preserving, non-crossing trajectories *within each set* (prompt or response), drastically shrinking the search over permutations. Concretely, for either set $S \in \{p, r\}$ and indices $1 \le i < j \le \ell^S$ in that set's ground-truth order, we have $z_t^{S,(i)} \le z_t^{S,(j)}$ for all $t \in [0, 1]$, so within-set crossings cannot occur (while inter-set prompt–response crossings remain allowed), avoiding the combinatorial explosion from arbitrary permutations.

(b) Preserves supervision signal. Small word-order changes can flip semantics; The OT coupling between z_T and z_0 preserves relative order throughout diffusion and thus maintains a stable learning signal from the ground-truth sequence rather than forcing the model to infer order from noisy, permuted trajectories.

(c) Prevents invalid permutations. In infilling tasks, the prompt's *relative order is part of the conditioning*; any output that reorders prompt tokens is an invalid infilling result. We feed the prompt tokens at the terminal state T in their given order. With OT coupling, the forward $0 \to T$ paths are order-preserving within the prompt set (non-crossing), so the reverse diffusion returns the same within-set order at $t = 0$. Without coupling, trajectories can cross and swap prompt order, yielding invalid infills.

Assignment formulation (per set). We compute the optimal transport coupling *within* prompt and non-prompt sets. From the noisy positions $z_T \in \mathbb{R}^L$, sample ℓ^p entries to form the prompt slice z_T^p; the remaining $L-\ell^p$ entries form the non-prompt slice z_T^{np}. We compare ground-truth positions (rescaled to $[-1,1]$) directly to these slices.

Prompt OT (balanced). One-to-one match: each of the ℓ^p prompt positions at $t=0$ is assigned to exactly one sampled prompt position at $t=T$, minimizing total displacement:

$$\pi^p = \arg\min_{\pi \in S_{\ell^p}} \sum_{i=1}^{\ell^p} \left| \tfrac{L}{\ell}\, z^p_{0,i} - z^p_{T,\pi(i)} \right|, \quad \tilde{z}^p_T = z^p_T[\pi^p]. \tag{6}$$

Where S_{ℓ^p} is the set of permutations of $[\ell^p]$ and, for any vector v, $v[\pi]$ denotes reindexing by π: $(v[\pi])_i := v_{\pi(i)}$.

Response OT (unbalanced). Injective match: the ℓ^r response positions at $t=0$ are assigned into the non-prompt pool z_T^{np} of size $L-\ell^p$, each slot used at most once; unassigned slots become pads:

$$\pi^r = \arg\min_{\pi \in \Pi_{\ell^r}^{L-\ell^p}} \sum_{i=1}^{\ell^r} \left| \tfrac{L}{\ell}\, z^r_{0,i} - z^{\text{np}}_{T,\pi(i)} \right|, \quad \tilde{z}^r_T = z^{\text{np}}_T[\pi^r]. \tag{7}$$

Here S_{ℓ^p} is the set of all permutations of $[\ell^p]$, and $\Pi_{\ell^r}^{L-\ell^p}$ is the set of injective maps $[\ell^r] \hookrightarrow [L-\ell^p]$.

Pads (leftover positions). The $L-\ell$ non-prompt positions *not* selected by π^r constitute z_T^{pad}. Pads are excluded from OT and follow the stationary path in Equation 5.

After solving the assignments, we reorder the sampled positions and use the reordered slices for interpolation:

$$z_t^S = (1-t)\, z_0^S + t\, \tilde{z}_T^S, \qquad S \in \{p, r\}. \tag{8}$$

This per-set OT coupling induces order-preserving, non-crossing trajectories within prompt and response (by construction of $\tilde{z}_T^S$ and linear paths), while allowing *inter-set* crossings between prompt and response. See subsection A.4 for visualizations.

Computation. All couplings are computed *per sequence* in one dimension, using absolute distance between rescaled ground-truth positions and sampled positions. This choice lets us avoid forming dense cost matrices that are typically used when calculating OT and enables simple, fast implementations that scale favorably in the sequence length.

Prompt OT computation. For the prompt set we must match ℓ^p ground-truth positions z_0^p to ℓ^p sampled positions z_T^p in a one-to-one manner. In 1D with convex costs, the optimal assignment is equivalent to sorting both vectors and pairing by index. Therefore, we sort z_0^p and z_T^p once and obtain the permutation π^p that maps each sorted index in z_0^p to the corresponding rank in z_T^p. This procedure achieves the exact minimum from Equation 6, runs in $O(\ell^p \log \ell^p)$ time, and does not materialize a cost matrix.

Response OT computation. We sort $a = z_0^r$ and $b = z_T^{\text{np}}$ in ascending order, then compute the injective, order-preserving match with a standard global-alignment dynamic programming solution (Needleman and Wunsch, 1970). This exactly minimizes Equation 7 and runs in $O(\ell^r(L-\ell^p))$ time and $O(L-\ell^p)$ memory.

Complexity and practical overhead. The total per-sequence cost is $O(\ell^p \log \ell^p + \ell^r(L-\ell^p))$ time and $O(L)$ memory. For the context sizes considered ($L \le 1024$), the coupling time is negligible compared to a single denoiser forward pass. In practice we precompute several OT couplings in advance on the CPU and stream the dataset to the GPU, leading to negligible overhead, as seen in subsection 5.3

4.4 Training Objective

At each update we draw a time $t \sim \mathcal{U}(0,1)$, sample noisy positions $z_T \sim \mathcal{U}(-1,1)^L$, and form the matched targets $\tilde{z}_T = (\tilde{z}_T^p, \tilde{z}_T^r, z_T^{\text{pad}})$ by solving Equation 6–Equation 7 (pads follow Equation 5). We then construct the per-set paths (Equation 8). Independently, for token values we follow SEDD's forward process (Sec. 3) to obtain x_t from x_0 at the same t.

Position loss. We supervise the position head (Figure 2) on *all* L slots with a simple MSE toward the straight-path target:

$$\mathcal{L}_{\text{pos}}(\theta) = \mathbb{E}_t \Big[\big\| v_\theta(z_t, x_t, t) - \big(z_0 - \tilde{z}_T\big) \big\|_2^2 \Big]. \tag{9}$$

Token loss. For tokens we use the SEDD score-entropy objective from section 3, denoted $\mathcal{L}_{\text{tok}}(\theta)$.

Total loss. The final objective is a weighted sum

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{tok}}(\theta) + \lambda\, \mathcal{L}_{\text{pos}}(\theta), \tag{10}$$

with λ controlling the relative weight of position supervision.

