A Continuous-Time Markov Chain Framework for Insertion Language Models

Dhruvesh Patel
UMass Amherst

Benjamin Rozonoyer
UMass Amherst

Soumitra Das
UMass Amherst

Tahira Naseem
IBM Research

Tim G. J. Rudner
University of Toronto & Vijil

Andrew McCallum
UMass Amherst

Abstract

Insertion Language Models (ILMs) offer several advantages over left-to-right generation and mask-based generation. However, existing formulations of insertion-based generation have largely been ad-hoc. In this paper, we derive a diffusion-style denoising objective for ILMs from first principles by formulating the noising process as a continuous-time Markov chain on the space of variable-length sequences. We show that previous formulations of ILMs can be viewed as special cases of this denoising framework. Through empirical evaluation on a synthetic planning task, we show that the proposed approach retains the benefits of insertion-based generation over left-to-right generation and masked diffusion models. In language modeling, our diffusion-based approach is competitive with left-to-right generation and masked diffusion models, while offering additional flexibility in sampling compared to existing insertion language models.

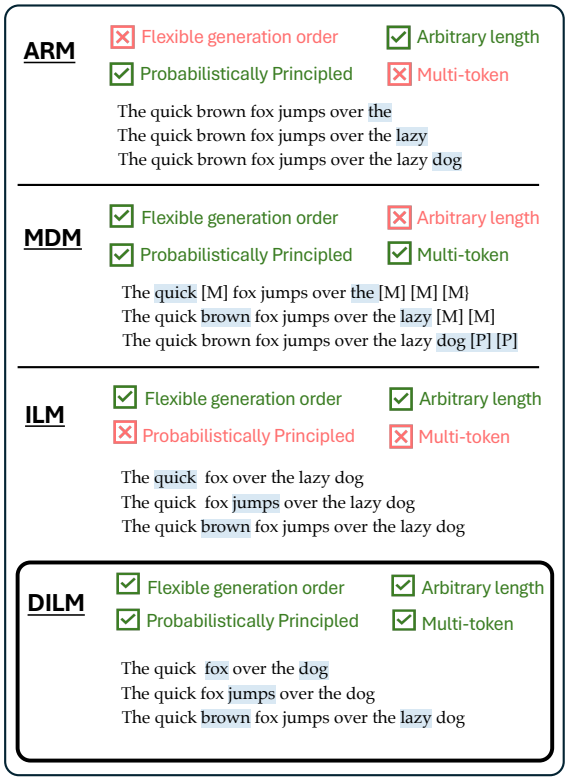


Figure 1: Diffusion-based Insertion Language Models (DILMs) retain the benefits of Insertion Language Models (ILMs) while offering principled training and sampling procedures based on diffusion using a continuous-time Markov chain framework.

1 INTRODUCTION

Autoregressive language models (ARMs) generate sequences by predicting tokens from left-to-right. They can naturally generate variable-length sequences by using an `EOS` token to mark the end of a sequence. However, certain tasks like planning and sub-goal based infilling require a more flexible approach for sequence generation (Bachmann and Nagarajan, 2024).

Masked diffusion models can generate sequences in arbitrary order while also generating multiple tokens per step (Lou et al., 2024; Sahoo et al., 2024; Shi et al., 2024). MDMs do not explicitly model the length of the sequence, but are trained to predict variable-length sequences by predicting the `PAD` tokens like ordinary tokens. This impacts the quality of generation and re-

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

stricts the sampling to block-based left-to-right sampling (Nie et al., 2025; Yang et al., 2026; Li et al., 2025; Wu et al., 2026). Insertion Language Models (ILMs) (Stern et al., 2019; Patel et al., 2025), on the other hand, generate sequences through expansion by iteratively inserting tokens at arbitrary positions. ILMs combine the benefits of ARMs and MDMs — they naturally generate variable-length sequences while maintaining the ability to generate in non-left-to-right order and using relative position constraints. ILMs perform better than MDMs and left-to-right autoregressive models (ARMs) on sub-goal based planning tasks and infilling tasks that require respecting relative ordering constraints (Patel et al., 2025).

Existing formulations of ILMs use ad-hoc training objectives and sampling procedures (Patel et al., 2025; Stern et al., 2019). In this work, we derive a principled diffusion-style denoising objective and sampling procedures for insertion-based generation. We formulate the noising process as a continuous-time Markov chain over the space of sequences that drops tokens uniformly with a time-dependent rate. We present two noising processes and respective transformer-based parameterizations of the rate matrix of the generative process (the reverse of the noising process). The first parameterization models the joint probability of vocabulary item and insertion location, and the probability of the length-to-go. The second parameterization models the rate matrix of the generative process using independent, per-dimension probabilities allowing one to sample multiple insertions at once. We derive the evidence lower bound on the data log-likelihood under the parameterized generative process, and discuss the necessary computational considerations for training and sampling.

The main contributions of this work are as follows:

1. We present a principled diffusion-style denoising framework for insertion-based generation and propose two variants of the generative process in this framework. Figure 1 shows how the proposed generative model relates to existing sequence generation models.
2. We show that the resulting sampling procedures unify the existing formulations of Insertion Language Models.
3. Empirical evaluation on synthetic planning tasks and language modeling demonstrates improved performance over existing ILMs and MDMs.

Code to reproduce our experiments is available at: https://github.com/dhruvdcoder/ctmc_dilm

2 BACKGROUND

In this section, we will state some elementary results from the theory of Markov chains to set up the framework for our method. We will also introduce some notation that will be used throughout the paper (see Appendix A for a summary of all notation).

Notation. Capital letters (e.g. $X, \boldsymbol{B}$) are used to denote (scalar or vector valued) random variables and subscripts are used to denote the time index of stochastic processes (e.g., X_t, $\boldsymbol{B}_t$, etc.). Boldface vectors and superscripts will be used to denote components of a vector, e.g., x^i, X_t^i, for the i-th component of $\boldsymbol{x}$ and $\boldsymbol{X}_t$, respectively. Double square brackets are used to denote the set of natural numbers up to a specific number, i.e., $[n] = \{1, 2, \ldots, n\}$. All stochastic processes are assumed to be continuous-time unless an underline is used, which denotes a discrete-time process (e.g. $\boldsymbol{X}_t, \underline{\boldsymbol{X}}_k$ are continuous- and discrete-time processes, respectively). We will use the shorthand $p_{s|t}(y \mid x)$ to denote the transition probability $\mathbb{P}(X_s = y \mid X_t = x)$, and in case of discrete time processes, we will use $\underline{p}_{r|k}(y \mid x)$.

Continuous-Time Markov Chains. Here we briefly discuss continuous-time Markov chains (CTMCs), and their characterization in terms of the rate matrix. In section 3, we will leverage the relationship between DTMCs and CTMCs to describe our noising process. Let X_t be a CTMC taking values in a countable state space $\mathbb{X}$ with right-continuous sample paths. Its dynamics can be described using a state- and time-dependent jump rate $\lambda_t(x)$ and a post-jump transition kernel $K_t(x, y) \geq 0$ satisfying $K_t(x, x) = 0$ and $\sum_{y \in \mathbb{X}} K_t(x, y) = 1$. The corresponding transition rate matrix of X_t is then given by

$$R_t(x, y) = \begin{cases} \lambda_t(x) K_t(x, y) & \text{if } x \neq y, \\ -\lambda_t(x) & \text{if } x = y, \end{cases} \quad \text{with}$$
$$K_t(x, x) = 0 \quad \text{and} \quad \sum_{y \in \mathbb{X}} K_t(x, y) = 1. \tag{1}$$

An informal description of the evolution of the transition probability in continuous time for small values of h is given by the first-order discretization

$$p_{t+h|t}(y \mid x) = \delta_x(y) + h\, R_t(x, y) + o(h), \tag{2}$$

where $\delta_x(y)$ is the indicator function that is 1 if $x = y$ and 0 otherwise. Intuitively, the tendency $\delta_x(y)$ to remain in the same state x is counteracted by the passage of time and the rate parameter $\lambda_t(x) = \sum_{y \neq x} R_t(x, y)$.

Time Reversal. On the time interval $[0, T]$, for any regular CTMC X_t on countable state space, with initial distribution p_0 and rate R_t, the time reversal $\hat{X}_t := X_{T-t}$ is also a CTMC with rate $\hat{R}_t(x, y) =$

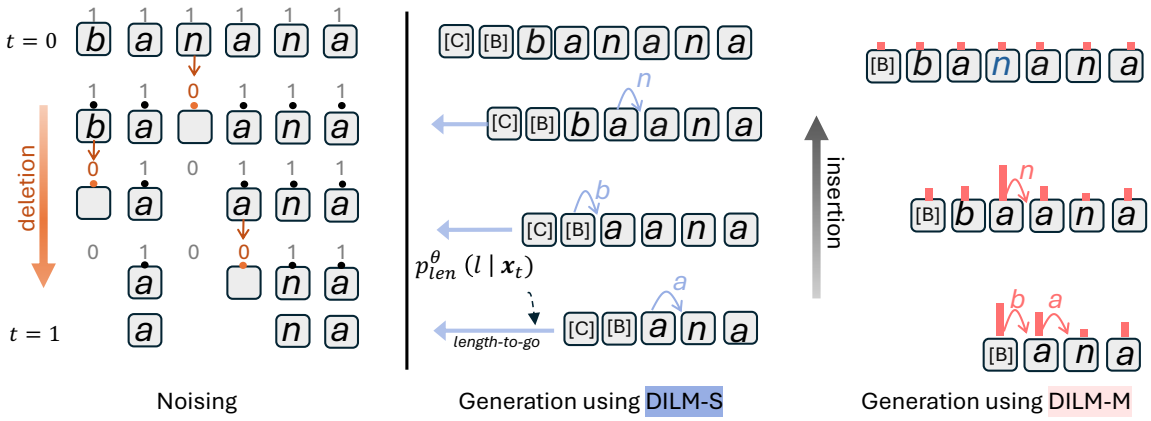


Figure 2: The noising process, shown on the left, is a continuous-time Markov chain that deletes one token at a time uniformly with the deletion events arriving with rate σ_t. The bit vectors show the deletion path w.r.t. the original sequence $\boldsymbol{x}_0 = (6, \text{banana})$. On the right we show two parameterizations of the generative process. DILM-S predicts the length-to-go and a joint probability $p^\theta_{\text{ins}}(i, w \mid \boldsymbol{x}_t)$ over insertion locations and vocabulary items. The generation stops when the predicted insertion rate modeled through the predicted *length-to-go* value, shown using the blue arrows pointing left, goes to zero. DILM-M uses per-position insertion rate, shown as the vertical blue bars, and $p^\theta_{\text{ins}}(w \mid \boldsymbol{x}_t, i)$. The generation stops when the insertion rate is low for all positions. `[C]` and `[B]` are special tokens, where the former is used by DILM-S to perform length-to-go prediction and the latter is used by both models as a beginning-of-sequence marker to predict the token at the first gap.

$\frac{p_t(y)}{p_t(x)} R_t(y, x)$. [1] Let $\hat{R}^\theta_t$ denote a parameterized rate for the time reversal $\hat{X}_t$ for some parameter value θ. Then the following proposition gives a lower bound on the log-likelihood of the data under the parameterized model.

Proposition 1 (ELBO). *For the time interval $[0, 1]$, let $p_0 = p_{\text{data}}$ and $p_1 = p_{\text{ref}}$ denote the terminal time marginals of the noising process X_t. Then for the reverse CTMC with initial distribution p_{ref}, parameterized rate $\hat{R}^\theta_t$, and the induced terminal marginal p^θ_0, the log-likelihood $\mathbb{E}\left[\log p^\theta_0(X_0)\right]$ under the model is bounded from below by*

$$\mathbb{E} \sum_{y \neq x} \left[-\hat{R}^\theta_t(x, y) + \frac{p_{t|0}(y|x_0)}{p_{t|0}(x|x_0)} R_t(y, x) \log \hat{R}^\theta_t(x, y) \right]$$

where the expectation is over $x_0 \sim p_{\text{data}}$, $t \sim \text{Uniform}[0, 1]$, $x \sim p_{t|0}(x \mid x_0)$.

The result follows by applying Dynkin's formula for the change of measure to the CTMC path measures (Hanson, 2007), followed by the data processing inequality. We provide an intuitive proof using elementary techniques in Appendix C.1.

[1] We have relabeled the time index of the reverse process $T - t$ as t to keep the notation simple. Also, at the jump points the time reversal's value is the left limit of the forward process.

3 DIFFUSION-BASED INSERTION LANGUAGE MODELS

For ease of exposition, we will make the sequence length explicit by incorporating it into the state-space, and therefore our domain will be the set of all sequences up to a maximum length $\mathbb{X} := \bigcup_{n=0}^{l_{\max}} (\{n\} \times \mathbb{V}^n)$, with a single example being the tuple $\boldsymbol{x} = (n, \dot{\boldsymbol{x}})$, where n is the sequence length and $\dot{\boldsymbol{x}} \in \mathbb{V}^n$ the actual sequence of tokens. We will omit the dot in $\dot{\boldsymbol{x}}$ and write it as $\boldsymbol{x}$ when the distinction is not needed. We will use the shorthand $\mathbb{X}_n$ to mean $\{n\} \times \mathbb{V}^n$ — the collection of all sequences of length n.

3.1 Noising Process

The noising process proceeds by deleting tokens from the ground-truth sequence $\boldsymbol{x}_0$. To provide intuition, we first formulate the deletions as a homogeneous discrete-time Markov chain with no self-transitions, and then embed it in continuous time.

Formally, any deletion process that does not depend on the content of the sequence can be described by expressing the transitions $(n, \dot{\boldsymbol{x}}) \to \cdots \to (m, \dot{\boldsymbol{y}})$ using bit vectors in $\mathbb{B}_n = \bigcup_{m=0}^{n} \mathbb{B}_{n,m}$, where $\mathbb{B}_{n,m}$ is the set of binary vectors of length n with exactly m ones, i.e. $\mathbb{B}_{n,m} = \{\boldsymbol{b} \in \{0, 1\}^n : \sum_{i=1}^{n} b^i = m\}$. Let $\text{Idx}(\boldsymbol{b}) := \{i \in [n] \mid b^i = 1\}$ be the set of indices of the ones in $\boldsymbol{b}$, and $\boldsymbol{x}[\boldsymbol{b}] := \boldsymbol{x}^{\text{Idx}(\boldsymbol{b})}$ be the subsequence of $\boldsymbol{x}$ that only contains elements corresponding to the ones in $\boldsymbol{b}$. For $\boldsymbol{b}_1, \boldsymbol{b}_2 \in \mathbb{B}_n = \bigcup_{m=0}^{n} \mathbb{B}_{n,m}$, we define a

partial order $\boldsymbol{b}_1 \succ \boldsymbol{b}_2$ if $b_1^i * b_2^i = b_2^i$ for all $i \in [n]$ and $\sum_{i=1}^n b_1^i > \sum_{i=1}^n b_2^i$, i.e. we call $\boldsymbol{b}_1$ a predecessor of $\boldsymbol{b}_2$ if the positions that are 1 in $\boldsymbol{b}_2$ are also 1 in $\boldsymbol{b}_1$, and the number of ones in $\boldsymbol{b}_2$ is less than the number of ones in $\boldsymbol{b}_1$, e.g., $11111 \succ 10101 \succ 10001$. Using this notation, a general r-step deletion path $(n_0, \dot{\boldsymbol{x}}_0) \to (n_1, \dot{\boldsymbol{x}}_1) \to \cdots \to (n_{r-1}, \dot{\boldsymbol{x}}_{r-1}) \to (n_r, \dot{\boldsymbol{x}}_r)$ can be expressed using r bit vectors $\boldsymbol{b}_1 \succ \cdots \succ \boldsymbol{b}_r$ in $\mathbb{B}_{n_0} = \bigcup_{k=0}^{n_0} \mathbb{B}_{n_0,k}$ such that $\boldsymbol{x}_k = \boldsymbol{x}_0[\boldsymbol{b}_k]$. To make this concrete, consider $\boldsymbol{x}_0 = (6, \text{banana})$, and the deletion path specified by the bit vectors $\boldsymbol{b}_1 = 110111, \boldsymbol{b}_2 = 010011$, resulting in $\boldsymbol{x}_2 = (3, \text{ana})$ as shown in Figure 2. Notice that there could be multiple deletion paths that lead from $(n, \dot{\boldsymbol{x}}_0)$ to $(m, \dot{\boldsymbol{x}}_r)$; in our example, $\boldsymbol{b}_1 = 011111 \succ \boldsymbol{b}_2 = 000111$, or just $\boldsymbol{b}_1 = 000111$, would also lead to $(3, \text{ana})$. Since the deletion process does not depend on the content of the state $\boldsymbol{x}$ but only on the length, the probability of one step of deletion has the form $\kappa(\boldsymbol{b}_{k+1} \mid \boldsymbol{b}_k) \propto \delta_{\{\boldsymbol{b}_k \succ \cdot\}}(\boldsymbol{b}_{k+1})\, u(\boldsymbol{b}_k, \boldsymbol{b}_{k+1})$, where u is an unnormalized joint probability mass function. In order to make the process analytically tractable, we further restrict the deletion process such that it deletes *one token at a time* uniformly at random, which implies that $u(\boldsymbol{b}_k, \boldsymbol{b}_{k+1}) = \delta_{|\boldsymbol{b}_k|-1}(|\boldsymbol{b}_{k+1}|)$, where $|\boldsymbol{b}_k| = \sum_j b_k^j$ is the number of 1s in $\boldsymbol{b}_k$. Putting these constraints together we get

$$\kappa_{\text{Uni}}(\boldsymbol{b}_{k+1} \mid \boldsymbol{b}_k) = \frac{1}{|\boldsymbol{b}_k|} \delta_{\{\boldsymbol{b}_k \succ \cdot\}}(\boldsymbol{b}_{k+1})\ \delta_{|\boldsymbol{b}_k|-1}(|\boldsymbol{b}_{k+1}|).$$

The following lemma shows that such a noising process is a homogeneous DTMC with transition kernel $K(\boldsymbol{x}, \boldsymbol{y})$ that produces a closed-form expression for the r-step transition probability $\underline{p}_{k+r|k}(\boldsymbol{y} \mid \boldsymbol{x})$.

Lemma 1 (Deletion DTMC)**.** *Let $\underline{\boldsymbol{X}}_k$ be a discrete-time stochastic process taking values in $\mathbb{X}$ with one-step transition probabilities governed by the deletion kernel κ_{Uni}. Then, letting $\hat{C}_m^n \coloneqq \frac{(n-m)!}{n(n-1)\ldots(m+1)}$, the process $\underline{\boldsymbol{X}}_k$ is a DTMC with one-step transition kernel and transition probability given by*

$$K((n, \dot{\boldsymbol{x}}), (m, \dot{\boldsymbol{y}})) = \frac{1}{n} \sum_{\boldsymbol{b} \in \mathbb{B}_{n,m}} \delta_{n-1}(m)\, \delta_{\dot{\boldsymbol{x}}[\boldsymbol{b}]}(\dot{\boldsymbol{y}}),$$

$$\underline{p}_{k+r|k}\left((m, \dot{\boldsymbol{y}}) | (n, \dot{\boldsymbol{x}})\right) = \hat{C}_m^n\ \delta_r(n-m) \sum_{\boldsymbol{b} \in \mathbb{B}_{n,m}} \delta_{\dot{\boldsymbol{x}}[\boldsymbol{b}]}(\dot{\boldsymbol{y}}).$$

Proof sketch. To get the intuition for the expression for transition probability, note that for κ_{Uni}, the probability of any $(n-m)$ step delete path $\boldsymbol{b}_{n-m} \succ \cdots \succ \boldsymbol{b}_1$ is the same and is equal to $\frac{1}{n}\frac{1}{n-1}\cdots\frac{1}{m+1}$. Meanwhile, the number of paths one can take from $\boldsymbol{b}_{n-m}$ down to $\boldsymbol{b}_1$ is $\binom{n-m}{1}\binom{n-m-1}{1}\ldots\binom{1}{1} = (n-m)!$. Therefore, the required probability is $\hat{C}_m^n = \frac{(n-m)!}{n(n-1)\ldots(m+1)} = \frac{1}{\binom{n}{n-m}}$. The formal proof is given in Appendix C.

Remark 1. Note that the transition probability is not uniform over the set of sequences reachable from $(n, \dot{\boldsymbol{x}})$ in r steps. Specifically, if there is more than one mask $\boldsymbol{b} \in \mathbb{B}_{n,n-r}$ such that $\dot{\boldsymbol{x}}[\boldsymbol{b}] = \dot{\boldsymbol{y}}$, then the transition probability to land on $\dot{\boldsymbol{y}}$ is the sum of the probabilities over all such cases. One can obtain a similar closed-form expression for the transition probability for any deletion kernel that only depends on the length and not the content of the sequence; for example, the right-to-left deletion is a special case, which leads to the usual left-to-right AR generative model.

Having described the deletion process using discrete steps, we now introduce the jump rate $\lambda_t(\boldsymbol{x})$ to complete the description of the continuous-time deletion process (eqs. (1) and (2)). Intuitively, $\underline{\boldsymbol{X}}_k$ is converted into a continuous-time process $\boldsymbol{X}_t$ by dispersing the deletion steps over the time interval $[0, 1]$ with deletion rate $\lambda_t(\boldsymbol{x})$ per unit time.

For efficient sampling from the noising process, we keep the rate $\lambda_t(\boldsymbol{x})$ independent of the state $\boldsymbol{x}$ except for the case when the length of the sequence is 0, in which case the rate is 0 as well, i.e., the rate is $\lambda_t((r, \dot{\boldsymbol{x}})) = \sigma_t\, \delta_{\{\cdot>0\}}(r)$, where $\sigma : [0, 1] \to \mathbb{R}_+$ is a scalar noise schedule. The following proposition gives the transition probability under this noising process.

Proposition 2 (Transition Probability)**.** *Let $\boldsymbol{X}_t$ be a continuous-time stochastic process described in equation 1, with transition rate $\lambda_t((r, \dot{\boldsymbol{x}})) = \sigma_t\, \delta_{\{\cdot>0\}}(r)$ and transitions governed by the transition kernel K that drops one token at a time uniformly at random. Then $\boldsymbol{X}_t$ is an inhomogeneous CTMC with transition probability for $s \le t$ given by*

$$p_{t|s}^{\text{joint}}\left((m, \dot{\boldsymbol{y}}) \mid (n, \dot{\boldsymbol{x}})\right) = \begin{cases} \frac{\exp(-\bar{\sigma}_{s,t})(\bar{\sigma}_{s,t})^{n-m}\ m!}{n!} \sum_{\boldsymbol{b} \in \mathbb{B}_{n,m}} \delta_{\dot{\boldsymbol{x}}[\boldsymbol{b}]}(\dot{\boldsymbol{y}}), & 0 < m \le n \\ 1 - \sum_{r=1}^{n} \frac{\exp(-\bar{\sigma}_{s,t})(\bar{\sigma}_{s,t})^{n-r}\ r!}{n!}, & m = 0 \end{cases}$$

where $\sigma : [0, T] \to \mathbb{R}_+$ is a scalar noise schedule, $\bar{\sigma}_{s,t} = \int_s^t \sigma(u)\, du$.

Proof sketch. We can decouple the number of tokens dropped at time t from the choice of tokens dropped, where the former has a Poisson distribution with mean rate $\bar{\sigma}_{s,t}$ and the latter has a distribution given in Lemma 1. We separately take care of the case when the number of tokens dropped is equal to the total number of tokens in the sequence. The complete proof is given in Appendix C.

Proposition 2 allows us to sample from the noising process by first sampling the number of positions to drop using the Poisson distribution with rate $\bar{\sigma}_{0,t}$, with maximum drops equal to the length of the sequence, and then sampling the positions to drop by permuting the indices of the sequence and picking the first $n - d$ positions. Sampling from this noising process, which we call Joint Noising, is described in Algorithm 1.

Independent Noising. We can also obtain uniform random deletions of positions as the noising process by giving *each position* an *independent* deletion rate σ_t. In particular, the corresponding transition probability for $s \leq t$ is

$$p^{\text{ind}}_{t|s}\big((m, \dot{\boldsymbol{y}}) \mid (n, \dot{\boldsymbol{x}})\big) = \rho^m_{s,t}(1-\rho_{s,t})^{n-m} \sum_{\boldsymbol{b} \in \mathbb{B}_{n,m}} \delta_{\dot{\boldsymbol{x}}[\boldsymbol{b}]}(\dot{\boldsymbol{y}}),$$

where $\rho_{s,t} = \exp(-\bar{\sigma}_{s,t})$. We can sample from this noising process by sampling the number of positions to drop using the Binomial$(n, \rho_{s,t})$ distribution instead of Poisson$(\bar{\sigma}_{s,t})$. The same procedure can be used to sample from the noising process as described earlier, but with the Poisson distribution replaced by the Binomial distribution. Appendix C.3 provides additional details and the analogue of Proposition 2 for the independent noising process.

Algorithm 1 Sampling from the Noising Process

Require: Sequence $(n, \dot{\boldsymbol{x}})$, rate function σ, NoisingType $\in$ {Joint, Independent}
1: $t \sim$ Uniform$[0, T]$
2: **if** NoisingType = Joint **then**
3: Sample $d \sim$ Poisson$(\bar{\sigma}_{0,t})$ ▷ # positions to drop
4: **else if** NoisingType = Independent **then**
5: Sample $d \sim$ Binomial$(n, 1 - \exp(-\bar{\sigma}_{0,t}))$
6: **end if**
7: $d \leftarrow \min(d, n)$
8: Permute the indices $(1, \ldots, n)$ of $\dot{\boldsymbol{x}}$ to get π.
9: Pick the first $n - d$ positions from π to get $\tilde{\pi} \leftarrow \pi([1 : n - d])$
10: Prepare mask $\boldsymbol{b} \in \mathbb{B}_{n,n-d}$ such that $\boldsymbol{b}[i] = 1$ if $i \in \tilde{\pi}$ and 0 otherwise.
11: $\dot{\boldsymbol{y}} \leftarrow \dot{\boldsymbol{x}}[\boldsymbol{b}]$ ▷ Remove the positions to drop
12: **return**$(\,(n - d, \dot{\boldsymbol{y}}),\, t\,)$

3.2 Generative Process

In our case, the time reversal of the noising CTMC is the generative process that constructs a sequence by inserting tokens. Let $\mathbb{X}_n$ denote $\{n\} \times \mathbb{V}^n$, ins : $\mathbb{X}_n \times [n+1] \times \mathbb{V} \to \mathbb{X}_{n+1}$ denote the insertion operation such that $\text{ins}(\boldsymbol{x}, i, a) = (n+1,\, x^1, \ldots x^{i-1}\, a\, x^i \ldots x^n)$, and $\text{Range}_{\text{ins}}(\boldsymbol{x}) = \{\text{ins}(\boldsymbol{x}, i, a) \mid i \in [n+1], a \in \mathbb{V}\}$.

We parameterize the reverse rate matrix directly as $\hat{R}^\theta_t$. Next, we discuss different options for parameterizing the rate $\hat{R}^\theta_t$ of the generative process. We also remark on how these parameterizations relate to some existing formulations of insertion language models.

3.2.1 Joint Insertion

For the joint noising process, we parameterize the rate using a joint probability distribution over positions and tokens as follows. Let $\hat{p}^\theta_t : \mathbb{X} \times [0, 1] \to \Delta^{n \times |\mathbb{V}|}$ be the parameterized joint probability distribution $(\boldsymbol{x}, t) \mapsto \hat{p}^\theta_t(i, w|\boldsymbol{x})$ that takes as input sequence $\boldsymbol{x} = (n, \dot{\boldsymbol{x}})$, time t, and an insertion location $i \in [n]$ and produces a joint over the insertion location $i \in [n]$ and vocab item w. Then the generative transition kernel, given that a transition out of $\boldsymbol{x}$ occurs is $\hat{K}^\theta_t(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{m+1} \sum_{w \in \mathbb{V}} \hat{p}^\theta_t(i, w \mid \boldsymbol{x})\, \delta_{\text{ins}(\boldsymbol{x},i,w)}(\boldsymbol{y})$, and the fully specified rate of the generative process is

$$\hat{R}^\theta_t(\boldsymbol{x}, \boldsymbol{y}) = \hat{\lambda}^\theta_t(\boldsymbol{x}) \sum_{i=1}^{m+1} \sum_{w \in \mathbb{V}} \hat{p}^\theta_t(i, w \mid \boldsymbol{x})\, \delta_{\text{ins}(\boldsymbol{x},i,w)}(\boldsymbol{y}),$$

where $\hat{\lambda}^\theta_t$ is the parameterized jump rate. Plugging this parameterization into the bound in Proposition 1 results in a tractable estimate of the ELBO as shown in the following corollary.

Corollary 1 (ELBO: Joint Insertion)**.** *A (biased) estimate of the upper bound on the negative log-likelihood in Proposition 1 for the joint parameterization of $\hat{R}^\theta_t$ is given by*

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{len}}(\theta) + \mathcal{L}_{\text{token}}(\theta), \textit{ where}$$

$$\mathcal{L}^{\text{len}}(\theta) = \mathbb{E}\left[\hat{\lambda}^\theta_t(\boldsymbol{x}_0[\boldsymbol{b}]) - \gamma^{\text{joint}}_t \log \hat{\lambda}^\theta_t(\boldsymbol{x}_0[\boldsymbol{b}])\right], \textit{ and}$$

$$\mathcal{L}^{\text{token}}(\theta) = -\,\mathbb{E}\left[\gamma^{\text{joint}}_t \sum_{i,w} q(i, w) \log \hat{p}^\theta_t(i, w \mid \boldsymbol{x}_0[\boldsymbol{b}])\right].$$

Here, the expectation is over $\boldsymbol{x}_0 \sim p_{\text{data}}$, $t \sim$ Uniform$[0, T]$, $(n-m) \sim$ Poisson$(\bar{\sigma}_{0,t})$, $\delta_{\{\cdot \geq 0\}}(m)$, $\boldsymbol{b} \sim$ Uniform$(\mathbb{B}_{n,m})$, the inner sum in $\mathcal{L}^{\text{token}}(\theta)$ is over $i \in [m+1], w \in \mathbb{V}$, and $\gamma^{\text{joint}}_t = \frac{\sigma_t\,(n-m)}{\bar{\sigma}_{0,t}\,\tilde{S}_{n,m,\sigma_t}}$ with $\tilde{S}_{n,m,\sigma_t} = [(1 - \delta_0(m)) + \delta_0(m) S_{n,\sigma_t}]$, $S_{n,\sigma_t} = \sum_{k=0}^{\infty} \frac{n!\,\bar{\sigma}^k_{0,t}}{(n+k)!}$, $\boldsymbol{x}_0 = (n, \dot{\boldsymbol{x}}_0)$, and $q(i, w) = \frac{\sum_{k \in s_i(\boldsymbol{b})} \delta_w(x^k_0)}{n-m}$ is the normalized count of the vocabulary item w occurring in $\boldsymbol{x}_0$ between the $(i-1)$-th and i-th 1s, with $s_i(\boldsymbol{b})$ being the set of indices of 0s in $\boldsymbol{b}$ that fall between the $(i-1)$-th and i-th 1s.

Remark 2. This estimate is biased because we replace $\delta_{\text{ins}(\boldsymbol{x},i,w)}(\boldsymbol{y}) \log K^\theta_t(\boldsymbol{x}, \boldsymbol{y})$ by $\log \hat{p}_t(i, w|\boldsymbol{x})$, where we have removed the summation present inside the log. This approximation is only active in the case where two insertions lead to the exact same sequence, which only happens in the case of contiguous repeated tokens. For example, if $\boldsymbol{x}$ = aaa, then $\text{ins}(\boldsymbol{x}, 1, \text{a}) = \text{ins}(\boldsymbol{x}, 2, \text{a}) = \text{aaaa}$.

Remark 3. Here we have obtained the estimator by fixing the latent alignment between the sequence $\boldsymbol{x}_0$ and the subsequence $\boldsymbol{x}_0[\boldsymbol{b}]$ to be equal to the one given by $\boldsymbol{b}$ itself, but we could marginalize over all $\boldsymbol{a} \in \mathbb{B}_{n,m}(\boldsymbol{x}_0, \boldsymbol{b})$ explicitly. This can be viewed as Rao-Blackwellizing the latent alignment variable $\boldsymbol{a}$, yielding a lower-variance estimator of the ELBO term.

However, this requires solving one dynamic programming instance per $(\boldsymbol{x}_0, \boldsymbol{b})$ pair to compute all possible alignments of the subsequence $\boldsymbol{x}_0[\boldsymbol{b}]$ and $\boldsymbol{x}_0$, which can introduce significant computational overhead during training. We leave an efficient implementation of this approach to future work.

A detailed discussion and the complete derivation of the corollary is provided in Appendix C.1. A method to efficiently compute S_{n,σ_t} on the GPU using `hyp1f1` is provided in Appendix E.2.

Neural Network.. The joint probability distribution over positions and tokens is parameterized using a standard transformer with time conditioning using adaptive layer-norm (Peebles and Xie, 2023). For representing each available gap, we prepend a special `BOS` (beginning of sequence) token to the sequence, so that $\tilde{\boldsymbol{x}}_t = (\texttt{BOS}, x_t^1, \dots, x_t^n)$ has length $n+1$ and each insertion location can be indexed by the token immediately preceding it. Let f^θ denote the transformer with bidirectional attention (without the final linear projection), which takes $\tilde{\boldsymbol{x}}_t$ and time $t \in [0,1]$ as input and returns $f^\theta(\tilde{\boldsymbol{x}}_t; t) \in \mathbb{R}^{(n+1)\times d}$. For each insertion location $i \in [n+1]$ the corresponding output of the transformer backbone $f^\theta(\tilde{\boldsymbol{x}}_t; t)^i \in \mathbb{R}^d$ is passed through the final linear layer $U^\theta : \mathbb{R}^d \to \mathbb{R}^{|\mathbb{V}|}$ to get a score for the pair (i, w).

$$s^\theta(i, w \mid \boldsymbol{x}_t) = U^\theta \left(f^\theta(\tilde{\boldsymbol{x}}_t; t)^i\right)^w$$

$$\hat{p}_t^\theta(i, w \mid \boldsymbol{x}_t) = \frac{\exp(s^\theta(i, w \mid \boldsymbol{x}_t))}{\sum_{i'=1}^{n+1} \sum_{w' \in \mathbb{V}} \exp(s^\theta(i', w' \mid \boldsymbol{x}_t))}.$$

The superscripts i, w above indicate position and token index, respectively. Parameterizing the insertion rate $\hat{\lambda}_t^\theta$ as a non-negative scalar is challenging (Campbell et al., 2023; Patel et al., 2025). However, we can use the length posterior to parameterize the insertion rate as shown in Proposition 3 in Appendix C. Specifically, denoting $p_{t,\text{len}}(l \mid \boldsymbol{x}) \coloneqq \mathbb{P}(N_0 - N_t = l \mid \boldsymbol{X}_t = \boldsymbol{x})$, with $\boldsymbol{x} = (m, \dot{\boldsymbol{x}})$, the corresponding insertion rate is given by

$$\hat{\lambda}_t^\theta(\boldsymbol{x}) = \begin{cases} \frac{\sigma_t}{\bar{\sigma}_{0,t}} \mathop{\mathbb{E}}_{l \sim \hat{p}^\theta_{t,\,\text{len}}(\cdot \mid \boldsymbol{x})} l & \text{if } m > 0 \\ \frac{\sigma_t}{\bar{\sigma}_{0,t}} \mathop{\mathbb{E}}_{l \sim \hat{p}^\theta_{t,\,\text{len}}(\cdot \mid \boldsymbol{x})} \frac{l}{S_{l,\sigma_t}} & \text{if } m = 0. \end{cases}$$

We parameterize $\hat{p}^\theta_{t,\text{len}}(l \mid \boldsymbol{x}_t)$ using a categorical distribution on $[n_{\max}]$ and a dedicated input position in the transformer backbone. In the implementation, the network outputs logits for the full distribution $\hat{p}^\theta_{t,\text{len}}(l \mid \boldsymbol{x}_t)$ and we use its mean

$$\hat{\Delta} l_t^\theta(\boldsymbol{x}_t) = \mathop{\mathbb{E}}_{l \sim \hat{p}^\theta_{t,\text{len}}(\cdot \mid \boldsymbol{x}_t)} [l].$$

This allows us to optimize the length term using the simpler objective of matching the observed length-to-go $n-m$ with the predicted mean $\hat{\Delta} l_t^\theta(\boldsymbol{x}_0[\boldsymbol{b}])$, namely

$$\mathcal{L}_{\text{mean}}^{\text{len}}(\theta) = \mathbb{E}\left[\hat{\Delta} l_t^\theta(\boldsymbol{x}_0[\boldsymbol{b}]) - (n-m) \log \hat{\Delta} l_t^\theta(\boldsymbol{x}_0[\boldsymbol{b}])\right],$$

which differs from the corresponding Poisson negative log-likelihood only by an additive constant independent of θ.

Sampling.. Given the rate $\hat{R}_t^\theta$, the generation is simulated using a simple first-order (Euler) discretization (eq. (2)) as described in Algorithm 2. We call the combination of joint noising and the Euler sampling **D**iffusion-based **I**nsertion **L**anguage Model with **S**ingle Insertions (DILM-S).

Stopping condition. The Insertion Language Model (ILM) of Patel et al. (2025), which uses a stopping classifier to determine when to stop generation, can be viewed as a time-agnostic special case where the Bernoulli probability $\hat{\lambda}_t^\theta(\boldsymbol{x})\Delta t$ in Algorithm 2 is replaced with a stopping probability $p^\theta_{\text{stop}}(\cdot \mid \boldsymbol{x})$. Note, however, that ILM does not use the time parameter. Moreover, it uses a hyper-parameter ζ to determine when to stop generation using the stopping condition $p^\theta_{\text{stop}}(\text{stop} \mid \boldsymbol{x}) < \zeta$. This means that there is no way to trade-off the quality of the generated sequence by reducing Δt, or equivalently, increasing the number of sampling steps. Our method, like most other diffusion-based generative models, allows for this trade-off.

3.2.2 Independent Insertion

For the case of independent noising, we parameterize the rate of the generative process also using independent, per-position rates as

$$\hat{R}_t^\theta(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{m+1} \sum_{w \in \mathbb{V}} \lambda_t^\theta(\boldsymbol{x}, i)\, \hat{p}_t^\theta(w \mid \boldsymbol{x}, i)\; \delta_{\text{ins}(\boldsymbol{x},i,w)}(\boldsymbol{y}).$$

In this case, with a slight abuse of notation, we use the same symbol $\hat{p}_t^\theta$ as before but now $\hat{p}_t^\theta : \mathbb{X} \times [0,1] \times [n] \to \mathbb{V}$ denotes the categorical distribution over vocabulary item w for gap i. Applying the same approximations as done for Corollary 1, we get the following estimate of the upper bound on the negative log-likelihood, which is biased for the same reasons as in Corollary 1.

Corollary 2 (ELBO: Independent Insertion)**.** *A (biased) estimate of the upper bound on the negative log-likelihood in Proposition 1 for the independent per-position parameterization of $\hat{R}_t^\theta$ is given by $\mathcal{L}(\theta) = \mathcal{L}_{\text{len}}(\theta) + \mathcal{L}_{\text{token}}(\theta)$,where the two loss terms are*

$$\mathbb{E}\left[\sum_{i=1}^{m+1} \hat{\lambda}_t^\theta(\boldsymbol{x}_0[\boldsymbol{b}], i) - \gamma_t^{ind} |s_i(\boldsymbol{b})| \log \hat{\lambda}_t^\theta(\boldsymbol{x}_0[\boldsymbol{b}], i)\right],$$

$$-\mathbb{E}\left[\gamma_t^{ind} \sum_{i=1}^{m+1} \sum_{k \in s_i(\boldsymbol{b})} \log \hat{p}_t^\theta(x_0^k \mid \boldsymbol{x}_0[\boldsymbol{b}], i)\right].$$

Here, the expectation is over $\boldsymbol{x}_0 \sim p_{\text{data}}$, $t \sim \text{Uniform}[0, T]$, $m \sim \text{Binomial}(n, \rho_t)$, $\boldsymbol{b} \sim \text{Uniform}(\mathbb{B}_{n,m})$, $\gamma_t^{ind} = \frac{\sigma_t \rho_t}{1-\rho_t}$ *with* $\rho_t = \exp(-\bar{\sigma}_{0,t})$, $\boldsymbol{x}_0 = (n, \dot{\boldsymbol{x}}_0)$, *and* $s_i(\boldsymbol{b})$ *is the set of indices of 0s in* $\boldsymbol{b}$ *that fall between the* $(i-1)$*-th and* i*-th 1s.*

As in the joint case, in the implementation we parameterize a distribution over the number of tokens inserted into each gap, $\hat{p}^\theta_{t,\text{len}}(l \mid \boldsymbol{x}_t, i)$, with $\hat{\Delta} l^\theta_t(\boldsymbol{x}_t, i) \coloneqq \mathbb{E}_{l \sim \hat{p}^\theta_{t,\text{len}}(\cdot \mid \boldsymbol{x}_t, i)}[l]$. The corresponding reverse-process insertion rate is then $\hat{\lambda}^\theta_t(\boldsymbol{x}_t, i) \coloneqq \gamma_t^{\text{ind}} \hat{\Delta} l^\theta_t(\boldsymbol{x}_t, i)$. Consequently, the length term can be optimized by matching the observed gap size $|s_i(\boldsymbol{b})|$ with the predicted mean using $\mathcal{L}^{\text{len}}_{\text{mean}}(\theta)$ given by

$$\mathbb{E}\left[\gamma_t^{\text{ind}} \sum_{i=1}^{m+1} \left(\hat{\Delta} l^\theta_t(\boldsymbol{x}_0[\boldsymbol{b}], i) - |s_i(\boldsymbol{b})| \log \hat{\Delta} l^\theta_t(\boldsymbol{x}_0[\boldsymbol{b}], i)\right)\right],$$

which recovers the same objective up to additive constants independent of θ.

Neural Network.. We use the same transformer architecture as in the case of Joint Insertion Parameterization, but now the token head is interpreted independently at each gap:

$$s^\theta(w \mid \boldsymbol{x}_t, i) = U^\theta \left(f^\theta(\tilde{\boldsymbol{x}}_t; t)^i\right)^w$$
$$\hat{p}^\theta_t(w \mid \boldsymbol{x}_t, i) = \frac{\exp(s^\theta(w \mid \boldsymbol{x}_t, i))}{\sum_{w' \in \mathbb{V}} \exp(s^\theta(w' \mid \boldsymbol{x}_t, i))}.$$

That is, unlike the joint parameterization, the softmax is taken only over vocabulary items independently for each gap. To parameterize the distribution over the length of each gap, we use an additional head that outputs $\hat{p}^\theta_{t,\text{len}}(l \mid \boldsymbol{x}_t, i)$.

Sampling.. Algorithm 3 shows an efficient constrained τ-leaping (Gillespie, 2001) based sampling procedure for the independent parameterization, which can insert multiple tokens per step across different gaps. Additionally, this algorithm can be applied on a mini-batch of sequences in parallel.[2] We call the parameterization DILM-M, where M stands for multiple insertions across gaps.

Algorithm 2 Sampling from DILM-S

Require: Sequence $\boldsymbol{x} = (m, (\boldsymbol{v}, \boldsymbol{u}))$, steps MaxSteps
$\Delta t \leftarrow \frac{1}{\text{MaxSteps}}$
while $t < 1$ **do**
  $e \leftarrow \text{Uniform}[0, 1]$
  $\hat{\lambda}^\theta_t(\boldsymbol{x}) \leftarrow \frac{\sigma_t}{\bar{\sigma}_{0,t}} \mathbb{E}_{l \sim p^\theta_{t,\text{len}}(\boldsymbol{x})} \frac{l}{\bar{S}_{l,m,\sigma_t}}$
  if $e \leq \hat{\lambda}^\theta_t(\boldsymbol{x})\, \Delta t$ **then**
    $i, w \sim \hat{p}^\theta_t(i, w \mid \boldsymbol{x})$
    $\boldsymbol{v}' \leftarrow \texttt{concat}(\boldsymbol{v}, w)$
    for $k = 1$ to $\texttt{len}(\boldsymbol{u})$ **do**
      if $\boldsymbol{u}[k] > i$ **then**
        $\boldsymbol{u}[k] \leftarrow \boldsymbol{u}[k] + 1$
      end if
    end for
    $\boldsymbol{u}' \leftarrow \texttt{concat}(\boldsymbol{u}, i+1)$
    $\boldsymbol{x} \leftarrow (\boldsymbol{v}', \boldsymbol{u}')$
  end if
  $t \leftarrow t + \Delta t$
end while
return $\boldsymbol{x}$

4 RELATED WORK

Discrete diffusion and flow matching.. Discrete diffusion models (Sahoo et al., 2024; Shi et al., 2024; Austin et al., 2021; Gong et al., 2024), inspired from diffusion models (Ho et al., 2020; Sohl-Dickstein et al., 2015; Song and Ermon, 2019), generate sequences by iteratively *denoising* the tokens of a noisy sequence by framing both the noising and the denoising processes as Markov chains on countable state spaces. The discrete diffusion models can only work with fixed length sequences. They generate variable length sequences by modeling special `PAD` token as an ordinary token. Our formulation is motivated by the discrete diffusion framework, but with some key distinctions. We do not assume independent noise per-component as done in Campbell et al. (2022) and subsequent works, instead our noising process acts directly on the sequence.

Insertion-based sequence generation.. There have been several works in the machine translation literature that explore insertion-style generation in sequence-to-sequence models (Stern et al., 2019; Gu et al., 2019; Ruis et al., 2020; Patel et al., 2025). Welleck et al. (2019) uses reinforcement learning to learn an insertion policy using the *learning to search* approach (Ross et al., 2011). The insertions in Welleck et al. (2019) are constrained to be level order traversals of a tree and cannot be arbitrary. Moreover, reinforcement learning can be orders of magnitude slower than our approach and is therefore not suitable for pretraining language models. The insertion transformer (Stern et al., 2019) uses an efficient denoising objective to train the insertion model. It can be viewed as implementing independent insertions rate matrix (section 3.2.2). However, it does not parameterize the insertion rate separately from the token probabilities. In general, deciding *when to stop the generation* is challenging for insertion-style models (Stern et al., 2019; Patel et al., 2025; Reid et al., 2022), and formalizing the process a continuous time Markov chain allows us parameterize the model in a way that separates token predictions from the stopping probability.