Initialization: DDOT-R vs. DDOT-U. During inference we manipulate how we sample z_T. By default (DDOT-R), we randomly sample a full length-L terminal vector $z_T \sim \mathcal{U}(-1,1)^L$, reserve any ℓ^p entries as the prompt slice z_T^p. However, random sampling can produce high-density regions that tend to map to pad tokens because there are fewer tokens in the corresponding ground-truth region.

To mitigate this, DDOT-U places terminal positions on uniform grids for each set:

$$z^p_{T,i} = \frac{2i-(\ell^p-1)}{\max\{1,\ \ell^p-1\}}, \qquad i=0,\dots,\ell^p-1,$$
$$z^{np}_{T,i} = \frac{2i-(\ell^{np}-1)}{\max\{1,\ \ell^{np}-1\}}, \qquad i=0,\dots,\ell^{np}-1. \tag{11}$$

Uniform spacing avoids random clustering, reduces pad spillover, and yields more stable placements.

Reverse-time updates. During sampling, positions follow near-straight trajectories because the learned velocity field approximates the constant ground-truth direction $z_0 - \tilde{z_T}$. We therefore update positions with a simple Euler step at each τ-leap of the SEDD reverse process, while keeping prompts clamped and pads stationary (Equation 5).

4.5 Simultaneous Text & Position Diffusion

DDOT performs discrete text and continuous position diffusion simultaneously, as these processes operate independently in continuous time. We therefore predict both token-value scores and position velocities in a single forward pass. This independence also enables simulation-free training by sampling token and position states independently at arbitrary time steps. We summarize the training procedure in algorithm 1.

Algorithm 1 DDOT Training

Require: batch (x_0, z_0); loss weight λ
1: Sample $t \sim \mathcal{U}(0,1)$
2: Build terminal positions: sample $z_T \sim \mathcal{U}(-1,1)^L$ and split (z_T^p, z_T^{np}), or use uniform grids from Equation 11
3: Compute within-set OT: π^p via 1D sort-match (Equation 6); π^r via injective matching (Equation 7); set $\tilde{z}_T^p = z_T^p[\pi^p]$, $\tilde{z}_T^r = z_T^{\text{np}}[\pi^r]$, leftovers $\to z_T^{\text{pad}}$
4: Form paths $z_t^S = (1-t)z_0^S + t\,\tilde{z}_T^S$ for $S \in \{p, r\}$; set pads via Equation 5
5: Sample token states x_t from the SEDD forward at time t
6: Predict $s_\theta(x_t, t)$ and $v_\theta(z_t, x_t, t)$
7: Compute $\mathcal{L}_{\text{tok}}$ (Equation 2) and $\mathcal{L}_{\text{pos}}$ (Equation 9)
8: Update θ on $\mathcal{L}_{\text{tok}} + \lambda\,\mathcal{L}_{\text{pos}}$ (Equation 10)

4.6 Implementation Details

We extend SEDD, which is based on the Diffusion Transformer architecture (Peebles and Xie, 2023), with two additional modules. First, we introduce a learnable type embedding applied directly after the token-embedding lookup. This embedding indicates whether a token is part of the prompt or the masked response ($x_i \in x^p$ or $x_i \in x^n p$), which is critical for assigning each token to the correct OT flow. Second, we add a linear head at the end of the Diffusion Transformer to compute $v_\theta(z_t, t)$. The model architecture is available in Figure 2.

To incorporate continuous positional information, we scale z_t from $[-1, 1]$ to match the context length of the original pretrained model (1024). We then use Rotary Position Embeddings (Su et al., 2024), a standard technique in discrete diffusion models. Implementation details can be found in Subsection A.3.2.

5 Experiments

5.1 Experimental Setup

Datasets We evaluate our approach on the One-Billion-Word and Yelp datasets, following the preprocessing steps outlined in prior work on infilling and lexically constrained generation (Miao et al., 2019; Zhang et al., 2020; Iso, 2024). These datasets consist of examples with 1 to 6 keywords that must be infilled while maintaining their relative order to generate coherent sentences. In addition to randomly masking positions, we also introduce a *block* masking method that masks a single continuous span of text ranging from 0 to $L/2$ tokens (32 for One-Billion-Word and Yelp; 512 for CodeParrot). Finally, we apply the aforementioned masking methods to the Python subset of the CodeParrot dataset. Table 1 illustrates examples of this lexically constrained generation task.

Keywords:	earned , cents , share , costs
One-Billion-Word-Random:	It earned 28 cents a share , excluding restructuring costs .
Keywords:	we are so incredibly wedding .
Yelp-Block:	we are so incredibly happy we chose this venue for our wedding .

Table 1: Example generations for the keywords-to-sentence generation on One-Billion-Word and Yelp.

Training Details To align the position-prediction modules, we first fine-tune SEDD with the added

Model	One Billion Word–Block				Yelp-Block				CodeParrot-Block			CodeParrot-Random		
	B2/B4	N2/N4	M	SR	B2/B4	N2/N4	M	SR	B2/B4	CB	SR	B2/B4	CB	SR
LC	55.1/47.9	4.38/4.40	37.4	54.7	59.1/51.7	7.77/8.48	37.9	60.3	51.0/49.4	53.3	**16.9**	15.4/24.8	19.4	0.
PoP	59.2/50.0	**8.43**/8.77	37.1	58.6	64.9/48.6	7.81/8.47	38.1	67.1	29.4/13.3	21.7	0.34	12.5/2.75	13.9	0.
DDOT-R	62.2/57.1	7.42/8.16	42.1	99.7	62.5/57.3	7.50/8.24	42.2	99.8	47.2/41.1	38.5	1.01	57.2/43.5	39.2	3.53
DDOT-U	**63.7/58.4**	8.40/**8.79**	**42.1**	**100.**	**64.9/59.5**	**8.06/8.86**	**42.9**	**100.**	**53.7/51.0**	**50.5**	10.8	**58.2**/40.4	**45.4**	**11.2**

Table 2: **DDOT outperforms diffusion baselines on standard sequences (0-32 prompt tokens).** Metrics are BLEU (B2, B4), NIST (N2, N4), METEOR (M), and Success Rate (SR). Top scores are **bolded**.