[2] We provide a simplified implementation of Algorithm 3 in PyTorch in Appendix E.4.1.

Algorithm 3 Sampling from DILM-M

Require: Sequence $\boldsymbol{x} = (x_0, \ldots, x_{m-1})$, time $t \in [0,1]$, steps MaxSteps
$\Delta t \leftarrow \frac{1}{\text{MaxSteps}}$
while $t < 1$ **do**
  Initialize $\mathbf{r} \in \{0,1\}^m$ and $\mathbf{w} \in \mathcal{V}^m$
  for $i \in \{0, 1, \ldots, m-1\}$ **do**
    $e \sim \text{Uniform}[0,1]$
    if $e \leq 1 - \exp(-\hat{\lambda}_t^\theta(\boldsymbol{x}, i)\,\Delta t)$ **then**
      $\mathbf{r}[i] \leftarrow 1$
      $\mathbf{w}[i] \sim \hat{p}_t^\theta(w \mid \boldsymbol{x}, i)$
    else
      $\mathbf{r}[i] \leftarrow 0$
    end if
  end for
  $\boldsymbol{x}' \leftarrow [\texttt{PAD}, \ldots, \texttt{PAD}]$
  $\mathbf{s} \leftarrow \text{RollRight}(\mathbf{r}, 1)$
  $\texttt{idx\_old} \leftarrow \text{Arange}(m) + \text{CumSum}(\mathbf{s})$
  $\texttt{idx\_ins} \leftarrow \text{Arange}(m) + \text{CumSum}(\mathbf{r})$
  $\boldsymbol{x}' \leftarrow \text{Scatter}(\boldsymbol{x}', \texttt{idx\_old}, \boldsymbol{x})$
  $\boldsymbol{x}'[\texttt{idx\_ins}[\mathbf{r}]] \leftarrow \mathbf{w}[\mathbf{r}]$
  $t \leftarrow t + \Delta t$; $\boldsymbol{x} \leftarrow \boldsymbol{x}'$
end while
return $\boldsymbol{x}$

Concurrent work.. More recently, Discrete flow matching (Campbell et al., 2024; Lipman et al., 2024) and stochastic interpolants (Albergo et al., 2023) have been proposed as simplified frameworks for generative modeling using Markov chains, wherein there is no noising process but only a generative Markov chain that is constrained to generate sequence of time marginals that converge to the true data distribution. We provide an extended discussion of the relationship between our work and these frameworks in the appendix D.

5 EMPIRICAL EVALUATION

We split our experiments into two parts. First, we demonstrate that the proposed approach retains the benefits of insertion-style generation by evaluating it on the variable-length planning task on the star graphs (Bachmann and Nagarajan, 2024; Patel et al., 2025). We then evaluate the usefulness of the proposed approach on language modeling using two language modeling corpora with different characteristics. We compare our results with the standard left-to-right autoregressive language models (ARMs), masked diffusion models (Ou et al., 2024; Zheng et al., 2025) and the Insertion Language Model (ILM) (Patel et al., 2025).

Model For DILM-S, we use the same transformer backbone as in Patel et al. (2025), with one change. We introduce the time parameter t using adaptive layer normalization (AdaLN-Zero) (Peebles and Xie, 2023). For DILM-S, we also use the first token position to model $\hat{\lambda}_t^\theta(\boldsymbol{x})$. For DILM-M, we model the per-position insertion rate $\lambda_t^\theta(\boldsymbol{x}, i)$ using an additional feedforward layer on top of the transformer backbone.

Table 1: Sequence-level exact match accuracy on the synthetic Star graph tasks.

Model	$\text{Star}_{\text{easy}}$	$\text{Star}_{\text{medium}}$	$\text{Star}_{\text{hard}}$
ARM	32.3	75.0	23.0
MDM	100.0	36.5	21.0
ILM	100.0	100.0	99.1
DILM-S	100.0	**100.0**	**99.3**
DILM-M	100.0	98.60	95.1

5.1 Planning Task

The simple star graph task (Bachmann and Nagarajan, 2024) and its harder variable-length variant (Patel et al., 2025) are minimal tasks designed to exemplify limitations of ARMs and MDMs. [3] This is a sequence-to-sequence task where the source sequence contains randomly permuted edges of a star-shaped graph, followed by the start and the target node. The model needs to generate the edges that connect the start and the target node going through the junction of the star. This task is difficult for ARMs trained to predict from left-to-right using teacher forcing (Bachmann and Nagarajan, 2024) because the model is overwhelmed by the training signal for the easy-to-predict non-junction edges and it never learns to perform the implicit lookahead required to generate the edge that immediately follows the junction. MDMs, which can model all the tokens simultaneously, can generate the correct sequence only when the arm length is fixed (Patel et al., 2025). As shown in Table 1, both DILM-S and DILM-M continue to enjoy the same benefits as ILM and achieve almost perfect accuracy on all three versions of the task.

5.2 Language Modeling

For all the language modeling experiments, we use a transformer backbone with 12 layers, 768 hidden dimensions, and 12 attention heads with ≈87 million non-embedding parameters and train it from scratch using the AdamW optimizer on 8 A100 GPUs.

LM1B. For language modeling on short sequences, we use the One Billion Word Benchmark (LM1B) (Chelba et al., 2013), which contains short sequences from the news domain. We preprocess that data by tokenizing it with the BERT tokenizer (Devlin et al., 2019) and pad to a maximum sequence length of 128. We use a

[3] We use the dataset from (Patel et al., 2025).

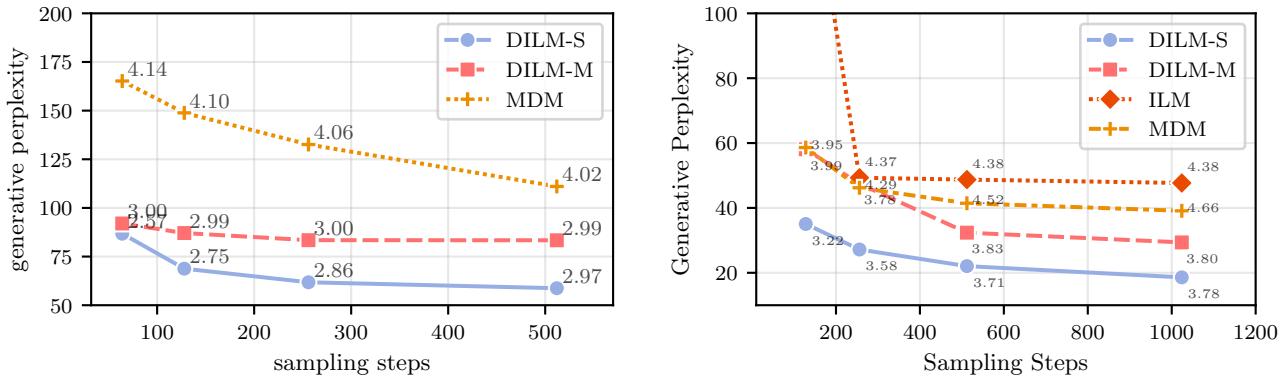


Figure 3: Generative perplexity vs. number of sampling steps for models trained on LM1B (left) and OpenWebText (right). The annotations show the token entropy values.

Table 2: Evaluation of unconditional generation quality using per-token NLL under Llama 3.2 3B. The rows with the dataset names contain the NLL and entropy of the examples in the training data.

	NLL↓	Ent↑	$\overline{\text{len}}$
LM1B	3.71	3.08	28
ARM	**3.94**	3.12	30
MDM (Sahoo et al., 2024)	4.81	3.70	85
ILM (Patel et al., 2025)	4.67	2.80	21
DILM-S (ours)	<u>4.65</u>	2.87	30
DILM-M (ours)	4.76	2.99	48
OpenWebText (padded)	2.59	5.01	533
MDM	3.96	4.64	320
ILM	4.65	5.46	860
DILM-S	**3.74**	4.33	267
DILM-M	<u>3.92</u>	4.54	478

learning rate of 10^{-4}, effective batch size of 512, and train for 1 million steps.

To evaluate unconditional generation we compute per-token negative log-likelihood (NLL) under Llama-3.2-3B (Grattafiori et al., 2024) for 1000 unconditionally generated sequences. As seen in Table 2, both DILM-S and DILM-M are competitive with the ILM (Patel et al., 2025), with DILM-S doing slightly better. The key advantage of the diffusion-based formulation is the ability to trade-off sample quality and number of forward passes. As shown in Figure 3, the generative perplexity of both DILM-S and DILM-M improves as we increase the number of sampling steps.

OpenWebText.. For language modeling on longer sequences, we use the OpenWebText (Gokaslan and Cohen, 2019) corpus. We preprocess the OpenWebText corpus by tokenizing it using the GPT-2 tokenizer, removing sequences that are longer than 1024 tokens, and padding to a maximum sequence length of 1024. We use a learning rate of 10^{-4}, effective batch size of 512, and train for 300 thousand steps. For evaluation, we compute the per-token negative log-likelihood under the GPT-2 Large model for 1000 unconditionally generated sequences from each model; Table 2 reports these values. We can see that DILM-S and DILM-M obtain better NLL than ILM and MDM. Figure 3 (right) shows the generative perplexity *vs.* number of sampling steps on OpenWebText. Here we observe again that the diffusion-based formulation allows for better trade-off between sample quality and number of forward passes. The generative perplexity for ILM is very high at low sampling steps (128); it falls rapidly at medium sampling steps (256), and then remains flat even as the sampling steps are increased. DILM-S and DILM-M both exhibit a smoother trade-off curve between the generative perplexity and the number of sampling steps. However, on longer sequences, we observe token repetitions, which show up in the lower entropy values.

6 CONCLUSION

We presented a diffusion-based approach for variable-length sequence generation, leading to two new parameterizations of the generative process. With clearly stated assumptions, we derived a low-variance learning objective for both parameterizations, and demonstrated their efficacy empirically on planning and language modeling tasks.

Limitations and future work.. While DILM-M can generate multiple tokens, it is still constrained to produce only one token per gap. This can hinder the ability of the neural network in leveraging correlations between tokens that occur in contiguous spans. A multi-token insertion model that can insert contiguous spans of tokens is a promising direction for further exploration. Scaling insertion-based generation to larger models and longer sequences also remains an interesting direction for future work.

Acknowledgments

We thank Michael Boratko and Tomas Geffner for helpful discussions and encouragement. This work was supported by IBM under IBM Research Collaboration Agreement No W1668553, and by National Science Foundation under the grant IIS-2106391. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IBM or NSF.

References

Albergo, M. S., Boffi, N. M., and Vanden-Eijnden, E. (2023). Stochastic interpolants: A unifying framework for flows and diffusions.

Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and van den Berg, R. (2021). Structured Denoising Diffusion Models in Discrete State-Spaces. In *Advances in Neural Information Processing Systems*.

Bachmann, G. and Nagarajan, V. (2024). The pitfalls of next-token prediction. In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F., editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 2296–2318. PMLR.

Campbell, A., Benton, J., Bortoli, V. D., Rainforth, T., Deligiannidis, G., and Doucet, A. (2022). A continuous time framework for discrete denoising models. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.

Campbell, A., Harvey, W., Weilbach, C., De Bortoli, V., Rainforth, T., and Doucet, A. (2023). Trans-Dimensional Generative Modeling via Jump Diffusion Models. *Advances in Neural Information Processing Systems*, 36:42217–42257.

Campbell, A., Yim, J., Barzilay, R., Rainforth, T., and Jaakkola, T. (2024). Generative Flows on Discrete State-Spaces: Enabling Multimodal Flows with Applications to Protein Co-Design. In *Proceedings of the 41st International Conference on Machine Learning*, pages 5453–5512. PMLR.

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., and Robinson, T. (2013). One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Cinlar, E. (2013). *Introduction to Stochastic Processes*. Dover Publications, Incorporated, Newburyport, 1st ed edition.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

Gillespie, D. T. (2001). Approximate accelerated stochastic simulation of chemically reacting systems. *The Journal of Chemical Physics*, 115(4):1716–1733.

Gokaslan, A. and Cohen, V. (2019). Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus.

Gong, S., Agarwal, S., Zhang, Y., Ye, J., Zheng, L., Li, M., An, C., Zhao, P., Bi, W., Han, J., Peng, H., and Kong, L. (2024). Scaling Diffusion Language Models via Adaptation from Autoregressive Models. In *The Thirteenth International Conference on Learning Representations*.

Grattafiori, A., Dubey, A., Jauhri, A., et al. (2024). The llama 3 herd of models.

Gu, J., Wang, C., and Zhao, J. (2019). Levenshtein Transformer. In Wallach, H., Larochelle, H., Beygelzimer, A., Alché-Buc, F. d., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Hanson, F. B. (2007). *Applied stochastic processes and control for jump-diffusions: modeling, analysis and computation*. SIAM.

Havasi, M., Karrer, B., Gat, I., and Chen, R. T. Q. (2025). Edit flows: Flow matching with edit operations.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc.

Kim, J., Cheuk-Kit, L., Domingo-Enrich, C., Du, Y., Kakade, S., Ngotiaoco, T., Chen, S., and Albergo, M. (2025). Any-order flexible length masked diffusion.

Li, J., Dong, X., Zang, Y., Cao, Y., Wang, J., and Lin, D. (2025). Beyond fixed: Training-free variable-length denoising for diffusion large language models.

Lipman, Y., Havasi, M., Holderrieth, P., Shaul, N., Le, M., Karrer, B., Chen, R. T. Q., Lopez-Paz, D., Ben-Hamu, H., and Gat, I. (2024). Flow matching guide and code.

Lou, A., Meng, C., and Ermon, S. (2024). Discrete diffusion modeling by estimating the ratios of the data distribution. In *Forty-first International Conference on Machine Learning*.

Nie, S., Zhu, F., Du, C., Pang, T., Liu, Q., Zeng, G., Lin, M., and Li, C. (2025). Scaling up masked diffusion models on text. In *The Thirteenth International Conference on Learning Representations*.

Ou, J., Nie, S., Xue, K., Zhu, F., Sun, J., Li, Z., and Li, C. (2024). Your Absorbing Discrete Diffusion Secretly Models the Conditional Distributions of Clean Data. In *The Thirteenth International Conference on Learning Representations*.

Patel, D., Maram, D. P., Chintha, S. S., Rozonoyer, B., and McCallum, A. (2026). xLM: A python package for non-autoregressive language models. In Croce, D., Leidner, J., and Moosavi, N. S., editors, *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 445–456, Rabat, Marocco. Association for Computational Linguistics.

Patel, D., Sahoo, A., Amballa, A., Naseem, T., Rudner, T. G. J., and McCallum, A. (2025). Insertion language models: Sequence generation with arbitrary-position insertions.

Peebles, W. and Xie, S. (2023). Scalable Diffusion Models with Transformers. pages 4195–4205.

Reid, M., Hellendoorn, V. J., and Neubig, G. (2022). Diffuser: Discrete diffusion via edit-based reconstruction.

Ross, S., Gordon, G., and Bagnell, D. (2011). A reduction of imitation learning and structured prediction to no-regret online learning. In Gordon, G., Dunson, D., and Dudík, M., editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635, Fort Lauderdale, FL, USA. PMLR.

Ruis, L., Stern, M., Proskurnia, J., and Chan, W. (2020). Insertion-deletion transformer.

Sahoo, S. S., Arriola, M., Gokaslan, A., Marroquin, E. M., Rush, A. M., Schiff, Y., Chiu, J. T., and Kuleshov, V. (2024). Simple and Effective Masked Diffusion Language Models.

Shi, J., Han, K., Wang, Z., Doucet, A., and Titsias, M. (2024). Simplified and generalized masked diffusion for discrete data. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. (2015). Deep Unsupervised Learning using Nonequilibrium Thermodynamics.

Song, Y. and Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. In Wallach, H., Larochelle, H., Beygelzimer, A., dAlché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Stern, M., Chan, W., Kiros, J., and Uszkoreit, J. (2019). Insertion Transformer: Flexible Sequence Generation via Insertion Operations. arXiv:1902.03249 [cs].

Welleck, S., Brantley, K., Iii, H. D., and Cho, K. (2019). Non-monotonic sequential text generation. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6716–6726. PMLR.

Wu, Z., Zheng, L., Xie, Z., Ye, J., Gao, J., Gong, S., Feng, Y., Li, Z., Bi, W., Zhou, G., and Kong, L. (2026). Dreamon: Diffusion language models for code infilling beyond fixed-size canvas.

Yang, J., Jiang, Y., and Shao, J. (2026). ρ-EOS: Training-free bidirectional variable-length control for masked diffusion llms.

Zheng, K., Chen, Y., Mao, H., Liu, M.-Y., Zhu, J., and Zhang, Q. (2025). Masked diffusion models are secretly time-agnostic masked models and exploit inaccurate categorical sampling. In *The Thirteenth International Conference on Learning Representations*.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Not Applicable]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [No]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes, we provide proofs for the new results and provide references for known results.]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes, the code used for this work is available at https://github.com/

dhruvdcoder/ctmc_dilm. Appendix E contains some key implementation details, which are also available in the code.]

(b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes, these are available in the code and in Appendix E.]

(c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Not Applicable]

(d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes, the details are available in Appendix E.]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:

(a) Citations of the creator If your work uses existing assets. [Yes]

(b) The license information of the assets, if applicable. [Not Applicable]

(c) New assets either in the supplemental material or as a URL, if applicable. [Yes, we provide the URL to the code repository, which also contains URLs for the model checkpoints.]

(d) Information about consent from data providers/curators. [Not Applicable]

(e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

(a) The full text of instructions given to participants and screenshots. [Not Applicable]

(b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Appendix for Continuous-Time Markov Chain Framework for Insertion Language Models

Contents

A Summary of Notation

Notation	Description
	General
$(\Omega, \mathcal{O}, P)$	Probability space from which all random variables are drawn.
ω	Sample point in Ω
$\boldsymbol{X}_t$, $\boldsymbol{B}_t$, D_t	Random variables (capital letters)
$\boldsymbol{x}$, $\boldsymbol{b}$, d	Values of random variables (lowercase)
$\boldsymbol{X}$, $\boldsymbol{x}$	Non-scalar random variables and values (boldface)
$\mathbb{X}$, $\mathbb{B}$	Sets (blackboard font)
$[n]$	Set of natural numbers $\{1, 2, \ldots, n\}$
Δ^n	Simplex of dimension n, i.e., $\{p \in \mathbb{R}^n \mid p \geq 0, \sum_{i=1}^n p_i = 1\}$
x^i, X^i	i-th component of $\boldsymbol{x}$ and $\boldsymbol{X}$ respectively
$\boldsymbol{X}_t$, $\boldsymbol{B}_t$, D_t	Continuous time stochastic processes
$\underline{\boldsymbol{X}}_k$, $\underline{\boldsymbol{B}}_k$	Discrete time stochastic processes
$p_{s\|t}(\boldsymbol{x} \mid \boldsymbol{x}')$ or $p_{t,s}(\boldsymbol{x}', \boldsymbol{x})$	Transition probability $\mathbb{P}(\boldsymbol{X}_s = \boldsymbol{x} \mid \boldsymbol{X}_t = \boldsymbol{x}')$
$\underline{p}_{k\|k'}(\boldsymbol{x} \mid \boldsymbol{x}')$	Transition probability $\mathbb{P}(\underline{\boldsymbol{X}}_k = \boldsymbol{x} \mid \underline{\boldsymbol{X}}_{k'} = \boldsymbol{x}')$
$\delta_{\mathbb{A}}(a)$	Indicator function for set $\mathbb{A}$: 1 if $a \in \mathbb{A}$, 0 otherwise
$\delta_b(a)$	Shorthand for $\delta_{\{b\}}(a)$
$\mathrm{Diag}(\lambda)$	Diagonal matrix with diagonal elements given by λ
	Specific variables
$\mathbb{V}$	Vocabulary of tokens
$\mathbb{V}^n \coloneqq \mathbb{V} \times \cdots \times \mathbb{V}$	Set of token sequences of length n. $\mathbb{V}^0 \coloneqq \emptyset$ by convention.
$\mathbb{X}_n$	$\{n\} \times \mathbb{V}^n$
$\mathbb{B}_{n,m}$	Set of all bit vectors of length n with exactly m 1s.
$\mathbb{B}_n$	$\bigcup_{m=0}^n \mathbb{B}_{n,m}$, i.e., set of all bit vectors of length n.
$\|\cdot\| : \mathbb{B}_n \to \mathbb{N}$	The number of 1s in a bit vector; $\|\boldsymbol{b}\| = \sum_{i=1}^n b^i$
$\mathrm{Idx}(\boldsymbol{b}) \coloneqq \{i \in [n] \mid b^i = 1\}$	The set of indices of the ones in $\boldsymbol{b}$
$\mathrm{del}(\boldsymbol{x}, i)$	$\mathrm{del}(x^1 \ldots x^i \ldots x^n, i) = x^1 \ldots x^{i-1} x^{i+1} \ldots x^n$
$\mathrm{ins}(\boldsymbol{x}, i, a)$	$\mathrm{ins}(x^1 \ldots x^i \ldots x^n, i, a) = x^1 \ldots x^{i-1} a\, x^{i+1} \ldots x^n$
$\mathrm{Range}_{\mathrm{ins}}(\boldsymbol{x})$	For $\boldsymbol{x} = (n, \dot{\boldsymbol{x}})$, $\mathrm{Range}_{\mathrm{ins}}(\boldsymbol{x}) = \{\mathrm{ins}(\boldsymbol{x}, i, a) \mid i \in [n+1], a \in \mathbb{V}\}$

Table 3: Summary of notation used throughout the paper.

B Background: Continuous-Time Markov Chains on Countable State Spaces

In this section, we will review some elementary results from the theory of CTMCs on countable state spaces that are used in the main paper text. Most of these results are well known and can be found in introductory textbooks on stochastic processes like Cinlar (2013).

B.1 Structure of CTMCs

Let X_t be a CTMC taking values in a countable state space $\mathbb{X}$.

Definition 1 (Waiting time)**.** The random variable $W_t(\omega) = \inf\{s > 0 : X_s(\omega) \neq X_t(\omega)\}$ is called the waiting time of the CTMC.

The following proposition states that the conditional distribution of the waiting time is exponential and only depends on the current state.

Fact 1 (Conditional Distribution of Waiting Time)**.** *For any $x \in \mathbb{X}$, and $t \geq 0$,*

$$\mathbb{P}(W_t > u \mid X_t = x) = \exp\left(-\int_t^{t+u} \lambda(x, s)\, ds\right), \quad u \geq 0,$$

where $\lambda(x,s) \in [0,\infty]$, with the convention that if $\lambda(x,s) = \infty$, then $\mathbb{P}(W_t > u \mid X_t = x) = 0$ for all $u \geq 0$.

Proof. Note that $\{W_t > u+v\} = \{W_t > u, W_{t+u} > v\}$ because $\omega \in \{W_t > u+v\}$ implies $X_t(\omega) = X_{t+(u+v)}(\omega) = X_{t+u}(\omega)$, which further implies that $\omega \in \{W_t > u\}$ and $\omega \in \{W_{t+u} > v\}$. Therefore,

$$\begin{aligned}
&\mathbb{P}(W_t > u+v \mid X_t = x) \\
&= \mathbb{P}(W_t > u,\ W_{t+u} > v \mid X_t = x) \\
&= \mathbb{P}(W_t > u \mid X_t = x)\mathbb{P}(W_{t+u} > v \mid X_t = x, W_t > u) \\
&= \mathbb{P}(W_t > u \mid X_t = x)\mathbb{P}(W_{t+u} > v \mid X_{t+u} = x) \quad \text{(Markov property)}
\end{aligned}$$

Denoting $\mathbb{P}(W_t > u \mid X_t = x)$ as $p(x,u;t)$, we have

$$p(x, u+v;\, t) = p(x,u;\, t)\ p(x,v;\, t+u). \tag{3}$$

Setting $u = 0$ we get $p(x,v;\,t) = p(x,0;\,t)\ p(x,v;\,t)$, which implies that $p(x,0;\,t) = 1$ since $p(x,v;\,t)$ is not identically zero. This, along with Equation 3, implies

$$\begin{aligned}
p(x,v;\,u) &= \frac{p(x,u+v;\,0)}{p(x,u;\,0)} \\
\implies \log p(x,v;\,u) &= \log p(x,u+v;\,0) - \log p(x,u;\,0) \\
\implies p(x,v;\,u) &= \exp\left(-\int_u^{u+v} \lambda(x,s)\,ds\right),
\end{aligned}$$

where $\lambda(x,s) \coloneqq -\frac{d}{ds}\log p(x,s;\,0)$, i.e., $\int_u^{u+v} \lambda(x,s)\,ds = \log p(x,u+v;\,0) - \log p(x,u;\,0)$. □

Next we will show that a CTMC with right-continuous sample paths can be decomposed into a product of Poisson process arrivals and DTMC transitions.

Definition 2 (Right-continuous CTMC)**.** A CTMC X_t is said to have right-continuous sample paths if for any $t \in [0,\infty)$,

$$\lim_{s \downarrow t} X_s = X_t.$$

Definition 3 (Types of States)**.** Let X_t be a CTMC with rate parameter $\lambda(x,s)$ as described in Proposition 1. Then a state $x \in \mathbb{X}$ is called

- *absorbing* if $\lambda(x,s) = 0$ for all $s \geq 0$,
- *stable* if $0 < \lambda(x,s) < \infty$ for all $s \geq 0$,
- *instantaneous* if $\lambda(x,s) = \infty$ for some $s \geq 0$.

Fact 2 (Right Continuity)**.** *A CTMC X_t has right-continuous sample paths if there are no instantaneous states.*

Figure 4 shows an example of a sample path of a right-continuous CTMC with states $\mathbb{X} = \{a,b,c\}$. Assume for the rest of this section that the CTMC is right-continuous, i.e., the mapping $t \mapsto X_t(\omega)$ is right-continuous for almost all $\omega \in \Omega$ and there are no instantaneous states. Moreover, $\mathbb{X}$ is countable and has discrete topology, Δ is the point at infinity if $\mathbb{X}$ is not finite. If $\mathbb{X}$ is finite, then we don't need Δ and the one-point compactification of $\mathbb{X}$ is the same as $\mathbb{X}$. Let $\underline{T}_n$ be the time of the n-th jump of the CTMC, with $\underline{T}_0 = 0$, and let $\underline{X}_n = X_{\underline{T}_n}$ when $\underline{T}_n < \infty$, and $\underline{X}_n = \underline{X}_{n-1}$ when $\underline{T}_n = \infty$, where the last condition is needed to handle the case of an absorbing state. Then the following holds for all $n \geq 1$

$$\underline{T}_0 = 0\ ; \quad W_\infty = \infty\ ; \quad \underline{T}_n = \underline{T}_{n-1} + W_{\underline{T}_{n-1}}; \quad \underline{X}_n = \begin{cases} X_{\underline{T}_n} & \text{if } \underline{T}_n < \infty, \\ \underline{X}_{n-1} & \text{if } \underline{T}_n = \infty. \end{cases}$$

The next proposition clarifies the structure of CTMC X_t in terms of the arrival times $\underline{T}_n$ and the transitions $\underline{X}_n$. Specifically, it shows that $\underline{X}_n$ is a DTMC and the waiting times $\underline{T}_n - \underline{T}_{n-1}$ depend only on the current state $\underline{X}_{n-1}$, and are independent of the past.

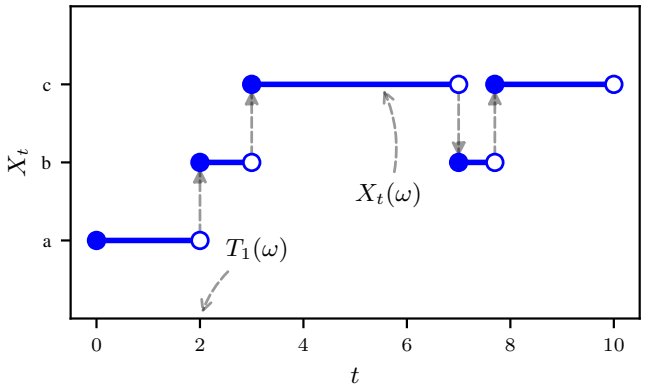


Figure 4: Sample path of a right-continuous CTMC with states $\mathbb{X} = \{a, b, c\}$.

Fact 3 (Structure of CTMC)**.** *Let X_t be a right-continuous CTMC with states $\mathbb{X}$. Then for all $n \in \mathbb{N}$, and $x' \in \mathbb{X}$, and $u \in \mathbb{R}_+$, we have*

$$\mathbb{P}(\underline{X}_{n+1} = x',\, \underline{T}_{n+1} - \underline{T}_n > u \mid \underline{X}_0, \ldots, \underline{X}_n, \underline{T}_0, \ldots, \underline{T}_n) = K_{t+u}(x' \mid x) \;\; e^{-\Lambda(x,u;\,t)},$$

if $\{\underline{X}_n = x, \;\; \underline{T}_n = t\}$ occurs. Here $\Lambda(x, u;\, t) = \int_t^{t+u} \lambda(x, s)\, ds$, $K_{t+u}(x' \mid x) \geq 0$, and $\sum_{x' \in \mathbb{X}} K_{t+u}(x' \mid x) = 1$.

Proof. First we will convert the desired probability into one that uses the continuous-time variables and then apply Proposition 1. First note that knowing $\underline{X}_0, \ldots, \underline{X}_n$ and $\underline{T}_0, \ldots, \underline{T}_n$ is equivalent to knowing the values of X_s for all $s \leq \underline{T}_n$. Therefore,

$$\begin{aligned}
&\mathbb{P}(\underline{X}_{n+1} = x',\, \underline{T}_{n+1} - \underline{T}_n > u \mid \underline{X}_0, \ldots, \underline{X}_n, \underline{T}_0, \ldots, \underline{T}_n) \\
&= \mathbb{P}(X_{\underline{T}_n + W_{\underline{T}_n}} = x',\, W_{\underline{T}_n} > u \mid X_s,\, s \leq \underline{T}_n)
\end{aligned}$$

Now we will use the fact that $\underline{T}_n$ is a stopping time, because the event $\{\underline{T}_n < t\}$ can be determined by knowing the values of X_s for $s \leq t$; specifically, $\{\underline{T}_n \leq t\}$ occurs if and only if there exists $0 < s_1, \ldots, s_n \leq t$ such that $X_{s_i} \neq X_{s_{i+1}}$ for all $i = 1, \ldots, n-1$, which can be determined by knowing the values of X_s for all $s \leq t$. Therefore, using the strong Markov property, we have

$$\begin{aligned}
&\mathbb{P}(X_{\underline{T}_n + W_{\underline{T}_n}} = x',\, W_{\underline{T}_n} > u \mid X_s,\, s \leq \underline{T}_n) \\
&= \mathbb{P}(X_{\underline{T}_n + W_{\underline{T}_n}} = x',\, W_{\underline{T}_n} > u \mid X_{\underline{T}_n}) \\
&= \mathbb{P}(X_{\underline{T}_n + W_{\underline{T}_n}} = x' \mid X_{\underline{T}_n},\, W_{\underline{T}_n} > u)\; \mathbb{P}(W_{\underline{T}_n} > u \mid X_{\underline{T}_n})
\end{aligned}$$

If $\underline{T}_n = t$ and $X_{\underline{T}_n} = \underline{X}_n = x$, then the rightmost expression is equal to $\exp(-\Lambda(x, u;\, t))$ by Proposition 1. Moreover, since $\{X_t = x,\, W_t > u\} = \{X_{t+s} = x,\, s \leq u\}$, we have

$$\begin{aligned}
\mathbb{P}(X_{t+W_t} = x' \mid X_t = x, W_t > u) &= \mathbb{P}(X_{t+u+W_{t+u}} = x' \mid X_{t+s} = x,\, s \leq u) \\
&= \mathbb{P}(X_{(t+u)+W_{t+u}} = x' \mid X_{t+u} = x) \\
&= K_{t+u}(x' \mid x)
\end{aligned}$$

□

Fact 4. *Denote $\mathbb{P}(X_{t+u} = x' \mid X_t = x)$ as $p_{t+u|t}(x' \mid x)$. Then,*

$$p_{t+u|t}(x' \mid x) = e^{-\Lambda(x,u;\,t)} \delta(x', x) + \int_t^{t+u} \lambda(x, s)\, e^{-\Lambda(x, s-t;\, t)} \sum_{y \in \mathbb{X}} K_s(y \mid x)\, p_{t+u|s}(x' \mid y)\, ds.$$

Proof. Assume that $n-1$ transitions have occurred by time t. Then,

$$\begin{aligned} p_{t+u|t}(x' \mid x) &= \mathbb{P}(X_{t+u} = x' \mid X_t = x) \\ &= \mathbb{P}(X_{t+u} = x', \underline{T}_n - t > u \mid X_t = x) + \mathbb{P}(X_{t+u} = x', \underline{T}_n - t \leq u \mid X_t = x) \end{aligned}$$

The first term on the right-hand side is equal to $e^{-\Lambda(x,u;\,t)}\delta(x', x)$ because

$$\begin{aligned} \mathbb{P}(X_{t+u} = x', \underline{T}_n - t > u \mid X_t = x) &= \mathbb{P}(X_{t+u} = x', \underline{T}_n - t > u \mid X_{\underline{T}_n} = x) \\ &= \mathbb{P}(X_{t+u} = x' \mid \underline{T}_n - t > u,\, X_{\underline{T}_n} = x)\ \mathbb{P}(\underline{T}_n - t > u \mid X_t = x) \\ &= \delta(x', x)\ e^{-\Lambda(x,u;\,t)} \quad \text{(By Proposition 1)} \end{aligned}$$

□

We can take the derivative of the expression above w.r.t. u to get Kolmogorov equations and show the form of the rate matrix decomposed into λ and K.

Fact 5 (Kolmogorov Forward Equation)**.** *Let X_t be a right-continuous CTMC with states $\mathbb{X}$. Then, for all $t \in [0, \infty)$, $x, x' \in \mathbb{X}$, and $v > t$,*

$$\frac{\partial}{\partial v} p_{v|t}(x' \mid x) = \sum_{y \in \mathbb{X}} p_{v|t}(x' \mid y)\ R_v(y \mid x),$$

where

$$R_v(y \mid x) = \begin{cases} \lambda(x, v) K_v(y \mid x) & \text{if } x \neq y, \\ -\lambda(x, v) & \text{if } x = y. \end{cases}$$

is the rate matrix of the CTMC, $K_v(y \mid x)$ is the transition kernel of the embedded DTMC, and $\lambda(x, v)$ is the rate parameter of the CTMC.

B.2 Time Reversal

Definition 4 (Time Reversal)**.** The time reversal of a *regular* CTMC X_t on a finite horizon $[0, T]$ is defined as $\hat{X}_t$ such that $\hat{X}_t = X_{T-t}$ for all $t \in [0, T]$ except the jump times. For the jump times, T_n, $\hat{X}_{T_n} = X_{T - T_n^-}$.

Fact 6 (Time Reversal of CTMC)**.** *The time reversal of a regular CTMC X_t is also a regular CTMC with the rate matrix*

$$\hat{R}_t(x, y) = \frac{p_{T-t}(y)}{p_t(x)} R_{T-t}(y, x).$$

C Proofs

C.1 ELBO

Proposition 1 (ELBO)**.** *For the time interval $[0, 1]$, let $p_0 = p_{\text{data}}$ and $p_1 = p_{\text{ref}}$ denote the terminal time marginals of the noising process X_t. Then for the reverse CTMC with initial distribution p_{ref}, parameterized rate $\hat{R}_t^\theta$, and the induced terminal marginal p_0^θ, the log-likelihood $\mathbb{E}\left[\log p_0^\theta(X_0)\right]$ under the model is bounded from below by*

$$\mathbb{E} \sum_{y \neq x} \left[-\hat{R}_t^\theta(x, y) + \frac{p_{t|0}(y|x_0)}{p_{t|0}(x|x_0)} R_t(y, x) \log \hat{R}_t^\theta(x, y) \right]$$

where the expectation is over $x_0 \sim p_{\text{data}}$,
$t \sim \text{Uniform}[0, 1]$, $x \sim p_{t|0}(x \mid x_0)$.

Proof. The result can be proved for any regular CTMC using Dynkin's formula (Hanson, 2007) for the change of measure on the CTMC path space followed by the data processing inequality as discussed in Lou et al. (2024).

In order to provide useful intuition for the result, we instead use elementary techniques to provide an informal argument in which we derive the continuous-time ELBO from the discrete-time ELBO, and then use that to establish the result. Assume that the time range is $[0, 1]$, and we discretize the time into K intervals of equal length $\Delta t = 1/K$. The endpoints of the intervals are $0 = t_0 < t_1 < \cdots < t_K = 1$. For brevity, we will use k instead of t_k throughout the derivation.

Before we begin, we recall a few useful facts that we will use multiple times.

1. Taylor expansion of log:

$$\begin{aligned}\log(1+x) &= x - \frac{x^2}{2} + \frac{x^3}{3} - \cdots \\ &= x - \frac{x^2}{2} + o(x^2)\end{aligned} \tag{4}$$

2. For $a, b > 0$,

$$\begin{aligned}\log(ab + o(a)) &= \log(a(b + o(1))) \\ &= \log(a) + \log(b + o(1)) \\ &= \log(a) + \log(b(1 + o(1))) \\ &= \log(a) + \log(b) + \log(1 + o(1)) \\ &= \log(a) + \log(b) + o(1)\end{aligned} \tag{5}$$

3. Bayes rule and the Markov property imply

$$\begin{aligned}q(x_k \mid x_{k+1}, x_0) &= \frac{q(x_k \mid x_0)}{q(x_{k+1} \mid x_0)} q(x_{k+1} \mid x_k) \\ &= r(k, k+1) q(x_{k+1} \mid x_k),\end{aligned} \tag{6}$$

   where we denote the ratio $\frac{q(x_k|x_0)}{q(x_{k+1}|x_0)}$ with $r(k, k+1)$.

With slight abuse of notation, we will denote the probability law of the noising process with q and its time-reversal with p. To begin, recall that the discrete-time ELBO (Ho et al., 2020) is given by

$$\log p_\theta(x_0) \geq \underbrace{\mathop{\mathbb{E}}_{x_1 \sim q_{x_1|x_0}} \log p_\theta(x_0|x_1)}_{-L_0} - \underbrace{\mathop{\mathbb{E}}_{x_{2:T} \sim q_{x_{2:T}|x_0}} \sum_{t=2}^{T} D_{\mathrm{KL}}\left[q(x_{t-1}|x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\right]}_{L_{1:T-1}} - \underbrace{D_{\mathrm{KL}}[q(x_T \mid x_0) \,\|\, p_\theta(x_T)]}_{L_T}$$

We will focus on one term of $L_{1:T-1}$, denoting it by L_k; we will also drop θ from p_θ for brevity. Taking the expectation inside the summation for $L_{1:T-1}$ and writing one term, we get

$$\begin{aligned}L_k(x_0) &= \mathop{\mathbb{E}}_{x_{k+1} \sim q_{k+1|0}} D_{\mathrm{KL}}\left[q(x_k \mid x_{k+1}, x_0) \,\|\, p(x_k \mid x_{k+1})\right] \\ &= \mathop{\mathbb{E}}_{x_{k+1} \sim q_{k+1|0}} -H[q(x_k \mid x_{k+1}, x_0)] + \mathrm{CE}[q(x_k \mid x_{k+1}, x_0) \,\|\, p(x_k \mid x_{k+1})]\end{aligned}$$

We will expand the entropy and the cross entropy terms by writing the conditional probabilities in terms of the forward and reverse rate matrices and making approximations knowing that we will ultimately apply the limit $\Delta t \to 0$, and $K \to \infty$. The approximations do not introduce any error but are a way to apply the limit throughout the derivation to manage the complexity of the expressions. The following are the expressions for the conditional probabilities in terms of the forward and reverse rate matrices:

$$\begin{aligned}q(x_k \mid x_{k+1}, x_0) &= r(k, k+1)\left(\delta_{x_k}(x_{k+1}) + R_k(x_k, x_{k+1})\Delta t + o(\Delta t)\right) \\ p(x_k \mid x_{k+1}) &= \delta_{x_k}(x_{k+1}) + \hat{R}_{k+1}(x_{k+1}, x_k)\Delta t + o(\Delta t)\end{aligned} \tag{7}$$

where we assume $r(k, k+1)$ exists and is finite, i.e., $q(x_{k+1} \mid x_0) > 0$. We will also suppress the indices involved in $r(k, k+1)$ and $\delta_{x_k}(x_{k+1})$ for brevity as they do not change in the following derivation. So unless stated explicitly, $r = r(k, k+1)$ and $\delta = \delta_{x_k}(x_{k+1})$ and $\bar{\delta} = 1 - \delta_{x_k}(x_{k+1})$.

$$
\begin{aligned}
&- \mathrm{CE}[q(x_k \mid x_{k+1}, x_0) \,\|\, p(x_k \mid x_{k+1})] \\
&= \sum_{x_k} q(x_k \mid x_{k+1}, x_0) \log p(x_k \mid x_{k+1}) \\
&= \sum_{x_k} r \left(\delta + R_k(x_k, x_{k+1})\Delta t + o(\Delta t)\right) \log \left(\delta + \hat{R}_{k+1}(x_{k+1}, x_k)\Delta t + o(\Delta t)\right)
\end{aligned}
$$

Note that

$$
\begin{aligned}
\log(\delta + \hat{R}_{k+1}(x_{k+1}, x_k)\Delta t + o(\Delta t)) &= \delta \log(1 + \hat{R}_{k+1}(x_{k+1}, x_k)\Delta t + o(\Delta t)) \\
&\quad + \bar{\delta} \log(\hat{R}_{k+1}(x_{k+1}, x_k)\Delta t + o(\Delta t)) \\
&= \delta(\hat{R}_{k+1}(x_{k+1}, x_k)\Delta t + o(\Delta t)) \\
&\quad + \bar{\delta} \log(\hat{R}_{k+1}(x_{k+1}, x_k)\Delta t) + o(\Delta t) \quad \text{(by 4)} \\
&= \delta(\hat{R}_{k+1}(x_{k+1}, x_k)\Delta t + o(\Delta t)) \\
&\quad + \bar{\delta} \log(\hat{R}_{k+1}(x_{k+1}, x_k)\Delta t) + o(\Delta t) \\
&= \delta(\hat{R}_{k+1}(x_{k+1}, x_k)\Delta t) \\
&\quad + \bar{\delta}(\log(\hat{R}_{k+1}(x_{k+1}, x_k)) + \log(\Delta t) + o(1)) + o(\Delta t) \quad \text{(by 5)} \qquad (8)
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
&- \mathrm{CE}[q(x_k \mid x_{k+1}, x_0) \,\|\, p(x_k \mid x_{k+1})] \\
&= \sum_{x_k} r \left\{\delta \hat{R}_{k+1}(x_{k+1}, x_k)\Delta t + \bar{\delta}\Delta t R_k(x_k, x_{k+1}) \log(\hat{R}_{k+1}(x_{k+1}, x_k)) + \Delta t \log(\Delta t) + o(\Delta t)\right\} \\
&= \sum_{x_k} r \left\{\delta \hat{R}_{k+1}(x_{k+1}, x_k)\Delta t + \bar{\delta}\Delta t R_k(x_k, x_{k+1}) \log(\hat{R}_{k+1}(x_{k+1}, x_k)) + o(\Delta t)\right\} \\
&= \Delta t \hat{R}_{k+1}(x_{k+1}, x_{k+1}) + \Delta t \sum_{x_k \neq x_{k+1}} r R_k(x_k, x_{k+1}) \log(\hat{R}_{k+1}(x_{k+1}, x_k)) + o(\Delta t) \qquad (9)
\end{aligned}
$$

Similarly,

$$
\begin{aligned}
-H[q(x_k \mid x_{k+1}, x_0)] &= \sum_{x_k} r \left\{\delta + R_k(x_k, x_{k+1})\Delta t + o(\Delta t)\right\} \log r \left\{\delta + R_k(x_k, x_{k+1})\Delta t + o(\Delta t)\right\} \\
&= \sum_{x_k} r \left\{\delta R_k(x_k, x_{k+1})\Delta t + \bar{\delta}\Delta t R_k(x_k, x_{k+1}) \log(R_k(x_k, x_{k+1})) + o(\Delta t)\right\} \\
&\quad + \sum_{x_k} \delta r \log r + r\Delta t R_k(x_k, x_{k+1}) \log r \\
&= \Delta t R_k(x_{k+1}, x_{k+1}) + \Delta t \sum_{x_k \neq x_{k+1}} r R_k(x_k, x_{k+1}) \log(R_k(x_k, x_{k+1})) + o(\Delta t) \\
&\quad + \Delta t \sum_{x_k \neq x_{k+1}} R_k(x_k, x_{k+1}) r \log r \quad (\text{since } r \log r = 0 \text{ when } x_k = x_{k+1}) \qquad (10)
\end{aligned}
$$

Putting the two together, we get

$$L_k(x_0) = \mathop{\mathbb{E}}_{x_{k+1}} \sum_{x_k} r\Delta t \left\{ \underbrace{\left[\delta R_k(x_k, x_{k+1}) + \bar{\delta} R_k(x_k, x_{k+1}) \log(R_k(x_k, x_{k+1}))\right] + \bar{\delta} R_k(x_k, x_{k+1}) \log r\right]}_{H \text{ term}} \right.$$
$$\left. - \underbrace{\left[\delta \hat{R}_{k+1}(x_{k+1}, x_k) + \bar{\delta} R_k(x_k, x_{k+1}) \log(\hat{R}_{k+1}(x_{k+1}, x_k))\right]}_{\text{CE term}} + \frac{o(\Delta t)}{\Delta t} \right\} \tag{11}$$

Recall that $r = \frac{q(x_k|x_0)}{q(x_{k+1}|x_0)}$, and the expectation is over $x_{k+1} \sim q_{k+1|0}(\cdot \mid x_0)$. The entropy is constant with respect to $\hat{R}$.

$$\sum_k L_k(x_0) = \sum_k \Delta t \sum_{x_{k+1}} q(x_{k+1} \mid x_0) \sum_{x_k} \frac{q(x_k \mid x_0)}{q(x_{k+1} \mid x_0)} \left\{ C(x_0, x_k, x_{k+1}) \right.$$
$$\left. - \underbrace{\left[\delta \hat{R}_{k+1}(x_{k+1}, x_k) + \bar{\delta} R_k(x_k, x_{k+1}) \log(\hat{R}_{k+1}(x_{k+1}, x_k))\right]}_{\text{CE term}} + \frac{o(\Delta t)}{\Delta t} \right\}$$