Model	One Billion Word–Random				Yelp-Random			
	B2/B4	N2/N4	M	SR	B2/B4	N2/N4	M	SR
Autoregressive Models								
GBS (Hokamp and Liu, 2017)	10.1/2.8	1.49/1.50	13.5	$\leq$ 100.	13.6/4.5	1.68/1.71	15.3	$\leq$ 100.
InstructGPT (Ouyang et al., 2022)	10.1/2.8	1.72/1.73	13.0	92.33	9.3/2.4	1.42/1.44	13.6	92.17
AutoTemplate-small (Iso, 2024)	16.4/6.1	3.11/3.15	15.5	100.	22.5/9.5	3.51/3.63	17.1	100..
AutoTemplate-base (Iso, 2024)	18.3/7.6	3.39/3.45	16.0	100.	23.7/10.8	3.62/3.76	17.8	100..
AutoTemplate-large (Iso, 2024)	18.9/8.1	3.49/3.54	16.2	100.	24.1/11.1	3.68/3.83	17.9	100..
Non-Autoregressive Models								
Traditional Models								
SeqBF (Mou et al., 2016)	4.4/0.7	0.62/0.62	7.0	$\leq$ 100.	6.9/2.1	0.52/0.53	8.7	$<$ 100.
CGMH (Miao et al., 2019)	9.9/3.5	1.15/1.17	13.1	**100.**	12.3/4.6	1.41/1.45	14.6	$\leq$ 100.
POINTER (Zhang et al., 2020)	8.7/1.6	2.11/2.12	14.3	**100.**	10.6/2.4	2.14/2.16	16.8	**100.**
CBART (He, 2021)	15.6/**6.6**	2.16/2.19	**15.2**	**100.**	19.4/9.0	2.54/2.64	17.4	**100.**
Diffusion Models								
LC	15.36/6.52	2.02/2.05	14.99	99.82	20.55/**9.66**	2.76/2.87	17.58	99.75
PoP	**16.59**/5.66	3.06/3.09	15.15	99.05	21.13/7.97	3.31/3.40	**17.90**	98.87
DDOT-R (Ours)	15.7/5.1	**3.17/3.21**	14.6	99.60	19.7/7.0	3.26/3.34	17.4	99.77
DDOT-U (Ours)	16.3/5.2	3.13/3.17	15.0	**100.**	**21.2**/7.9	**3.43/3.52**	17.7	**100.**

Table 3: **DDOT performs on-par with state-of-the-art NAR models on short sequences (1–6 prompt tokens).** Top NAR scores are **bold**; second-best are underlined. Since SOTA NAR backbones (e.g. diffusion) still lag behind AR backbones, we focus on NAR comparisons.

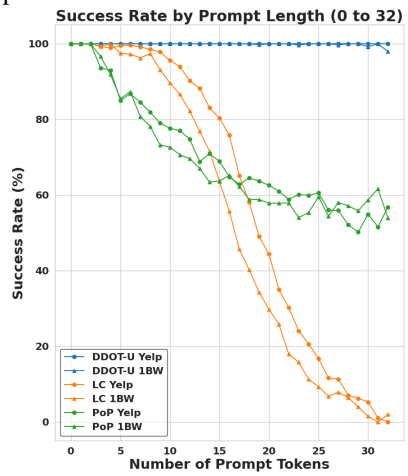


Figure 3: Success rate on block datasets. LC and PoP increasingly generate invalid responses (missing or swapping prompt tokens) as the number of prompt tokens grows.

modules on FineWeb-Edu (Penedo et al., 2024). Afterward, we further fine-tune on the One-Billion-Word and Yelp datasets. For simplicity, we keep all parameters unfrozen and optimize $\mathcal{L}_{\text{tok}}$ and $\mathcal{L}_{\text{pos}}$ simultaneously.

In line with SEDD, we train our model in two configurations: *small* (90M non-embedding parameters) and *medium* (320M non-embedding parameters). DDOT-medium is on the same scale as CBART (406M parameters) and AutoTemplate-base (220M parameters). Following SEDD, we use the AdamW optimizer with a learning rate of 3×10^{-5}. We set $\lambda = 10$ for position-loss weighting. For each experiment, we use either $48\times$ L40 (48 GB), $80\times$ A30 (24 GB), or $8\times$ A100 (80 GB) GPUs.

Baselines We compare our method against strong autoregressive (AR) and non-autoregressive (NAR) baselines. AutoTemplate (Iso, 2024), a state-of-the-art AR approach, leverages the T5 (Raffel et al., 2020) family of pretrained models and parses the lexically constrained generation task into a template that is generated autoregressively from left to right. The previous state-of-the-art NAR method, CBART (He, 2021), is built on the pretrained BART framework (Lewis et al., 2020) and iteratively inserts tokens into a sequence.

We also introduce two diffusion-based baselines that follow the same training procedure as DDOT. *Left Context (LC)* concatenates all prompt tokens to the left of the sequence and generates the response to the right of a separator token. *Position Prediction (PoP)* uses a SEDD model with a linear head that first predicts the positions of every token; this sequence is then fed through a fine-tuned fixed-position SEDD.

Distribution Annealing Many lexically constrained generation baselines, including AutoTemplate and CBART, use distribution-annealing methods such as top-p, top-k, greedy decoding, and beam search. To provide a parallel to greedy decoding—which always takes the top token probability—we anneal the token-value distribution during sampling to include only the most probable non-mask token. Specifically, given the predicted probability that a token is the mask, $\hat{p}(x^{\text{mask}})$, we assign $1 - \hat{p}(x^{\text{mask}})$ to the token value with the highest probability excluding the mask

token, and set the remaining token probabilities to 0. Greedy decoding in prior models (e.g., AR) is deterministic, collapsing the tree of all generation paths into a single path. However, our annealing process maintains generation diversity (A.2.2) because the model still samples from the annealed distribution over the top token value and the mask token. Whenever possible, we evaluate against the greedy-decoding baseline.

Metrics Following prior work (Miao et al., 2019; Zhang et al., 2020; Iso, 2024), we evaluate BLEU-2/4 (Papineni et al., 2002), NIST-2/4 (Doddington, 2002), METEOR-v1.5 (Denkowski and Lavie, 2014), and success rate (SR), which is the percentage of responses that are valid infilling results. A result is determined as invalid when any prompt tokens in the generated sequence do not appear in the same relative order that they were provided.

5.2 Main Results

We present lexically constrained generation results in Table 2. Our approach uses greedy annealing and is compared against greedy decoding wherever applicable, including the CBART greedy-decoding baseline. Our method achieves competitive performance with previous NAR models, approaching AR performance. Notably, our model achieves state-of-the-art performance on most metrics among the diffusion baselines. Our method performs well on block infilling, which may be more useful in real-world applications. Furthermore, we observe that DDOT scales well to longer sequences: it frequently generates valid responses that include all prompt words in the same relative order, as reflected in the SR. In contrast, **diffusion baselines quickly generate invalid responses** as the number of prompt tokens increases (Table 3, Table 2). However, in benchmarks with six or fewer prompt tokens, diffusion baselines maintain high SR (Table 3). This may be because fixed-position models have room to correct generation when the ratio of prompt to response tokens is low.