Now recall that δ and $\bar{\delta}$ are shorthand for $\delta_{x_t}(y)$ and $1 - \delta_{x_t}(y)$, respectively, and therefore

$$\sum_y \frac{q(y|x_0)}{q(x_t|x_0)} \delta_{x_t}(y) \hat{R}_t(x_t, y) = \hat{R}_t(x_t, x_t) = -\sum_{y \neq x_t} \hat{R}_t(x_t, y). \tag{12}$$

As $\Delta t \to 0, K \to \infty$, we get $x_{k+1} \to x_{t_{k+1}} = x_t$ for $t \in [0, 1]$, and the summation becomes an integral (since the Markov chain is assumed to be regular, the limit is well defined). Denoting $\lim_{K\to\infty} \sum_k L_k(x_0)$ as $\mathcal{L}(x_0)$ and using Equation (12), we get

$$\begin{aligned}
\mathcal{L}(x_0) &= \int_0^1 \sum_{x_t} q(x_t \mid x_0) \sum_y \frac{q(y \mid x_0)}{q(x_t \mid x_0)} \left[\delta \hat{R}_t(x_t, y) + \bar{\delta} R_t(y, x_t) \log(\hat{R}_t(x_t, y))\right] dt + C(x_0) \\
&= \mathop{\mathbb{E}}_{t, x_t} \sum_{y \neq x_t} \left[-\hat{R}_t(x_t, y) + \frac{q(y \mid x_0)}{q(x_t \mid x_0)} R_t(y, x_t) \log(\hat{R}_t(x_t, y))\right] dt + C(x_0)
\end{aligned}$$

□

Corollary 1 (ELBO: Joint Insertion)**.** *A (biased) estimate of the upper bound on the negative log-likelihood in Proposition 1 for the joint parameterization of $\hat{R}_t^\theta$ is given by*

$$\begin{aligned}
&\mathcal{L}(\theta) = \mathcal{L}_{\text{len}}(\theta) + \mathcal{L}_{\text{token}}(\theta), \textit{where} \\
&\mathcal{L}^{\text{len}}(\theta) = \mathbb{E}\left[\hat{\lambda}_t^\theta(\boldsymbol{x}_0[\boldsymbol{b}]) - \gamma_t^{\text{joint}} \log \hat{\lambda}_t^\theta(\boldsymbol{x}_0[\boldsymbol{b}])\right], \textit{and} \\
&\mathcal{L}^{\text{token}}(\theta) = -\mathbb{E}\left[\gamma_t^{\text{joint}} \sum_{i,w} q(i, w) \log \hat{p}_t^\theta(i, w \mid \boldsymbol{x}_0[\boldsymbol{b}])\right].
\end{aligned}$$

Here, the expectation is over $\boldsymbol{x}_0 \sim p_{\text{data}}$, $t \sim \text{Uniform}[0, T]$, $(n - m) \sim \text{Poisson}(\bar{\sigma}_{0,t})$, $\delta_{\{\cdot \geq 0\}}(m)$, $\boldsymbol{b} \sim \text{Uniform}(\mathbb{B}_{n,m})$, the inner sum in $\mathcal{L}^{\text{token}}(\theta)$ is over $i \in [m + 1], w \in \mathbb{V}$, and $\gamma_t^{\text{joint}} = \frac{\sigma_t (n-m)}{\bar{\sigma}_{0,t} \tilde{S}_{n,m,\sigma_t}}$ with $\tilde{S}_{n,m,\sigma_t} = [(1 - \delta_0(m)) + \delta_0(m) S_{n,\sigma_t}]$, $S_{n,\sigma_t} = \sum_{k=0}^{\infty} \frac{n!\, \bar{\sigma}_{0,t}^k}{(n+k)!}$, $\boldsymbol{x}_0 = (n, \dot{\boldsymbol{x}}_0)$, and $q(i, w) = \frac{\sum_{k \in s_i(\boldsymbol{b})} \delta_w(x_0^k)}{n-m}$ is the normalized count of the vocabulary item w occurring in $\boldsymbol{x}_0$ between the $(i-1)$-th and i-th 1s, with $s_i(\boldsymbol{b})$ being the set of indices of 0s in $\boldsymbol{b}$ that fall between the $(i-1)$-th and i-th 1s.

Proof. Replacing the rate matrices in Proposition 1 with their respective components, we get

$$\begin{aligned}
&\sum_{\boldsymbol{y}\neq\boldsymbol{x}}\left[-\hat{R}_t(\boldsymbol{x},\boldsymbol{y})+\frac{p_{t|0}(\boldsymbol{y}\mid\boldsymbol{x}_0)}{p_{t|0}(\boldsymbol{x}\mid\boldsymbol{x}_0)}R_t(\boldsymbol{y},\boldsymbol{x})\log\hat{R}_t(\boldsymbol{x},\boldsymbol{y})\right]\\
&=\sum_{\boldsymbol{y}\neq\boldsymbol{x}}\left[-\hat{\lambda}_t(\boldsymbol{x})\,\hat{K}_t(\boldsymbol{x},\boldsymbol{y})+\frac{p_{t|0}(\boldsymbol{y}\mid\boldsymbol{x}_0)}{p_{t|0}(\boldsymbol{x}\mid\boldsymbol{x}_0)}\lambda_t(\boldsymbol{y})\,K_t(\boldsymbol{y},\boldsymbol{x})\log\left(\hat{\lambda}_t(\boldsymbol{x})\,\hat{K}_t(\boldsymbol{x},\boldsymbol{y})\right)\right]\\
&=\underbrace{-\hat{\lambda}_t(\boldsymbol{x})+\log(\hat{\lambda}_t(\boldsymbol{x}))\sum_{\boldsymbol{y}\neq\boldsymbol{x}}K_t(\boldsymbol{y},\boldsymbol{x})\frac{p_{t|0}(\boldsymbol{y}\mid\boldsymbol{x}_0)}{p_{t|0}(\boldsymbol{x}\mid\boldsymbol{x}_0)}\lambda_t(\boldsymbol{y})}_{\text{the length term}(A)}\\
&\quad+\underbrace{\sum_{\boldsymbol{y}\neq\boldsymbol{x}}K_t(\boldsymbol{y},\boldsymbol{x})\frac{p_{t|0}(\boldsymbol{y}\mid\boldsymbol{x}_0)}{p_{t|0}(\boldsymbol{x}\mid\boldsymbol{x}_0)}\lambda_t(\boldsymbol{y})\log\left(\hat{K}_t(\boldsymbol{x},\boldsymbol{y})\right)}_{\text{the token term}(B)}
\end{aligned}$$

The length term (A):

Let $\boldsymbol{x}=(m,\dot{\boldsymbol{x}})$, $\boldsymbol{y}=(r,\dot{\boldsymbol{y}})$, and $\boldsymbol{x}_0=(n,\dot{\boldsymbol{x}}_0)$ for some $n>r>m$. Then

$$\frac{p_{t|0}(\boldsymbol{y}\mid\boldsymbol{x}_0)}{p_{t|0}(\boldsymbol{x}\mid\boldsymbol{x}_0)}=\frac{\mathbb{P}(\underline{N}_t=r\mid\underline{N}_0=n)\mathbb{P}(\dot{\underline{\boldsymbol{X}}}_{n-r}=\dot{\boldsymbol{y}}\mid\dot{\underline{\boldsymbol{X}}}_0=\dot{\boldsymbol{x}}_0)}{\mathbb{P}(\underline{N}_t=m\mid\underline{N}_0=n)\mathbb{P}(\dot{\underline{\boldsymbol{X}}}_{n-m}=\dot{\boldsymbol{x}}\mid\dot{\underline{\boldsymbol{X}}}_0=\dot{\boldsymbol{x}}_0)}$$

For $t>s$, and $m>0$, from equation 19 we have

$$\begin{aligned}
\frac{\mathbb{P}(N_t=m+1\mid N_s=n)}{\mathbb{P}(N_t=m\mid N_s=n)}&=\frac{e^{-\bar{\sigma}_{s,t}}(\bar{\sigma}_{s,t})^{n-m-1}}{(n-m-1)!}\frac{(n-m)!}{e^{-\bar{\sigma}_{s,t}}(\bar{\sigma}_{s,t})^{n-m}}\\
&=\frac{n-m}{\bar{\sigma}_{s,t}}.
\end{aligned}$$

For $m=0$, we need the tail of the Taylor series for the exponential to compute the denominator:

$$\begin{aligned}
\frac{\mathbb{P}(N_t=m+1\mid N_s=n)}{\mathbb{P}(N_t=m\mid N_s=n)}&=\frac{\mathbb{P}(N_t=1\mid N_s=n)}{\mathbb{P}(N_t=0\mid N_s=n)}\\
&=\frac{e^{-\bar{\sigma}_{s,t}}(\bar{\sigma}_{s,t})^{n-1}}{(n-1)!}\frac{1}{1-\sum_{k=0}^{n-1}\frac{e^{-\bar{\sigma}_{s,t}}(\bar{\sigma}_{s,t})^k}{k!}}\\
&=\frac{\bar{\sigma}_{s,t}^{n-1}}{(n-1)!}\frac{1}{e^{\bar{\sigma}_{s,t}}-\sum_{k=0}^{n-1}\frac{(\bar{\sigma}_{s,t})^k}{k!}}\\
&=\frac{\bar{\sigma}_{s,t}^{n-1}}{(n-1)!}\frac{1}{\sum_{k=0}^{\infty}\frac{(\bar{\sigma}_{s,t})^{n+k}}{(n+k)!}}\\
&=\frac{n}{\bar{\sigma}_{s,t}}\frac{1}{\sum_{k=0}^{\infty}\frac{n!(\bar{\sigma}_{s,t})^k}{(n+k)!}}
\end{aligned}$$

Using $S_{n,\sigma_t}=\sum_{k=0}^{\infty}\frac{n!(\bar{\sigma}_{s,t})^k}{(n+k)!}$, and $\tilde{S}_{n,m,\sigma_t}=(1-\delta_0(m))+\delta_0(m)S_{n,\sigma_t}$, we can combine the two cases to get

$$\frac{\mathbb{P}(N_t=m+1\mid N_s=n)}{\mathbb{P}(N_t=m\mid N_s=n)}=\frac{n-m}{\bar{\sigma}_{s,t}}\frac{1}{\tilde{S}_{n,m,\sigma_t}}\tag{13}$$

For $r = m + 1$, we have

$$K_t(\boldsymbol{y}, \boldsymbol{x}) \frac{p_{t|0}(\boldsymbol{y} \mid \boldsymbol{x}_0)}{p_{t|0}(\boldsymbol{x} \mid \boldsymbol{x}_0)}$$
$$= \frac{n-m}{\bar{\sigma}_{0,t}\, \tilde{S}_{n,m,\sigma_t}} \frac{K(\boldsymbol{y}, \boldsymbol{x}) \mathbb{P}(\underline{\boldsymbol{X}}_{n-m-1} = \dot{\boldsymbol{y}} \mid \underline{\boldsymbol{X}}_0 = \dot{\boldsymbol{x}}_0)}{\sum_{\dot{\boldsymbol{z}} \in \mathbb{V}^{m+1}} \mathbb{P}(\underline{\boldsymbol{X}}_{n-m-1} = \dot{\boldsymbol{z}} \mid \underline{\boldsymbol{X}}_0 = \dot{\boldsymbol{x}}_0) K(\boldsymbol{z}, \boldsymbol{x})}$$

Due to the sum over $\boldsymbol{y}$, the length term simplifies to

$$- \hat{\lambda}_t(\boldsymbol{x}) + \sigma_t \frac{(n-m)}{\bar{\sigma}_{0,t}\, \tilde{S}_{n,m,\sigma_t}} \log \hat{\lambda}_t(\boldsymbol{x}) \frac{\sum_{\boldsymbol{y} \neq \boldsymbol{x}} K(\boldsymbol{y}, \boldsymbol{x}) \mathbb{P}(\underline{\boldsymbol{X}}_{n-m-1} = \dot{\boldsymbol{y}} \mid \underline{\boldsymbol{X}}_0 = \dot{\boldsymbol{x}}_0)}{\sum_{\dot{\boldsymbol{z}} \in \mathbb{V}^{m+1}} \mathbb{P}(\underline{\boldsymbol{X}}_{n-m-1} = \dot{\boldsymbol{z}} \mid \underline{\boldsymbol{X}}_0 = \dot{\boldsymbol{x}}_0) K(\boldsymbol{z}, \boldsymbol{x})}$$
$$= -\hat{\lambda}_t(\boldsymbol{x}) + \sigma_t \frac{(n-m)}{\bar{\sigma}_{0,t}\, \tilde{S}_{n,m,\sigma_t}} \log \hat{\lambda}_t(\boldsymbol{x}) \tag{14}$$

The token term (B):

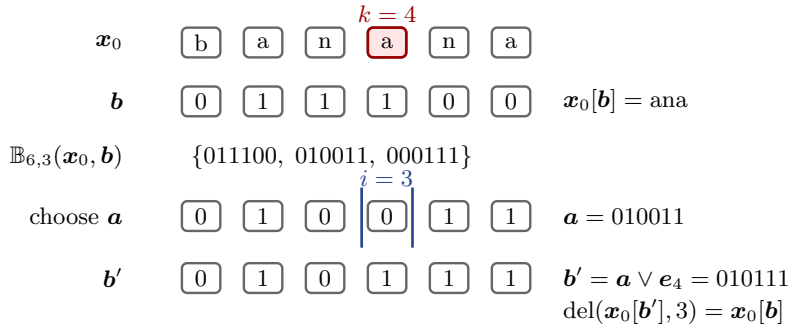
Now we write the token term (B) along with the outer expectation, where we will denote $\mathbb{P}(\underline{\boldsymbol{X}}_r = \dot{\boldsymbol{y}} \mid \underline{\boldsymbol{X}}_0 = \dot{\boldsymbol{x}}_0)$ as $\underline{p}_{r|0}(\dot{\boldsymbol{y}} \mid \dot{\boldsymbol{x}}_0)$.

$$\mathop{\mathbb{E}}_{\boldsymbol{x} \sim p_{t|0}(\cdot|\boldsymbol{x}_0)} \sum_{\boldsymbol{y} \neq \boldsymbol{x}} \frac{\mathbb{P}(\underline{N}_t = r \mid \underline{N}_0 = n)}{\mathbb{P}(\underline{N}_t = m \mid \underline{N}_0 = n)} \frac{K(\boldsymbol{y}, \boldsymbol{x}) \underline{p}_{n-m-1|0}(\dot{\boldsymbol{y}} \mid \dot{\boldsymbol{x}}_0)}{\sum_{\dot{\boldsymbol{z}} \in \mathbb{V}^{m+1}} \underline{p}_{n-m-1|0}(\dot{\boldsymbol{z}} \mid \dot{\boldsymbol{x}}_0) K(\boldsymbol{z}, \boldsymbol{x})} \lambda_t(\boldsymbol{y}) \log \left(\hat{K}_t(\boldsymbol{x}, \boldsymbol{y}) \right)$$
$$= \frac{\sigma_t}{\bar{\sigma}_{0,t}} \mathop{\mathbb{E}}_{\boldsymbol{x} \sim p_{t|0}(\cdot|\boldsymbol{x}_0)} \frac{(n-m)}{\tilde{S}_{n,m,\sigma_t}} \sum_{\boldsymbol{y} \in \mathbb{V}^{m+1}} \frac{K(\boldsymbol{y}, \boldsymbol{x}) \underline{p}_{n-m-1|0}(\dot{\boldsymbol{y}} \mid \dot{\boldsymbol{x}}_0)}{\sum_{\dot{\boldsymbol{z}} \in \mathbb{V}^{m+1}} \underline{p}_{n-m-1|0}(\dot{\boldsymbol{z}} \mid \dot{\boldsymbol{x}}_0) K(\boldsymbol{z}, \boldsymbol{x})} \log \left(\hat{K}_t(\boldsymbol{x}_0[\boldsymbol{b}], \boldsymbol{y}) \right)$$
$$= \frac{\sigma_t}{\bar{\sigma}_{0,t}} \mathop{\mathbb{E}}_{m,\boldsymbol{b}} \frac{(n-m)}{\tilde{S}_{n,m,\sigma_t}} \sum_{\boldsymbol{y} \in \mathbb{V}^{m+1}} \frac{\underline{p}_{n-m-1|0}(\dot{\boldsymbol{y}} \mid \dot{\boldsymbol{x}}_0)\, K(\boldsymbol{y}, \boldsymbol{x}_0[\boldsymbol{b}])}{\sum_{\dot{\boldsymbol{z}} \in \mathbb{V}^{m+1}} \underline{p}_{n-m-1|0}(\dot{\boldsymbol{z}} \mid \dot{\boldsymbol{x}}_0)\, K(\boldsymbol{z}, \boldsymbol{x}_0[\boldsymbol{b}])} \log \left(\hat{K}_t(\boldsymbol{x}_0[\boldsymbol{b}], \boldsymbol{y}) \right)$$
$$= \frac{\sigma_t}{\bar{\sigma}_{0,t}} \mathop{\mathbb{E}}_{m,\boldsymbol{b}} \frac{(n-m)}{\tilde{S}_{n,m,\sigma_t}} \sum_{\dot{\boldsymbol{y}} \in \mathbb{V}^{m+1}} \frac{\sum_{\boldsymbol{b}' \in \mathbb{B}_{n,m+1}} \frac{\delta_{\dot{\boldsymbol{x}}_0[\boldsymbol{b}']}(\dot{\boldsymbol{y}})}{\binom{n}{m+1}} \sum_{i=1}^{m+1} \frac{\delta_{\mathrm{del}(\dot{\boldsymbol{y}}, i)}(\boldsymbol{x}_0[\boldsymbol{b}])}{m+1}}{\sum_{\dot{\boldsymbol{z}} \in \mathbb{V}^{m+1}} \sum_{\boldsymbol{b}' \in \mathbb{B}_{n,m+1}} \frac{\delta_{\dot{\boldsymbol{x}}_0[\boldsymbol{b}']}(\dot{\boldsymbol{z}})}{\binom{n}{m+1}} \sum_{i=1}^{m+1} \frac{\delta_{\mathrm{del}(\dot{\boldsymbol{z}}, i)}(\boldsymbol{x}_0[\boldsymbol{b}])}{m+1}} \log \left(\hat{K}_t(\boldsymbol{x}_0[\boldsymbol{b}], \boldsymbol{y}) \right)$$
$$= \frac{\sigma_t}{\bar{\sigma}_{0,t}} \mathop{\mathbb{E}}_{m,\boldsymbol{b}} \frac{(n-m)}{\tilde{S}_{n,m,\sigma_t}} \frac{\sum_{\boldsymbol{b}' \in \mathbb{B}_{n,m+1}} \sum_{i=1}^{m+1} \delta_{\mathrm{del}(\boldsymbol{x}_0[\boldsymbol{b}'], i)}(\boldsymbol{x}_0[\boldsymbol{b}]) \log \left(\hat{K}_t(\boldsymbol{x}_0[\boldsymbol{b}], \boldsymbol{x}_0[\boldsymbol{b}']) \right)}{\sum_{\boldsymbol{b}' \in \mathbb{B}_{n,m+1}} \sum_{i=1}^{m+1} \delta_{\mathrm{del}(\boldsymbol{x}_0[\boldsymbol{b}'], i)}(\boldsymbol{x}_0[\boldsymbol{b}])}$$

Now let's look at the numerator:

$$\sum_{\boldsymbol{b}' \in \mathbb{B}_{n,m+1}} \sum_{i=1}^{m+1} \delta_{\mathrm{del}(\boldsymbol{x}_0[\boldsymbol{b}'], i)}(\boldsymbol{x}_0[\boldsymbol{b}]) \log \left(\hat{K}_t(\boldsymbol{x}_0[\boldsymbol{b}], \boldsymbol{x}_0[\boldsymbol{b}']) \right)$$
$$= \sum_{\boldsymbol{b}' \in \mathbb{B}_{n,m+1}} \sum_{i=1}^{m+1} \left[\prod_{j=0 \neq i}^{m+1} \delta_{\boldsymbol{x}_0[b']^j}(\boldsymbol{x}_0[\boldsymbol{b}]^j) \right] \left[\sum_{w \in \mathbb{V}} \delta_w(\boldsymbol{x}_0[\boldsymbol{b}']^i) \right] \log \left(\hat{K}_t(\boldsymbol{x}_0[\boldsymbol{b}], \boldsymbol{x}_0[\boldsymbol{b}']) \right)$$
$$= \sum_{i=1}^{m+1} \sum_{w \in \mathbb{V}} \sum_{\boldsymbol{b}' \in \mathbb{B}_{n,m+1}} \left[\prod_{j=0 \neq i}^{m+1} \delta_{\boldsymbol{x}_0[b']^j}(\boldsymbol{x}_0[\boldsymbol{b}]^j) \right] \delta_w(\boldsymbol{x}_0[\boldsymbol{b}']^i) \log \left(\hat{K}_t(\boldsymbol{x}_0[\boldsymbol{b}], \boldsymbol{x}_0[\boldsymbol{b}']) \right)$$
$$= \sum_{i=1}^{m+1} \sum_{w \in \mathbb{V}} \sum_{\boldsymbol{a} \in \mathbb{B}_{n,m}(\boldsymbol{x}_0, \boldsymbol{b})} \sum_{k \in s_i(\boldsymbol{a})} \delta_w(\boldsymbol{x}_0^k) \log \left(\hat{K}_t(\boldsymbol{x}_0[\boldsymbol{b}], \boldsymbol{x}_0[\boldsymbol{a} \vee \boldsymbol{e}_k]) \right)$$
$$= \sum_{\boldsymbol{a} \in \mathbb{B}_{n,m}(\boldsymbol{x}_0, \boldsymbol{b})} \sum_{i=1}^{m+1} \sum_{k \in s_i(\boldsymbol{a})} \sum_{w \in \mathbb{V}} \delta_w(\boldsymbol{x}_0^k) \log \left(\hat{K}_t(\boldsymbol{x}_0[\boldsymbol{b}], \boldsymbol{x}_0[\boldsymbol{a} \vee \boldsymbol{e}_k]) \right)$$

where $\boldsymbol{e}_k \in \mathbb{B}_{n,1}$ is the vector of length n with all 0s except for the k-th position, $\mathbb{B}_{n,m}(\boldsymbol{x}_0, \boldsymbol{b}) = \{\boldsymbol{a} \in \mathbb{B}_{n,m} : \boldsymbol{x}_0[\boldsymbol{a}] = \boldsymbol{x}_0[\boldsymbol{b}]\}$, and $s_i(\boldsymbol{a})$ is the set of indices of zeros in $\boldsymbol{a}$ that fall between the $(i-1)$-th and i-th 1s. The re-indexing in the third line above is justified by a bijection between the pairs $(\boldsymbol{b}', i)$ satisfying $\delta_{\mathrm{del}(\boldsymbol{x}_0[\boldsymbol{b}'],i)}(\boldsymbol{x}_0[\boldsymbol{b}]) = 1$ and the triples $(\boldsymbol{a}, i, k)$ with $\boldsymbol{a} \in \mathbb{B}_{n,m}(\boldsymbol{x}_0, \boldsymbol{b})$ and $k \in s_i(\boldsymbol{a})$. To see this, fix such a pair $(\boldsymbol{b}', i)$. Since deleting the i-th token from $\boldsymbol{x}_0[\boldsymbol{b}']$ yields $\boldsymbol{x}_0[\boldsymbol{b}]$, there is a unique original index $k \in [n]$ that is present in $\boldsymbol{b}'$ but not in the subsequence corresponding to $\boldsymbol{x}_0[\boldsymbol{b}]$. Let $\boldsymbol{a} \coloneqq \boldsymbol{b}' \wedge (\boldsymbol{1}_n - \boldsymbol{e}_k)$, so that $\boldsymbol{b}' = \boldsymbol{a} \vee \boldsymbol{e}_k$ and $\boldsymbol{x}_0[\boldsymbol{a}] = \boldsymbol{x}_0[\boldsymbol{b}]$. Moreover, the index k must lie in the gap of $\boldsymbol{a}$ corresponding to insertion location i, and therefore $k \in s_i(\boldsymbol{a})$. Conversely, given any triple $(\boldsymbol{a}, i, k)$ with $\boldsymbol{a} \in \mathbb{B}_{n,m}(\boldsymbol{x}_0, \boldsymbol{b})$ and $k \in s_i(\boldsymbol{a})$, defining $\boldsymbol{b}' \coloneqq \boldsymbol{a} \vee \boldsymbol{e}_k$ gives a unique predecessor mask in $\mathbb{B}_{n,m+1}$ such that deleting the i-th token from $\boldsymbol{x}_0[\boldsymbol{b}']$ yields $\boldsymbol{x}_0[\boldsymbol{b}]$. Therefore, summing over valid pairs $(\boldsymbol{b}', i)$ is equivalent to summing over triples $(\boldsymbol{a}, i, k)$. For example, the following schematic illustrates a repeated-token case with $\boldsymbol{x}_0 = \text{banana}$ and $\boldsymbol{x}_0[\boldsymbol{b}] = \text{ana}$, where multiple masks $\boldsymbol{a}$ give the same subsequence:



Note that the denominator is the same as the numerator except for the log term, and therefore the overall token term is an expectation over the uniform probability mass function $\pi(\boldsymbol{a}, i, k \mid \boldsymbol{x}_0, \boldsymbol{b})$ on the set

$$\Omega(\boldsymbol{x}_0, \boldsymbol{b}) \coloneqq \{(\boldsymbol{a}, i, k) \mid \boldsymbol{a} \in \mathbb{B}_{n,m}(\boldsymbol{x}_0, \boldsymbol{b}),\ i \in [m+1],\ k \in s_i(\boldsymbol{a})\}.$$

At this point, before introducing any approximation to the reverse kernel, the token term is

$$\frac{\sigma_t}{\bar{\sigma}_{0,t}} \mathop{\mathbb{E}}_{m,\boldsymbol{b}} \frac{(n-m)}{\tilde{S}_{n,m,\sigma_t}} \mathop{\mathbb{E}}_{\boldsymbol{a},i,k \sim \pi(\cdot \mid \boldsymbol{x}_0, \boldsymbol{b})} \log\left(\hat{K}_t(\boldsymbol{x}_0[\boldsymbol{b}], \boldsymbol{x}_0[\boldsymbol{a} \vee \boldsymbol{e}_k])\right).$$

Therefore, we get an unbiased estimate of the token term by taking a single sample $(\boldsymbol{a}, i, k) \sim \pi(\cdot \mid \boldsymbol{x}_0, \boldsymbol{b})$. In fact, up to this point we can also fix $\boldsymbol{a} = \boldsymbol{b}$ without introducing bias, provided we keep the outer expectation over $\boldsymbol{b} \sim \mathrm{Uniform}(\mathbb{B}_{n,m})$. To see this, for any subsequence $\dot{\boldsymbol{y}} \in \mathbb{V}^m$, define the equivalence class

$$C(\dot{\boldsymbol{y}}) \coloneqq \{\boldsymbol{a} \in \mathbb{B}_{n,m} \mid \boldsymbol{x}_0[\boldsymbol{a}] = \dot{\boldsymbol{y}}\},$$

and define

$$h(\boldsymbol{a}, \dot{\boldsymbol{y}}) \coloneqq \frac{1}{n-m} \sum_{i=1}^{m+1} \sum_{k \in s_i(\boldsymbol{a})} \log\left(\hat{K}_t(\dot{\boldsymbol{y}}, \boldsymbol{x}_0[\boldsymbol{a} \vee \boldsymbol{e}_k])\right).$$

The exact estimator obtained from the expectation over $\pi(\cdot \mid \boldsymbol{x}_0, \boldsymbol{b})$ can then be written as

$$\frac{1}{|\mathbb{B}_{n,m}|} \sum_{\dot{\boldsymbol{y}}} \sum_{\boldsymbol{b} \in C(\dot{\boldsymbol{y}})} \frac{1}{|C(\dot{\boldsymbol{y}})|} \sum_{\boldsymbol{a} \in C(\dot{\boldsymbol{y}})} h(\boldsymbol{a}, \dot{\boldsymbol{y}}) = \frac{1}{|\mathbb{B}_{n,m}|} \sum_{\dot{\boldsymbol{y}}} \sum_{\boldsymbol{a} \in C(\dot{\boldsymbol{y}})} h(\boldsymbol{a}, \dot{\boldsymbol{y}}) = \frac{1}{|\mathbb{B}_{n,m}|} \sum_{\boldsymbol{b} \in \mathbb{B}_{n,m}} h(\boldsymbol{b}, \boldsymbol{x}_0[\boldsymbol{b}]).$$

Therefore, averaging over all (i, k) while fixing $\boldsymbol{a} = \boldsymbol{b}$ yields another unbiased estimator. The samples are correlated, but the correlation does not introduce bias. Thus, we can equivalently use $(n-m)$ correlated samples

by fixing $\boldsymbol{a}$ to be the same for all samples and equal to $\boldsymbol{b}$, which gives the exact estimator

$$\begin{aligned}
&\frac{\sigma_t}{\bar{\sigma}_{0,t}} \mathop{\mathbb{E}}_{m,\boldsymbol{b}} \frac{(n-m)}{\tilde{S}_{n,m,\sigma_t}} \frac{1}{n-m} \sum_{i=1}^{m+1} \sum_{k \in s_i(\boldsymbol{b})} \sum_{w \in \mathbb{V}} \delta_w(\boldsymbol{x}_0^k) \log \left(\hat{K}_t(\boldsymbol{x}_0[\boldsymbol{b}], \boldsymbol{x}_0[\boldsymbol{b} \vee \boldsymbol{e}_k]) \right) \\
&\overset{(*)}{\approx} \frac{\sigma_t}{\bar{\sigma}_{0,t}} \mathop{\mathbb{E}}_{m,\boldsymbol{b}} \frac{(n-m)}{\tilde{S}_{n,m,\sigma_t}} \frac{1}{n-m} \sum_{i=1}^{m+1} \sum_{k \in s_i(\boldsymbol{b})} \sum_{w \in \mathbb{V}} \delta_w(\boldsymbol{x}_0^k) \log \left(\hat{p}^{\theta}_{\text{ins}}(i, x_0^k \mid \boldsymbol{x}_0[\boldsymbol{b}]) \right) \\
&= \frac{\sigma_t}{\bar{\sigma}_{0,t}} \mathop{\mathbb{E}}_{m,\boldsymbol{b}} \frac{(n-m)}{\tilde{S}_{n,m,\sigma_t}} \sum_{\substack{i \in [m+1], \\ w \in \mathbb{V}}} p^{\text{target}}_{\text{ins}}(i, w \mid \boldsymbol{x}_0, \boldsymbol{b}) \log \left(\hat{p}^{\theta}_{\text{ins}}(i, w \mid \boldsymbol{x}_0[\boldsymbol{b}]) \right). \qquad (15)
\end{aligned}$$

(*) Here, in the second step, we invoke the following approximation. We fix the alignment between the positions of the current sequence $\boldsymbol{x}_0[\boldsymbol{b}]$ and a predecessor state, and identify the predecessor state $\boldsymbol{x}_0[\boldsymbol{b} \vee \boldsymbol{e}_k]$ by the insertion location i and token x_0^k. Under this approximation, the reverse kernel $\hat{K}_t(\boldsymbol{x}_0[\boldsymbol{b}], \boldsymbol{x}_0[\boldsymbol{b} \vee \boldsymbol{e}_k])$ is replaced by the parameterized insertion probability $\hat{p}^{\theta}_{\text{ins}}(i, x_0^k \mid \boldsymbol{x}_0[\boldsymbol{b}])$. This is exact whenever the predecessor state determines a unique insertion location-token pair, and becomes approximate in the repeated-token case discussed below. If $\boldsymbol{x} = \text{aa}$ and $\boldsymbol{y} = \text{aaa}$, then the same predecessor state $\boldsymbol{y}$ can be obtained from $\boldsymbol{x}$ by inserting the token a at any of the three insertion locations. Therefore, $\hat{K}_t(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{3} \hat{p}^{\theta}_{\text{ins}}(i, \text{a} \mid \boldsymbol{x})$, and replacing $\log \left(\hat{K}_t(\boldsymbol{x}, \boldsymbol{y}) \right)$ with a single aligned term $\log \left(\hat{p}^{\theta}_{\text{ins}}(i, \text{a} \mid \boldsymbol{x}) \right)$ is an approximation in this case.

Finally, $p^{\text{target}}_{\text{ins}}(i, w \mid \boldsymbol{x}_0, \boldsymbol{b}) = \frac{1}{n-m} \sum_{k \in s_i(\boldsymbol{b})} \delta_w(\boldsymbol{x}_0^k)$ can be seen as the target joint probability distribution over positions and tokens. To see this, note that $\sum_{i=1}^{m+1} s_i(\boldsymbol{b}) = n - m$. Putting eq. (14) and eq. (15) together, we get the final expression. □

Remark 4 (Rao-Blackwellization). Instead of using the unbiased estimator obtained by fixing $\boldsymbol{a} = \boldsymbol{b}$, we could marginalize over all $\boldsymbol{a} \in \mathbb{B}_{n,m}(\boldsymbol{x}_0, \boldsymbol{b})$ explicitly. This can be viewed as Rao-Blackwellizing the latent alignment variable $\boldsymbol{a}$, yielding a lower-variance estimator of the ELBO term. However, this requires solving one dynamic programming instance per $(\boldsymbol{x}_0, \boldsymbol{b})$ pair to compute all possible alignments of the subsequence $\boldsymbol{x}_0[\boldsymbol{b}]$ and $\boldsymbol{x}_0$, which can introduce significant computational overhead during training. We leave an efficient implementation of this approach to future work.

Proposition 3 (Insertion rate from the length posterior)**.** *Fix t and a current state $\boldsymbol{x} = (m, \dot{\boldsymbol{x}})$. Let $L \coloneqq N_0 - N_t$ and define*

$$p_{t,\text{len}}(l \mid \boldsymbol{x}) \coloneqq \mathbb{P}(L = l \mid \boldsymbol{X}_t = \boldsymbol{x}).$$

Then the pointwise minimizer of the conditional length term in Corollary 1 is

$$\hat{\lambda}_t^{\star}(\boldsymbol{x}) = \frac{\sigma_t}{\bar{\sigma}_{0,t}} \mathbb{E}\left[\frac{L}{\tilde{S}_{m+L,m,\sigma_t}} \,\middle|\, \boldsymbol{X}_t = \boldsymbol{x} \right] = \frac{\sigma_t}{\bar{\sigma}_{0,t}} \sum_{l \geq 0} p_{t,\text{len}}(l \mid \boldsymbol{x}) \frac{l}{\tilde{S}_{m+l,m,\sigma_t}}.$$

Proof. By Equation (14), for a sample with current state $\boldsymbol{X}_t = \boldsymbol{x} = (m, \dot{\boldsymbol{x}})$ and initial length $N_0 = n$, the length contribution to the ELBO is

$$\hat{\lambda}_t(\boldsymbol{x}) - \sigma_t \frac{(n-m)}{\bar{\sigma}_{0,t}\, \tilde{S}_{n,m,\sigma_t}} \log \hat{\lambda}_t(\boldsymbol{x}).$$

On the event $\{\boldsymbol{X}_t = \boldsymbol{x}\}$ we have $N_t = m$, hence $n - m = L$ and $n = m + L$. Therefore,

$$\hat{\lambda}_t(\boldsymbol{x}) - \sigma_t \frac{(n-m)}{\bar{\sigma}_{0,t}\, \tilde{S}_{n,m,\sigma_t}} \log \hat{\lambda}_t(\boldsymbol{x}) = \hat{\lambda}_t(\boldsymbol{x}) - \frac{\sigma_t}{\bar{\sigma}_{0,t}} \frac{L}{\tilde{S}_{m+L,m,\sigma_t}} \log \hat{\lambda}_t(\boldsymbol{x}).$$

Taking the conditional expectation with respect to L given $\boldsymbol{X}_t = \boldsymbol{x}$ yields

$$\mathbb{E}\left[\hat{\lambda}_t(\boldsymbol{x}) - \frac{\sigma_t}{\bar{\sigma}_{0,t}} \frac{L}{\tilde{S}_{m+L,m,\sigma_t}} \log \hat{\lambda}_t(\boldsymbol{x}) \,\middle|\, \boldsymbol{X}_t = \boldsymbol{x} \right] = \hat{\lambda}_t(\boldsymbol{x}) - \frac{\sigma_t}{\bar{\sigma}_{0,t}} \mathbb{E}\left[\frac{L}{\tilde{S}_{m+L,m,\sigma_t}} \,\middle|\, \boldsymbol{X}_t = \boldsymbol{x} \right] \log \hat{\lambda}_t(\boldsymbol{x}).$$

Note that if $m > 0$, then $\tilde{S}_{m+L,m,\sigma_t} = 1$, so this reduces to

$$\hat{\lambda}_t(\boldsymbol{x}) - \frac{\sigma_t}{\bar{\sigma}_{0,t}} \mathbb{E}\left[L \mid \boldsymbol{X}_t = \boldsymbol{x}\right] \log \hat{\lambda}_t(\boldsymbol{x}).$$

More generally, let

$$c_t(\boldsymbol{x}) \coloneqq \frac{\sigma_t}{\bar{\sigma}_{0,t}} \mathbb{E}\left[\frac{L}{\tilde{S}_{m+L,m,\sigma_t}} \,\middle|\, \boldsymbol{X}_t = \boldsymbol{x}\right].$$

The conditional objective is $f(a) = a - c_t(\boldsymbol{x}) \log a$ for $a > 0$. Its derivative is $f'(a) = 1 - \frac{c_t(\boldsymbol{x})}{a}$ and $f''(a) = \frac{c_t(\boldsymbol{x})}{a^2} \geq 0$, so the unique minimizer for $c_t(\boldsymbol{x}) > 0$ is $a = c_t(\boldsymbol{x})$, and if $c_t(\boldsymbol{x}) = 0$ the minimum is attained at $a = 0$ by continuity. Therefore,

$$\hat{\lambda}_t^\star(\boldsymbol{x}) = c_t(\boldsymbol{x}) = \frac{\sigma_t}{\bar{\sigma}_{0,t}} \mathbb{E}\left[\frac{L}{\tilde{S}_{m+L,m,\sigma_t}} \,\middle|\, \boldsymbol{X}_t = \boldsymbol{x}\right].$$

Expanding the conditional expectation over the posterior of L gives

$$\hat{\lambda}_t^\star(\boldsymbol{x}) = \frac{\sigma_t}{\bar{\sigma}_{0,t}} \sum_{l \geq 0} p_{t,\text{len}}(l \mid \boldsymbol{x}) \frac{l}{\tilde{S}_{m+l,m,\sigma_t}}.$$

□

C.2 Joint Noising

Lemma 1 (Deletion DTMC)**.** *Let $\boldsymbol{X}_k$ be a discrete-time stochastic process taking values in $\mathbb{X}$ with one-step transition probabilities governed by the deletion kernel κ_{Uni}. Then, letting $\hat{C}_m^n \coloneqq \frac{(n-m)!}{n(n-1)\ldots(m+1)}$, the process $\underline{\boldsymbol{X}}_k$ is a DTMC with one-step transition kernel and transition probability given by*

$$K((n, \dot{\boldsymbol{x}}), (m, \dot{\boldsymbol{y}})) = \frac{1}{n} \sum_{\boldsymbol{b} \in \mathbb{B}_{n,m}} \delta_{n-1}(m)\, \delta_{\dot{\boldsymbol{x}}[\boldsymbol{b}]}(\dot{\boldsymbol{y}}),$$

$$\underline{p}_{k+r|k}\left((m, \dot{\boldsymbol{y}}) | (n, \dot{\boldsymbol{x}})\right) = \hat{C}_m^n\ \delta_r(n-m) \sum_{\boldsymbol{b} \in \mathbb{B}_{n,m}} \delta_{\dot{\boldsymbol{x}}[\boldsymbol{b}]}(\dot{\boldsymbol{y}}).$$

Remark 5. Let $\mathbf{1}_n$ be the vector of length n with all 1s. Assume that $\boldsymbol{z} = (l, \dot{\boldsymbol{z}})$ and $\boldsymbol{x} = (n, \dot{\boldsymbol{x}})$ such that $\boldsymbol{x} = \boldsymbol{z}[\boldsymbol{b}]$ for some $\boldsymbol{b} \in \mathbb{B}_{l,n}$ with $l \geq n$. Then there exists a bijection $\phi_{\boldsymbol{b}} : \mathbb{B}_{l,n-1} \to \mathbb{B}_{n,n-1}$ such that $\kappa_{\text{Uni}}(\boldsymbol{a} \mid \boldsymbol{b}) = \kappa_{\text{Uni}}(\phi_{\boldsymbol{b}}(\boldsymbol{a}) \mid \mathbf{1}_n)$, and $\boldsymbol{z}[\boldsymbol{a}] = \boldsymbol{x}[\phi_{\boldsymbol{b}}(\boldsymbol{a})]$ for all $\boldsymbol{a} \in \mathbb{B}_{l,n-1}$ with $\kappa_{\text{Uni}}(\boldsymbol{a} \mid \boldsymbol{b}) > 0$. This is all to say that at any point during the deletion process, we can freely replace the current sequence $(n, \dot{\boldsymbol{x}})$ with some longer sequence $(l, \dot{\boldsymbol{z}})$ and vice versa, such that $\boldsymbol{x} = \boldsymbol{z}[\boldsymbol{b}]$, and it will not change anything about the subsequent deletion process.

Now note that

$$\begin{aligned}
&\sum_{\boldsymbol{x} \in \mathbb{X}_n} \sum_{\boldsymbol{b} \in \mathbb{B}_{l,n}} \sum_{\boldsymbol{c} \in \mathbb{B}_{n,n-1}} \delta_{\dot{\boldsymbol{z}}[\boldsymbol{b}]}(\dot{\boldsymbol{x}})\ \delta_{\dot{\boldsymbol{x}}[\boldsymbol{c}]}(\dot{\boldsymbol{y}}) \\
&\overset{(a)}{=} \sum_{\boldsymbol{b} \in \mathbb{B}_{l,n}} \sum_{\boldsymbol{c} \in \mathbb{B}_{n,n-1}} \delta_{\dot{\boldsymbol{z}}[\boldsymbol{b}][\boldsymbol{c}]}(\dot{\boldsymbol{y}}) \\
&\overset{(b)}{=} \sum_{\boldsymbol{b} \in \mathbb{B}_{l,n}} \sum_{\boldsymbol{a} \in \mathbb{A}_{l,n-1,\boldsymbol{b}}} \delta_{\dot{\boldsymbol{z}}[\boldsymbol{a}]}(\dot{\boldsymbol{y}}) \\
&= \sum_{\boldsymbol{a} \in \mathbb{B}_{l,n-1}} (l - n + 1) \delta_{\dot{\boldsymbol{z}}[\boldsymbol{a}]}(\dot{\boldsymbol{y}})
\end{aligned} \tag{16}$$

Here, (a) follows from the fact that for a specific $\boldsymbol{b}$, there exists a unique $\boldsymbol{x}$ such that $\boldsymbol{x} = \boldsymbol{z}[\boldsymbol{b}]$. Therefore, we replace $\boldsymbol{x}$ with $\boldsymbol{z}[\boldsymbol{b}]$. In (b), $\mathbb{A}_{l,n-1,\boldsymbol{b}} = \{\boldsymbol{a} \in \mathbb{B}_{l,n-1} \mid \kappa_{\text{Uni}}(\boldsymbol{a} \mid \boldsymbol{b}) > 0\}$. In the final step, we get rid of the sum over $\boldsymbol{b}$ by first noting that $\bigcup_{\boldsymbol{b} \in \mathbb{B}_{l,n}} \mathbb{A}_{l,n-1,\boldsymbol{b}} = \mathbb{B}_{l,n-1}$, i.e., while going over the double sum, every $\boldsymbol{a} \in \mathbb{B}_{l,n-1}$ appears at least once. Using this we flip the order of the summation and note that for a specific $\boldsymbol{a} \in \mathbb{B}_{l,n-1}$, there will be $l - (n-1)$ choices for $\boldsymbol{b}$ such that $\kappa_{\text{Uni}}(\boldsymbol{a} \mid \boldsymbol{b}) > 0$ because there are $l - (n-1)$ positions with zero in $\boldsymbol{a}$ and exactly one of them has to become 1 to obtain a valid $\boldsymbol{b}$.

Proof. The transition kernel is given by

$$\begin{aligned}
K((n,\dot{\boldsymbol{x}}),(m,\dot{\boldsymbol{y}})) &\coloneqq \mathbb{P}(\underline{\boldsymbol{X}}_{k+1} = (m,\dot{\boldsymbol{y}}) \mid \underline{\boldsymbol{X}}_k = (n,\dot{\boldsymbol{x}})) \\
&= \sum_{\boldsymbol{b}\in\mathbb{B}_n} \kappa_{\text{Uni}}(\boldsymbol{b} \mid \mathbf{1}_n)\ \delta_{\dot{\boldsymbol{x}}[\boldsymbol{b}]}(\dot{\boldsymbol{y}}) \\
&= \sum_{\boldsymbol{b}\in\mathbb{B}_n} \frac{1}{|\mathbf{1}_n|}\delta_{\{\mathbf{1}_n \succ \cdot\}}(\boldsymbol{b})\,\delta_{|\mathbf{1}_n|-1}(|\boldsymbol{b}|)\ \delta_{\dot{\boldsymbol{x}}[\boldsymbol{b}]}(\dot{\boldsymbol{y}}) \quad \text{(by definition of } \kappa_{\text{Uni}}) \\
&= \sum_{\boldsymbol{b}\in\mathbb{B}_{n,m}} \frac{1}{n}\,\delta_{n-1}(m)\,\delta_{\dot{\boldsymbol{x}}[\boldsymbol{b}]}(\dot{\boldsymbol{y}}) \\
&= \sum_{i=1}^{n} \frac{1}{n}\,\delta_{\text{del}(\boldsymbol{x},i)}(\boldsymbol{y})
\end{aligned}$$

The process is clearly Markovian because the transition probabilities only depend on the current state and any history does not change the one-step transition probability, and therefore any future transition probabilities.