Table 3 compares results with previous work in lexically constrained generation. Since pretrained diffusion models currently lag behind AR models—an issue not unique to DDOT—we focus on NAR models. DDOT performs on par with previous state-of-the-art models and achieves higher SR than all diffusion baselines. Although DDOT underperforms LC and PoP on some metrics, we argue that the One-Billion-Word–Random and Yelp–Random settings over-index on the unrealistic task of generating text from only 1–6 randomly spaced tokens. Furthermore, Table 3 shows the broader trend when DDOT is scaled to a larger number of prompt tokens.

5.3 Analysis

In this section, we investigate the effect of random versus uniform position initialization, the inclusion of OT coupling, and the impact of varying the number of sampling steps.

Position Initialization In Table 2 and Table 3, we also explore the difference between DDOT-R and DDOT-U. In DDOT-R, pad tokens tend to cluster in high-density regions because OT finds no matches for them; in DDOT-U, pad tokens tend to be evenly spaced. We find that DDOT-U consistently outperforms DDOT-R.

OT Coupling To demonstrate the importance of OT coupling between source and target positions, we retrain the small version of DDOT without OT coupling and provide a quantitative comparison in Table 4. Models trained with OT coupling consistently outperform those using independent (random) coupling. We hypothesize that OT coupling provides a stronger signal about token ordering throughout the diffusion process. Specifically, DDOT guarantees that the relative ordering of the prompt and generated tokens at any time step matches the original ordering. In contrast, independent coupling requires the model to infer the original ordering—a challenging task given the numerous plausible orderings that can result from interspersed prompt tokens.

OT?	B2	B4	N2	N4	M
One-Billion-Word					
No OT	13.2	4.6	1.60	1.62	14.1
OT	**15.7**	**5.1**	**3.2**	**3.2**	**14.6**
Yelp					
No OT	16.39	5.94	2.05	2.10	16.0
OT	**18.6**	**6.9**	**3.13**	**3.21**	**16.6**

Table 4: **Our OT coupling drastically improves preformance across all metrics.** Ablation on OT coupling with small model size.

Position Over Time We qualitatively compare ground-truth token paths during training in Figure 4. With OT coupling, token trajectories exhibit significantly fewer crossings, maintaining relative order throughout the generation process;

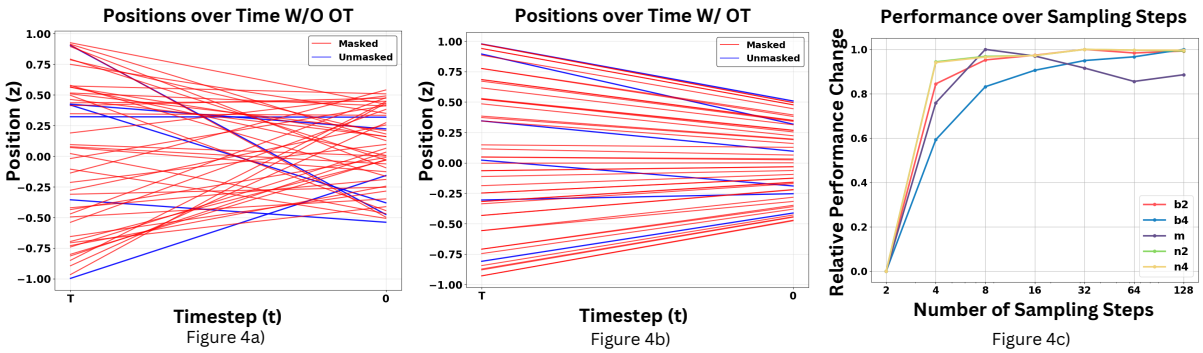


Figure 4a) Figure 4b) Figure 4c)

Figure 4: (a) and (b) show ground-truth token positions over time. Without OT (a), many line crossings indicate unstable permutations, whereas with OT coupling (b), trajectories are nearly straight throughout the denoising process. (c) Performance tends to increase with more sampling steps.

in contrast, independent coupling frequently permutes tokens. Visualizations of token paths during inference are available in subsection A.4.

Number of Sampling Steps One advantage of diffusion models over autoregressive models is the ability to trade compute for accuracy by varying the number of inference steps. Figure 4 shows how the number of sampling steps influences lexically constrained generation performance: as the number of sampling steps increases, performance improves.

Wall-Time Analysis We evaluate the inference speed of DDOT against the diffusion baselines on the One-Billion-Word. Table 5 presents the wall-clock time per batch alongside BLEU-2 and BLEU-4 scores for increasing numbers of sampling steps.

DDOT demonstrates significantly better efficiency. For any given number of sampling steps, DDOT is not only faster than LC and competitive with PoP in raw speed, but also achieves substantially higher BLEU scores. Notably, LC must regenerate prompt tokens and therefore requires up to double the input sequence length. PoP also requires an additional forward pass to predict initial positions.

Efficiency Considerations The added modules that enable position prediction are lightweight, consisting of a linear head and two type embeddings. By contrast, the LC baseline requires double the context length of DDOT because it must regenerate prompt tokens.

The OT calculation is highly efficient, taking 16 minutes, 11 seconds on an Intel Xeon 8462Y+ 64-core processor for the 10-billion-token subset of FineWeb-Edu. In practice, we stream the dataset, caching several OT couplings in advance without needing to preprocess them. With caching, it takes 4 minutes, 30 seconds to run 1,000 training steps on an L40 GPU with a batch size of 256. Without caching, it takes 4 minutes, 27 seconds—a negligible difference.

Model	Number of Sampling Steps					
	2	4	8	16	32	64
LC (Left Context)						
Time (s/batch)	0.71	1.43	2.86	5.72	11.5	22.9
BLEU-2	38.3	44.3	50.7	53.3	54.5	55.1
BLEU-4	27.5	33.9	41.9	45.3	47.0	47.9
PoP (Position Prediction)						
Time (s/batch)	0.53	0.90	1.64	3.13	6.09	12.0
BLEU-2	45.4	59.5	59.5	59.4	59.3	59.2
BLEU-4	40.7	50.3	50.3	50.2	50.1	50.0
DDOT-Uniform (Ours)						
Time (s/batch)	**0.44**	**0.81**	**1.54**	**3.03**	**5.98**	**12.0**
BLEU-2	**59.4**	**61.2**	**62.4**	**63.1**	**63.5**	**63.7**
BLEU-4	**55.5**	**56.8**	**57.5**	**58.0**	**58.2**	**58.4**

Table 5: **DDOT achieves superior BLEU scores with faster inference times.** Inference speed (seconds per batch) and BLEU scores on One-Billion-Word for varying numbers of sampling steps.