To get the r-step transition probability we can use the Chapman-Kolmogorov equation, which in the case of a DTMC simply means that we take the product of the one-step transition probabilities and marginalize over the intermediate states.

Denoting $\boldsymbol{z}_0 = \boldsymbol{x} = (n,\dot{\boldsymbol{x}})$, $\boldsymbol{z}_r = \boldsymbol{y} = (m,\dot{\boldsymbol{y}})$, and $\boldsymbol{b}_0 = \mathbf{1}_n$, and using the expression for K from above, we get the following expression for one step of marginalization.

$$\begin{aligned}
&\sum_{\boldsymbol{z}_1} K(\boldsymbol{z}_0,\boldsymbol{z}_1)K(\boldsymbol{z}_1,\boldsymbol{z}_2) \\
&= \sum_{\boldsymbol{z}_1} \left[\sum_{\boldsymbol{b}_1\in\mathbb{B}_{n_0,n_1}} \frac{1}{n_0}\,\delta_{n_0-1}(n_1)\,\delta_{\dot{\boldsymbol{x}}[\boldsymbol{b}_1]}(\dot{\boldsymbol{z}}_1)\right]\left[\sum_{\boldsymbol{b}_2\in\mathbb{B}_{n_1,n_2}} \frac{1}{n_1}\,\delta_{n_1-1}(n_2)\,\delta_{\dot{\boldsymbol{z}}_1[\boldsymbol{b}_2]}(\dot{\boldsymbol{z}}_2)\right].
\end{aligned}$$

As in Remark 5, we note that for a specific $\boldsymbol{b}_1$, there exists a unique $\boldsymbol{z}_1$ such that $\boldsymbol{z}_1 = \boldsymbol{x}[\boldsymbol{b}_1]$. Therefore, we can replace $\boldsymbol{z}_1$ with $\boldsymbol{x}[\boldsymbol{b}_1]$ in the above expression and get rid of the sum over $\boldsymbol{z}_1$. We can also apply the conditions on n_1 and n_2 to write them in terms of n_0.

$$\begin{aligned}
&= \sum_{\boldsymbol{b}_1\in\mathbb{B}_{n_0,n_0-1}} \frac{1}{n_0} \sum_{\boldsymbol{b}_2\in\mathbb{B}_{n_0-1,\,n_0-2}} \frac{1}{n_0-1}\delta_{(\dot{\boldsymbol{z}}_0[\boldsymbol{b}_1])[\boldsymbol{b}_2]}(\dot{\boldsymbol{z}}_2) \\
&= \sum_{\boldsymbol{b}\in\mathbb{B}_{n,n-2}} \frac{1\cdot 2}{n\,(n-1)}\delta_{\dot{\boldsymbol{x}}[\boldsymbol{b}]}(\dot{\boldsymbol{z}}_2) \qquad (17)
\end{aligned}$$

In the final step, we removed the summation over $\boldsymbol{b}_1$ using the same argument as we used in Equation (16). We also renamed $\boldsymbol{b}_2$ to $\boldsymbol{b}$ to simplify the notation, and used the fact that $n_0 = n$ and $\boldsymbol{x} = \boldsymbol{z}_0$. We can now repeat this procedure $r-1$ times to get the following expression for the r-step transition probability, which concludes the proof.

$$\begin{aligned}
&\mathbb{P}(\underline{\boldsymbol{X}}_{k+r} = (m,\dot{\boldsymbol{y}}) \mid \underline{\boldsymbol{X}}_k = (n,\dot{\boldsymbol{x}})) \\
&= \sum_{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_{r-1}} \prod_{i=1}^{r} K(\boldsymbol{z}_{i-1},\boldsymbol{z}_i) \\
&= \sum_{\boldsymbol{z}_{r-1}} K(\boldsymbol{z}_{r-1},\boldsymbol{z}_r) \sum_{\boldsymbol{z}_{r-2}} \cdots \sum_{\boldsymbol{z}_2} K(\boldsymbol{z}_2,\boldsymbol{z}_3) \sum_{\boldsymbol{z}_1} K(\boldsymbol{z}_0,\boldsymbol{z}_1)K(\boldsymbol{z}_1,\boldsymbol{z}_2) \\
&= \sum_{\boldsymbol{b}\in\mathbb{B}_{n,m}} \frac{r!}{n\,(n-1)\cdots(n-(r-1))}\delta_{\dot{\boldsymbol{x}}[\boldsymbol{b}]}(\dot{\boldsymbol{y}})\delta_r(n-m)
\end{aligned}$$

□

Proposition 2 (Transition Probability)**.** *Let $\boldsymbol{X}_t$ be a continuous-time stochastic process described in equation 1, with transition rate $\lambda_t((r, \dot{\boldsymbol{x}})) = \sigma_t \, \delta_{\{\cdot>0\}}(r)$ and transitions governed by the transition kernel K that drops one token at a time uniformly at random. Then $\boldsymbol{X}_t$ is an inhomogeneous CTMC with transition probability for $s \le t$ given by*

$$p^{\text{joint}}_{t|s}\left((m, \dot{\boldsymbol{y}}) \mid (n, \dot{\boldsymbol{x}})\right) = \begin{cases} \frac{\exp(-\bar{\sigma}_{s,t})(\bar{\sigma}_{s,t})^{n-m}\ m!}{n!} \sum_{b \in \mathbb{B}_{n,m}} \delta_{\dot{\boldsymbol{x}}[\boldsymbol{b}]}(\dot{\boldsymbol{y}}), & 0 < m \le n \\ 1 - \sum_{r=1}^{n} \frac{\exp(-\bar{\sigma}_{s,t})(\bar{\sigma}_{s,t})^{n-r}\ r!}{n!}, & m = 0 \end{cases}$$

where $\sigma : [0, T] \to \mathbb{R}_+$ is a scalar noise schedule, $\bar{\sigma}_{s,t} = \int_s^t \sigma(u)\, du$.

Proof. An equivalent way to represent the continuous-time and discrete-time noising processes is $\boldsymbol{X}_t = (N_t, \dot{\boldsymbol{X}}_t)$, and $\underset{\sim}{\boldsymbol{X}}_k = (\underset{\sim}{N}_k, \underset{\sim}{\dot{\boldsymbol{X}}}_k)$, respectively, where N_t and $\underset{\sim}{N}_k$ are the lengths of the sequences at time t and k, respectively. Using this, we can write the transition probability as

$$\begin{aligned} &\mathbb{P}(\boldsymbol{X}_t = (m, \dot{\boldsymbol{y}}) \mid \boldsymbol{X}_s = (n, \dot{\boldsymbol{x}})) \\ &= \mathbb{P}(N_t = m, \dot{\boldsymbol{X}}_t = \dot{\boldsymbol{y}} \mid N_s = n, \dot{\boldsymbol{X}}_s = \dot{\boldsymbol{x}}) \\ &= \mathbb{P}(\dot{\boldsymbol{X}}_t = \dot{\boldsymbol{y}} \mid \dot{\boldsymbol{X}}_s = \dot{\boldsymbol{x}}, N_s = n, N_t = m)\mathbb{P}(N_t = m \mid N_s = n, \dot{\boldsymbol{X}}_s = \dot{\boldsymbol{x}}) \\ &\overset{(a)}{=} \mathbb{P}(\underset{\sim}{\dot{\boldsymbol{X}}}_{n-m} = \dot{\boldsymbol{y}} \mid \underset{\sim}{\dot{\boldsymbol{X}}}_0 = \dot{\boldsymbol{x}}, \underset{\sim}{N}_0 = n)\mathbb{P}(N_t = m \mid N_s = n) \\ &\overset{(b)}{=} \mathbb{P}(\underset{\sim}{\boldsymbol{X}}_{n-m} = (m, \dot{\boldsymbol{y}}) \mid \underset{\sim}{\boldsymbol{X}}_0 = (n, \dot{\boldsymbol{x}}))\mathbb{P}(N_t = m \mid N_s = n) \end{aligned} \tag{18}$$

Here, in (a), $\mathbb{P}(\dot{\boldsymbol{X}}_t = \dot{\boldsymbol{y}} \mid \dot{\boldsymbol{X}}_s = \dot{\boldsymbol{x}}, N_s = n, N_t = m) = \mathbb{P}(\underset{\sim}{\dot{\boldsymbol{X}}}_{n-m} = \dot{\boldsymbol{y}} \mid \underset{\sim}{\dot{\boldsymbol{X}}}_0 = \dot{\boldsymbol{x}}, \underset{\sim}{N}_0 = n)$ because the embedded DTMC is time-homogeneous, and given the initial and final lengths, the CTMC only depends on the transitions of the embedded DTMC. Now note that deletions arrive with rate $\lambda_t((r, \dot{\boldsymbol{x}})) = \sigma_t \delta\{r > 0\}$, and therefore the number of deletions until there is nothing left to delete has a Poisson distribution with rate $\bar{\sigma}_{s,t} = \int_s^t \sigma(u)\, du$. The general expression for $\mathbb{P}(N_t = m \mid N_s = n)$ can be obtained by creating a special case for the absorbing state $m = 0$ and noting that the total probability is 1:

$$\mathbb{P}(N_t = m \mid N_s = n) = \begin{cases} \frac{e^{-\bar{\sigma}_{s,t}}(\bar{\sigma}_{s,t})^{n-m}}{(n-m)!} & \text{if } 0 < m \le n \\ 1 - \sum_{k=1}^{n} \frac{e^{-\bar{\sigma}_{s,t}}(\bar{\sigma}_{s,t})^{n-k}}{(n-k)!} & \text{if } m = 0. \end{cases} \tag{19}$$

We obtain the transition probability for the CTMC by substituting the expression for $\mathbb{P}(\underset{\sim}{\boldsymbol{X}}_{n-m} = (m, \dot{\boldsymbol{y}}) \mid \underset{\sim}{\boldsymbol{X}}_0 = (n, \dot{\boldsymbol{x}}))$ from Lemma 1, and for $\mathbb{P}(N_t = m \mid N_s = n)$ from Equation (19), in Equation (18). □

C.3 Independent Noising

We can also obtain uniformly random deletion of positions by giving each position an independent deletion clock with rate σ_t. So the probability that a position is *retained* at time t given it was present at time s is $\rho_{s,t} \coloneqq \exp(-\bar{\sigma}_{s,t})$, with $\bar{\sigma}_{s,t} = \int_s^t \sigma(u)\ du$. At time t the number of retained positions is therefore Binomial$(n, \rho_{s,t})$, and conditional on length m, the mask is uniform over $\mathbb{B}_{n,m}$.

Proposition 4 (Transition probability for Independent Noising)**.** *Let $\boldsymbol{X}_t$ be the continuous-time process where each of the n positions is deleted independently with rate σ_t. Then for $s \le t$,*

$$p_{t|s}\big((m, \dot{\boldsymbol{y}}) \mid (n, \dot{\boldsymbol{x}})\big) = \binom{n}{m} \rho_{s,t}^m (1 - \rho_{s,t})^{n-m} \frac{1}{\binom{n}{m}} \sum_{\boldsymbol{b} \in \mathbb{B}_{n,m}} \delta_{\dot{\boldsymbol{x}}[\boldsymbol{b}]}(\dot{\boldsymbol{y}}) = \rho_{s,t}^m (1 - \rho_{s,t})^{n-m} \sum_{\boldsymbol{b} \in \mathbb{B}_{n,m}} \delta_{\dot{\boldsymbol{x}}[\boldsymbol{b}]}(\dot{\boldsymbol{y}}),$$

for $0 \le m \le n$, with $\rho_{s,t} = \exp(-\bar{\sigma}_{s,t})$. The empty sequence $(0, \emptyset)$ has mass $(1 - \rho_{s,t})^n$.

Proof. For each position $i \in [n]$, let $b^i \in \{0, 1\}$ indicate whether that position is retained at time t given it was present at time s. Since the deletion clock at each position has rate σ_t, the survival probability over $[s, t]$ is

$$\mathbb{P}(b^i = 1) = \rho_{s,t} = \exp(-\bar{\sigma}_{s,t}),$$

and the indicators are independent across positions. Hence the retention mask $\boldsymbol{b} = (b^1, \ldots, b^n)$ satisfies

$$\mathbb{P}(\boldsymbol{b}) = \rho_{s,t}^{|\boldsymbol{b}|}(1-\rho_{s,t})^{n-|\boldsymbol{b}|}.$$

Therefore, for any $\dot{\boldsymbol{y}}$ of length m,

$$\begin{aligned}
&p_{t|s}\big((m, \dot{\boldsymbol{y}}) \mid (n, \dot{\boldsymbol{x}})\big) \\
&= \sum_{\boldsymbol{b}\in\mathbb{B}_n} \mathbb{P}(\boldsymbol{b})\, \delta_{|\boldsymbol{b}|}(m)\, \delta_{\dot{\boldsymbol{x}}[\boldsymbol{b}]}(\dot{\boldsymbol{y}}) \\
&= \sum_{\boldsymbol{b}\in\mathbb{B}_{n,m}} \rho_{s,t}^m(1-\rho_{s,t})^{n-m}\, \delta_{\dot{\boldsymbol{x}}[\boldsymbol{b}]}(\dot{\boldsymbol{y}}) \\
&= \rho_{s,t}^m(1-\rho_{s,t})^{n-m} \sum_{\boldsymbol{b}\in\mathbb{B}_{n,m}} \delta_{\dot{\boldsymbol{x}}[\boldsymbol{b}]}(\dot{\boldsymbol{y}}).
\end{aligned}$$

Equivalently, since every mask in $\mathbb{B}_{n,m}$ has the same probability, the number of retained positions is Binomial$(n, \rho_{s,t})$, and conditional on $|\boldsymbol{b}| = m$, the mask is uniform over $\mathbb{B}_{n,m}$, which gives the first factorization in the proposition.

□

Corollary 3 (Independent Noising)**.** *A (biased) estimate of the upper bound on the negative log-likelihood in Proposition 1 for the independent per-position parameterization of $\hat{R}_t^\theta$, when the noising process is per-position deletion (Proposition 4), is given by*

$$\begin{aligned}
&\mathcal{L}(\theta) = \mathcal{L}_{\text{len}}(\theta) + \mathcal{L}_{\text{token}}(\theta), \text{ where} \\
&\mathcal{L}^{\text{len}}(\theta) = \mathbb{E}\left[\sum_{i=1}^{m+1} \hat{\lambda}_t^\theta(\boldsymbol{x}_0[\boldsymbol{b}], i) - \gamma_t^{ind}|s_i(\boldsymbol{b})| \log \hat{\lambda}_t^\theta(\boldsymbol{x}_0[\boldsymbol{b}], i)\right], \\
&\mathcal{L}^{\text{token}}(\theta) = -\,\mathbb{E}\left[\gamma_t^{ind} \sum_{i=1}^{m+1} \sum_{k\in s_i(\boldsymbol{b})} \log \hat{p}_t^\theta(x_0^k \mid \boldsymbol{x}_0[\boldsymbol{b}], i)\right].
\end{aligned}$$

Here, the expectation is over $\boldsymbol{x}_0 \sim p_{\text{data}}$, $t \sim$ Uniform$[0,T]$, $m \sim$ Binomial(n, ρ_t), $\boldsymbol{b} \sim$ Uniform$(\mathbb{B}_{n,m})$, $\gamma_t^{ind} = \frac{\sigma_t\,\rho_t}{1-\rho_t}$ with $\rho_t = \exp(-\bar{\sigma}_{0,t})$, $\boldsymbol{x}_0 = (n, \dot{\boldsymbol{x}}_0)$, and $s_i(\boldsymbol{b})$ is the set of indices of 0s in $\boldsymbol{b}$ that fall between the $(i-1)$-th and i-th 1s.

Proof. For $\boldsymbol{x} = (m, \dot{\boldsymbol{x}})$, write

$$\hat{R}_t^\theta(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{m+1} \sum_{w\in\mathbb{V}} \hat{\lambda}_t^\theta(\boldsymbol{x}, i)\, \hat{p}_t^\theta(w \mid \boldsymbol{x}, i)\ \delta_{\text{ins}(\boldsymbol{x},i,w)}(\boldsymbol{y}).$$

Fix $\boldsymbol{x}_0 = (n, \dot{\boldsymbol{x}}_0)$ and $\boldsymbol{x} = \boldsymbol{x}_0[\boldsymbol{b}]$. As in the proof of Corollary 1, we plug this factorization into Proposition 1 and separate the contribution into the negative rate term and the predecessor-weighted log term.

The negative term is exact:

$$\sum_{\boldsymbol{y}\neq\boldsymbol{x}} \hat{R}_t^\theta(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{m+1} \sum_{w\in\mathbb{V}} \hat{\lambda}_t^\theta(\boldsymbol{x}, i)\hat{p}_t^\theta(w \mid \boldsymbol{x}, i) = \sum_{i=1}^{m+1} \hat{\lambda}_t^\theta(\boldsymbol{x}, i).$$

For the log term, if $\boldsymbol{y} = \text{ins}(\boldsymbol{x}, i, x_0^k)$ for some $k \in s_i(\boldsymbol{b})$, then Proposition 4 gives

$$\frac{p_{t|0}(\boldsymbol{y} \mid \boldsymbol{x}_0)}{p_{t|0}(\boldsymbol{x} \mid \boldsymbol{x}_0)} = \frac{\rho_t^{m+1}(1-\rho_t)^{n-m-1}}{\rho_t^m(1-\rho_t)^{n-m}} = \frac{\rho_t}{1-\rho_t},$$

and the forward deletion rate from $\boldsymbol{y}$ to $\boldsymbol{x}$ is σ_t. Hence every missing original index contributes the same coefficient

$$\gamma_t^{\text{ind}} = \frac{\sigma_t\,\rho_t}{1-\rho_t}.$$

As in Corollary 1, we now use the same approximation for the log term: for each gap i, we pair the deleted indices $k \in s_i(\boldsymbol{b})$ with the factorized reverse rate and replace

$$\log \hat{R}_t^\theta(\boldsymbol{x}_0[\boldsymbol{b}], \mathrm{ins}(\boldsymbol{x}_0[\boldsymbol{b}], i, x_0^k))$$

by

$$\log \hat{\lambda}_t^\theta(\boldsymbol{x}_0[\boldsymbol{b}], i) + \log \hat{p}_t^\theta(x_0^k \mid \boldsymbol{x}_0[\boldsymbol{b}], i).$$

With this approximation, the contribution to the upper bound on the negative log-likelihood becomes

$$\mathbb{E}\left[\sum_{i=1}^{m+1} \hat{\lambda}_t^\theta(\boldsymbol{x}_0[\boldsymbol{b}], i) - \gamma_t^{\mathrm{ind}} \sum_{i=1}^{m+1} |s_i(\boldsymbol{b})| \log \hat{\lambda}_t^\theta(\boldsymbol{x}_0[\boldsymbol{b}], i) \right.$$
$$\left. -\gamma_t^{\mathrm{ind}} \sum_{i=1}^{m+1} \sum_{k \in s_i(\boldsymbol{b})} \log \hat{p}_t^\theta(x_0^k \mid \boldsymbol{x}_0[\boldsymbol{b}], i)\right],$$

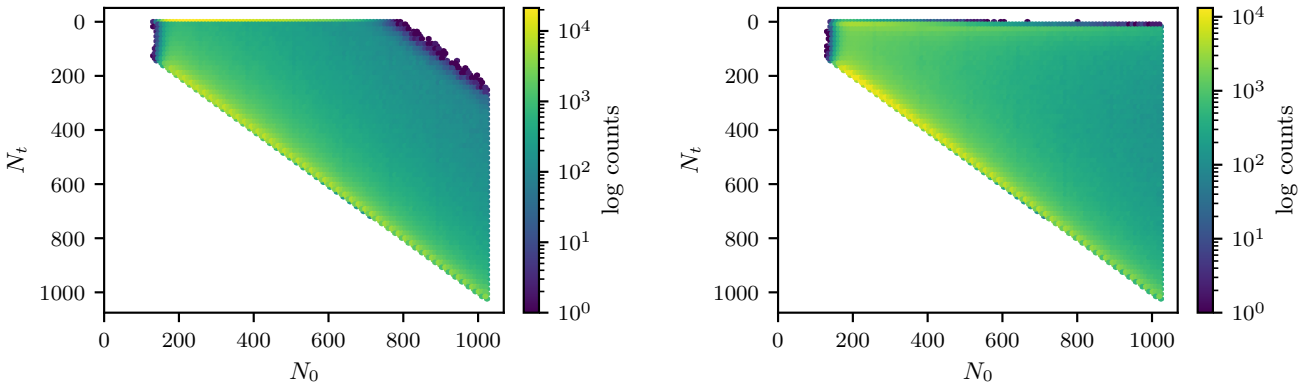
where the expectation is over $\boldsymbol{x}_0 \sim p_{\mathrm{data}}$, $t \sim \mathrm{Uniform}[0, T]$, $m \sim \mathrm{Binomial}(n, \rho_t)$, and $\boldsymbol{b} \sim \mathrm{Uniform}(\mathbb{B}_{n,m})$. Grouping the first two terms as $\mathcal{L}_{\mathrm{len}}(\theta)$ and the last term as $\mathcal{L}_{\mathrm{token}}(\theta)$ gives the stated result. □

D Extended Discussion on Related Work

Relation to discrete flow matching and stochastic interpolants.. Our work is closely related in spirit to recent discrete flow matching and stochastic interpolant approaches for sequence generation (Campbell et al., 2024; Albergo et al., 2023). In those frameworks, one typically specifies data-conditional target rates $R_t(x_t, \cdot \mid x_0)$ directly and trains a model $\hat{R}_t^\theta(x_t, \cdot)$ to match them, often using a Bregman divergence. In contrast, our starting point is an explicit diffusion-style noising process: a deletion CTMC on variable-length sequences whose reverse-time dynamics define the generative insertion process. Conceptually, the target-rate construction in flow matching plays a role analogous to the choice of noising process in diffusion, but the modeling perspective is different: in our case the reverse process is obtained from a concrete forward deletion mechanism, rather than being specified directly.

Comparison with Edit Flows.. Havasi et al. (2025) also study continuous-time generation on discrete sequences, but they do so through conditional discrete flow matching with *edit* operations. The high-level similarity is that both approaches use Markovian sequence evolution in continuous time and both can be trained with closely related rate-matching objectives. The main difference is that our construction is tailored to insertion-only generation. We begin with a deletion process that acts directly on complete sequences and then derive the reverse insertion model, whereas Edit Flows directly parameterizes a generative chain over a broader class of edit operations: insertions, deletions, and substitutions. The presence of deletion operations in the generative process necessitates that the noising process insert spurious tokens thereby making the explicit expectation over the alignments completely intractable. Under an insertion-only restriction our formulation provides a way to perform Rao-Blackwellized estimation of the ELBO objective using dynamic programming as stated in remark 4. If the Edit Flows operations were restricted to only insertions, and the alignments were sampled as we currently do in our implementation, the Bregman divergence for insertion-only rate matching, after ignoring the constant terms, would produce a training objective similar to the one given in our corollary 3.

Comparison with FlexMDM.. FlexMDM (Kim et al., 2025) is a concurrent work that also generates sequences by growing a partial sequence over time using insertions. The main distinction between DILM and FlexMDM lies in the choice of *what* is inserted and *how* the forward process is defined. In our model, a generative step inserts a vocabulary token directly into a gap. In FlexMDM, a generative step inserts a fresh `MASK` token into a gap and separately allows already-present `MASK` tokens to be unmasked. Thus both methods are insertion-based, but ours chooses token identity at insertion time, whereas FlexMDM defers that decision to a later unmasking step.