6 Conclusion

In this work, we introduce DDOT, the first discrete diffusion model capable of flexible-length text infilling by jointly denoising token values and positions. By incorporating optimal-transport coupling, DDOT preserves token order while enabling dynamic position adjustment, addressing limitations in existing text diffusion models. Our experiments show that DDOT outperforms diffusion baselines and matches state-of-the-art non-autoregressive models.

Limitations

DDOT inherits many drawbacks from previous discrete text diffusion methods (Lou et al., 2024; Austin et al., 2021; Sahoo et al., 2024). First, diffusion backbones still lag behind autoregressive backbones. Since diffusion for text is an emerging field, this performance gap is expected. Similarly, the fine-tuned versions of these models (e.g., DDOT vs. AutoTemplate) exhibit the same gap. Second, unlike autoregressive models, existing diffusion models cannot use a KV cache to store previous activations. DDOT also inherits risks common to large-scale text models.

Ethical Considerations

Artifacts The artifacts we used have public-use licenses. Furthermore, we plan to release our code artifacts for public use after acceptance.

Dataset Considerations We use publicly available datasets that underwent safety checks, such as FineWeb-Edu.

Documentation of Artifacts Our work generates English text, and our codebase is primarily in Python.

Use of AI Assistants We used AI assistants to help write our code and revise our paper.

Package Usage We use NLTK for BLEU, NIST, METEOR, and n-gram diversity. We use `https://pypi.org/project/codebleu/` for CodeBLEU.

Acknowledgements

We thank Virginia Tech's Advanced Research Computing (ARC) for computational resources and technical support, and the anonymous reviewers for their constructive feedback.

References

Michael S Albergo and Eric Vanden-Eijnden. 2023. Building normalizing flows with stochastic interpolants. In *11th International Conference on Learning Representations, ICLR 2023*.

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.

Hang Cao, Zhiquan Cao, Chi Hu, Baoyu Hou, Tong Xiao, and Jingbo Zhu. 2023. Improving autoregressive grammatical error correction with non-autoregressive models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12014–12027, Toronto, Canada. Association for Computational Linguistics.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*, pages 376–380.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. 2024. Discrete flow matching. *Advances in Neural Information Processing Systems*, 37:133345–133385.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, et al. 2024. Scaling diffusion language models via adaptation from autoregressive models. *CoRR*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Xingwei He. 2021. Parallel refinements for lexically constrained text generation with BART. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8653–8666, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546.

Hayate Iso. 2024. Autotemplate: A simple recipe for lexically constrained text generation. In *Proceedings of the 17th International Natural Language Generation Conference*, pages 1–12.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart:

Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Xingchao Liu, Chengyue Gong, and Qiang Liu. 2023. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations (ICLR)*.

Aaron Lou, Chenlin Meng, and Stefano Ermon. 2024. Discrete diffusion modeling by estimating the ratios of the data distribution. In *Proceedings of the 41st International Conference on Machine Learning*, pages 32819–32848.

Sidi Lu, Tao Meng, and Nanyun Peng. 2022. Insnet: An efficient, flexible, and performant insertion-based text generation model. In *Advances in Neural Information Processing Systems*, volume 35, pages 7011–7023. Curran Associates, Inc.

Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. 2019. Cgmh: Constrained sentence generation by metropolis-hastings sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6834–6842.

Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3349–3358.

Saul B Needleman and Christian D Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

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.

William Peebles and Saining Xie. 2023. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, et al. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849.

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. *Journal of machine learning research*, 21(140):1–67.

Yinuo Ren, Haoxuan Chen, Grant M Rotskoff, and Lexing Ying. 2024. How discrete and continuous diffusion meet: Comprehensive analysis of discrete diffusion models via a stochastic integral framework. *CoRR*.

Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. 2024. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37:103131–103167.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *International Conference on Machine Learning*, pages 5976–5985. PMLR.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. 2024. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, pages 1–34.

Cédric Villani et al. 2009. *Optimal transport: old and new*, volume 338. Springer.

Yizhe Zhang, Guoyin Wang, Chunyuan Li, Zhe Gan, Chris Brockett, and William B Dolan. 2020. Pointer: Constrained progressive text generation via insertion-based generative pre-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8649–8670.

A Appendix

A.1 Overview

This appendix provides additional results, diversity analyses, hyperparameter sensitivity, path visualizations, and short implementation notes for reproducibility. We reuse notation from the main text and make one explicit addition used below:

$$\ell^{\mathrm{np}} \;=\; L - \ell^{p}$$

denoting the number of non-prompt slots (response + pads). This matches the usage of z_T^{np} in the main text.

A.2 Additional Results

A.2.1 Size Comparison

Table 6 reports BLEU-2/4, NIST-2/4, and METEOR for DDOT small/medium and for DDOT-R vs. DDOT-U on One-Billion-Word–Random and Yelp–Random. Uniform z_T initialization (DDOT-U) consistently outperforms random (DDOT-R), and scaling to the medium model further improves most metrics.

Model	**One-Billion-Word-Random**					**Yelp-Random**				
	B2	B4	N2	N4	M	B2	B4	N2	N4	M
DDOT-R small (Ours)	15.7	5.1	3.16	3.19	14.6	18.6	6.9	3.13	3.21	16.6
DDOT-U small (Ours)	15.5	4.9	2.94	2.97	**15.0**	18.9	7.1	2.99	3.07	17.0
DDOT-R medium (Ours)	15.7	5.1	**3.17**	**3.21**	14.6	19.7	7.0	3.26	3.34	17.4
DDOT-U medium (Ours)	**16.3**	**5.2**	3.13	3.17	**15.0**	**21.2**	**7.9**	**3.43**	**3.52**	**17.7**

Table 6: Results on One-Billion-Word and Yelp. Metrics are BLEU (B2, B4), NIST (N2, N4), and METEOR (M). Best results are highlighted in **bold**.

A.2.2 Generation Diversity

Table 7 compares diversity using D2/D4 (unique bigrams/four-grams). DDOT-U yields competitive diversity relative to strong NAR baselines while maintaining ordering constraints.

Model	**One-Billion-Word-Random**		**Yelp-Random**	
	D2	D4	D2	D4
CBART (He, 2021)	**49.2**	82.1	**38.1**	**92.6**
LC	37.5	89.4	21.5	69.0
PoP	38.85	**91.4**	19.9	68.3
DDOT-R (Ours)	30.7	85.3	20.2	67.7
DDOT-U (Ours)	34.03	88.2	18.6	66.6

Table 7: Results on One-Billion-Word and Yelp dataset for generation diversity. Metrics include Diversity (D2, D4). Best results are highlighted in **bold**.

A.3 Hyperparameter Sensitivity and Training Details

A.3.1 Loss Weight λ

We sweep λ to balance $\mathcal{L}_{\text{tok}}$ and $\mathcal{L}_{\text{pos}}$. Results in Table 8 show stable performance across a range; $\lambda \approx 10$ is a good default used in the main results.

A.3.2 Training Hyperparameters

Table 9 lists optimizer, learning rate, weight decay, βs, batch sizes (by GPU), and schedules. Unless noted, we follow SEDD defaults and fine-tune all parameters jointly.

λ	B2	B4	N2	N4	M
3	63.07	58.27	7.56	8.32	42.64
10	63.04	58.23	7.55	8.30	42.85
30	63.64	58.63	7.73	8.50	42.66
100	63.05	58.18	7.56	8.32	42.6
300	63.25	58.24	7.63	8.39	42.50
1000	63.53	58.46	7.74	8.51	42.50
3000	62.95	57.95	7.59	8.34	42.27

Table 8: Impact of different λ values on model performance metrics.

Hyperparameter	Value
Batch size	128 (A30) / 256 (L40) / 512 A(100)
λ	300 (with scaling)
Weight decay	0.0
Learning rate	$3e-4$
Optimizer	AdamW
β_1	0.9
β_2	0.999

Table 9: Training hyperparameter specifications.

A.4 Path Visualizations

We visualize token-position *trajectories* during both **inference** (what the model actually does at test time) and **training** (the straight paths used as supervision). We show two coordinate systems: **scaled** to $[-1, 1]$ (used in losses; cf. Equation 3) and **unscaled** in the model's native context length (for intuition about absolute movement). Unless noted, we split by set $S \in \{p, r\}$ and index tokens within a set by their ground-truth order.

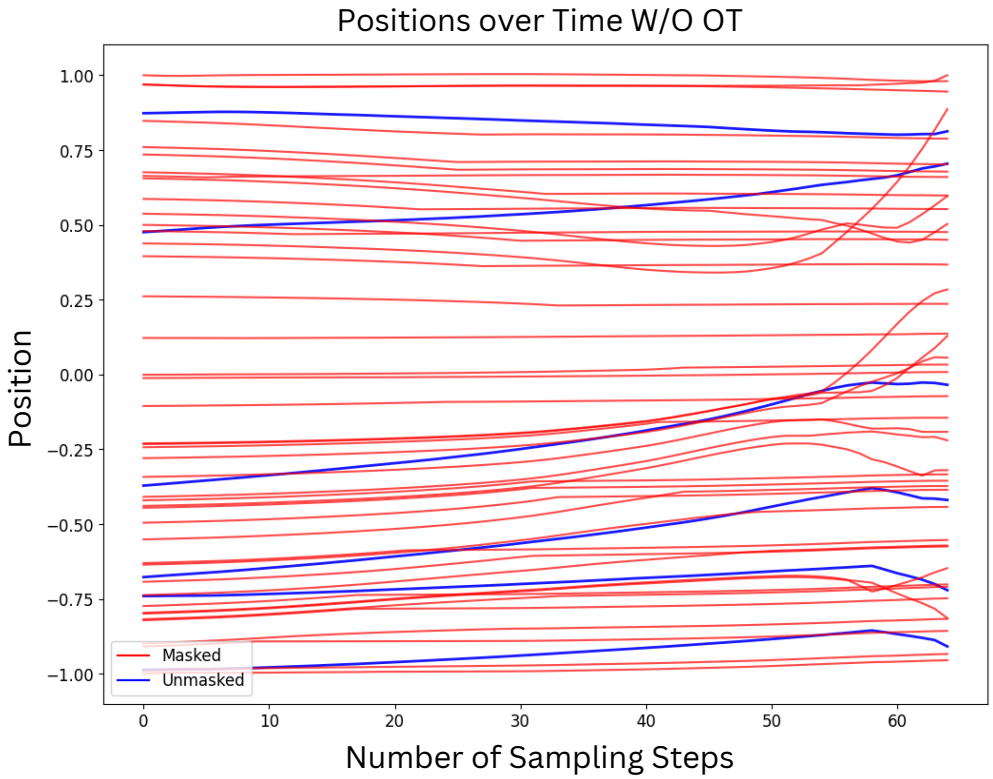
Plot Explanation. Let $\{t_u\}_{u=0}^{K}$ be the reverse-time grid (left-to-right on the x-axis), with $t_0 = T$ and $t_K = 0$. For each token index i in set S, we plot its 1D position $z_{t_u}^{S,(i)}$ across u. Colors denote the token's state (masked vs. unmasked) at each step. During inference, $z_{t_{u+1}}$ is obtained by an Euler update using the learned velocity $v_\theta(z_{t_u}, x_{t_u}, t_u)$ while clamping prompts and keeping pads stationary (Equation 5). During training, targets follow straight paths $z_t^S = (1-t)z_0^S + t\,\tilde{z}_T^S$ from Equation 8.

Analysis. Within a *set* (prompt or response), desirable behavior is (i) **order preservation** (no within-set crossings) and (ii) **near-straight trajectories** (little curvature). *Inter-set* crossings (prompt vs. response) are allowed—and often required—to place response tokens between prompts during infilling.