(a) Joint Poisson noising.

(b) Independent noising.

Figure 6: Empirical joint distributions of initial length N_0 and noised length N_t on OpenWebText training documents. For each sequence we draw t from the same uniform distribution on $[\varepsilon, 1]$ used in training ($\varepsilon = 10^{-4}$). In Figure 6a, we sample $D \sim \text{Poisson}(\bar{\sigma}_{0,t})$ with the log-linear schedule $(\sigma_{\min}, \sigma_{\max}) = (50, 3000)$ and set $N_t = \max(N_0 - D, 0)$. In Figure 6b, we sample $D \sim \text{Binomial}(N_0, 1 - \exp(-\bar{\sigma}_{0,t}))$ with $(\sigma_{\min}, \sigma_{\max}) = (0.1, 20)$ and set $N_t = N_0 - D$. Both panels show hexagonal-bin densities with log-scaled counts (darker is more mass). Joint noising yields additive shrinkage, with visible mass near $N_t = 0$ for short inputs, whereas independent noising is closer to multiplicative shrinkage, so the typical noised length tracks the original length more closely.

E Implementation Details

E.1 Choosing the Noise Schedule

Unlike fixed-length MDMs, where the noise schedule is not as critical (Zheng et al., 2025; Ou et al., 2024), picking the right noise schedule is important for variable-length generation. We want the generator to generalize across different sequence lengths, and therefore we need to pick a noise schedule that provides sufficient coverage of all final lengths in the noised sequences.

E.1.1 Joint Noising

We use the *log-linear* schedule for joint noising defined as

$$\sigma(t) = \sigma_{\min} \left(\frac{\sigma_{\max}}{\sigma_{\min}} \right)^t ,$$

where $0 < \sigma_{\min} < \sigma_{\max}$ are hyperparameters (in particular $\sigma(0) = \sigma_{\min}$ and $\sigma(1) = \sigma_{\max}$). The corresponding integrated rate from proposition 2 is

$$\bar{\sigma}_{0,t} = \int_0^t \sigma(u)\, du = \frac{\sigma_{\min}}{\ln(\sigma_{\max}/\sigma_{\min})} \left(\left(\frac{\sigma_{\max}}{\sigma_{\min}} \right)^t - 1 \right).$$

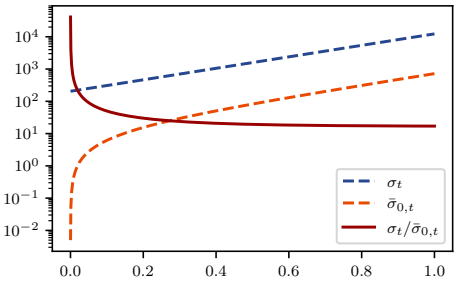


Figure 5: Log-linear noise schedule.

During training we sample t uniformly from $[\varepsilon, 1]$, with ε small (e.g. $\varepsilon = 10^{-4}$ in the configuration). The log-linear schedule has the desirable property that the process slows down near $t = 0$. We did not explore other schedules in our experiments.

Under joint Poisson noising (algorithm 1), let N_t denote the length at time t given $N_0 = n$. Equivalently, sample $D \sim \text{Poisson}(\bar{\sigma}_{0,t})$ for the number of positions to drop and set $N_t = \max(n - D, 0)$ (after the min with n in the

algorithm). Then for $m \in \{0, 1, \ldots, n\}$,

$$\mathbb{P}(N_t = m \mid N_0 = n) = \begin{cases} 1 - \sum_{k=0}^{n-1} \frac{e^{-\bar{\sigma}_{0,t}}\, \bar{\sigma}_{0,t}^{k}}{k!}, & m = 0, \\ \frac{e^{-\bar{\sigma}_{0,t}}\, \bar{\sigma}_{0,t}^{n-m}}{(n-m)!}, & m \in \{1, \ldots, n\}. \end{cases}$$

Its conditional expectation is

$$\mathbb{E}[N_t \mid N_0 = n] = a_n(\bar{\sigma}_{0,t})\, n - b_n(\bar{\sigma}_{0,t})\, \bar{\sigma}_{0,t},$$

where

$$a_n(\lambda) = \sum_{k=0}^{n-1} \frac{e^{-\lambda}\lambda^k}{k!}, \qquad b_n(\lambda) = \sum_{k=0}^{n-2} \frac{e^{-\lambda}\lambda^k}{k!}.$$

Thus joint noising is additive in the sense that the expected length is a linear combination of the original length n and the integrated rate $\bar{\sigma}_{0,t}$, with truncation-dependent coefficients. When truncation is negligible, $a_n(\bar{\sigma}_{0,t}) \approx b_n(\bar{\sigma}_{0,t}) \approx 1$, recovering the heuristic scale $n - \bar{\sigma}_{0,t}$.

Figure 6a illustrates how these choices interact with the heavy-tailed length distribution of OpenWebText: the model trains on a wide range of (N_0, N_t) pairs rather than a single fixed shrinkage factor, which motivates tuning the schedule so that typical noised lengths still cover the range needed at generation time.

E.1.2 Independent Noising

Under independent noising, each position survives until time t with probability

$$\rho_{0,t} = \exp(-\bar{\sigma}_{0,t}),$$

so Algorithm 1 samples

$$D \sim \text{Binomial}(n, 1 - \rho_{0,t}),$$

and returns $N_t = n - D$. Equivalently, conditional on $N_0 = n$, the noised length has marginal

$$\mathbb{P}(N_t = m \mid N_0 = n) = \binom{n}{m} \rho_{0,t}^{m} \left(1 - \rho_{0,t}\right)^{n-m}, \qquad m \in \{0, 1, \ldots, n\}.$$

Hence

$$\mathbb{E}[N_t \mid N_0 = n] = n\rho_{0,t},$$

so independent noising induces multiplicative shrinkage of sequence length. This contrasts with joint Poisson noising, where the number of deletions is additive and, before truncation at zero, independent of n. Figure 6b shows the resulting behavior on OpenWebText: the noised lengths remain broadly distributed, but their typical scale tracks the original length more closely than in the joint-noising case.

E.2 Computing the diffusion coefficients

Computing the diffusion coefficient γ_t^{ind} in the independent noising case is straightforward, but this is not the case for the joint noising case. Here we provide details on how to numerically compute the diffusion coefficient for the joint noising case. For the joint noising case, with $t > s$, and $m > 0$, from equation 19 we have

$$\frac{\mathbb{P}(N_t = m+1 \mid N_s = n)}{\mathbb{P}(N_t = m \mid N_s = n)} = \frac{e^{-\bar{\sigma}_{s,t}} (\bar{\sigma}_{s,t})^{n-m-1}}{(n-m-1)!} \frac{(n-m)!}{e^{-\bar{\sigma}_{s,t}} (\bar{\sigma}_{s,t})^{n-m}} = \frac{n-m}{\bar{\sigma}_{s,t}}.$$

For the case of $m = 0$, we need to compute the tail of the exponential for the denominator.

$$\begin{aligned}\frac{\mathbb{P}(N_t = m+1 \mid N_s = n)}{\mathbb{P}(N_t = m \mid N_s = n)} &= \frac{\mathbb{P}(N_t = 1 \mid N_s = n)}{\mathbb{P}(N_t = 0 \mid N_s = n)} \\ &= \frac{e^{-\bar{\sigma}_{s,t}}(\bar{\sigma}_{s,t})^{n-1}}{(n-1)!} \frac{1}{1 - \sum_{k=0}^{n-1} \frac{e^{-\bar{\sigma}_{s,t}}(\bar{\sigma}_{s,t})^k}{k!}} \\ &= \frac{e^{-\bar{\sigma}_{s,t}}(\bar{\sigma}_{s,t})^{n-1}}{(n-1)!} \frac{1}{\Phi(n, \bar{\sigma}_{s,t})} \\ &= \frac{n}{\bar{\sigma}_{s,t}} \frac{1}{S(n, \bar{\sigma}_{s,t})}\end{aligned}$$

where $S(n, \bar{\sigma}_{s,t}) = \frac{n!\, e^{\bar{\sigma}_{s,t}}}{(\bar{\sigma}_{s,t})^n} \Phi(n, \bar{\sigma}_{s,t})$ and $\Phi(n, \bar{\sigma}_{s,t}) = (1 - \frac{\Gamma(n, \bar{\sigma}_{s,t})}{\Gamma(n)})$ is the regularized lower incomplete gamma function. We vectorize the computation in PyTorch in a numerically stable manner using Algorithm 4, where lgamma and gammainc are `torch.special.lgamma` and `torch.special.gammainc`, respectively.

Algorithm 4 Computing $\frac{\mathbb{P}(N_t = m+1 | N_s = n)}{\mathbb{P}(N_t = m | N_s = n)}$

Require: $n, m, \bar{\sigma}_{s,t}$
$u \leftarrow \frac{n-m}{\bar{\sigma}_{s,t}}$
if $m = 0$ **then**
  $S \leftarrow \exp\left[\text{lgamma}(n+1) + \bar{\sigma}_{s,t} - n \log \bar{\sigma}_{s,t} + \text{gammainc}(n, \bar{\sigma}_{s,t})\right]$
  $u \leftarrow \frac{u}{S}$
end if
return u

For large values of n, the regularized lower incomplete gamma function can quickly approach values less than 10^{-50}, which leads to numerical instability. Another alternative is to use the confluent hypergeometric function.

$$\begin{aligned}S(n, \bar{\sigma}) &= \frac{n!\, e^{\bar{\sigma}}}{\bar{\sigma}^n} \Phi(n, \bar{\sigma}) \\ &= \frac{n!\, e^{\bar{\sigma}}}{\bar{\sigma}^n} \left(1 - \sum_{k=0}^{n-1} \frac{e^{-\bar{\sigma}} \bar{\sigma}^k}{k!}\right) \\ &= \frac{n!}{\bar{\sigma}^n} \sum_{k=n}^{\infty} \frac{\bar{\sigma}^k}{k!} \\ &= \frac{n!}{\bar{\sigma}^n} \sum_{k=0}^{\infty} \frac{\bar{\sigma}^{k+n}}{(k+n)!} \\ &= \sum_{k=0}^{\infty} \frac{n!}{(n+k)!} \bar{\sigma}^k \\ &= \text{hyp1f1}(1, n+1; \bar{\sigma})\end{aligned}$$

Since `hyp1f1` is not available in PyTorch, we can simply use the series expansion.

Listing: Computing `hyp1f1` using series expansion

```
def hyp1f1_1_nplus1_vec(x, n, K=500):
    # x: scalar tensor, n: (batch,) tensor
    device = x.device
    n = n.unsqueeze(1).to(torch.float64)  # shape (batch, 1)
    x = x.unsqueeze(1).to(torch.float64)  # shape (batch, 1)

    # create matrix of denominators of shape (*batch, K), where *batch is the leading dimensions
    ↪  of x
    ks = torch.arange(K, dtype=x.dtype, device=device).unsqueeze(
```

```
        0
    )  # k=0..K-1, shape (1, K)
    den = n + 1 + ks  # shape (batch, K)

    # factors = x / (n+1+k)
    factors = x / den  # shape (batch, K)

    # compute cumulative product along k to get T_k/T_0
    cumfac = torch.cumprod(factors, dim=-1)  # shape (batch, K)

    # prepend T_0=1 to align
    T = torch.cat([torch.ones_like(n), cumfac], dim=-1)  # (batch, K+1)

    # sum over k
    return T.sum(dim=-1)  # shape (batch,)
```

E.3 Efficient Training

Efficient training of insertion language models requires careful consideration of engineering details. Here we discuss some low-level details that are essential for making the training practical.

Multi-worker collation on CPU using sparse tensors. Computing the token loss requires, for each gap in the corrupted sequence, identifying all original tokens that were removed from that gap. Equivalently, each gap must be paired with a vocabulary-sized multi-hot target vector that has 1s exactly on the tokens belonging to that gap. Since the number and identity of removed tokens vary irregularly across gaps and examples, these targets are most naturally assembled during collation by scanning each example using a for-loop and writing only the nonzero entries into a sparse representation.

In practice, this work is best done by CPU dataloader workers in parallel, while the GPU is reserved for the forward and backward passes. It is also important to keep these targets sparse until they have been transferred to the GPU: materializing the dense tensors of shape `(batch, seq_len, vocab)` on CPU leads to prohibitively high CPU memory usage in multi-worker settings and wastes CPU-GPU transfer bandwidth on tensors that are almost entirely zero, slowing down training. We therefore construct sparse targets during CPU collation and expand them only on the GPU when evaluating the loss.

With these optimizations, the training of insertion language models is as efficient as MDMs and ARMs in terms of examples per second.

E.4 Sampling

E.4.1 Multi-token insertions

Figure 7 visualizes how multiple insertions can be performed in a single tensorized step without for-loops using cumulative sums to determine shifted positions of existing tokens and the destinations of newly inserted tokens. The listing that follows shows a minimal PyTorch implementation of this core multi-token insertion update.

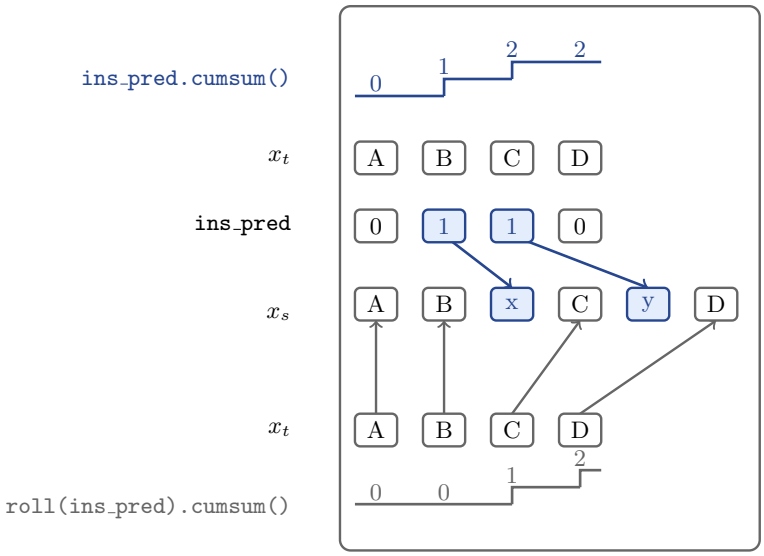


Figure 7: Illustration of the insertion update. The top panel isolates the insertion slots: `ins_pred.cumsum()` indexes the new positions that receive the inserted tokens. The bottom panel isolates the movement of existing tokens: `roll(ins_pred).cumsum()` counts earlier insertions and therefore determines how far each original token shifts to the right.

Listing: Core multi-token insertion update in PyTorch

```python
def insert_sampled_tokens(
    x_t: torch.Tensor,         # (batch, seq_len)
    ins_pred: torch.Tensor,    # bool mask: insert after this position?
    ins_tokens: torch.Tensor, # sampled token for each active insertion
    pad_token_id: int = 0,
) -> torch.Tensor:
    batch_size, seq_len = x_t.shape
    x_s = torch.full_like(x_t, pad_token_id)

    inc_positions = torch.arange(seq_len, device=x_t.device)
    inc_positions = inc_positions.unsqueeze(0).expand(batch_size, -1)

    # Existing tokens shift right by the number of insertions before them.
    existing_tokens_new_positions = (
        inc_positions + ins_pred.roll(shifts=1, dims=-1).int().cumsum(dim=-1)
    ).clamp(max=seq_len - 1)

    # Inserted tokens go in the newly opened slots.
    inserted_tokens_new_positions = (
        inc_positions + ins_pred.int().cumsum(dim=-1)
    ).clamp(max=seq_len - 1)

    batch_index = torch.arange(batch_size, device=x_t.device)
    batch_index = batch_index.unsqueeze(-1).expand(-1, seq_len)

    x_s.scatter_(dim=-1, index=existing_tokens_new_positions, src=x_t)
    x_s[batch_index[ins_pred], inserted_tokens_new_positions[ins_pred]] = (
        ins_tokens[ins_pred]
    )
    return x_s
```

E.5 Data preprocessing

Graph traversal The data for the graph traversal task is obtained from Patel et al. (2025).

Language modeling Following Patel et al. (2025), we train one model on LM1B tokenized with BERT tokenizer, with sequences truncated to a maximum length of 128 tokens. The other model is trained on OpenWebText-1024 split, which is the same as OpenWebText but tokenized using GPT-2 tokenizer and keeping sequences of length 1024 tokens or less. This preserves about 80% of the corpus. We use the xLM library (Patel et al., 2026) for the training and evaluation boilerplate code.

E.5.1 Hyperparameters

Table 4 lists the key training hyperparameters.

	LM1B	OpenWebText-1024	STAR (easy)	STAR (medium)	STAR (hard)
Per-device batch size	128	32	64	64	64
Num. GPUs	8	8	1	1	1
Global batch size	512	512	64	64	64
Output length	128	1024	14	16	28
Input length	—	—	28	36	116
Noise schedule ($\sigma_{\max}, \sigma_{\min}, \varepsilon$)	500, 5, 10^{-4}	3000, 50, 10^{-4}	30, 3, 10^{-3}	30, 3, 10^{-3}	50, 5, 10^{-3}
Training steps	$1000k$	$300k$	$80k$	$80k$	$80k$
Learning rate	5×10^{-5}	10^{-5}	10^{-4}	10^{-4}	10^{-4}
Weight decay	0.03	0.03	0.01	0.01	0.01
Adam ε	5×10^{-6}	5×10^{-6}	—	—	—
LR schedule	Cosine + min LR	Cosine + min LR	Constant	Constant	Constant
Minimum LR (cosine floor)	10^{-6}	10^{-6}	—	—	—
LR warmup steps	2000	2000	500	500	500
Cosine cycles	4.0	4.0	—	—	—

Table 4: Training hyperparameters for all models (noise schedule only applicable to DILM-S and DILM-M).

We picked reasonable defaults for the hyperparameters, and it is possible that the results could be improved with more careful hyperparameter tuning.

E.5.2 Computing infrastructure

For the language modeling experiments on LM1B and OpenWebText-1024, we trained all the models on 8 A100-80GB GPUs with 'torch.compile' using the hyperparameters in Table 4. The training for 1 million steps on LM1B took approximately 28–30 hours, and on OpenWebText-1024 training for 300 thousand steps took approximately 72–80 hours. For the experiments on star graphs, we use a single A100-80GB GPU, with the training taking 2–3 hours.

F Unconditional Generation Examples

F.1 DILM-S on OpenWebText

The following are unconditional samples from DILM-S on OpenWebText trained for 300,000 steps.

```
A small woman said on Tuesday that a very great, very bad man had been looking for him
with her husband, who has been playing with his son, he said.
"He's not really going to have done his whole job in this case, and he has always said
that, but he's not talking about him," he said.
"He's got the great man working up with him, and getting him back out there, and he is not
talking about himself, because he is on the edge of him, even when you have to call him
up, and it's all the right thing.'
"And he's not sitting on the good man, which is that he has been a really good son, but
that his son has always come out of him, and that he is his son, his son, and his wife."
"I think that he is going to have him, right now, that he's not like me, because he's not
looking for me.
```

"I think that he's not looking at him, that they are just going to play with him, and that they are not looking for his parents, and when I'm not running up with him, this is a lot of a so-called good man. He's going to be a really good kid, and because they're not getting out at me, he's a very good man, and it is really great that he is going to have it, but I just feel like being a very bad man."
"He's just right here in that room, because he has never been a really good guy, and as he goes on, he's just going to be getting out behind his son," he said, looking at a very bad man, and saying that he's always been told that he is only running up on his side," he said.
"He said that, "I will all see what he is going into being out there."
"He's not really going to see him, and he is not going to be the same guy, and he wouldn't be going to have to see him.
"I'd like to know that, and he's going to be out there, and he's not saying that, and he really knows that, and for sure, you're not in that room."

The next day, I don't have signed up for the first time now. You are "signing it up in here, and you know that there is not anything going on from it, and that the company that's ever caught up with me being a great group," or any other part of that episode (which it just fades around) seems to be going off to make my view of this event, and some of you have been living in that episode, even after bringing it up here, in some time.
"The only way we are going to be doing it is that you are going to go down right here, and we know that you are not really stepping up with them."
Because I don't think that it's just going on in all the good news, and as if it is actually going to go down here? "And if you will go back to thinking about it," it's always going to find it up again.

F.2 DILM-S on LM1B

Dataset: LM1B; NLL = 2.36; Entropy = 3.42
the first quarter of 2008. million of in the fourth quarter 2007. the first quarter of 2009. per share for the quarter of the year. second quarter 2009 results are currently reported. was 346. 4 million. from $ 15. 7 million in the second quarter of 2009 and the first quarter of 2008. for the second quarter ended june,. quarter as compared to the same period in 2008. million from $ 46. 6 million for the same period in 2007. total revenue of $ 50. 9 million, an increase of 3. 1 % compared. % compared to the company ' s for the 2008 third quarter.

Dataset: LM1B; NLL = 2.58; Entropy = 3.73
year - to - quarter operating expenses in the hyatt quarter increased by 1. 6 percent in the first three months of 2008, and for the third quarter 2009 of $ 7. 7 million, this was a 2. 7 % increase from the third quarter of 2008 due to net operating expenses of $ 22. 7 million, an increase of $ 38. 9 million due to higher operating costs and other related to general and administrative costs. reversing an rise in net income in the fiscal year. in the same period 2008, total revenues of $ 12 1 million for the second quarter was increased by 12. 8 %.

Dataset: LM1B; NLL = 3.80; Entropy = 3.94
" if they want to do that, since the end of 2008, they will know that the global economic downturn will be running at between 4 % and only 4 % this year, and that the world economy will be struggling, " said fernando santiago korobattimimo, the head of argentina ' s national environment and environment administration, at the official press conference in brazil, one of the largest developed economies of these two countries.

Dataset: LM1B; NLL = 4.06; Entropy = 4.18

the thing on the west end of this debate, especially, given all the important events that has already happened in south carolina is that we don ' t vote on the president ' s own performance at the democratic levels as well) - - and, compared with a lot of voters and sales voters in arizona in the past elections, we know that, as the report showed, we aren ' t going to do the same for all the delegates in and florida because they won in massachusetts, they looked on to change america and weather the storm even before obama was about to change his strategy and win in iowa and lead in the general election.

Dataset: LM1B; NLL = 4.52; Entropy = 4.11

the move to create this fuel cell product, because of the fast fuel supply chain that has cut gas costs, improved fuel efficiency and enough fueling capacity to replace the smart car vehicles, will have touched off clear assurances made to the local gas industry that if it was no longer made more efficient by the threat of economic and climate changes in japan and other large developed nations, they would be seen as a high earner because of a drop in fuel - like demand.

Dataset: LM1B; NLL = 4.96; Entropy = 4.03

and in their time, with so much talk of an interest rate cut, the concern that fed ' s aggressive rate cuts, including a dramatic cut in interest rates, and a slow recovery for an already troubled economy, will bring many economic analysts to a close eye on them in a conference call friday - - hours after news earlier this week - - that the fed could keep raising interest rates and cut borrowing and cut spending through this burden at the same time as holding a " smart line, " to start with the economy, the economy, the economy and the economy, which will each increase demand for the economy for years to come.