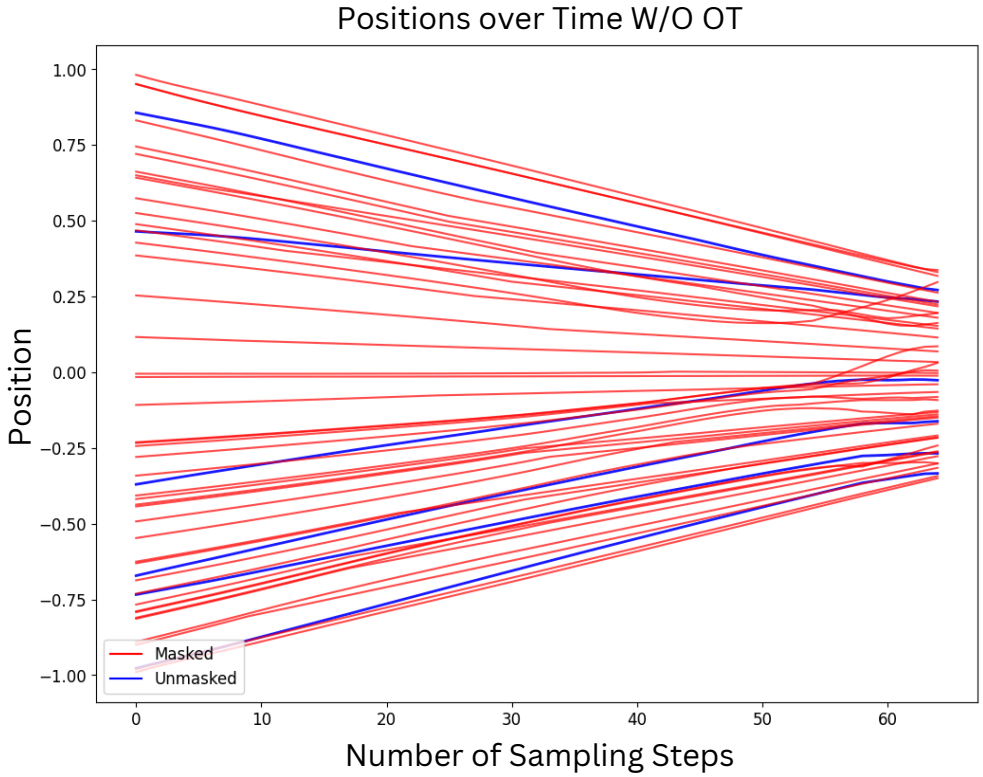
A.5 Implementation Notes

Computing OT efficiently. Prompt matching is solved exactly in 1D by sorting z_0^p and z_T^p and pairing by rank; response matching uses order-preserving injective alignment via global-alignment DP against z_T^{np}. Both avoid dense cost matrices and run in $O(\ell^p \log \ell^p + \ell^r(L - \ell^p))$ time and $O(L)$ memory.

Caching & streaming. For large corpora, precompute couplings on CPU and stream mini-batches; we observed negligible overhead whether caching or computing couplings on the fly.

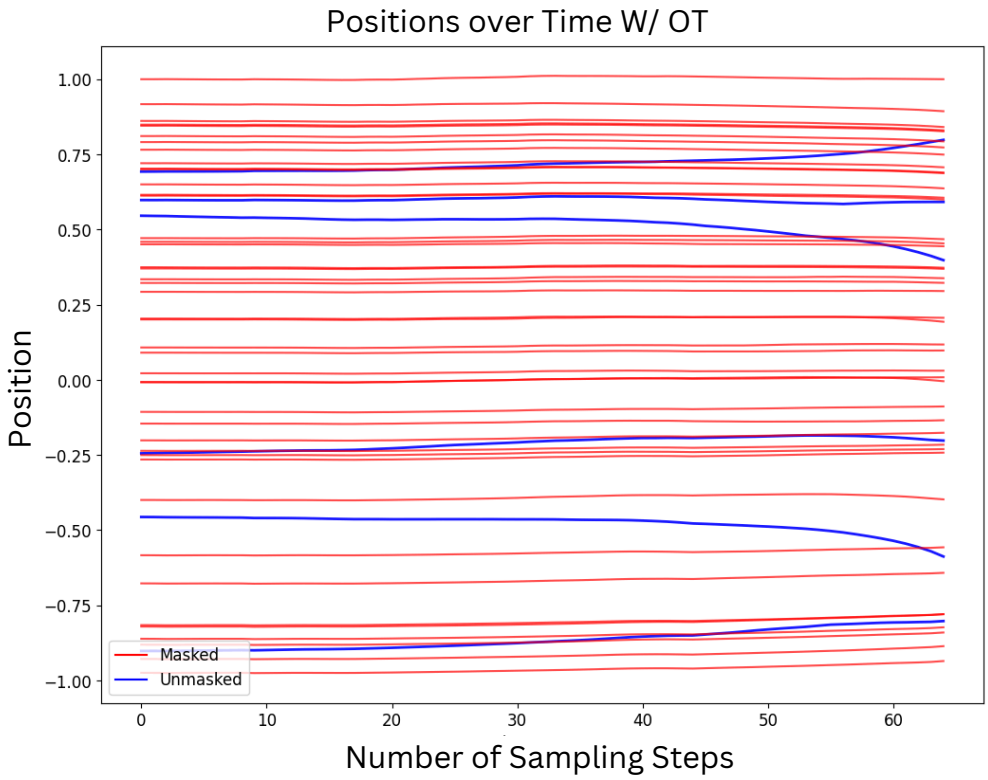


(a) Scaled positions over time without OT.

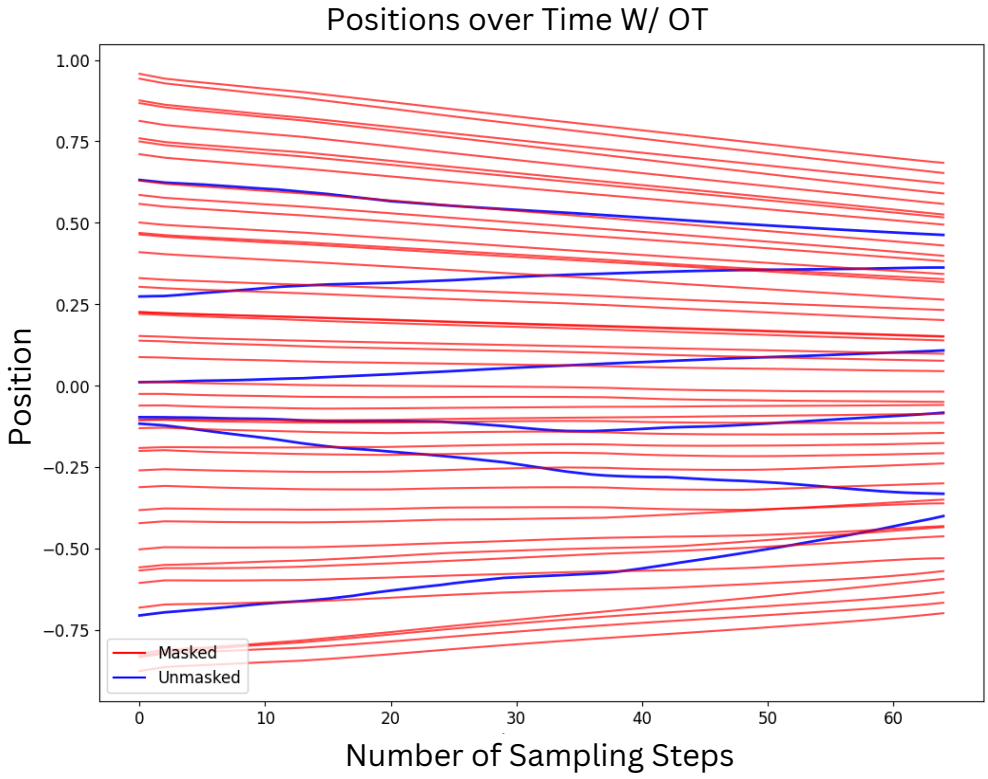


(b) Unscaled positions over time without OT.

Figure 5: Ablations on number of sampling steps without OT coupling. Paths curve and cross more often, complicating learning and inference.

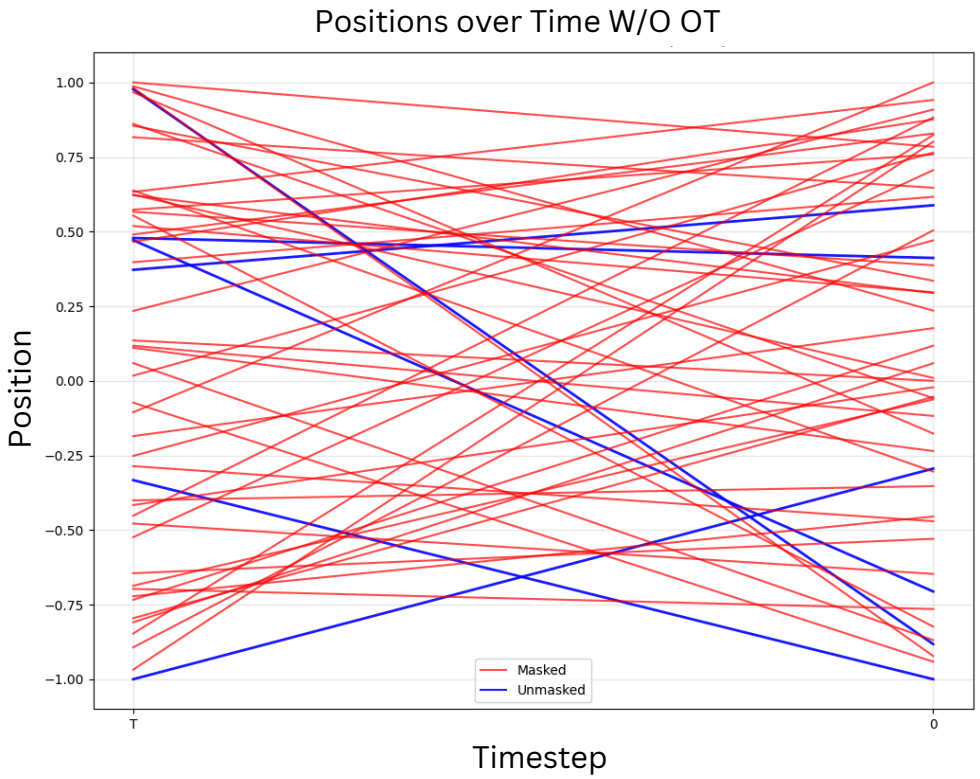


(a) Scaled positions over time with OT.

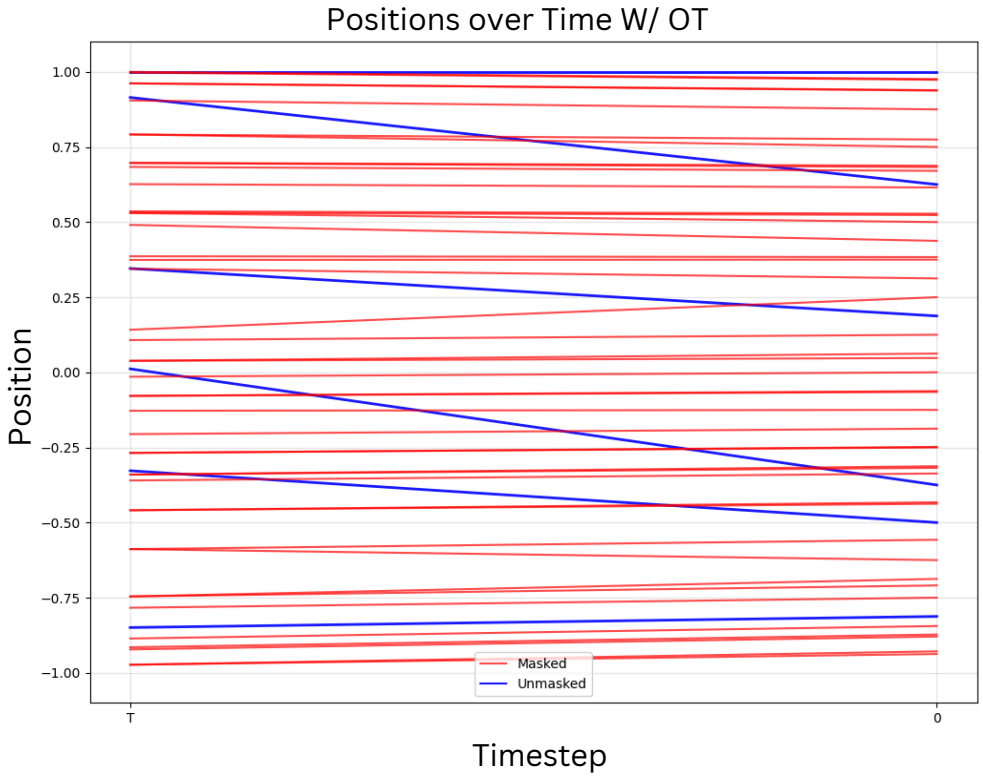


(b) Unscaled positions over time with OT.

Figure 6: With per-set OT coupling, trajectories are straighter and order-preserving within each set.



(a) Scaled training targets without OT: many crossings.



(b) Scaled training targets with OT: straighter, non-crossing (within-set).

Figure 7: Ground-truth token paths (scaled to $[-1, 1]$) used as supervision. Without OT (left), many crossings indicate unstable matching; with OT (right), trajectories are nearly straight and order-preserving within each set.